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

Romain Gay[†] Aayush Jain[‡] Huijia Lin[§] Amit Sahai[¶]

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).[1] 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 subject to extensive cryptanalysis since much before our work. 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 introduce a simple new technique for proving leakage resilience when polynomial-size noise is used to hide small secrets (for example, to hide LWE-based FHE decryption errors).
- 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.

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 security amplification, 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 $i\mathcal{O}$.

[*]Please note that this paper subsumes the ePrint article published by three of the present authors in [JLS19].

[†]Cornell Tech. Email: `romain.rgay@gmail.com`.

[‡]UCLA and NTT Research. Email: `aayushjain@cs.ucla.edu`.

[§]UW. Email: `rachel@cs.washington.com`.

[¶]UCLA. Email: `sahai@cs.ucla.edu`.

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

Contents

1 Introduction

This paper studies the notion of indistinguishability obfuscation ($i\mathcal{O}$) for general programs computable in polynomial time [BGI$^+$01, GKR08, GGH$^+$13b], 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 [BGI$^+$01] in 2001. However, until 2013, there was not even a single candidate construction known. This changed with the breakthrough work of [GGH$^+$13b]. Soon after, the floodgates opened and a flurry of over 100 papers were published reporting applications of $i\mathcal{O}$ (e.g. [SW14, BFM14, GGG$^+$14, HSW13, KLW15, BPR15, CHN$^+$16, GPS16, HJK$^+$16]). 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* [GGH$^+$13b]: 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, [BPR15] constructed hard instances for the End Of the Line (EOL) problem assuming subexponentially secure $i\mathcal{O}$ and one-way functions.

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, as summarized in Table 1, and discussed further below. So far, despite of a lot of progress, before our work, no known $i\mathcal{O}$ constructions were based on assumptions that have all above features.

Our new assumption. In this work, 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 [Reg05], and subexponentially secure SXDH and bilateral DLIN assumptions over bilinear maps [Jou00, BF01]. Let us now describe, informally, our new assumption. In this introductory description, we will omit discussion of parameter choices; however, they are

	Complex Assumptions[‡]		Simple-to-State Assumptions[‡]	
MMap	poly-deg MMap	[GGH+13b] ... [§]	poly-deg Mmap*	[GLSW14, PST14a]
	O(1)-deg MMap	[Lin16, AS17]	O(1)-deg MMap*	[LV16, Lin17, LT17]
No MMap	Direct Construction	[GJK18, BIJ+20a]	This work	
	Noisy Linear FE	[Agr19, AP20a]		
	ΔRG (or PFG)[†]	[AJL+19, JLMS19]		
	Split FHE	[BDGM20]		

[‡] In this table, every assumption categorized as complex is instance dependent and/or consists of a family of an exponential number of assumptions; every assumption categorized as simple is falsifiable, instance independent, and truly a single assumption.

[*] These assumptions over MMaps, even with degree 3, currently either are broken, or quite complex [MZ18]. Note that this is important because the description of the MMap must be a part of description of the assumption.

[†] The security of ΔRG and LWE with leakage on errors through ΔRG in [AJL+19, JLMS19] are families of exponentially many assumptions. With a simple modification, they can be reduced to families of polynomially many assumptions. Here, we categorize these works according to assumptions stated in the papers.

[§] See introduction for an extensive list of references.

Table 1: Summary of IO constructions and key cryptographic objects they rely on.

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[2] 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 [Gol00] 20 years ago. The complexity theory and cryptography communities have jointly developed a rich body of literature on the cryptanalysis and theory of constant-locality Boolean PRGs [Gol00, MST03, ABR12, BQ12, App12, OW14, AL16, CDM+18]. 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 last decade [MP13, AG11, CTA19, CSA20]. Our new assumption, combining Boolean PRGs and LWBE, is as follows:

The G-LWEleak-security assumption (informal).

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

$$\approx \quad \Big(\{\boldsymbol{a}_i, \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle + e_i \bmod p\}_{i \in [n]}, \qquad G, \boldsymbol{r} \Big) \quad // \boldsymbol{r} \leftarrow \{0,1\}^m \tag{2}$$

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

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 Section 5.4, 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 [AG11], and also avoid all known algebraic attacks [CTA19]. Indeed, the parameter choices we make are not possible using the previous work of [JLMS19], and the parameters used in [JLMS19] would render LWBE insecure.

Comparison to previous $i\mathcal{O}$ constructions. We now elaborate on Table 1 with an overview of the assumptions underlying previous constructions of $i\mathcal{O}$, and how these compare with our work.

Initial works [GGH13a, GGH$^+$13b, BGK$^+$14, BR14, PST14b, AGIS14, BMSZ16, CLT13, CLT15, GGH15, CHL$^+$15, BWZ14, CGH$^+$15, HJ15, BGH$^+$15, Hal15, CLR15, MF15, MSZ16, DGG$^+$16] constructed candidate $i\mathcal{O}$ using high-degree multilinear maps with heuristic or "generic model" arguments of security, and studied attacks on these candidates [CHL$^+$15, BWZ14, CGH$^+$15, HJ15, BGH$^+$15, Hal15, CLR15, MF15, MSZ16].

The work of [GLSW14] proposed clean and instance-independent assumptions in the context of multilinear maps, which unfortunately was found to be broken when instantiated with then-known multilinear map candidates [CHL$^+$15, CLR15, CGH$^+$15, BWZ14]. The work of [PST14b] formulated the semantic security of multilinear map, which is falsifiable and instance independent, but nevertheless similar in spirit to the Uber assumption. On the other hand, multilinear maps of degree 2 – bilinear maps – are well-understood objects that have been used extensively in cryptography, and for which we have standard computational hardness assumptions. Naturally, research focused on decreasing the degree of the multilinear map used to build $i\mathcal{O}$, down to a constant [Lin16, LV16, AS17, Lin17, LT17] — note that prior constructions required a multilinear map whose degree grew with the size of the obfuscated circuits. This line of work, initiated by Lin [Lin16], led to the work of [LT17], which builds $i\mathcal{O}$ from a natural assumption called SXDH over 3-linear maps. Again unfortunately, even this qualitatively weaker assumption is known to be broken when instantiated with existing multilinear map candidates [BWZ14, CHL$^+$15, CLR15, CGH$^+$15]. Alternatively, one can instantiate the multilinear maps in this line of works with the complex candidate multilinear maps of [MZ18] that are themselves based on "immunized obfuscation" techniques and "weak generic multilinear map models" of [MSZ16, DGG$^+$16], but this would involve incorporating the complex multilinear map candidates into the hardness assumptions.

A number of recent works [GJK18, AJS18, Agr19, LM18, JLMS19, BIJ$^+$20b, AP20b, BDGM20] circumvent the use of multilinear maps. The works of [GJK18, BIJ$^+$20b] gave direct constructions of $i\mathcal{O}$ using new mathematics, but with only heuristic security arguments – where essentially the underlying assumption is that the $i\mathcal{O}$ scheme itself is secure. The works of [Agr19, AP20b] and [BDGM20] proposed new primitives called noisy linear FE and split FHE respectively, which are sufficient for $i\mathcal{O}$ when combined with standard assumptions, and gave heuristic instantiations of these new primitives. While noisy linear FE and split FHE are significantly simpler and apparently weaker than $i\mathcal{O}$, their security is not known to rely on a simple, instance-independent, single assumption.

Noisy linear FE [Agr19] allows encrypting a vector $\boldsymbol{v}$ in a ciphertext ct and releasing many secret keys sk_i, each of which associated with a vector $\boldsymbol{u}_i$, such that decryption reveals the inner product $\langle \boldsymbol{v}, \boldsymbol{u}_i \rangle + \mathsf{noise}_i$ perturbed by some noise dependent on $\boldsymbol{v}$ and $\boldsymbol{u}_i$. Security guarantees that ciphertexts for two different vectors $\boldsymbol{v}$ and $\boldsymbol{v}'$ are indistinguishable as long as they have approximately the same (instead of exactly the same) inner product with the vectors tied to the secret keys, i.e., $|\langle \boldsymbol{v}, \boldsymbol{u}_i \rangle - \langle \boldsymbol{v}', \boldsymbol{u}_i \rangle| \leq B$ for some fixed polynomial bound B. As such, the security corresponds to a family of exponentially many assumptions, one for each possible combination of vectors $\boldsymbol{v}, \boldsymbol{v}'$ and $\boldsymbol{u}_i$'s satisfying the constraint. We note that [Agr19], when combined with techniques from [AJL$^+$19, JLMS19] or this work, also points to a pathway to $i\mathcal{O}$ if there exists 2-block-local PRG with appropriate stretch that is not ruled out by existing attacks [BBKK17, LV17a]. However, there are currently no unbroken instantiation of such 2-block-local PRGs (and hence omitted in Table 1).

On the other hand, the notion of split FHE proposed by [BDGM20] is as follows: Using an FHE scheme, one can homomorphically evaluate many circuits $C_1, \cdots, C_n$ on ciphertext ct' of a message m and obtain ciphertext ct of outputs $y_1, \cdots, y_n$. In a split FHE, decryption contains two syntactical steps: i) the first secret step uses the secret key, circuits $C_1, \cdots, C_n$, and the ciphertext ct, to produce a decryption hint ρ, whose length is sub-linear in the length of the outputs (e.g., $|\rho| = |y_1, \cdots, y_n|^{1-\epsilon}$ for some $\epsilon > 0$), then ii) the second public step recovers the outputs from the decryption hint and the ciphertext ct. Importantly, the decryption hint ρ which is made pubic should not hurt the semantic security of ct nor ct'. More precisely, for any two messages m_0, m_1 that produce the same outputs through $C_1, \cdots C_n$, their ciphertexts should be indistinguishable given the hint ρ for the the output ciphertext. This security, again, corresponds to a family of exponentially many assumptions, one for each combination of messages and circuits.

Finally, closest to our work is the line of works by [AJL$^+$19, Agr19, LM18, JLMS19], which gets us close to having simple assumptions. They proposed a new way to construct $i\mathcal{O}$ without multilinear maps, but instead by conjecturing and leveraging novel pseudorandomness properties of low-degree polynomials over the *integers*. In the most recent work by [JLMS19], $i\mathcal{O}$ is constructed from a new assumption, in addition to three standard assumptions: (1) subexponential security of succinct assumptions over bilinear maps, (2) subexponential security of constant-locality Boolean pseudorandom generators with polynomial expansion, and (3) subexponential security of LWE.

The new assumption of [JLMS19] postulates that there exist polynomials $Q : \mathbb{Z}^n \to \mathbb{Z}^m$ of constant-degree and polynomial stretch (i.e, $m = n^{1+\epsilon}$) satisfying a weak pseudo-randomness property, called weak perturbation-resilience: the outputs $\boldsymbol{r} = Q(\boldsymbol{x})$ can be used as "flooding" noises to partially hide a smaller vector $\boldsymbol{v}$ by considering $\boldsymbol{r} + \boldsymbol{v}$. This property is then combined with the LWE assumption as follows: for every small integer vector $\boldsymbol{v} \in \mathbb{Z}^m$ that is B bounded (i.e., $\|\boldsymbol{v}\|_\infty \leq B$), no efficient adversary can distinguish the following two distributions with larger than 0.99 advantage, where χ is a narrow discrete Gaussian distribution, and λ is the security parameter that is polynomially related with, but much smaller than n.

$$\begin{aligned} &\Big(\{\boldsymbol{a}_i, \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle + e_i \bmod p\}_{i \in [n]}, \qquad Q, Q(\boldsymbol{e}) \text{ over } \mathbb{Z} \Big) \quad // \boldsymbol{e} = (e_1, \ldots, e_n) \leftarrow \chi^n, \boldsymbol{a}_i, \boldsymbol{s} \leftarrow \mathbb{Z}_p^\lambda \\ \overset{\text{weakly}}{\approx} \; &\Big(\{\boldsymbol{a}_i, \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle + e_i \bmod p\}_{i \in [n]}, \qquad Q, Q(\boldsymbol{e}) + \boldsymbol{v} \text{ over } \mathbb{Z} \Big) \quad // \boldsymbol{v} \in ([-B, B] \cap \mathbb{Z})^m \end{aligned}$$

The assumption of [JLMS19] is succinct, falsifiable, and instance independent, however, is a family of exponentially many assumptions, one for each vector $\boldsymbol{v}$ with small magnitude. In addition, the assumption has several novel, and therefore relatively unstudied, aspects:

1. It gives the adversary LWE samples with "leakage" on their noises $Q(\boldsymbol{e})$ through a function with some pseudo-randomness property.

2. It postulates pseudo-randomness property of constant-degree integer polynomials. Usually, cryptographic pseudo-random objects are defined over a finite field like $\mathbb{Z}_p$ for a prime p. Consequently, there were no previous cryptanalysis literature to rely on when selecting candidate polynomials. Moreover, computation over the integers may open the door to more attacks.

For instance, degree-2 integer polynomials were successfully attacked using Sum-Of-Square algorithms [BBKK18, BHJ+19].

3. The integer polynomials satisfy perturbation-resilience, which is a new, and therefore relatively unstudied, weak pseudo-randomness property. The weakness of the pseudorandomness property is required because no known constant-degree polynomial over the integers satisfy the usual strong pseudorandomness properties satisfied by standard PRGs.

While it is interesting and important to study the pseudo-randomness properties of integer polynomials and the security of the combined assumptions above, at this stage in the development of $i\mathcal{O}$, a primary goal is diversifying the set of assumptions sufficient for $i\mathcal{O}$ and basing $i\mathcal{O}$ on hard computational problems that have as rich a history of study as possible. To this end, we formulate our new assumption (Equation (1)) that is qualitatively different from the above assumption, and replaces integer polynomials with the more standard notion of a Boolean PRG when combined with LWE with binary errors, and show that it is sufficient for $i\mathcal{O}$.

COMPARISON BETWEEN OUR WORK AND [JLMS19]. Let us compare our assumption with the assumption used in [JLMS19]. Our new assumption retains the unusual aspect (1) that the adversary sees LWBE samples with leakage on the noises, now through a PRGs. However, it mitigates the unusual aspect (2) by replacing the use of constant-degree integer polynomials with constant-degree Boolean PRGs, which has a rich history of study. It also addresses the unusual aspect (3) by eliminating the need for a new notion of weak pseudo-randomness, and replace it with standard pseudorandomness. Both of the two ingredients, namely, the security of Goldreich's PRG and the security of LWE with binary errors have been studied for over a decade. While studies on each ingredient individually do not directly justify the security of our new assumption (which combines both), the rich literature on the cryptanalysis of Goldreich's PRG [Gol00, MST03, ABR12, BQ12, App12, OW14, AL16, CDM+18] and LWE with binary error [ACF+15, MP13, AG11, CTA19] provide ample techniques for attacks, defenses, and analysis. Guided by them, we suggest concrete candidates PRGs and LWBE parameters, and verify that the resulting assumption withstands a rich body of cryptanalysis techniques. In comparison, cryptanalysis on integer polynomials started only after the recent works (see [BHJ+19]).

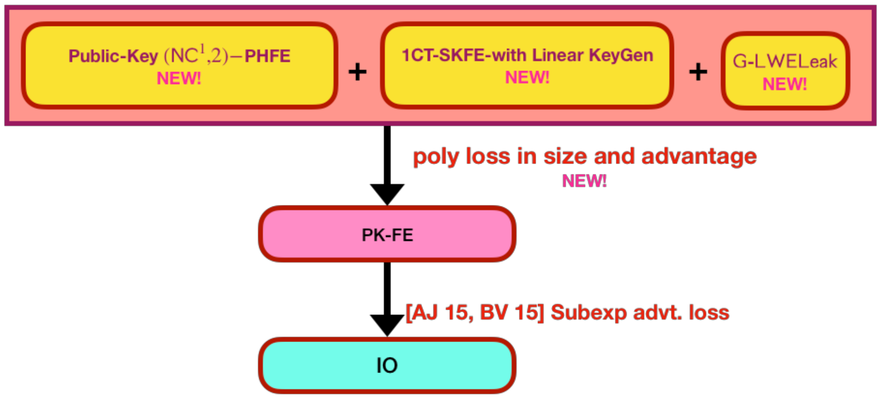
Public-Key (NC^1,2)−PHFE
NEW!
+
1CT-SKFE-with Linear KeyGen
NEW!
+
G-LWELeak
NEW!
poly loss in size and advantage
NEW!
PK-FE
[AJ 15, BV 15] Subexp advt. loss
IO

Figure 1: Our Framework.

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

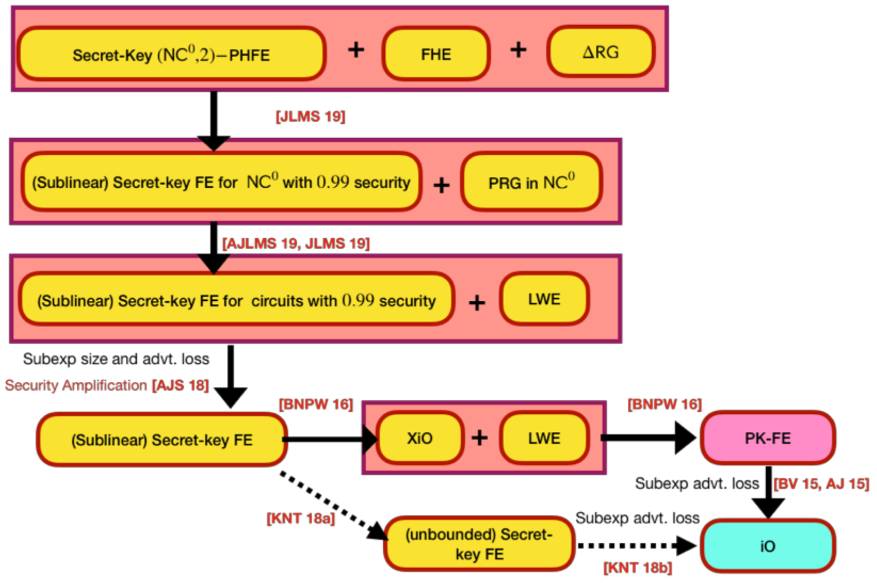
Secret-Key (NC^0,2)−PHFE
+
FHE
+
ΔRG
[JLMS 19]
(Sublinear) Secret-key FE for NC^0 with 0.99 security
+
PRG in NC^0
[AJLMS 19, JLMS 19]
(Sublinear) Secret-key FE for circuits with 0.99 security
+
LWE
Subexp size and advt. loss
Security Amplification [AJS 18]
[BNPW 16]
(Sublinear) Secret-key FE
XiO
+
LWE
[BNPW 16]
PK-FE
Subexp advt. loss
[BV 15, AJ 15]
[KNT 18a]
(unbounded) Secret-key FE
Subexp advt. loss
[KNT 18b]
iO

Figure 2: Framework of the construction [JLMS19] to achieve functional encryption and obfuscation.

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 [AJ15, BV15], which shows that $i\mathcal{O}$ can be build from Functional Encryption [SW05], a primitive that was originally formulated by [BSW11, O'N10]. 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 $i\mathcal{O}$ [AJ15, BV15]. For example, as illustrated in Figure 2, the work of [JLMS19] 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. Figure 2 depicts the blueprint of $i\mathcal{O}$ construction in [JLMS19].

An important factor that contributed to the complexity is the weakness of the pseudo-randomness property of integer polynomials – it only partially hides, hence partially leaks, secret values (denoted by $\boldsymbol{v}$ above) to be protected. To compensate for the leakage, previous constructions rely on heavy machinery, such as dense model theorems and advanced secret sharing schemes where it is possible to compute directly functions over individual shares to obtain shares of the outputs.

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.

Our contributions in a nutshell. We provide a simpler, more direct construction of $i\mathcal{O}$. We do this by formulating a new assumption, together with the standard assumptions of subexponential LWE and subexponentially secure bilinear maps. Our new assumption is built upon computational problems that are qualitatively different from and more extensively studied than that used in prior works. In particular, we replace the use of constant-degree polynomials over the integers having weak pseudorandomness properties, with simply constant-degree Boolean PRGs, which has been studied since 2000 [Gol00]. We also rely on the LWE assumption with *binary* errors, a natural strengthening of the standard LWE (with small integer errors) that has been studied for the last decade, see for instance [MP13, AG11, CTA19]. We combine them into a new assumption that is simple to state, and instance-independent, and use it to prove $i\mathcal{O}$ security. On the front of simplifying $i\mathcal{O}$ constructions, we give a direct construction of full-fledged FE needed by previous works [AJ15, BV15] for the implication to $i\mathcal{O}$. Notably, our direct construction gives an FE that i) is public-key ii) handles the generation of function keys associated with functions computable by *any polynomial-size circuit*, iii) guarantees *standard security* from the *polynomial hardness* of the underlying assumptions. Hence, we circumvent the costly generic transformations for "lifting" the function class supported and the security level applied in prior constructions, and avoid heavy machinery such as dense model theorems and advanced secret sharing. This leads to simpler constructions of both FE and $i\mathcal{O}$, whose blueprints are depicted in Figure 1.

1.1 Our Results

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

Theorem 1.1. *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 LWE 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.

New technique for hiding errors using polynomially bounded noises. A common technical problem encountered in previous $i\mathcal{O}$ constructions is: how to hide a vector of small integer values $\boldsymbol{v} \in \mathbb{Z}^m$ of some bounded magnitude B', using another vector $\boldsymbol{r} \in \mathbb{Z}^m$ of larger but still polynomially bounded magnitude, by adding them together $\boldsymbol{r} + \boldsymbol{v}$. Information theoretically, the sum does not hide $\boldsymbol{v}$ completely. In this work, leveraging our new assumption, we use $\boldsymbol{r}$ that is uniformly distributed in $([0, B] \cap \mathbb{Z})^m$, where B is polynomially related to B', and show that this suffices (in reality this $\boldsymbol{r}$ will be generated using a Boolean PRG). We do so by proving a simple Bounded Leakage Resilience Lemma (see Lemma 2.1), which informally says the following: suppose the vector to be protected is statistically determined by some other value $\boldsymbol{c}$, that is, $\boldsymbol{v} = V(\boldsymbol{c})$ with respect to a potentially inefficient function V. Then, the sum $\boldsymbol{r} + \boldsymbol{v}$ can be efficiently simulated using $\boldsymbol{c}$ alone, that is, $(\boldsymbol{c}, \boldsymbol{v} + \boldsymbol{r})$ and $(\boldsymbol{c}, \mathsf{Sim}(\boldsymbol{c}))$ are indistinguishable w.r.t. an efficient simulator. This means if $\boldsymbol{c}$ computationally hides $\boldsymbol{v}$, it suffices to use polynomially bounded vector $\boldsymbol{r}$ to hide $\boldsymbol{v}$. We believe this simple lemma may be of independent interest.

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, 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 [AR17a, Agr19, LM18]. 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 (Theorem 7.2). 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 [SS10a, GVW12a], and several applications (e.g., [ABSV15]). 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 [GVW12b] where it was used to bootstrap FE schemes. A similar notion of partially-hiding predicate encryption was proposed and constructed by [GVW15]. PHFE beyond the case of predicate encryption was first constructed by [AJS18] 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 $\boldsymbol{x}$ (Theorem 8.1). 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 [JLMS19].

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 [ACGU20]. 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.

Simplification. We considerably simplify the path to construct public-key functional encryption and obfuscation. The overall framework in the prior works is given in Figure 2. In contrast, our framework is more direct and arguably simpler. This is depicted in Figure 1. The detailed explanation of these figures can be found in Section 2.7.

A tantalizing open question. Looking ahead, consider the following possibility: suppose it is possible to "separate" 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. This would give the first construction of $i\mathcal{O}$ relying on well-studied assumptions. We are optimistic about this possibility based in part on the beautiful work of [GKPV10], which showed that assuming separately LWE and sufficiently strong one-wayness, it is possible to establish leakage resilience of LWE where the leakage is on the LWE secret $\boldsymbol{s}$. What we would need is to find an analogue of this result for LWBE, that considers classes of leakage functions over the errors $(e_1, \dots, e_n)$ that contain Goldreich's PRGs.

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 [Agr19, AJL+19, JLMS19]. 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 [SS10b, GKP+13], or are limited to basic functions — namely, degree-2 polynomials at most, [BCFG17, Gay20] for the public-key setting, [Lin17, AS17] for the private-key setting[3]Furthermore, known HE schemes have very simple

[3]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

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 [ABDP15, ALS16], sublinearly compact FE that also computes the rounding are currently our of reach. Omitting this rounding would reveal $f(\boldsymbol{x})$, but also $\boldsymbol{e}_f$, 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 [AR17b], 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)

$$\begin{array}{rcl} \mathsf{fesk}(f) \text{ contains} & : & \mathsf{2fsk}(g) \\ \mathsf{fect}(\boldsymbol{x}) \text{ contains} & : & \mathsf{hect}(\boldsymbol{s}, \boldsymbol{x}),\ \ \mathsf{2fct}(\boldsymbol{s}||\mathsf{sd}) \end{array}$$

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 2.2,2.3,2.5,2.4, 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

still limited to degree 2.

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 [AJL+19, JLMS19] 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:

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

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. We explain this next.

Challenge 2: How to Hide Errors using Polynomial-sized Noises? The role of the noise generator NG is expanding out many smudging noises $\boldsymbol{r}$ to hide errors $\boldsymbol{e}$ as $\boldsymbol{r} + \boldsymbol{e}$. However, under the constraint that $\boldsymbol{r}$ is *polynomially* bounded, $\boldsymbol{r} + \boldsymbol{e}$ is noticably far from $\boldsymbol{r}$, meaning that $\boldsymbol{e}$ cannot be completely hidden (e.g., one can distinguish whether $\boldsymbol{e}$ is zero or non-zero with noticeable probability). Previous works [AJL+19, JLMS19] formulated weak NG security notions, perturbation resilience [AJS18] and pseudo-flawed smudging [LM18], to capture that $\boldsymbol{r} + \boldsymbol{e}$ only partially hides $\boldsymbol{e}$. In all known constructions, this is a source of inefficiencies. Typically one uses security amplification transformations such as the one in [AJS18], to deal with such security properties. Further, this also is a source of making stronger versions of standard assumption as in order to argue security the hardness amplification transformations typically lose a subexponential factor in the size of the adversary.

On the other hand, using our degree-2.5 Boolean PRG PRG, we show how to hide errors using poly-sized noises, through a much simpler *bounded leakage resilience technique*[4], so that our FE

[4]Although we still use ideas from the Dense Model Theorem.

construction does not need to rely on a general purpose amplification theorem. Using the new technique, we achieve standard polynomial security for our FE construction based on polynomial hardness. More specifically, suppose the errors are B'-bounded, given a random Boolean vector $\boldsymbol{r}'$ (which will be generated by our degree-2.5 PRG), we hide errors by choosing a sufficiently large B that is polynomially related to B' and m (the length of $\boldsymbol{e}$) and compute:

$$\boldsymbol{e}+\boldsymbol{r} \qquad \text{where } r_j = \Sigma_{k=0}^{\log B-1} 2^k r'_{(j-1)\log B+k} \text{ where } r'_l \text{ is the } l\text{'th bit in } \boldsymbol{r}' \ .$$

Since the $\boldsymbol{r}_j$'s are independently and randomly distributed over $[0, B-1] \cap \mathbb{Z}$, it can be shown that at only a constant number of coordinates j, e_j is leaked, and at all other coordinates, $e_j + r_j$ information-theoretically hides e_j. From here, we prove the following bounded leakage resilience lemma, which says that if additionally there is a "commitment" $\boldsymbol{c}$ that statistically binds $\boldsymbol{e}$, then the leakage can be efficiently simulated using $\boldsymbol{c}$ alone. Hence, $\boldsymbol{e}$ is hidden as long as $\boldsymbol{c}$ hides it.

Lemma 2.1 (Bounded Leakage Resilience Lemma)**.** *Let $B', m, s \in \mathbb{N}$, $\epsilon > 0$. Let $B \geq (B'+m)^c$ for a sufficiently large constant c. Then, for every distribution $D_{\boldsymbol{c}}$ over $\{0,1\}^k$ and function $V : \{0,1\}^k \to ([-B', B'] \cap \mathbb{Z})^m$ (both potentially inefficient), there exists a simulator* Sim*, such that:*

1. Sim *has size bounded by $s' = \mathsf{poly}(B,m) \cdot \epsilon^{-2} \cdot s$, and*
2. *The following two distributions are (s,ϵ)-indistinguishable*[5]

$$\{\boldsymbol{c} \leftarrow D_{\boldsymbol{c}},\ \boldsymbol{e} \leftarrow V(\boldsymbol{c}),\ \boldsymbol{r} \leftarrow ([0,B-1]\cap\mathbb{Z})^m \ :\ \boldsymbol{c},\ \boldsymbol{e}+\boldsymbol{r}\ \} \qquad \text{and} \qquad \{\boldsymbol{c} \leftarrow D_{\boldsymbol{c}} \ :\ \boldsymbol{c},\ \mathsf{Sim}(\boldsymbol{c})\}$$

We emphasize again that the magnitude of the smudging noise $\boldsymbol{r}$ is polynomial $B = \mathsf{poly}(B', m)$. Moreover, simulation is relatively efficient comparing with the distinguishers, with a $\mathsf{poly}(B,m)\cdot\epsilon^{-2}$ factor slowdown. Therefore if $\boldsymbol{c}$ computationally hides $\boldsymbol{e}$ against $(\mathsf{poly}(B,m)\epsilon^{-2}s)$-size adversaries, $\boldsymbol{c}, \boldsymbol{e}+\boldsymbol{r}$ computationally hides $\boldsymbol{e}$ against s-size adversaries. Consider a more concrete example where $\boldsymbol{c} = \mathsf{hect}(\boldsymbol{s}, \boldsymbol{x})$ and $\boldsymbol{e} = \boldsymbol{e}_f$. Since the former statistically binds the latter (as hect binds $\boldsymbol{s}, \boldsymbol{x}$ and $\boldsymbol{e}_f$ is a function of hect, $\boldsymbol{s}, \boldsymbol{x}$, and f), by our lemma, as long as hect is sufficiently hiding, smudging with poly-sized noises $\boldsymbol{e}+\boldsymbol{r}$ suffices to hides $\boldsymbol{e}$ completely.

Challenge 3: 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 2FE. The notion of Partially-Hiding Functional Encryption PHFE, originally introduced by [GVW15] 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 [AJL+19, JLMS19] 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 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 2.1 (Public-key (NC^1, 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.*

[5]That is, ϵ-indistinguishable to all s sized distinguishers.

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 4: How to Ensure Integrity? Now that we have replaced 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 P with *linear key generation*, denoted as 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 $\mathsf{1LGFE.ct}$
$\mathsf{KeyGen}(\mathsf{pp}, \boldsymbol{s}, f)$	:	$\mathsf{pp}_f \leftarrow \mathsf{EvalPP}(\mathsf{pp}, f)$, $\boldsymbol{r} \leftarrow ([0, B-1] \cap \mathbb{Z})^m$, output f and secret key $\mathsf{1LGFE.sk}(f) = \langle \mathsf{pp}_f, \boldsymbol{s}\rangle - \boldsymbol{r}$
$\mathsf{Dec}(\mathsf{1LGFE.ct}, (f, \mathsf{1LGFE.sk}))$	:	$\mathsf{1LGFE.ct}_f \leftarrow \mathsf{EvalCT}(\mathsf{1LGFE.ct}, f)$ output $\frac{q}{2}\boldsymbol{y} + \boldsymbol{e}_f + \boldsymbol{r} \leftarrow \mathsf{1LGFE.ct} - \mathsf{1LGFE.sk}$, $\lvert \boldsymbol{e}_f \rvert_\infty \le B'$

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 $\mathsf{1LGFE.ct}$ *to obtain an intermediate ciphertext* $\mathsf{1LGFE.ct}_f$*, and then simply subtracts the secret key from* $\mathsf{1LGFE.ct}_f$*, 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 $\mathsf{1LGFE.ct}(\boldsymbol{s}, \boldsymbol{x})$ of 1LGFE. By the correctness and security of 1LGFE, revealing $\mathsf{1LGFE.sk}(f)$ 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.

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

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

One more technical caveat is that known pairing-based PHFE schemes actually compute the secret key $\mathsf{1LGFE.sk}$ in the exponent of a target group element, which we denote by $[\mathsf{1LGFE.sk}]_T$, 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 1LGFE — namely, it is linear in $\mathsf{1LGFE.sk}$ — these group elements are sufficient for decryption. A decryptor can first compute $\mathsf{1LGFE.ct}_f$ from $\mathsf{1LGFE.ct}(\boldsymbol{s}, \boldsymbol{x})$ and f in the clear, then perform the decryption by subtracting $[\mathsf{1LGFE.ct}_f - \mathsf{1LGFE.sk}]_T$ 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 [Agr19, LM18]. 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 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 two benefits. First, previous works constructed SHE by modifying the Brakerski-Vankuntanathan FHE scheme [BV11], but are limited to supporting NC^1 computations based on RLWE [AR17b], and NC_0 based on LWE [AR17b, LM18]. Instead, the FE lens led us to search for ideas in the predicate encryption literature. We show how to construct 1LGFE for P from LWE by modifying the predicate encryption scheme of [GVW15]. This new construction allowed us to construct FE for P directly without invoking any bootstrapping theorem from weaker function classes.

Theorem 2.2 (1LGFE from LWE, informal)**.** *There is a construction of a single-ciphertext FE for* P *with linear key generation as described above, from LWE.*

Second, constructing 1LGFE already requires us to resolve the challenge of hiding errors $\boldsymbol{e}_f$ with only poly-sized smudging noises $\boldsymbol{r}$. Indeed, we apply our bounded leakage resilience lemma (Lemma 2.1) in the construction of this simpler primitive to argue that poly-sized $\boldsymbol{r}$ is sufficient. This leads to a simpler and more modular proof for the overall FE construction.

In summary, putting all the pieces together, our construction of FE for P 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, no security amplification step).

Our FE for P Construction

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

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

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 [AG11] and then by [MP13, ACF+15, BGPW16, CTA19]. 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 Section 5, 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 [AL16, CDM+18].

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

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 $c > 0$ [KMOW17]. 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 Section 5 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 [LV17b] 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 [LV17b], 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 Section 5, 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.

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

We show in Section 5, 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.

Please refer to Table 2 for a summary of attacks on all these predicates as well as the combined assumption.

2.3 How to Hide Errors using Polynomial-sized Noises

Now we describe how to prove lemma 2.1. We recall it below.

Lemma 2.2 (Bounded Leakage Resilience Lemma)**.** *Let $B', m, s \in \mathbb{N}$, $\epsilon > 0$. Let $\mathsf{Bound} \geq B' \cdot m^3$. Then, for every distribution $D_{\boldsymbol{c}}$ over $\{0,1\}^k$ and function $V : \{0,1\}^k \to ([-B', B'] \cap \mathbb{Z})^m$ (both potentially inefficient), and for every $c \in \mathbb{N}$ there exists a simulator Sim, such that:*

1. *Sim has size bounded by $s' = O(\mathsf{poly}_c(m, B') \cdot \epsilon^{-2} \cdot s)$, and*
2. *The following two distributions are $(s, \epsilon + O(\frac{1}{m^c}))$-indistinguishable[6]*

$$\{\boldsymbol{c} \leftarrow D_{\boldsymbol{c}},\ \boldsymbol{e} \leftarrow V(\boldsymbol{c}),\ \boldsymbol{r} \leftarrow ([0, B-1] \cap \mathbb{Z})^m\ :\ \boldsymbol{c},\ \boldsymbol{e} + \boldsymbol{r}\ \} \quad \text{and} \quad \{\boldsymbol{c} \leftarrow D_{\boldsymbol{c}}\ :\ \boldsymbol{c},\ \mathsf{Sim}(\boldsymbol{c})\}$$

We now describe a sketch of the proof here. Let $\boldsymbol{c}$ be sampled as described in the lemma above using the distribution $D_{\boldsymbol{c}}$. Let $\boldsymbol{e} \leftarrow V(\boldsymbol{c})$. Denote $\boldsymbol{e} = (e_1, \ldots, e_m)$. Now the idea is that we consider the following process:

- Sample $r_i \leftarrow [0, \mathsf{Bound}]$ for $i \in [m]$ for some bound Bound which we set later.
- Set $t_i = e_i + r_i$ for $i \in [m]$. Set $\mathbf{T} = (t_1, \ldots, t_m)$. Output $(\boldsymbol{c}, \mathbf{T})$.

Our goal is to simulate this distribution efficiently. First we make the following compression argument.

Information compression. Since Bound is much bigger than B' and r_i is uniform in $[0, \mathsf{Bound}]$, sampling $t_i = e_i + r_i$ is equivalent to sampling uniformly from $[e_i, e_i + \mathsf{Bound}]$. This is also equivalent to sampling from.

- Sample t_i uniformly from $I = [B' + 1, \mathsf{Bound} - B' - 1]$ with probability $\alpha = \frac{\mathsf{Bound} - 2B' - 1}{\mathsf{Bound} + 1} = 1 - O(\frac{B'}{\mathsf{Bound}})$ and with probability $1 - \alpha$ from $[e_i, e_i + \mathsf{Bound}] \setminus I$.

Notice that if $\mathsf{Bound} >> B'$ then α is very large. We set $\frac{\mathsf{Bound}}{B'} = m^3$. Thus, using this we build another machine Mach that samples $\mathbf{T}$ as follows. It computes $\boldsymbol{e} = V(\boldsymbol{c})$. Then, it initializes a list $\mathbf{L}$ to be empty.

- Sample coins $\boldsymbol{\beta} \leftarrow \{0,1\}^m$ where each $\beta_i = 1$ with probability α.
- If $\beta_i = 0$ we sample uniformly $t_i \leftarrow [e_i, e_i + \mathsf{Bound}] \setminus I$ and append (i, t_i) into $\mathbf{L}$. Mach outputs $\mathbf{L}$.

Notice that $\mathbf{L}$ is the only information that one needs to sample $\mathbf{T}$ efficiently and identically to the original procedure as one can set $t_i = \ell_i$ if (i, ℓ_i) is in the list $\mathbf{L}$, otherwise set it to be a uniform sample from I. We call this polynomial time procedure as Samp. Thus $\mathbf{T} = \mathsf{Samp}(\mathsf{Mach}(\boldsymbol{c}))$

However, notice that:

$$\Pr[|\mathbf{L}| \geq k] \leq \binom{m}{k} \cdot (1 - \alpha)^k \leq O(\frac{1}{m^{2k}}) \cdot m^k$$

Thus, $\Pr[|\mathbf{L}| \geq c] \leq O(\frac{1}{m^c})$. Which means, that with very high probability the output of Mach is small.

[6]That is, ϵ-indistinguishable to all s sized distinguishers.

Why Information Compression Helps? We now recall the following theorem.

Theorem 2.3 (Imported Theorem [CCL18a]). *Let $k, t \in \mathbb{N}, \epsilon > 0$, and $\mathcal{C}_{leak}$ be a family of distinguisher circuits from $\{0,1\}^k \times \{0,1\}^t \rightarrow \{0,1\}$ of size $s(k)$. Then, for every distribution (X, Z) over $\{0,1\}^k \times \{0,1\}^t$, there exists a simulator $h : \{0,1\}^k \rightarrow \{0,1\}^t$ such that:*

1. *h is a circuit computable in size $s' = O(s \cdot 2^t \epsilon^{-2})$*
2. *(X, Z) and $(X, h(Z))$ are indistinguishable by $\mathcal{C}_{leak}$. That is, for every $C \in \mathcal{C}_{leak}$,*

$$\left| \Pr_{(x,z) \leftarrow (X,Z)}[C(x,z) = 1] - \Pr_{x \leftarrow X, h}[C(x, h(x)) = 1] \right| \leq \epsilon$$

Notice that the theorem above allows one to simulate auxiliary information Z about any distribution X. Crucially, the size of h is only slightly bigger than the size s if the length of Z is small. The idea is that, we can use this theorem to simulate the machine Mach where the size of the list $\mathbf{L}$ is constrained to be less than c (otherwise, the machine just gives up). We call this machine as $\mathsf{Mach}^{\leq c}$. Consider $\mathbf{T}' = \mathsf{Samp}(\mathsf{Mach}^{\leq c}(\boldsymbol{c}))$. Since the size of $\mathbf{L}$ is greater than c with probability less than $O(\frac{1}{m^c})$, the statistical distance between $\mathbf{T}$ and $\mathbf{T}'$ is bounded by $O(\frac{1}{m^c})$. Now we can invoke the theorem 7.6 above. We replace $\mathsf{Mach}^{\leq c}$ with h. Observe that size of output is bounded by $c \cdot (1 + 3 \cdot \log_2 m + \log_2 B')$. Thus, the size of h is $O(m^{3c} \cdot B'^c \cdot s \cdot \epsilon^{-2})$. Our required simulator Sim is $\mathsf{Samp}(h(\boldsymbol{c}))$. The claim follows because Samp is a polynomial time procedure.

2.4 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, that have the simple structure outlined in Section 2, denoted as $\mathsf{1LGFE}$, 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.

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

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 [AR17b, LM18]. Previous SHE schemes are constructed by modifying existing homomorphic encryption schemes of [BV11, BGV12]. 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 [GVW15] (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 [GVW15] 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 [BGV12]. 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 $\mathsf{1LGFE.sk} = \langle \mathsf{pp}_f \boldsymbol{s}\rangle - \boldsymbol{r}$ where $\boldsymbol{r}$ is a random vector of smudging noises of sufficiently large but still polynomially bounded magnitude. The decryption algorithm decrypts a ciphertext $\mathsf{1LGFE.ct} = (\{\boldsymbol{c}_i\}, \{\boldsymbol{d}_j\})$ using a functional key $\mathsf{1LGFE.sk}$ 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 $\mathsf{1LGFE.ct}_f = \lfloor \frac{q}{p}\boldsymbol{c}_{f,1}\rceil$, it then subtracts $\mathsf{1LGFE.sk}$ from it, yielding $f(\boldsymbol{x})\lfloor q/2\rceil + \lfloor \frac{q}{p}e_f\rceil + \mathsf{error} + \boldsymbol{r}$ as desired.

2.5 Our $(\mathsf{NC}^1, \text{deg-}2)$ Partially Hiding Functional Encryption

We construct 1-key PHFE with fully compact ciphertext of size linear in the input length n, for functions $F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ of the following form, from standard assumptions on asymmetric pairings. F maps three vectors $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{Z}_p^n$ to a (potentially longer) output vector in $\mathbb{Z}_p^m$ (our construction can handle any (polynomial) unbounded m), where each output element is computed by a function $f = F_k$ for $k \in [m]$ as the following matrix product:

$$f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = f^0 f^1(\boldsymbol{x}) f^2(\boldsymbol{x}) \cdots f^\ell(\boldsymbol{x}) f^{\ell+1}(\boldsymbol{y} \otimes \boldsymbol{z}), \tag{3}$$

where $f^0 \in \mathbb{Z}_p^{1\times w}$, for all $i \in [\ell]$, f^i takes as input a vector $\boldsymbol{x} \in \mathbb{Z}^n$ and outputs a matrix $f^i(\boldsymbol{x}) \in \mathbb{Z}_p^{w\times w}$, the function $f^{\ell+1}$ takes as input the vector $\boldsymbol{y} \otimes \boldsymbol{z} \in \mathbb{Z}^{n^2}$ and outputs a vector $f^{\ell+1}(\boldsymbol{y} \otimes \boldsymbol{z})$. Here, w denotes the width of the branching program, ℓ its length. The function f^i are affine, for all $i \in [\ell + 1]$. Such functions f can express computations such as $L(g(\boldsymbol{x}), \boldsymbol{y} \otimes \boldsymbol{z})$, where g is a Boolean circuit in NC^1, and L is a bilinear function, with degree one in $\boldsymbol{y} \otimes \boldsymbol{z}$.

Computing degree-2 polynomials on the private inputs.

Roughly speaking, we encrypt the private inputs $\boldsymbol{y}$ and $\boldsymbol{z}$ using encryption schemes with homomorphic properties that lets users manipulate the ciphertexts to obtain a new ciphertext, which encrypts the value $f^{\ell+1}(\boldsymbol{y} \otimes \boldsymbol{z})$, under a public key $\mathsf{pk}_{f^{\ell+1}}$ that depends on the function $f^{\ell+1}$. This manipulation can be performed publicly for arbitrary linear function $f^{\ell+1}$. At this point, providing the secret key associated to $\mathsf{pk}_{f^{\ell+1}}$ would reveal the value $f^{\ell+1}(\boldsymbol{y} \otimes \boldsymbol{z})$, and nothing else about the private inputs $\boldsymbol{y}$, $\boldsymbol{z}$. Otherwise stated, this would constitute a valid functional encryption scheme for degree-2 polynomials.

We implement this paradigm using cyclic groups $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{G}_T$ equipped with a pairing $e : \mathbf{G}_1 \times \mathbf{G}_2 \to \mathbf{G}_T$, and respectively generated by g_1, g_2, and $e(g_1, g_2)$. For any exponent $a \in \mathbb{Z}_p$, we denote by $[a]_T = e(g_1, g_2)^a \in \mathbf{G}_T$. To encrypt $\boldsymbol{y}$ and $\boldsymbol{z}$, we make generic use of a function-hiding inner product FE: the encryption of $\boldsymbol{y}$ comprises $\mathsf{IPFE.Enc}\begin{pmatrix} g_1^{y_i} \\ g_1^{r \cdot \alpha_i} \end{pmatrix}$ for all coordinates of $\boldsymbol{y}$, where $g_1^{\alpha_i}$ is a random group elements from $\mathbf{G}_1$ that is part of the public key, $r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$ is some fresh randomness, sampled at encryption time, and $\mathsf{IPFE.Enc}$ is the encryption algorithm of IPFE. The encryption of $\boldsymbol{z}$ comprises $\mathsf{IPFE.KeyGen}\begin{pmatrix} g_2^{z_j} \\ g_2^{\beta_j} \end{pmatrix}$ for all coordinate of $\boldsymbol{z}$, where $g_2^{\beta_j}$ is a random group elements from $\mathbf{G}_2$ that is part of the public key, and $\mathsf{IPFE.KeyGen}$ is the key generation algorithm of IPFE. Correctness of IPFE yields the products $[y_i z_j + r\alpha_i\beta_j]_T$ for all $i, j \in [n]$. Because IPFE is secure and function-hiding, these products are the only information revealed on the private inputs $\boldsymbol{y}$ and $\boldsymbol{z}$. It is possible to compute for any linear function $f^{\ell+1}$ the elements: $[f^{\ell+1}(\boldsymbol{y} \otimes \boldsymbol{z}) + r f^{\ell+1}(\alpha \otimes \beta)]_T$, which can be seen as an encryption of the value $f^{\ell+1}(\boldsymbol{y} \otimes \boldsymbol{z})$ under the public key $\mathsf{pk}_{f^{\ell+1}} = [f^{\ell+1}(\alpha \otimes \beta)]_T$. Because the parameters of the scheme IPFE are generated freshly during the encryption, even if IPFE is private-key —this is necessary for all function-hiding FE— the PHFE is public-key.

Computing branching programs on the public input.

We want to additionally force a specific computation on the public input $\boldsymbol{x} \in \mathbb{Z}^n$ before decryption. To do so, we produce re-encryption tokens, each of which computes one step of the matrix branching program directly on the ciphertext. That is, the token associated with the i-th product transform an encryption of $f^{i+1}(\boldsymbol{x}) \cdots f^{\ell}(\boldsymbol{x}) f^{\ell+1}(\boldsymbol{y} \otimes \boldsymbol{z})$ under $\mathsf{pk}_{f^{i+1} \dots f^{\ell+1}}$ into an encryption $f^{i}(\boldsymbol{x}) \cdots f^{\ell}(\boldsymbol{x}) f^{\ell+1}(\boldsymbol{y} \otimes \boldsymbol{z})$ under $\mathsf{pk}_{f^{i} \dots f^{\ell+1}}$, which we denote by ct_i. Finally, we release the secret key associated with the public key $\mathsf{pk}_{f^0 \dots f^{\ell+1}}$. To recover a meaningful information on the encrypted data, decryption is forced to perform the computation that precisely corresponds to the function $f^1 \cdots f^{\ell+1}$ encoded in the secret key.

The challenge is to realize these re-encryptions without blowing up the size of the ciphertext exponentially with the length ℓ. Concretely, the public keys will be of the form $\mathsf{pk}_{f^{i} \dots f^{\ell+1}} = [f^i(\boldsymbol{u}_i) \cdots f^{\ell}(\boldsymbol{u}_\ell) f^{\ell+1}(\alpha \otimes \beta)]_T$, where the vectors $\boldsymbol{u}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^n$ are part of the master secret key. These keys encode the last $\ell - i$ steps of the computation. Crucially, these keys do not grow with the length of the branching program, only its width. So is the case of the corresponding re-encryptions: we can handle polynomially large length efficiently. The i-th re-encryption token is of the form: $[r(f^i(\boldsymbol{u}_i) - f^i(\boldsymbol{x})) f^{i+1}(\boldsymbol{u}_{i+1}) \cdots f^{\ell+1}(\alpha \otimes \beta)]_T$, which allows the decryption to transition from ct_{i-1} to ct_i. Ultimately, the final ciphertext $\mathsf{ct}_\ell = [f^0 f^1(\boldsymbol{x}) \cdots f^{\ell+1}(\boldsymbol{y} \otimes \boldsymbol{z}) + r f^0 f^1(\boldsymbol{u}_1) \cdots f^{\ell+1}(\alpha \otimes \beta)]_T$, is obtained. To decrypt it, we simply need a mechanism to recover the mask $[r f^0 f^1(\boldsymbol{u}_1) \cdots f^{\ell+1}(\alpha \otimes \beta)]_T$. Providing $[r]_1$ on the encryption side, and $[f^0 f^1(\boldsymbol{u}_1) \cdots f^{\ell+1}(\alpha \otimes \beta)]_2$ as the functional secret key would already give a scheme secure in the generic-group model (and idealized model that captures attacks that do not rely on the algebraic structure of the underlying group). To obtain security from standard assumptions, we encrypt $[r]_1$ using an inner-product FE. The functional key is the inner product FE key associated with the value $[f^0 f^1(\boldsymbol{u}_1)\cdots f^{\ell+1}(\alpha \otimes \beta)]_2$. This way, decrypting the inner-product FE yields the mask to decrypt the PHFE. Note that the function is described as $[f^0 f^1(\boldsymbol{u}_1)\cdots f^{\ell+1}(\alpha \otimes \beta)]_2$ in $\mathbf{G}_2$, and not in $\mathbb{Z}$; revealing the value in $\mathbb{Z}$ would be detrimental for the security of the PHFE.

Remains to find a way to generate these re-encryption tokens. To do so, we provide an encoding of the public input $\boldsymbol{x}$ as part of the PHFE ciphertext — note that we choose the word encoding rather than encryption, since the input $\boldsymbol{x}$ must not be hidden. This encoding is used with the functional secret key to produce the tokens. We leverage the simple structure of each computational step of the branching program. Namely, we use the fact that all the functions f^i are affine. Thus, we can use an inner-product FE encryption to generate the tokens. The encoding of $\boldsymbol{x}$ is an inner-product FE encryption of $[r, r\boldsymbol{x}]_1$, and the keys are associated with the appropriate functions depending on the f^i and the vectors $[\boldsymbol{u}_i]_2$, $[\alpha]_2$, $[\beta]_2$. The challenging part is to prove security even when the values $[\boldsymbol{u}_i]_2$, $[\alpha]_2$, $[\beta]_2$ are revealed. Indeed, such is the case when using a vanilla inner-product FE, as opposed to function-hiding FE, where these values would be hidden, but which would intrinsically be private-key.

Putting things together.

Each PHFE ciphertext contains $\mathsf{IPFE.Enc}\begin{pmatrix} g_1^{y_i} \\ g_1^{r\cdot\alpha_i}\end{pmatrix}$ and $\mathsf{IPFE.KeyGen}\begin{pmatrix} g_2^{z_j} \\ g_2^{\beta_j}\end{pmatrix}$ for all $i, j \in [n]$, from which can be computed the encryption of $f^{\ell+1}(\boldsymbol{y} \otimes \boldsymbol{z})$ under an associated public key $\mathsf{pk}_{f^{\ell+1}}$, for all linear functions $f^{\ell+1}$. The scheme IPFE is function-hiding, and is generated freshly by the encryption. The PHFE ciphertext also contains another inner-product FE encryption of the values $[r, r\cdot\boldsymbol{x}]_1$. These are used with functional secret keys associated with f^i, $[\boldsymbol{u}_i]_2$, $[\alpha]_2$ and $[\beta]_2$, to generate tokens. The latter transform the encryption of $f^{\ell+1}(\boldsymbol{y}\otimes\boldsymbol{z})$ into and encryption of $f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ under a public key that encodes the matrix branching program. This transformation is performed step by step. At last, the mask of the form $[rf^0 f^1(\boldsymbol{u}_1)\cdots f^{\ell}(\boldsymbol{u}_\ell)f^{\ell+1}(\alpha \otimes \beta)]_T$ is recovered exactly as the tokens, using the inner-product FE encryption of $[r]_1$ with a functional key associated with $[f^0 f^1(\boldsymbol{u}_1)\cdots f^{\ell+1}(\alpha \otimes \beta)]_2$.

2.6 Alternative Instantiation Using Polynomials over Integers

Our FE construction can be easily modified to use the noise generator, denoted as $\Delta\mathsf{RG}$, implied by the assumption of [JLMS19] that combines LWE with constant-degree polynomials over the integers, and even simplifying the set of assumptions needed in previous works [AJS18, AJL$^+$19, JLMS19]. As discussed above, the outputs of $\Delta\mathsf{RG}$ are exactly the output $\boldsymbol{r}$ of a constant-degree polynomial computed over the integers and satisfy the notion of perturbation resilience that $\boldsymbol{r} + \boldsymbol{v}$ is 0.9-indistinguishable from $\boldsymbol{r}$. In our construction, we can directly replace the 2.5 degree PRG given by our new $\mathsf{G\text{-}LWEleak}$ assumption, with $\Delta\mathsf{RG}$. The resulting FE scheme inherits the weakness of perturbation resilience, and only satisfies weak indistinguishability security that ciphertexts of different messages cannot be distinguished with advantage over 0.9 (when the secret keys do not separate them). Then, using the general purpose FE security amplification in [AJS18], we can bootstrap this functional encryption scheme to a fully secure functional encryption scheme while preserving sublinearity, which implies $i\mathcal{O}$ under subexponential security loss. Importantly, since our construction directly gives a weakly secure FE for all polynomial-sized circuits. This circumvents the use of FE bootstrapping theorem for "lifting" the function class, and eliminates the need for constant-locality Boolean PRGs used in previous constructions. Therefore, we simplify the set of assumptions for obtaining $i\mathcal{O}$ comparing with previous works [AJS18, AJL$^+$19, JLMS19]. See the formal theorem statements in Section 6.1.

2.7 Simplification

In comparison with the prior state-of -the-art work [JLMS19], our construction is arguably simpler and more direct. Refer to Figure 2.7. The figure depicts the route to construct public-key functional encryption and obfuscation considered in [AJL$^+$19, JLMS19]. The big blocks contain the primitives used in each step of the bootstrapping. The first step used secret-key $(\mathsf{NC}^0, \mathbf{deg}2) - \mathsf{PHFE}$ along with a homomorphic encryption scheme and the $\Delta\mathsf{RG}$ assumption to construct a sublinear secret-key functional encryption for NC^0 with weak security. Then, this construction is bootstrapped to a secret-key sublinear functional encryption scheme for all circuits with weak security. Then, an expensive security amplification step is performed using the theorem in [AJS18, AJL$^+$19]. This step loses subexponential factor in the size of the adversary as well as the advantage. After that, one can construct public-key functional encryption relying on the result of [BNPW16b] and then obfuscation using the result of [AJ15, BV15]. Alternatively, one can construct obfuscation directly using the transformation in [KNT18]. However, this transformation also includes two steps and is even more inefficient in comparison to the route described via [BNPW16b], .

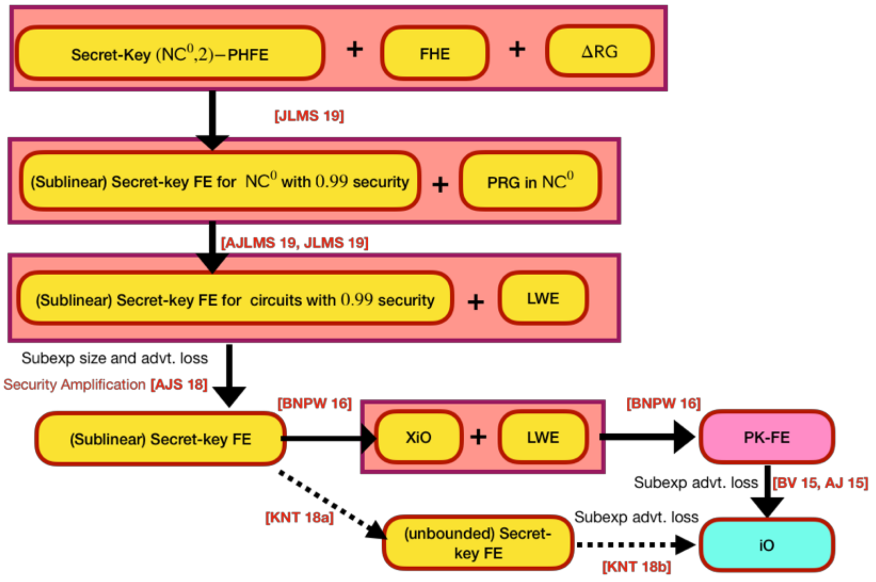
Secret-Key (NC^0,2)−PHFE
+
FHE
+
ΔRG
[JLMS 19]
(Sublinear) Secret-key FE for NC^0 with 0.99 security
+
PRG in NC^0
[AJLMS 19, JLMS 19]
(Sublinear) Secret-key FE for circuits with 0.99 security
+
LWE
Subexp size and advt. loss
Security Amplification [AJS 18]
[BNPW 16]
(Sublinear) Secret-key FE
XiO
+
LWE
[BNPW 16]
PK-FE
Subexp advt. loss
[BV 15, AJ 15]
[KNT 18a]
(unbounded) Secret-key FE
Subexp advt. loss
[KNT 18b]
iO

Figure 3: Framework of the construction [JLMS19] to achieve functional encryption and obfuscation.

On the other hand, our framework is presented in Figure 2.7. We construct sublinear public-key functional encryption scheme directly relying on the ingredients we build (public-key $(\mathsf{NC}^1, 2) - \mathsf{PHFE}$ and a single-ciphertext secret-key functional encryption with linear key generation) and our new assumption. Unlike prior works [AJL$^+$19, JLMS19], our constructions construct functional encryption incurring only polynomial loss in security (advantage of the adversary as well as the size). This can be bootstrapped to $i\mathcal{O}$ relying on the result of [AJ15, BV15].

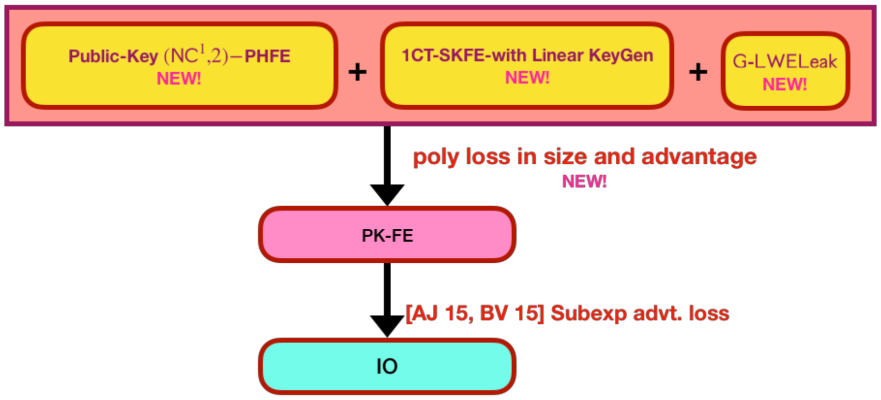
Public-Key (NC^1,2)−PHFE
NEW!
+
1CT-SKFE-with Linear KeyGen
NEW!
+
G-LWELeak
NEW!
poly loss in size and advantage
NEW!
PK-FE
[AJ 15, BV 15] Subexp advt. loss
IO

Figure 4: Our Framework.

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 $a \in \mathbb{F}_p$ 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. We now describe the following theorem that have been used in many works before our work. We cite the version from [AJL$^+$12].

Theorem 3.1. *Let B_1 and B_2 be two positive integers with $B_2 > B_1$ and let $e_1 \in [-B_1, B_1]$ be a fixed integer. Consider two distributions:*

- **Distribution 1.** *Sample $e_2 \leftarrow [0, B_2]$. Output $e_1 + e_2$.*
- **Distribution 2.** *Sample $e_2 \leftarrow [0, B_2]$. Output e_2.*

Then, the statistical distance (or the total variation distance) between the distributions is bounded by $O(B_1/B_2)$.

We also recall the following lemma from hardness amplification literature which will form a crucial pillar of our work from [JP14, CCL18b].

Theorem 3.2 (Imported Theorem [CCL18b])**.** *Let $k, t \in \mathbb{N}, \epsilon > 0$, and $\mathcal{C}_{leak}$ be a family of distinguisher circuits from $\{0,1\}^k \times \{0,1\}^t \to \{0,1\}$ of size $s(k)$. Then, for every distribution (X, Z) over $\{0,1\}^k \times \{0,1\}^t$, there exists a simulator $h : \{0,1\}^k \to \{0,1\}^t$ such that:*

1. *h is a circuit computable in size bounded by $s' = O(s \cdot 2^t \cdot \epsilon^{-2})$*
2. *(X, Z) and $(X, h(Z))$ are indistinguishable by $\mathcal{C}_{leak}$. That is, for every $C \in \mathcal{C}_{leak}$,*

$$\left| \Pr_{(x,z)\leftarrow(X,Z)}[C(x,z) = 1] - \Pr_{x \leftarrow X, h}[C(x, h(x)) = 1] \right| \leq \epsilon$$

Now we recall the definitions of some of the primitives central to this work.

3.1 Pairing Groups

Let PGGen be a PPT algorithm that on input the security parameter 1^λ, returns a description $\mathcal{PG} = (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, p, P_1, P_2, e)$ where for all $s \in \{1, 2, T\}$, $\mathbf{G}_s$ is an additive cyclic group of order p for a 2λ-bit prime p. $\mathbf{G}_1$ and $\mathbf{G}_2$ are generated by P_1 and P_2 respectively, and $e : \mathbf{G}_1 \times \mathbf{G}_2 \to \mathbf{G}_T$ is an efficiently computable (non-degenerate) bilinear map. Define $P_T := e(P_1, P_2)$, which is a generator of $\mathbf{G}_T$, of order p. We use implicit representation of group elements. For $s \in \{1, 2, T\}$ and $a \in \mathbb{Z}_p$, define $[a]_s = a \cdot P_s \in \mathbf{G}_s$ as the implicit representation of a in G_s. More generally, for a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_p^{n\times m}$ we define $[\mathbf{A}]_s$ as the implicit representation of $\mathbf{A}$ in $\mathbf{G}_s$:

$$[\mathbf{A}]_s := \begin{pmatrix} a_{11} \cdot P_s & \dots & a_{1m} \cdot P_s \\ \\ a_{n1} \cdot P_s & \dots & a_{nm} \cdot P_s \end{pmatrix} \in \mathbf{G}_s^{n\times m}.$$

Given $[a]_1$ and $[b]_2$, one can efficiently compute $[a \cdot b]_T$ using the pairing e. For matrices $\mathbf{A}$ and $\mathbf{B}$ of matching dimensions, define $e([\mathbf{A}]_1, [\mathbf{B}]_2) := [\mathbf{AB}]_T$. For any matrix $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_p^{n\times m}$, any group $s \in \{1, 2, T\}$, we denote by $[\mathbf{A}]_s + [\mathbf{B}]_s = [\mathbf{A} + \mathbf{B}]_s$.

For any prime p, we define the following distribution. The DDH distribution over $\mathbb{Z}_p^2$: Sample $a \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, output $\boldsymbol{a} := \binom{1}{a}$. The DLIN distribution over $\mathbb{Z}_p^{3\times 2}$: $a, b \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, outputs $\mathbf{A} := \begin{pmatrix} a & 0 \\ 0 & b \\ 1 & 1 \end{pmatrix}$.

Definition 3.1 (DDH assumption)**.** *For any adversary $\mathcal{A}$, any group $s \in \{1, 2, T\}$ and any security parameter λ, let*

$$\mathsf{adv}^{\mathsf{DDH}}_{\mathbf{G}_s,\mathcal{A}}(\lambda) := |\Pr[1 \leftarrow \mathcal{A}(\mathcal{PG}, [\boldsymbol{a}]_s, [\boldsymbol{a}r]_s)] - \Pr[1 \leftarrow \mathcal{A}(\mathcal{PG}, [\boldsymbol{a}]_s, [\boldsymbol{u}]_s)]|,$$

where the probabilities are taken over $\mathcal{PG} \leftarrow_R \mathsf{PGGen}(1^\lambda)$, $\boldsymbol{a} \leftarrow_R \mathsf{DDH}$, $r \leftarrow_R \mathbb{Z}_p$, $\boldsymbol{u} \leftarrow_R \mathbb{Z}_p^2$, and the random coins of $\mathcal{A}$. We say DDH holds in $\mathbf{G}_s$ if for all PPT adversaries $\mathcal{A}$, $\mathsf{adv}^{\mathsf{DDH}}_{\mathbf{G}_s,\mathcal{A}}(\lambda)$ is a negligible function of λ.

Definition 3.2 (SXDH assumption)**.** *For any security parameter λ and any pairing group $\mathcal{PG} = (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, p, P_1, P_2, e) \leftarrow_R \mathsf{PGGen}(1^\lambda)$, we say SXDH holds in $\mathcal{PG}$ if DDH holds in $\mathbf{G}_1$ and $\mathbf{G}_2$.*

Definition 3.3 (Bilateral DLIN assumption)**.** *For any adversary $\mathcal{A}$, any security parameter λ any pairing group $\mathcal{PG} \leftarrow \mathsf{PGGen}(1^\lambda)$, let*

$$\mathsf{adv}^{\mathsf{DLIN}}_{\mathcal{PG},\mathcal{A}}(\lambda) := \left|\Pr\left[1 \leftarrow \mathcal{A}\left(\mathcal{PG}, \{[\mathbf{A}]_s, [\mathbf{A}\boldsymbol{r}]_s\}_{s\in[1,2]}\right)\right] - \Pr\left[1 \leftarrow \mathcal{A}\left(\mathcal{PG}, \{[\mathbf{A}]_s, [\boldsymbol{u}]_s\}_{s\in[1,2]}\right)\right]\right|,$$

where the probabilities are taken over $\mathcal{PG} \leftarrow_R \mathsf{PGGen}(1^\lambda)$, $\mathbf{A} \leftarrow_R \mathsf{DLIN}$, $\boldsymbol{r} \leftarrow_R \mathbb{Z}_p^2$, $\boldsymbol{u} \leftarrow_R \mathbb{Z}_p^3$, and the random coins of $\mathcal{A}$. We say bilateral DLIN holds in $\mathcal{PG}$ if for all PPT adversaries $\mathcal{A}$, $\mathsf{adv}^{\mathsf{DLIN}}_{\mathcal{PG},\mathcal{A}}(\lambda)$ is a negligible function of λ.

3.2 Lattice Preliminaries

A full-rank m-dimensional integer lattice $\Lambda \subset \mathbb{Z}^m$ is a discrete additive subgroup whose linear span is $\mathbb{R}^m$. The basis of Λ is a linearly independent set of vectors whose integer linear combinations are exactly Λ. Every integer lattice is generated as the $\mathbb{Z}$-linear combination of linearly independent vectors $\mathbf{B} = \{\boldsymbol{b}_1, ..., \boldsymbol{b}_m\} \subset \mathbb{Z}^m$. For a matrix $\mathbf{A} \in \mathbb{Z}_p^{\mathsf{d} \times m}$, we define the "$p$-ary" integer lattices:

$$\Lambda_p^{\perp} = \{\boldsymbol{e} \in \mathbb{Z}^m | \mathbf{A}\boldsymbol{e} = \mathbf{0} \bmod p\}, \qquad \Lambda_p^{\mathbf{u}} = \{\boldsymbol{e} \in \mathbb{Z}^m | \mathbf{A}\boldsymbol{e} = \boldsymbol{u} \bmod q\}$$

It is obvious that $\Lambda_p^{\boldsymbol{u}}$ is a coset of $\Lambda_p^{\perp}$.

Let Λ be a discrete subset of $\mathbb{Z}^m$. For any vector $\boldsymbol{c} \in \mathbb{R}^m$, and any positive parameter $\sigma \in \mathbb{R}$, let $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x}) = \exp(-\pi||\boldsymbol{x} - \boldsymbol{c}||^2/\sigma^2)$ be the Gaussian function on $\mathbb{R}^m$ with center $\boldsymbol{c}$ and parameter σ. Next, we let $\rho_{\sigma,\boldsymbol{c}}(\Lambda) = \sum_{\boldsymbol{x}\in\Lambda} \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})$ be the discrete integral of $\rho_{\sigma,\boldsymbol{x}}$ over Λ, and let $\mathcal{D}_{\Lambda,\sigma,\boldsymbol{c}}(\boldsymbol{y}) := \frac{\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{y})}{\rho_{\sigma,\boldsymbol{c}}(\Lambda)}$. We abbreviate this as $\mathcal{D}_{\Lambda,\sigma}$ when $\boldsymbol{c} = \mathbf{0}$. We note that $\mathcal{D}_{Z^m,\sigma}$ is $\sqrt{m}\sigma$-bounded.

Let S^m denote the set of vectors in $\mathbb{R}^m$ whose length is 1. The norm of a matrix $\mathbf{R} \in \mathbb{R}^{m\times m}$ is defined to be $\mathsf{sup}_{\boldsymbol{x}\in S^m}||\mathbf{R}\boldsymbol{x}||$. The LWE problem was introduced by Regev [Reg05], who showed that solving it *on average* is as hard as (quantumly) solving several standard lattice problems *in the worst case.*

Definition 3.4 (LWE Assumption)**.** *For an integer $p = p(\mathsf{d}) \geq 2$, and an error distribution $\chi = \chi(\mathsf{d})$ over $\mathbb{Z}_p$, the* Learning With Errors assumption $\mathsf{LWE}_{\mathsf{d},m,p,\chi}$ *holds if it is hard to distinguish between the following pairs of distributions:*

$$\{\mathbf{A}, \boldsymbol{s}^{\mathbf{T}}\mathbf{A} + \boldsymbol{x}^{\mathbf{T}}\} \text{ and } \{\mathbf{A}, \boldsymbol{u}^{\mathbf{T}}\}$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{\mathsf{d}\times m}$, $\boldsymbol{s} \leftarrow \mathbb{Z}_p^{\mathsf{d}}$, $\boldsymbol{u} \leftarrow \mathbb{Z}_p^m$, *and* $\boldsymbol{x} \leftarrow \chi^m$.

Gadget matrix. The gadget matrix described below is proposed in [MP12].

Definition 3.5. *Let $m = \mathsf{d} \cdot \lceil \log p \rceil$, and define the gadget matrix $\mathbf{G} = \boldsymbol{g} \otimes \mathbf{I}_{\mathsf{d}} \in \mathbb{Z}_p^{\mathsf{d}\times m}$, where the vector $\boldsymbol{g} = (1, 2, 4, ..., 2^{\lfloor \log p \rfloor}) \in \mathbb{Z}_p^{\lceil \log p \rceil}$. We will also refer to this gadget matrix as "powers-of-two" matrix. We define the inverse function $\mathbf{G}^{-1} : \mathbb{Z}_p^{\mathsf{d}\times m} \to \{0, 1\}^{m\times m}$ which expands each entry $a \in \mathbb{Z}_p$ of the input matrix into a column of size $\lceil \log p \rceil$ consisting of the bits of binary representations. We have the property that for any matrix $\mathbf{A} \in \mathbb{Z}_p^{\mathsf{d}\times m}$, it holds that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$.*

4 Functional Encryption Definitions

In this section, we define functional encryption notions to be used and constructed in our work along with the efficiency and security properties. Throughout, we denote functionality by $\mathcal{F} : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$. The functionality ensemble $\mathcal{F}$ as well as the message ensembles $\mathcal{X}$ and $\mathcal{Y}$ are indexed by two parameters: n and λ (for example $\mathcal{F}_{n,\lambda}$), where λ is the security parameter and n is a length parameter and can be viewed as a function of λ. We define the syntax of a partially hiding functional encryption PHFE which is a generalization of functional encryption. The syntax of functional FE encryption is essentially identical with the change that in a functional encryption scheme the ensemble $\mathcal{X}$ is empty for all n and λ.

Definition 4.1. *(Syntax of a PHFE/FE Scheme.) A partially hiding functional encryption scheme,* PHFE*, for the functionality $\mathcal{F} : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ consists of the following polynomial time algorithms:*

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

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

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

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

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

Remark 4.1. (On Secret Key Schemes.) Above we define the syntax of a public key scheme. A secret key scheme just has one change over the syntax above. The Encryption algorithm takes as input the master secret key instead of the public key. Also, the setup does not produce any public key.

Remark 4.2. (On FE vs PHFE.) The syntax of the functional encryption scheme is identical to a partially hiding functional encryption scheme described above except that $\mathcal{X}$ is the empty set for a functional encryption scheme, as there is no public input.

Definition 4.2. *(Correctness of a PHFE scheme.) A partially hiding functional encryption scheme,* PHFE*, for the functionality $\mathcal{F} : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ is correct if for every $\lambda \in \mathbb{N}$ and every polynomial $n(\lambda) \in \mathbb{N}$, for every $(x, y) \in \mathcal{X}_{n,\lambda} \times \mathcal{Y}_{n,\lambda}$ and every $f \in \mathcal{F}_{n,\lambda}$, we have:*

$$\Pr\begin{bmatrix} \mathsf{PPGen}(1^\lambda, 1^n) \to \mathsf{crs} \\ \mathsf{Setup}(\mathsf{crs}) \to (\mathsf{pk}, \mathsf{sk}) \\ \mathsf{Enc}(\mathsf{pk}, (x, y)) \to (x, \mathsf{ct}) \\ \mathsf{KeyGen}(\mathsf{sk}, f) \to \mathsf{sk}_f \\ \mathsf{Dec}(\mathsf{sk}_f, x, \mathsf{ct})) \neq f(x, y) \end{bmatrix} \leq \mathsf{negl}(\lambda)$$

for some negligible function negl.

The correctness of a functional encryption scheme is defined similarly. Now we define the security notions associated with PHFE and FE.

4.1 Security Definitions

We discuss two security notions. The first one is the notion of simulation security. We define it for a PHFE scheme.

Definition 4.3 (Simulation security)**.** *For any public-key PHFE scheme,* PHFE*, for functionality $\mathcal{F} : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, any security parameter λ, any length parameter n, any PPT stateful adversary $\mathcal{A}$, and any PPT simulator $\mathcal{S} := (\widetilde{\mathsf{Setup}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{KeyGen}})$, we define the following two experiments.*

```
$\underline{\mathsf{Real}^{\mathsf{PHFE}}_{\mathcal{A}}(1^\lambda, 1^n)}$:
$(x, y) \in \mathcal{X}_{n,\lambda} \times \mathcal{Y}_{n,\lambda} \leftarrow \mathcal{A}(1^\lambda, 1^n)$
$\mathsf{crs} \leftarrow \mathsf{PPGen}(1^\lambda, 1^n)$
$(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{crs})$
$(x, \mathsf{ct}) \leftarrow \mathsf{Enc}(\mathsf{pk}, (x, y))$
$\alpha \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}(\cdot)}(\mathsf{ct}, \mathsf{pk})$
```

```
$\underline{\mathsf{Ideal}^{\mathsf{PHFE}}_{\mathcal{A},\mathcal{S}}(1^\lambda, 1^n)}$:
$(x, y) \in \mathcal{X}_{n,\lambda} \times \mathcal{Y}_{n,\lambda} \leftarrow \mathcal{A}(1^\lambda, 1^n)$
$\mathsf{crs} \leftarrow \mathsf{PPGen}(1^\lambda, 1^n)$
$(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{msk}}) \leftarrow \widetilde{\mathsf{Setup}}(\mathsf{crs})$
$(x, \widetilde{\mathsf{ct}}) \leftarrow \widetilde{\mathsf{Enc}}(\widetilde{\mathsf{msk}}, x)$
$\alpha \leftarrow \mathcal{A}^{\mathcal{O}_{\widetilde{\mathsf{KeyGen}}}(\cdot)}(\widetilde{\mathsf{ct}}, \widetilde{\mathsf{pk}})$
```

In the real experiment, the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$, when given as input $f \in \mathcal{F}_{n,\lambda}$, returns $\mathsf{KeyGen}(\mathsf{msk}, f)$*. In the ideal experiment, the key generation oracle $\mathcal{O}_{\widetilde{\mathsf{KeyGen}}}$, when given as input $f \in \mathcal{F}_{n,\lambda}$, computes $f(x, y)$, and returns* $\widetilde{\mathsf{KeyGen}}(\widetilde{\mathsf{msk}}, f, f(x, y))$*.*

We say that PHFE *is SIM secure if there exists a PPT simulator $\mathcal{S} := (\widetilde{\mathsf{Setup}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{KeyGen}})$ such that for any PPT adversary $\mathcal{A}$, any constant $c > 0$, any large enough security parameter λ, any polynomial $n(\lambda) \in \mathbb{N}$:*

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

Next, we discuss the standard indistinguishability security for a functional encryption scheme.

Definition 4.4 (Indistinguishability security)**.** *For any FE scheme* FE *for functionality $\mathcal{F} : \mathcal{Y} \to \mathcal{Z}$, any security parameter λ, any length parameter n, any PPT stateful adversary $\mathcal{A}$, we define the following experiment.*

```
$\underline{\mathsf{IND}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 1^n)}$:
$\{x_0^i, x_1^i\}_{i \in [Q_{\mathsf{ct}}]}, \{f^j\}_{j \in [Q_{\mathsf{sk}}]} \leftarrow \mathcal{A}(1^\lambda, 1^n)$
$\mathsf{crs} \leftarrow \mathsf{PPGen}(1^\lambda, 1^n)$
$(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{crs})$, $b \leftarrow_R \{0, 1\}$
$\forall i \in [Q_{\mathsf{ct}}] : \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, x_b^i)$, $\forall j \in [Q_{\mathsf{sk}}] : \mathsf{sk}_j \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f^j)$
$b' \leftarrow \mathcal{A}(\{\mathsf{ct}_i\}_{i \in [Q_{\mathsf{ct}}]}, \{\mathsf{sk}_j\}_{j \in [Q_{\mathsf{sk}}]}, \mathsf{pk})$
Return 1 if $b = b'$ and $\forall\ i \in [Q_{\mathsf{ct}}],\ j \in [Q_{\mathsf{sk}}], f^j(x_0^i) = f^j(x_1^i)$, 0 otherwise.
```

We say FE *is IND secure if for any PPT adversary $\mathcal{A}$, any constant $c > 0$, any large enough security parameter λ, any polynomial $n(\lambda) \in \mathbb{N}$:*

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

Similarly, we can also define secret-key function hiding FE as follows.

Definition 4.5 (Function Hiding Indistinguishability security)**.** *For any secret-key FE scheme* FE *for functionality $\mathcal{F} : \mathcal{Y} \to \mathcal{Z}$, any security parameter λ, any length parameter n, any PPT stateful adversary $\mathcal{A}$, we define the following experiment.*

```
$\underline{\mathsf{IND-FH}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 1^n)}$:
$\{x_0^i, x_1^i\}_{i \in [Q_{\mathsf{ct}}]}, \{f_0^j, f_1^j\}_{j \in [Q_{\mathsf{sk}}]} \leftarrow \mathcal{A}(1^\lambda, 1^n)$
$\mathsf{crs} \leftarrow \mathsf{PPGen}(1^\lambda, 1^n)$
$\mathsf{msk} \leftarrow \mathsf{Setup}(\mathsf{crs})$, $b \leftarrow_R \{0, 1\}$
$\forall i \in [Q_{\mathsf{ct}}] : \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{msk}, x_b^i)$, $\forall j \in [Q_{\mathsf{sk}}] : \mathsf{sk}_j \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_b^j)$
$b' \leftarrow \mathcal{A}(\{\mathsf{ct}_i\}_{i \in [Q_{\mathsf{ct}}]}, \{\mathsf{sk}_j\}_{j \in [Q_{\mathsf{sk}}]})$
Return 1 if $b = b'$ and $\forall\ i \in [Q_{\mathsf{ct}}],\ j \in [Q_{\mathsf{sk}}], f_0^j(x_0^i) = f_1^j(x_1^i)$, 0 otherwise.
```

We say FE *is IND-FH secure if for any PPT adversary $\mathcal{A}$, any constant $c > 0$, any large enough security parameter λ, any polynomial $n(\lambda) \in \mathbb{N}$:*

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

Remark 4.3. In both the above games, the ciphertext and key queries are not required to be bounded. We also consider Functional Encryption with $(Q_{\mathsf{ct}}, Q_{\mathsf{sk}})-$indistinguishability security where the number of key and the ciphertext queries are bounded by Q_{ct} and Q_{sk} respectively, where Q_{ct} and Q_{sk} are some polynomials in the security parameter. If there is no bound on the number of keys or the ciphertext we will set the corresponding parameter by $\mathsf{poly}(\lambda)$ indicating that it could be an arbitrary polynomial. For example, $(Q_{\mathsf{ct}}, \mathsf{poly}(\lambda))-$IND secure scheme denotes an FE scheme with unbounded key queries but bounded ciphertext queries bounded by Q_{ct}.

Remark 4.4 (On $(s, \epsilon)-$security)**.** Above, we give the definitions of security using standard indistinguishability. At times, we will also use $(s, \epsilon)-$security, where it will mean that the corresponding distinguishing advantage is bounded by $\epsilon(\lambda)$ for any adversary of size bounded by $s(\lambda)$. Standard subexponential security means that in this notation ϵ is inverse subexponential for all polynomial sized circuits.

4.2 Efficiency Features

We now define various efficiency variants that a PHFE/FE scheme may satisfy. First we define the notion of linear efficiency of a PHFE scheme, PHFE, but the definition for an FE scheme is identical except that the set $\mathcal{X}$ is empty.

Definition 4.6. *(linear efficiency of a PHFE/FE scheme) We say a* PHFE *for the functionality $\mathcal{F} : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ satisfies linear efficiency if for any security parameter $\lambda \in \mathbb{N}$, any polynomial $n(\lambda) \in \mathbb{N}$, any message $(x, y) \in \mathcal{X}_{n,\lambda} \times \mathcal{Y}_{n,\lambda}$ the following holds:*

- $\mathsf{crs} \leftarrow \mathsf{PPGen}(1^\lambda, 1^n)$
- *Let* $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{crs})$.
- *Compute* $(x, \mathsf{ct}) \leftarrow \mathsf{Enc}(\mathsf{pk}, (x, y))$.

Then the size of of ct *is bounded by $n \cdot \mathsf{poly}(\lambda)$ for a fixed polynomial in λ.*

Now we define the notion of sublinearity. It was shown in a series of works [AJ15, BV15, BNPW16a] that such FE schemes for P/poly imply obfuscation (assuming subexponential security).

Definition 4.7. *(Sublinearity of a PHFE/FE scheme) We say a functional encryption scheme* FE *for the functionality $\mathcal{F} : \mathcal{Y} \to \mathcal{Z}$ satisfies sub-linear efficiency if for any security parameter $\lambda \in \mathbb{N}$, any polynomial $n(\lambda) \in \mathbb{N}$, any message $y \in \mathcal{Y}_{n,\lambda}$ the following holds:*

- $\mathsf{crs} \leftarrow \mathsf{PPGen}(1^\lambda, 1^n)$
- *Let* $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{crs})$.
- *Compute* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, y)$.

Let $s_{\mathcal{F}}$ denote the maximum size of the circuit in $\mathcal{F}_{n,\lambda}$. Then the size of of ct *is bounded by $(s_{\mathcal{F}}^{1-\epsilon} + n) \cdot \mathsf{poly}(\lambda)$ for a fixed polynomial in λ and for some constant $\epsilon > 0$. Further, we say that the scheme is compaact if $\epsilon = 1$.*

We also define the notion of output sublinearity, which is a strengthening of the notion above.

Definition 4.8. *(Output Sublinearity of an FE scheme) We say a functional encryption scheme* FE *for the functionality* $\{\mathcal{F}_{n,\lambda,\ell} : \mathcal{Y}_{n,\lambda} \to \{0,1\}^{\ell}\}_{\lambda,n,\ell\in\mathbb{N}}$ *satisfies output sub-linear efficiency if for any security parameter* $\lambda \in \mathbb{N}$*, any polynomials* $n(\lambda) \in \mathbb{N}$ *and* $\ell(\lambda)$ *and any message* $y \in \mathcal{Y}_{n,\lambda}$ *the following holds:*

- $\mathsf{crs} \leftarrow \mathsf{PPGen}(1^{\lambda}, 1^{n})$
- *Let* $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{crs})$.
- *Compute* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, y)$.

Then the size of of ct *is bounded by* $(\ell^{1-\epsilon} + n) \cdot \mathsf{poly}(\lambda)$ *for a fixed polynomial in* λ *and for some constant* $\epsilon > 0$.

The functional encryption we describe in Section 6 actually satisfies the notion of sublinearity above.

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 4.9. *(Special Structure.) We say that a functional encryption scheme* FE *satisfies special structure if:*

- *(CRS Syntax.) The* crs *generated by the* $\mathsf{PPGen}(1^{\lambda}, 1^{n})$ *algorithm consists of a modulus* p *(which is a* λ^{c} *bit modulus for some constant* $c > 0$*).*
- *(Linear secret key structure.) The master secret key is a vector in* $\boldsymbol{s} \in \mathbb{Z}_p^{\mathsf{poly}(\lambda)}$ *for some polynomial* poly*. For any function* $f \in \mathcal{F}_{n,\lambda}$*, the functional secret key is of the form* $\langle \mathsf{crs}_f, \boldsymbol{s}\rangle + e \mod p$ *where* crs_f *is some deterministic polynomial time computable function of the* crs *and* e *is a randomly chosen field element from some distribution over* $\mathbb{Z}_p$*. Further* $|e| < p/16$.
- *(Linear + Round Decryption.) We require that for any ciphertext* ct*, the decryption for a circuit* f *proceeds by first computing a deterministic (possibly complex) function of* ct *to output* ct_f*. Finally if* ct *was an honest encryption of* m*, then, given the secret key* $\mathsf{sk}_f = \langle \mathsf{crs}_f, \boldsymbol{s}\rangle + e \mod p$ *the decryption computes* $\mathsf{ct}_f - \mathsf{sk}_f \mod p = f(m)\cdot\lceil p/2\rceil + e_f - e$ *where* e_f *is polynomially bounded in the security parameter in absolute value and* $|e| < p/16$*. The decryption algorithm rounds and recover the output.*

We also define the notion of Special Structure* where we additionally require that the decryption noise is polynomially bounded. More formally, consider the following definition.

Definition 4.10. *(Special Structure*.) We say that a functional encryption scheme* FE *satisfies special structure if:*

- *(CRS Syntax.) The* crs *generated by the* $\mathsf{PPGen}(1^{\lambda}, 1^{n})$ *algorithm consists of a modulus* p *(which is a* λ^{c} *bit modulus for some constant* $c > 0$*).*

- *(Linear secret key Structure.) The master secret key is a vector in* $\boldsymbol{s} \in \mathbb{Z}_p^{\mathsf{poly}(\lambda)}$ *for some polynomial* poly*. For any function* $f \in \mathcal{F}_{n,\lambda}$*, the functional secret key is of the form* $\langle \mathsf{crs}_f, \boldsymbol{s}\rangle + e \mod p$ *where* crs_f *is some deterministic polynomial time computable function of the* crs *and* e *is a randomly chosen field element from some distribution over* $\mathbb{Z}_p$*. Further* $|e| < \mathsf{poly}(n, \lambda)$ *for some polynomial* poly*.*
- *(Linear + Round Decryption with polynomial decryption error.) We require that for any ciphertext* ct*, the decryption for a circuit* f *proceeds by first computing a deterministic (possibly complex) function of* ct *to output* ct_f*. Finally if* ct *was an honest encryption of* m*, then, given the secret key* $\mathsf{sk}_f = \langle \mathsf{crs}_f, \boldsymbol{s}\rangle + e \mod p$ *the decryption computes* $\mathsf{ct}_f - \mathsf{sk}_f \mod p = f(m) \cdot \lceil p/2 \rceil + e_f - e$ *where* e_f *is polynomially bounded in the security parameter in absolute value and* $|e| < \mathsf{poly}(n, \lambda)$ *for some polynomial* poly*. The decryption algorithm rounds and recover the output.*

5 New Assumption

In this section, we describe our new assumption. We begin with some definitions.

Definition 5.1. *(Pseudorandom Generator.) A stretch-*$m(\cdot)$ *pseudorandom generator is a Boolean function* $\mathsf{PRG} : \{0,1\}^* \to \{0,1\}^*$ *mapping* n*-bit inputs to* $m(n)$*-bit outputs that is computable by a uniform p.p.t. machine, and for any non-uniform p.p.t adversary* $\mathcal{A}$ *there exist a negligible function* negl *such that, for all* $n \in \mathbb{N}$

$$\left| \Pr_{r \leftarrow \{0,1\}^n}[\mathcal{A}(\mathsf{PRG}(r)) = 1] - \Pr_{z \leftarrow \{0,1\}^m}[\mathcal{A}(z) = 1] \right| < \mathsf{negl}(n).$$

Definition 5.2. *($\mathbb{Z}$-degree of a* PRG*.) Consider a stretch-*$m(\cdot)$ *pseudorandom generator* PRG*. For all* $n \in \mathbb{N}$*, and every* $i \in [m(n)]$*, we denote by* $\mathsf{PRG}_{n,i} : \{0,1\}^n \to \{0,1\}$ *the function that outputs the* i*'th bit of the computation* $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^m$*, and* $d_{n,i}$ *its* $\mathbb{Z}$*-degree, that is, the degree of the unique multi-linear polynomial over* $\mathbb{Z}[X_1, ..., X_n]$ *that agrees with* $\mathsf{PRG}_{n,i}$ *on* $\{0,1\}^n$*. We define the* $\mathbb{Z}$*-degree of* PRG *as* $d(n) = \max_{i \in [m]} d_{n,i}$*.*

From now, by degree of a stretch-$m(\cdot)$ pseudorandom generator G, we mean the $\mathbb{Z}$ degree of G unless specified otherwise. We refer by $\mathbb{F}-$degree, the degree of the polynomial over $\mathbb{F}$.

Our new assumption is stronger than the one describe next. The assumption is widely known in cryptography as the LWE with binary error assumption.

Definition 5.3 ($\mathsf{LWBE}_{\epsilon,\rho}$ Assumption)**.** *For any constants* $\epsilon > 0$ *and* $\rho > 0$*, we say that the assumption* $\mathsf{LWBE}_{\epsilon,\rho}$ *holds if for every modulus* $p = O(2^{n^\rho})$ *the following happens. We define two distributions below. The assumption requires that the following distributions are computationally indistinguishable:*

```
PseudoR_A(1^n):
  $\boldsymbol{s} \leftarrow \mathbb{Z}_p^{n^{0.5+\epsilon}}$; $\boldsymbol{a}_i \leftarrow \mathbb{Z}_p^{n^{0.5+\epsilon}}$;
  $e_i \leftarrow \{0,1\}\ \forall\ i \in [n]$;
  Output $\left(\{\boldsymbol{a}_i, \langle \boldsymbol{a}_i, \boldsymbol{s}\rangle + e_i \mod p\}_{i \in [n]}\right)$
```

```
Random_A(1^n):
  $\boldsymbol{s} \leftarrow \mathbb{Z}_p^{n^{0.5+\epsilon}}$; $\boldsymbol{a}_i \leftarrow \mathbb{Z}_p^{n^{0.5+\epsilon}}$;
  $r_i \leftarrow \mathbb{Z}_p \forall\ i \in [n]$;
  Output $\left(\{\boldsymbol{a}_i, r_i\}_{i \in [n]}\right)$
```

Formally, we require that $\mathsf{LWBE}_{\epsilon,p}$ *holds if:*

$$\mathsf{adv}_{\mathcal{A}}^{\mathsf{LWBE}_{\epsilon,p}}(1^n) := |\Pr[\mathcal{A}(z_1) = 1] - \Pr[\mathcal{A}(z_2) = 1]| < \mathsf{negl}(n),$$

where $z_1 \leftarrow \mathsf{PseudoR}_{\mathcal{A}}(1^n)$ *and* $z_2 \leftarrow \mathsf{Random}_{\mathcal{A}}(1^n)$*.*

We discuss the state of this assumption in Section 5.4. Next, we describe our main new assumption which can be seen as an assumption arising from the interplay between the two assumptions described above (the assumption of LWBE and that of a pseudorandom generator with large enough stretch). We discuss the plausibility of this assumption too in Section 5.4.

Definition 5.4 ($\mathsf{G\text{-}LWEleak}_{d,\epsilon,\rho}$ Security)**.** *For any constant integer $d > 0$, constants $\epsilon > 0$ and $\rho \in (0, 0.5)$, we say that a degree d pseudorandom generator G of stretch at least $m(n) \geq n^{\lceil \frac{d}{2} \rceil \cdot (0.5+\epsilon)+\rho}$ satisfies $\mathsf{G\text{-}LWEleak}_{d,\epsilon,\rho}$−security if for any modulus $p = O(2^{n^{\rho}})$, the following to distributions are computationally indistinguishable:*

```
PseudoR^G_A(1^n):
$\boldsymbol{s} \leftarrow \mathbb{Z}_p^{n^{0.5+\epsilon}}$; $\boldsymbol{a}_i \leftarrow \mathbb{Z}_p^{n^{0.5+\epsilon}}$;
$e_i \leftarrow \{0,1\}\ \forall\ i \in [n]$;

Output $\left(\{\boldsymbol{a}_i, \langle \boldsymbol{a_i}, \boldsymbol{s} \rangle + e_i \mod p\}_{i \in [n]}, \mathsf{G}(\boldsymbol{e})\right)$
```

```
Random^G_A(1^n):
$\boldsymbol{s} \leftarrow \mathbb{Z}_p^{n^{0.5+\epsilon}}$; $\boldsymbol{a}_i \leftarrow \mathbb{Z}_p^{n^{0.5+\epsilon}}$;
$e_i \leftarrow \{0,1\}\ \forall\ i \in [n]$;
$r \leftarrow \{0,1\}^m$;
Output $\left(\{\boldsymbol{a}_i, \langle \boldsymbol{a_i}, \boldsymbol{s} \rangle + e_i \mod p\}_{i \in [n]}, r\right)$
```

Formally, we say that G satisfies $\mathsf{LWEleak}_{d,\epsilon,p}$ if:

$$\mathsf{adv}^{\mathsf{LWEleak}_{d,\epsilon,\rho}}_{\mathsf{G},\mathcal{A}}(1^n) := |\Pr[\mathcal{A}(z_1) = 1] - \Pr[\mathcal{A}(z_2) = 1]| < \mathsf{negl}(n),$$

where $z_1 \leftarrow \mathsf{PseudoR}^{\mathsf{G}}_{\mathcal{A}}(1^n)$ and $z_2 \leftarrow \mathsf{Random}^{G}_{\mathcal{A}}(1^n)$.

5.1 A Survey of the PRG Candidates

We consider Goldreich PRG candidates [Gol00]. We recall the definition of a hypergraph first.

Definition 5.5. *We define an (n, m, d)−hypergraph H to be a hypergraph with n vertices and m hyperedges of cardinality d. Each hyperedge σ_i for $i \in [m]$ is of the form $\sigma_i = \{\sigma_{i,1}, \ldots, \sigma_{i,d}\}$ where each $\sigma_{i,j_1} \in [n]$ is distinct from $\sigma_{i,j_2} \in [n]$ for every $i \in [m]$ and $j_1 \neq j_2$. Also, we assume that each σ_i is an ordered set.*

We now define Goldreich PRG candidates.

Definition 5.6. *Goldreich's candidate d-local PRG $\mathsf{G}_{H,\mathsf{P}}$ forms a family of local PRG candidates where $\mathsf{G}_{H,\mathsf{P}} : \{0,1\}^n \to \{0,1\}^m$ is parameterized by an (n, m, d)−hypergraph $H = (\sigma_1, \ldots, \sigma_m)$ and a boolean predicate $\mathsf{P} : \{0,1\}^d \to \{0,1\}$. The functionality is defined as follows: On input $\boldsymbol{x} \in \{0,1\}^n$, $\mathsf{G}_{H,\mathsf{P}}$ return m-bit strings: $(\mathsf{P}(x_{\sigma_{1,1}}, \ldots, x_{\sigma_{1,d}}), \ldots, \mathsf{P}(x_{\sigma_{m,1}}, \ldots, x_{\sigma_{m,d}}))$.*

Typically P is some predicate satisfying some nice properties, d is a constant integer greater than equal to 5, and H is a randomly chosen graph from some distribution. The security should hold with high probability over the choice of this graph.

Coming back to our assumption, intuitively, our assumption suggests that as long as other parameters are chosen appropriately, any Goldreich PRG predicate of constant degree d admitting a stretch of $\Omega(n^{\frac{1}{2} \cdot \lceil \frac{d}{2} \rceil + c})$ for any constant $c > 0$ can potentially form a nice choice to instantiate our assumption. Traditionally Goldreich's PRG has been a subject of extensive study (For example, see [Gol00, MST03, ABR12, BQ12, App12, OW14, AL16, CDM+18].). The standard complexity measure for a Goldreich's PRG is locality of the predicate (and not the $\mathbb{Z}$−degree.). Locality of the predicate is the number of bits that the predicate takes as input. Since the predicate in a Goldreich PRG is a boolean function, the locality of the predicate forms an upper bound on the $\mathbb{Z}$-degree of the predicate. We now survey some known results below and we will remark about both locality and $\mathbb{Z}-$degree of the predicate. Analysis of the PRG predicates in literature has focused mainly, on the following broad classes of attacks:

- $\mathbb{F}_2$ linear bias distinguishing attacks.
- Attacks from optimization literature such as (e.g. SoS based SDP algorithms.).
- Algebraic attacks that include, e.g. Gröbner Basis Attacks.
- Guess and Determine Attacks.

It is known from the work of [MST03], that in order to construct a PRG with polynomial stretch the minimum locality needs to be 5. For such a locality, [OW14] proved an optimal stretch of $m(n) = n^{1.5-\epsilon}$ for the Goldreich PRG instantiated with the TSA predicate[7], for any constant $\epsilon > 0$, against subexponential SDP adversaries and $\mathbb{F}_2$ linear bias adversaries.

This understanding can be generalized.

SoS Attacks. In fact for attacks relying on Semi-Definite Programming (SDP), there is a very powerful infrastructure to prove systematic lower bounds. This is captured by the *sum-of-squares* (SoS) hierarchy [Sho87, Par00, Nes00, Las01]. It was proven in [KMOW17] that the Goldreich PRG with a stretch $m(n) = n^{1+(\frac{k}{2}-1)(1-\delta)}$ for some constant $\delta > 0$, when instantiated using a random hypergraph and a predicate P that is $k-$wise independent[8], will require an SoS program of level $O(n^{\delta})$ for deriving refutations. This translates (very roughly) to an SDP that requires $2^{O(n^{\delta})}$ time to solve. This shows that for the TSA predicate with stretch of $n^{1.5-c}$, the SDP approach will take at least $2^{O(n^{2c})}$ time perform refutations/inversion.

$\mathbb{F}_2$ Linear Bias. These attacks are distinguishing attacks. $\mathbb{F}_2$ linear bias security consists of proving the following. For outputs $y_1, \ldots, y_m$ of the PRG, it requires that for every non-empty set $S \subseteq [n]$, it holds that $\left|\mathbb{E}[\oplus_{i \in S} y_i] - 0.5\right| \leq 2^{-n^{\epsilon}}$ for some constant $\epsilon > 0$. Usually this is a very hard property to prove in general. In fact, we only have sound analysis of very few predicates [MST03, ABR12, OW14, AL16]. The analysis in [AL16] is the first incident where a general degree d of the predicate is considered. Unfortunately, the analysis there can't be applied in our case because the parameters they achieve are not good enough for our setting. Unless a theorem already exists, we won't be discussing about these attacks for most of our candidates.

Algebraic Attacks / Guess and Determine Attacks. Algebraic attacks consists of resolution style attacks where some equations are set up and then they are manipulated until a search or refutation is made. This class of attacks capture the Gröbner Basis Attacks. In order to avoid the algebraic attacks with the stretch $m(n) = n^s$, as outlined by [AL16], the predicate should have a rational degree [9] greater than s. The reason for that is that, if the rational degree is lower than s, then the following happens. Write $\mathsf{P} \cdot \mathsf{Q} = \mathsf{R}$ where Q and R are degree $e < s$ functions. Given samples $(y_1, \ldots, y_m)$ one can write $y_i \cdot \mathsf{Q}(\boldsymbol{x}_{S_i}) = \mathsf{R}(\boldsymbol{x}_{S_i})$ where S_i is the corresponding indices on which the predicate P was applied to obtain y_i. Note that these are m degree e equations. This

[7]Recall, $\mathsf{TSA}(x_1, \ldots, x_5) = x_1 \oplus x_2 \oplus x_3 \oplus \mathsf{AND}(x_4, x_5)$.

[8]A predicate is $k-$wise independent if for any set S of size at most $k-1$, $\mathbb{E}[P(x_1, \ldots, x_d) \oplus_{i \in S} x_i] = 0.5$.

[9]Recall that the rational degree of P is the minimum degree e such that there exist degree e predicates Q and R such that $PQ = R$. Rational Degree is also known as algebraic immunity.

system can be linearized if $s > e$. In [AL16], the authors also prove lower bounds for subexponential algorithms in this model but unfortunately they are too weak to be applied here. However, in a very interesting work [CDM$^+$18], this attack was further improved where the authors considered rational degrees of predicates obtained by fixing some bits of the input called the bit-fixing algebraic immunity (hence the name *Guess and Determine.*). Thereby, under reasonable heuristic assumptions fine-tuned trade-offs of stretch vs running time were obtained. Refer to Proposition 5, 7 and 8 in [CDM$^+$18] for details. The paper is also an excellent source on the concrete security of various candidates and a survey of state-of-the-art attacks. For our candidates, we estimate running times of these algorithms by relying on the theorems from this work. All known attacks for our candidates and required parameters require subexponential time. We discuss the state of some of the major known algorithms and how they fare against our candidates in Table 2.

We now discuss our candidates below and how each of the attacks discussed above fare for these candidates.

5.2 The $\mathsf{XORMAJ}_{\ell,\ell}$ Predicate

As suggested earlier, for a general degree, there is a gap between provable security against the classes of attacks discussed above and actual attacks known in practice. While for a general degree d, the best known analysis in [AL16] only constructs a PRG predicate that has a provable stretch [10] of $\Omega(n^{d/38})$. As pointed out it in Corollary 2, and Proposition 8 in [CDM$^+$18], any Goldreich PRG instantiated with a predicate of the form (e.g. the $\mathsf{XOR}_\ell\mathsf{MAJ}_k$ predicates.)

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

for a non-linear balanced predicate g of locality k, can be broken in polynomial time (under a heuristic assumption) if the stretch of the PRG is more than $\widetilde{O}(n^{\lceil \frac{k}{2} \rceil + 1})$. The predicate above if g is balanced, is $(\ell + 1)-$wise independent. Thus, in light of these attacks and the SDP atttacks, to design a predicate of this form in general, one needs $\frac{\ell+1}{2} > \frac{1}{2} \cdot \lceil \frac{k+\ell}{2} \rceil$, because of the SDP condition, and $\lceil \frac{k}{2} \rceil + 1 > \frac{1}{2} \cdot \lceil \frac{k+\ell}{2} \rceil$ because of the attacks in [CDM$^+$18]. This leaves us with a tight margin to develop predicates in this manner. One might choose $k = \ell$, where ℓ is odd. Then, in the first equation $\frac{\ell+1}{2} > \frac{\ell}{2}$ and in the second equation, $\frac{\ell+3}{2} > \frac{\ell}{2}$. Thus, for an odd $\ell \geq 3$ define:

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

This predicate above has been widely studied, and has been regarded as the gold standard PRG predicate owing to the fact that Majority has the optimal rational degree [AL16].

SoS Attacks. We consider a stretch of $n^{\frac{\ell+1}{2} - c}$ for some constant $c > 0$. Under such circumstances we can show an SoS lower bound relying on the result of [KMOW17] against subexponential sized SoS programs. The exact parameters are computed in Table 2.

Algebraic Attacks. Unfortunately, we can't use the theorems in [AL16] to argue provable security against such attacks, we show that these as well as the improved attacks in [CDM$^+$18] approximately take subexponential time for our parameter setting. The exact parameters are computed in Table 2. In the table we rely on Proposition 5, 7 and 8 in [CDM$^+$18] to populate the parameters.

[10] Actually the result holds for locality.

5.3 Low-Degree High-Locality Predicates

As pointed out in the previous section, in general, we just have small room of parameters to build predicates with the stretch $n^{\frac{\ell}{4}+\epsilon}$ where ℓ is the locality in the way described above.

That points us to the following issue. Much of the research has been done in optimizing locality of the PRG predicates vs the stretch. However, in this work, we actually do not care much about the locality. For us, it is the degree of the predicate of the integers that is crucial. This allows us to design clever predicates that has much lower degree than the locality.

For example, consider the predicate proposed by Lombardi and Vaikunthanathan [LV17b] that has a locality of 5, but a degree of just 3!

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

At first sight, it does not appear to have a degree of 3, but on careful examination we can indeed show this. We also extend this observation and design a family of predicates that have much lower $\mathbb{Z}$ degree than the locality. We now discuss the status of known attacks for this particular predicate.

- SoS **Attacks.** Since the predicate is $3-$wise independent, relying on the result of [KMOW17] it can be shown that for a stretch of $m(n) \leq n^{1.5-c}$ for any constant $c \in (0, 0.5)$, the predicate provably resists attacks via the *sum-of-squares* paradigm running in time $O(2^{n^{2c}})$.

- **Linear Bias Attacks.** In [LV17b] it was proven that for a stretch of $n^{1.25-c}$ for any $c > 0$, the predicate provably resists linear bias distinguishing attacks relying on the dichotomy theorem of [ABR12]. Also, authors conjecture, that for this candidate by a tighter analysis even a stretch of $n^{1.5-c}$ should be possible against linear bias attacks.

- **Algebraic and [CDM+18] Style Attacks.** First observe that the rational degree of TSA and TSPA is the same because the variables are just related by an invertible linear transformation. Namely,

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

  Thus many of the ideas used to analyze TSA can be applied here. We work out the running time of the known attacks as a function of stretch in Table 2 for these attacks.

Next, we consider the following instantiation inspired by the TSPA predicate above. We suggest a general approach using which we construct a predicate of locality $2 \cdot k + 1$, and a $\mathbb{Z}-$degree of just $k+1$ for any constant integer $k > 0$. The predicate additionally satisfies $(k+1)-$wise independence. Further, the non-linear part will have an $\mathbb{F}_2$ degree of k. This allows us to enlarge the margin in parameters for constructing useful predicates as discussed above. Consider g, a non-linear boolean function of $\mathbb{F}_2$ degree k. Then, the predicate is simply:

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

Put it simply, this can also be written in the template above:

$$\mathsf{P}_g(x_1 \ldots, x_{2k+1}) = x_1 \oplus g'(x_2, \ldots x_{2k+1}),$$

where,

$$g'(x_2 \ldots, x_{2k+1}) = x_2 \oplus \ldots \oplus x_{k+1} \oplus g(x_{k+2} \oplus x_2, \ldots, x_{k+1} \oplus x_{2k+1}).$$

Now we argue $(k+1)-$wise independence. The predicate above is $(k+1)-$wise independent.

The reason for that is that in Fourier notation[11]:

$$\widehat{\mathsf{P}_g}(X_1 \ldots, X_{2k+1}) = \Pi_{i \in [k+1]} X_i \cdot g(X_{k+2} \cdot X_2, \ldots, X_{2k+1} \cdot X_{k+1}).$$

Also observe that in the Fourier expansion:

$$\widehat{g}(Y_1 \ldots, Y_k) = \sum_{S \subseteq [k]} \widehat{g}_S \chi_S(Y_1, \ldots, Y_k).$$

We substitute $Y_i = X_{i+1} \cdot X_{k+i+1}$. Thus, we get:

$$\widehat{\mathsf{P}_g}(X_1 \ldots, X_{2k+1}) = \Pi_{i \in [k+1]} X_i \cdot \sum_{S \subset [k]} \widehat{g}_S \chi_S(X_2 \cdot X_{k+2}, \ldots, X_{2k+1} \cdot X_{k+1}).$$

Thus, the Fourier expansion of P_g is a homogeneous polynomial of degree $k+1$. Hence, the predicate is $(k+1)-$wise independent. From the above, it is also clear that $\mathbb{Z}$ degree of P_g is also $k+1$. Infact, TSPA is obtained as a special case of this compiler where g is just the AND function. For an odd $k \geq 3$, we consider $\mathsf{P}_{\mathsf{MAJ}_k}$ as one of our candidate. For this candidate, consider:

- SoS **Attacks.** Since the predicate is $(k+1)-$wise independent, relying on the result of [KMOW17] it can be shown that for a stretch of $m(n) \leq n^{\frac{k+1}{2}-c}$ for any constant $c > 0$), the predicate provably resists attacks via the *sum-of-squares* paradigm running in subexponential time.

- **Algebraic and [CDM+18] Style Attacks.** First observe that the rational degree of $\mathsf{P}_{\mathsf{MAJ}_k}$ and $\mathsf{XORMAJ}_{k+1,k}$ is same because the variables are just related by an invertible linear transformation. Thus many of the ideas used to analyze XORMAJ can be applied here. We work out the running time of the known attacks as a function of stretch in Table 2 for these attacks.

5.4 Justifying Security of the Combined Assumptions

We now discuss the plausibility of our assumptions along with the binary LWE leakage part. The first category of attacks we discuss consists of attacks targeting the binary LWE part alone. Since the standalone PRG security has been discussed above, we do not discuss it here. Then we discuss the third category of attacks that consists of algebraic attacks over $\mathbb{F}_p$ that utilize both the LWE samples and the PRG leakage on the error of the LWE samples.

5.4.1 Binary LWE Security

Binary LWE has been a subject of study in quite a few number of works [MP13, ACF+15, AG11, CTA19]. Let n denote the dimension of the secret. While the problem is provably hard, and backed by a security reduction from worst case lattice problems, when the the number of samples $m(n) = n(1+\Omega(\frac{1}{\log_2 n}))$ [MP13], the problem is easy when $m(n) \geq \Omega(n^2)$, as shown by [AG11]. We

[11]Recall that for any boolean function $f : \{0,1\}^n \to \{0,1\}$, $f(x_1, \ldots, x_n)$, the fourier expansion of f, denoted by $\widehat{f} : \{-1,+1\}^n \to \{-1,+1\}$, is related as:

$$\widehat{f}(X_1, \ldots, X_n) = \sum_{S \subseteq [n]} \widehat{f}_S \cdot \chi_S(X_1, \ldots, X_n).$$

Here $\widehat{f}(X_1, \ldots, X_n) = 1 - 2 \cdot f(x_1, \ldots, x_n)$ and each $X_i = 1 - 2 \cdot x_i$. For any set S, $\chi_S(X_1, \ldots, X_n) = \prod_{i \in S} X_i$.

work in the regime when the number of samples $m(n) = n^s$ for some $s \in (1, 2)$. Under this regime, there are two kinds of algorithms that are studied.

Gröbner Basis Attacks: Arora-Ge algorithm [AG11] is a special case of a whole family of algebraic algorithms that consider all degree D algebraic constraints implied by the given equations for some large enough D so that the ideal generated by the unique solution can be recovered. Depending on the constraints, the degree defines the running time of the algorithm. The running time of these algorithm typically roughly grows like $n^{O(D)}$. In [CTA19], it was proven that Gröbner basis algorithm require $2^{O(n^{\epsilon})}$ time to run assuming that the number of samples are given by $m(n) = n^{2-\epsilon}$ for some $\epsilon > 0$. We will discuss this aspect again when we talk about the third category of attacks.

Lattice Attacks: The only attacks based on lattice reduction techniques that we are aware of apply to LWE more generally, and not just to binary-error LWE. The most relevant attack reduces the LWE instance to a BDD problem and then use the BKZ algorithm [Sch94] to solve it (see, e.g., [Ste] for details). With our setting of parameters, the time complexity of this attack would be $\Omega(2^{n^{0.5+\epsilon-\rho}})$. Because $\rho < 0.5$, this yields at best a subexponential attack.

5.4.2 Algebraic Attacks on the Combined Assumption

A natural approach to combine the information from both the PRG and LWE samples can be to form all equations that one can and then compute the Gröbner basis of the system generated by the equations. Recall a typical instance of our assumption contains:

- LWE samples $\{\boldsymbol{a}_i, b_i = \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle + e_i \mod p\}$ for $i \in [n]$. Here, $\boldsymbol{s}$ has dimension $n^{0.5+\epsilon}$ for some $\epsilon > 0$.
- Degree-d PRG evaluations: $\boldsymbol{y} = \mathsf{G}(e_1, \ldots, e_n) = (\mathsf{G}_{n,1}(\boldsymbol{e}), \ldots, \mathsf{G}_{n,m(n)}(\boldsymbol{e}))$ where $m(n) = n^{\lceil \frac{d}{2} \rceil \cdot (0.5+\epsilon)+\rho}$.

This means, that one can form the following equations.

$$(b_i - \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle)^2 = (b_i - \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle) \; \forall i \in [n],$$

$$y_i = \mathsf{G}_{n,i}(b_1 - \langle \boldsymbol{a}_1, \boldsymbol{s} \rangle, \ldots, b_n - \langle \boldsymbol{a}_n, \boldsymbol{s} \rangle) \; \forall i \in [m].$$

Here, the first equation is result of booleanity of the errors e_i. We now consider an example of this case when $d = 3$, and G is the Goldreich PRG instantiated with the TSPA predicate. We set $\epsilon = 0.1$, $\rho = 0.04$ and $m = n^{1.24} = n^{\lceil \frac{3}{2} \rceil \cdot (0.5+\epsilon)+\rho}$. This enforces the dimension to be $n^{0.6} = n^{0.5+\epsilon}$. Thus we have $\ell = m + n$ equations. m of them are degree 3 equations and n of them are degree 2. Let us denote these equations as $\{q_i(\boldsymbol{s}) = 0\}_{i \in [\ell]}$. A quick and dirty way to approximately gauge the performance of Gröbner basis algorithm is to fix a degree D, and then collect all equations of the form:

$$h(\boldsymbol{s}) \cdot q_i(\boldsymbol{s}) = 0,$$

for all monomials h of degree upto $D - \deg(q_i)$. Finally, if degree D is large enough, and there exists a unique solution, there will exist a D at which point, we can perform gaussian elimination in $n^{0.6 \cdot O(D)}$ variables (variables corresponding to all monomials of degree less than or equal to D generated by $\boldsymbol{s}$) to recover the secret $\boldsymbol{s}$.

For this strategy to succeed we want that the number of monomials of degree less than or equal to D in $\boldsymbol{s}$ to be lesser than the number of equations formed. This happens when:

$$n \cdot \binom{n^{0.6}+D-2}{D-2} + n^{1.24} \cdot \binom{n^{0.6}+D-3}{D-3} \geq \binom{n^{0.6}+D}{D}.$$

Which means that $D \geq n^{0.1}$. We can also do a similar analysis for a general degree d, which will require:

$$n \cdot \binom{n^{0.5+\epsilon}+D-2}{D-2} + m(n) \cdot \binom{n^{0.5+\epsilon}+D-d}{D-d} \geq \binom{n^{0.5+\epsilon}+D}{D}.$$

Here, $m = n^{\lceil \frac{d}{2} \rceil \cdot (0.5+\epsilon)+\rho}$. This requires $D \geq O(\min(n^{\epsilon}, n^{\frac{1}{d} \cdot (\lfloor \frac{d}{2} \rfloor - \rho)}))$. In fact, the above approach is really simplified and ignores many subtle issue but gives a lower bound on the actual degree D that should be considered. For a brief discussion about this, please refer [CTA19]. We will use this calculation to denote running times for various predicates under the column GB in Table 2.

5.5 Summary: Our Assumptions

We start with a table of comparison of our three instantiations where we list four kinds of attacks. SoS represent the sum-of-squares attacks applicable only to the PRG part of the instance. BKZ represent the running time obtained by using BKZ algorithm only the binary LWE part of the instance. GB represent an approximation of the running time of the algebraic attacks over $\mathbb{F}_p$ on the combined assumption discussed in the previous section. Finally in the last column we compute the running time for attacks on the PRG predicates using Propositions 5, 7 and 8 in [CDM+18]. We make the following assumption:

Assumption 5.1 (TSPA-LWEleak Assumption). *The Goldreich pseudorandom generator construction instantiated with the* TSPA *predicate satisfies* $\mathsf{TSPA\text{-}LWEleak}_{3,\epsilon,\rho}$ *security for some constants* $\epsilon > 0$ *and* $\rho > 0$.

Similarly, we make the following assumptions:

Assumption 5.2 ($\mathsf{XORMAJ}_{\ell,\ell}$-LWEleak Assumption). *The Goldreich pseudorandom generator construction instantiated with the* $\mathsf{XORMAJ}_{\ell,\ell}$ *predicate for an odd integer* $\ell \geq 3$ *satisfies* $\mathsf{XORMAJ\text{-}LWEleak}_{2 \cdot \ell,\epsilon,\rho}$ *security for some constants* $\epsilon > 0$ *and* $\rho > 0$.

Assumption 5.3 ($\mathsf{P}_{\mathsf{MAJ}_k}$-LWEleak Assumption). *The Goldreich pseudorandom generator construction instantiated with the* $\mathsf{P}_{\mathsf{MAJ}_k}$ *predicate for an odd integer* $k \geq 3$ *satisfies* $\mathsf{P}_{\mathsf{MAJ}_k}\text{-}\mathsf{LWEleak}_{k+1,\epsilon,\rho}$ *security for some constants* $\epsilon > 0$ *and* $\rho > 0$.

6 Construction of Functional Encryption

In this section, we construct a sublinear public-key functional encryption scheme FE for circuit class $\mathcal{C}_{n,\lambda,\gamma}$ which consists of all circuits with n input bits, depth bounded by λ and number of output bits bounded by $\ell = n^{1+\gamma}$ for some constant $\gamma > 0$. We need following ingredients to build such a scheme:

P	d	m_1	m_2	SoS	BKZ	GB	[CDM+18]
TSPA	3	$n^{1.45}$	n^{1+c}	$n^{0.10}$	$n^{0.71}$	$n^{0.22}$	$n^{0.4}$
$\mathsf{XORMAJ}_{5,5}$	10	$n^{2.95}$	$n^{2.5+c}$	$n^{0.025}$	$n^{0.582}$	$n^{0.082}$	$n^{0.5125}$
$\mathsf{P}_{\mathsf{MAJ}_5}$	6	$n^{2.95}$	$n^{1.5+c}$	$n^{0.025}$	$n^{0.97}$	$n^{0.48}$	$n^{0.51}$

Table 2: Running time for various known inversion attacks. Above P is a predicate of degree d. m_1 denotes the considered stretch, m_2 is the minimum stretch required in order to construct obfuscation via our assumption. $c > 0$ is arbitrary constant. SoS denotes the attacks known via the Sum-of-Squares paradigm. BKZ denotes the running time of the attacks via the BKZ lattice reduction algorithm. GB denotes the algebraic attacks on the combined assumption based on the Gröbner Basis algorithm. The last column denotes the running time from the subexponential time algorithm in [CDM+18] (Propositions 5,7 and 8). The cells represent $\widetilde{O}(\log_2(\cdot))$ of the running times where we hide logarithmic factors. The value of ρ is chosen to be 0.01, and so the modulus is $p = O(2^{n^{0.01}})$. We set ϵ so that, $m_1 = n^{\lceil \frac{d}{2} \rceil \cdot (0.5+\epsilon)+\rho}$.

- (Single Ciphertext FE with Linear Key Generation:) We use the secret key functional encryption scheme, denoted by 1LGFEB, constructed in Section 7.4. Note the following properties of that scheme:
  - The function class is $\mathcal{C}_{n,\lambda}$. This consists of all polynomial sized circuits with n bit inputs, depth bounded by λ, and with one bit output.
  - Special structure*: The scheme satisfies special structure*. In particular, in that construction as with all the constructions in this paper, there is an algorithm PPGen which outputs crs that is used by all schemes in this paper. In particular, $\mathsf{PPGen}(1^\lambda, 1^n)$ outputs a string crs that contains a bilinear map description $\mathcal{PG} = (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, p, P_1, P_2, e)$ where the order of the group is p which is $\mathsf{poly}(\lambda)$ bit prime modulus for some polynomial poly. The crs also consists of another modulus p_1 along with a string PE.PK.
  - The scheme can be instantiated to satisfy $(1, Q_{\mathsf{sk}})-$ indistinguishability security. Here Q_{sk} is set to be equal to ℓ, the number of output bits for circuits in $\mathcal{C}_{n,\lambda,\gamma}$.
  - The noise used to generate the function secret keys is sampled from $[0, \mathsf{Bound}_{smdg}]$ where Bound_{smdg} is some polynomial in ℓ, λ and n.
- (Pseudorandom Generator G satisfying $\mathsf{G\text{-}LWEleak}_{d,\epsilon,\rho}$:) Another ingredient is a pseudorandom generator G that satisfies $\mathsf{G\text{-}LWEleak}_{d,\epsilon,\rho}$ assumption for a constant integer $d > 0$, and some constants $\epsilon \in (0, 1)$ and $\rho \in (0, 1)$. The modulus p that we use for this assumption is the same as the order of the bilinear map. The modulus p is a $\mathsf{poly}(\lambda)$ bit modulus (instantiated in Section 7.2), which for sufficiently large $n(\lambda)$ is less than 2^{n^ρ}. The constant γ will be set as some function of ϵ and ρ later.
- (PHFE for $\mathcal{F}_{O(n),d,p}$:) We require a simulation secure, public-key, partially hiding functional encryption scheme PHFE with linear efficiency. The function class $\mathcal{F}_{O(n),d,p}$ consists of all functions f that takes an input of the form $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{Z}_p^{O(n)} \times \mathbb{Z}_p^{O(n)}$ and computes $f(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j,k} f_{j,k}(\boldsymbol{x}) \cdot y_j \cdot y_k$ where $f_{j,k}$ is a degree d polynomial over $\boldsymbol{x}$. Finally given an encryption of

$(\boldsymbol{x}, \boldsymbol{y})$ and a function secret key for f, the decryption reveals $[f(\boldsymbol{x}, \boldsymbol{y})]_T$ in the target group. Such a scheme is constructed in Section 8. The actual length denoted by $O(n)$ here will be described later.

We now describe the construction.

Parameters: We now describe the setting of the parameters. These parameters will only be referred to in the proof of security and sublinearlity.

- The parameter instantiation for the modulus p and corresponding parameters for $\mathsf{1LGFEB}$ can be found in Section 7.2. In particular, p is a $\mathsf{poly}(\lambda)$ bit prime modulus for some polynomial poly.
- We will refer to a parameter n'. This will be the length of input of G. We set $n' = \lceil n^{\frac{1}{0.5+\epsilon} \cdot \frac{1}{\lceil \frac{d}{2} \rceil}} \rceil$. This setting ensures that the ciphertext size grows linearly in n.
- Using the properties above, we prove that the number of output bits allowed, $\ell \geq n^{1+\frac{\rho}{4d}}$. Thus, $\gamma > \frac{\rho}{4d}$. Therefore, by making a stronger assumption, we can obtain keys for circuits with larger number of output bits.

Construction: Please refer to the construction in Figure 6.

Correctness: Now we argue the correctness of the construction. Consider any message $m \in \{0,1\}^n$ and a circuit $C \in \mathcal{C}_{n,\lambda,\ell}$. Let $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ be an honest encryption of m. Also let $\mathsf{sk}_C = (\mathsf{sk}_{C,1}, \ldots, \mathsf{sk}_{C,\ell})$ denote the function secret key for C. Let us now revisit the decryption steps.

- Using the special structure* property of $\mathsf{1LGFEB}$, compute $\mathsf{ct}_{C,i}$ by evaluating ct_1 using the circuit C_i.
- Compute $g_T^{w_i} \leftarrow \mathsf{PHFE.Dec}(\mathsf{sk}_{C_i}, \mathsf{ct}_2)$.
- Compute $z_i = g_T^{\mathsf{ct}_{C_i} - w_i}$.
- Try to bruteforce recover exponent of z_i. If it is bounded by $100 \cdot \mathsf{Bound}_{smdg}$ in absolute value, set y_i to be 0 and otherwise, if the recovery fails, set $y_i = 1$.
- Output $(y_1, ..., y_\ell)$.

We now describe correctness for all $i \in [\ell]$. Let's first revisit decryption procedure for $\mathsf{1LGFEB}$. In $\mathsf{1LGFEB}$, due to the linear + round decryption property of $\mathsf{1LGFEB}$ given ct_1 and $\mathsf{1LGFEB.sk}_{C_i} \leftarrow \mathsf{1LGFEB.KeyGen}(\boldsymbol{s}_1, C_i, \ell)$, the following holds. Let ct_{C_i} denote the evaluated ct_1. Then, $\mathsf{ct}_{C_i} - \mathsf{1LGFEB.sk}_{C_i} = C_i(m)\lceil \frac{p}{2} \rceil + \mathsf{err}$ for some err which is bounded by $2 \cdot \mathsf{Bound}_{smdg}$ in absolute value. Unfortunately, we are not given $\mathsf{1LGFEB.sk}_{C_i}$. We are given sk_{C_i} which is a PHFE secret key for a function that computes something in the exponent of g_T that is close to the secret key $\mathsf{1LGFE.sk}_{C_i}$. In particular, if $\mathsf{1LGFEB.sk}_{C_i} = \langle \mathsf{crs}_{C_i}, \boldsymbol{s}_2 \rangle + \mathsf{err} \mod p$ where $\mathsf{err} \leftarrow [0, \mathsf{Bound}_{smdg}]$, sk_{C_i} allows one to compute $g_T^{w_i} = g_T^{\langle \mathsf{crs}_{C_i}, \boldsymbol{s}_2 \rangle + \sum_{j \in [t]} 2^{j-1} \cdot \mathsf{G}_{(i-1)\cdot t+j}(e_1, ..., e_n)}$. Observe that, $\sum_{j \in [t]} 2^{j-1} \cdot \mathsf{G}_{(i-1)\cdot t+j}(e_1, ..., e_n)$ allows one to compute some string in $[0, 2^t - 1]$. Since t is logarithmic in $\lceil \log_2 \mathsf{Bound}_{smdg} \rceil + 1$, this range is within $[0, 4 \cdot \mathsf{Bound}_{smdg}]$. Thus by decryption equation above,

$$|\mathsf{ct}_{C_i} - w_i - C_i(m)\lceil \frac{p}{2} \rceil| \leq 6 \cdot \mathsf{Bound}_{smdg}$$

Since $\mathsf{Bound}_{smdg} << p$, this proves the claim.

$\underline{\mathsf{FE.PPGen}(1^\lambda, 1^n):}$ Run $\mathsf{1LGFEB.PPGen}(1^\lambda, 1^n) \rightarrow \mathsf{crs}$. Implicit in the crs is a bilinear map description and a modulus p.

$\underline{\mathsf{FE.Setup}(\mathsf{crs}):}$ Run $\mathsf{PHFE.Setup}(\mathsf{crs}) \rightarrow (\mathsf{PHFE.pk}, \mathsf{PHFE.msk})$. Sample vectors $\boldsymbol{a}_i \leftarrow \mathbb{Z}_p^{n'^{0.5+\epsilon}}$ for $i \in [n']$ for n' specified later. Set $\mathsf{FE.pk} = (\{\boldsymbol{a}_i\}_{i\in[n']}, \mathsf{PHFE.pk})$ and $\mathsf{FE.msk} = \mathsf{PHFE.msk}$.

$\underline{\mathsf{FE.Enc}(\mathsf{FE.pk}, m \in \{0,1\}^n):}$ Run $\boldsymbol{s}_1 \leftarrow \mathsf{1LGFEB.Setup}(\mathsf{crs})$ and sample $\boldsymbol{s}_2 \leftarrow \mathbb{Z}_p^{n'^{0.5+\epsilon}}$. Then perform the following steps.

- Compute $\mathsf{ct}_1 \leftarrow \mathsf{1LGFEB.Enc}(\boldsymbol{s}_1, m)$.
- Sample $e_i \leftarrow \{0,1\}$ for $i \in [n']$. Then compute $b_i = \langle \boldsymbol{a}_i, \boldsymbol{s}_2 \rangle + e_i \mod p$.
- Compute $\mathsf{S} = (\boldsymbol{s}_2, 1)^{\otimes \lceil \frac{d}{2} \rceil}$. In other words, S consists of all monomials generated from $\boldsymbol{s}_2$ of degree less than or equal to $\lceil \frac{d}{2} \rceil$.
- Denote $\boldsymbol{b} = (b_1, ..., b_{n'})$
- Parse $\mathsf{FE.pk} = (\{\boldsymbol{a}_i\}_{i\in[n']}, \mathsf{PHFE.pk})$. Compute $\mathsf{ct}_2 \leftarrow \mathsf{PHFE.Enc}(\mathsf{PHFE.pk}, (\boldsymbol{b}, (\boldsymbol{s}_1, \mathsf{S})))$. Here the public component of the ciphertext is $\boldsymbol{b}$. Output $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$.

$\underline{\mathsf{FE.KeyGen}(\mathsf{FE.msk}, C):}$ On input a circuit $C : \{0,1\}^n \rightarrow \{0,1\}^\ell$ where $\ell = n^{1+\gamma}$ do the following. Denote $C = (C_1, \ldots, C_\ell)$ where each C_i is the circuit computing i^{th} bit of the circuit evaluation.

- For each $i \in [\ell]$, from the linear key generation structure of $\mathsf{1LGFEB}$ let crs_{C_i} denote the coefficient vector over $\mathbb{Z}_p$, computable from crs determinstically, that is used to generate secret key for circuit C_i.
- Compute $\mathsf{sk}_{C,i} \leftarrow \mathsf{PHFE.KeyGen}(\mathsf{PHFE.msk}, f_i)$ where f_i is described in Figure 6. Intuitively, this function allows one to generate $\mathsf{1LGFEB}$ secret keys for function C_i using the master secret key $\boldsymbol{s}_1$. The noise for this is sampled using the pseudorandom generator G evaluated on the error vector used to construct samples $\boldsymbol{b}$. Output $\mathsf{sk}_C = (\mathsf{sk}_{C_1}, \ldots \mathsf{sk}_{C_\ell})$

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

- Using the special structure* property of $\mathsf{1LGFEB}$, compute $\mathsf{ct}_{C,i}$ using the ciphertext ct_1.
- Compute $g_T^{w_i} \leftarrow \mathsf{PHFE.Dec}(\mathsf{sk}_{C_i}, \mathsf{ct}_2)$.
- Compute $z_i = g_T^{\mathsf{ct}_{C_i} - w_i}$.
- Try to bruteforce recover exponent of z_i. If it is bounded by $100 \cdot \mathsf{Bound}_{smdg}$ in absolute value, set y_i to be 0 and otherwise, if the recovery fails, set $y_i = 1$. Output $(y_1, ..., y_\ell)$.

Figure 5: Construction of Functional Encryption Scheme FE.

$f_i(\boldsymbol{b}, (\boldsymbol{s}_1, \mathsf{S}))$
Hardwired :$\{\boldsymbol{a}_i\}_{i \in [n']}, \mathsf{crs}_{C_i}, t = \lceil \log_2 \mathsf{Bound}_{smdg} \rceil + 1$

1. Let $\mathsf{G} : \{0,1\}^{n'} \to \{0,1\}^m$ be the PRG with stretch m. Assume $m \geq t \cdot \ell$. Let G_i for $i \in [m(n')]$ denote the function that computes i^{th} bit of the PRG output.
2. Output $f_i(\boldsymbol{b}, \boldsymbol{s}_1, \mathsf{S}) = \langle \mathsf{crs}_{C_i}, \boldsymbol{s}_1 \rangle + \sum_{j \in [1,t]} 2^{j-1} \cdot \mathsf{G}_{(i-1)\cdot t + j}(e_1, ..., e_{n'})$

Computability in $\mathcal{F}_{O(n),d,p}$:

- n' is set such that $\mathsf{S} \in \mathbb{Z}_p^n$. Note that S has $({n'}^{0.5+\epsilon} + 1)^{\lceil \frac{d}{2} \rceil}$ field elements. Note that n' is set so that S has $O(n)$ field elements. This means that $n' < n$. Dimension of $\boldsymbol{s}_1$ is also a fixed polynomial in the security parameter. Thus the input consists of $O(n)$ field elements.
- $\langle \mathsf{crs}_{C_i}, \boldsymbol{s}_1 \rangle$ is just a linear function in $\boldsymbol{s}_1$. Then, consider $\mathsf{G}_j(e_1, ..., e_{n'})$. This function is a boolean function and has a unique representation over $\mathbb{Z}[e_1, ..., e_{n'}]$, which can be obtained by arithmetizing the function. Namely, replace $e_1 \oplus e_2$ with $e_1 \cdot (1 - e_2) + e_2 \cdot (1 - e_1)$ and replace $e_1 \wedge e_2$ with $e_1 \cdot e_2$. Further, the $\mathbb{Z}$ degree of G is d. Thus, G_j can be written as a polynomial over $\mathbb{Z}[e_1, ..., e_{n'}]$ which has a canonical representation over $\mathbb{Z}_p[e_1, ..., e_{n'}]$.
- Finally, we note that $e_i = b_i - \langle a_i, \boldsymbol{s}_2 \rangle \mod p$ for all $i \in [n']$. This means that $\mathsf{G}_j(e_1, ..., e_{n'}) = \mathsf{G}_j(b_1 - \langle \boldsymbol{a}_1, \boldsymbol{s}_2 \rangle, ..., b_{n'} - \langle \boldsymbol{a}_{n'}, \boldsymbol{s}_2 \rangle) \mod p$. Thus, this can be implemented as a polynomial that is degree 2 in S and degree d in $\boldsymbol{b}$.

Figure 6: Circuit f_i used in FE key generation procedure.

Sublinearity: We now bound the size of the ciphertext. Assume in the analysis below that the size of the modulus p is some fixed polynomial in security parameter, as instantiated in Section 7.2. Since the ciphertext have two components, ct_1 and ct_2, the size of the ciphertext ct is the sum of $|\mathsf{ct}_1|$ and $|\mathsf{ct}_2|$. By compactness of $\mathsf{1LGFEB}$ scheme, $|\mathsf{ct}_1| \leq n \cdot \mathsf{poly}(\lambda)$ for some fixed polynomial poly. Now, by linear efficiency of PHFE scheme, $|\mathsf{ct}_2| \leq (|\boldsymbol{s}_2| + |\boldsymbol{b}| + |\mathsf{S}|) \cdot \mathsf{poly}(\lambda)$ for some fixed polynomial. Now, $|\boldsymbol{s}_2|$ is some fixed polynomial in the security parameter.

Observe that $|\boldsymbol{b}| \leq n' \cdot \log_2 p$ and $|\mathsf{S}| \leq (2 \cdot n')^{(0.5+\epsilon)\cdot\lceil \frac{d}{2} \rceil} \cdot \log_2 p = O((2 \cdot n')^{(0.5+\epsilon)\cdot\lceil \frac{d}{2} \rceil} \cdot \mathsf{poly}(\lambda))$. In order to guarantee, sublinearity, we set n' as follows. Set n' so that:

$$(2 \cdot n')^{(0.5+\epsilon)\cdot\lceil \frac{d}{2} \rceil} \approx n$$
$$n' \approx n^{\frac{1}{0.5+\epsilon}\cdot\frac{1}{\lceil \frac{d}{2} \rceil}} \cdot \alpha.$$

Here α is some constant greater than 0. Now, with this n', let us find out the value of γ and ℓ. Observe that the stretch $m(n') \geq n'^{\frac{1}{2}\cdot\lceil \frac{d}{2} \rceil + d\cdot\epsilon+\rho}$. This can be written as:

$$\begin{aligned} m \geq & n'^{\frac{1}{2}\cdot\lceil \frac{d}{2} \rceil\cdot(0.5+\epsilon)+\rho} \\ \geq & \beta \cdot n^{\frac{1}{\lceil \frac{d}{2} \rceil\cdot(0.5+\epsilon)}\cdot(\lceil \frac{d}{2} \rceil\cdot(0.5+\epsilon)+\rho)} \\ \geq & \beta \cdot n^{1+\frac{\rho}{3\cdot d}}. \end{aligned}$$

Above β is some constant greater than 0. Here the last inequality requires that $d \geq 3$ and $\epsilon < 0.5$.

Now the number of output bits can be lower bounded by $\ell \geq m/t$, since for every key query t PRG output bits are used. Since $t \leq O(\log_2 \lambda)$, we have $\ell \geq n^{1+\frac{\rho}{4\cdot d}}$. Thus γ can be set as $\frac{\rho}{4\cdot d}$. This proves sublinearity as this shows that the length of the ciphertext is $O(n \cdot \mathsf{poly}(\lambda))$ and the number of output bits for circuits in $\mathcal{C}_{n,\lambda,\ell}$ tolerated are atleast $\ell \geq n^{1+\frac{\rho}{4d}}$. In fact, our scheme places no restriction on the size of the circuit C, only the length of the output needs to be lesser than ℓ bits. Thus it satisfies the notion of output sublinearity.

Security: We now prove security. Let the parameters be set as described in the construction. Then, we prove the following:

Theorem 6.1. *Assume that the following assumptions holds for some constants $\epsilon, \rho \in (0, 0.5)$, a constant integer $d \geq 3$ and the parameters described in Section 7.2.*

- *There exists a PRG G satisfying Definition 5.4 (instantiable using Assumption 5.1, or Assumption 5.2 or Assumption 5.3),*
- *LWE assumption, and,*
- *Standard assumptions over bilinear groups (SXDH and the Bilateral DLIN assumptions),*

then, there exists a (output) sublinearly efficient public-key Functional Encryption scheme.

Since PHFE can be built assuming the SXDH and bilateral DLIN assumptions and $\mathsf{1LGFEB}$ can be built using the LWE assumption, the result follows from the following lemma:

Lemma 6.1. *Assuming that there exists constant integer $d \geq 3$ and constants ρ and ϵ in $(0, 1)$ such that:*

- G *satisfies* $\mathsf{G\text{-}LWEleak}_{d,\epsilon,\rho}$ *security,*
- $\mathsf{1LGFEB}$ *is a* $(1, n^{1+\gamma}) - \mathsf{IND}$ *secure secret key functional encryption scheme where* $\gamma = \frac{\rho}{4 \cdot d}$*, and,*
- PHFE *is a public key simulation secure functional encryption scheme,*

the construction above is a secure single key sublinear public key Functional Encryption scheme.

Proof Overview: The proof of the construction is straightforward. First, we start simulating the PHFE ciphertext and the function keys. In doing this, the view of the adversary no longer consists of S. Then, we start hardwiring the output values of the decryption using the randomness sampled from a truly uniform distribution from $[0, 2^t - 1]$ as opposed to using the actual PRG output. This jump is indistinguishable and follows from $\mathsf{G\text{-}LWEleak}_{d,\epsilon,\rho}$ security. Then, we invoke the security of $\mathsf{1LGFEB}$ to go to a hybrid independent of the challenge bit. We now write hybrids and argue indistinguishability between them.
Hybrid$_0$:

- The adversary outputs $m_0, m_1 \in \{0,1\}^n$ along with a circuit $C \in \mathcal{C}_{n,\lambda,\gamma}$ such that $C(m_0) = C(m_1)$.
- The challenger runs $\mathsf{PPGen}(1^\lambda, 1^n) \rightarrow \mathsf{crs}$. Note that crs has a modulus p.
- Sample $\boldsymbol{a}_i \leftarrow \mathbb{Z}_p^{n'^{0.5+\epsilon}}$ for $i \in [n']$.
- Run $\mathsf{PHFE.Setup}(\mathsf{crs}) \rightarrow (\mathsf{PHFE.pk}, \mathsf{PHFE.msk})$. Set $\mathsf{FE.pk} = (\{\boldsymbol{a}_i\}_{i \in [n']}, \mathsf{PHFE.pk})$ and $\mathsf{FE.msk} = \mathsf{PHFE.msk}$.
- Sample a bit $\mu \leftarrow \{0,1\}$. Compute the challenge ciphertext as follows. Run $\boldsymbol{s}_1 \leftarrow \mathsf{1LGFEB.Setup}(\mathsf{crs})$ and sample $\boldsymbol{s}_2 \leftarrow \mathbb{Z}_p^{n'^{0.5+\epsilon}}$. Then perform the following steps.
  - Compute $\mathsf{ct}_1 \leftarrow \mathsf{1LGFEB.Enc}(\boldsymbol{s}_1, m_\mu)$.
  - Sample $e_i \leftarrow \{0,1\}$ for $i \in [n']$. Then compute $b_i = \langle \boldsymbol{a}_i, \boldsymbol{s}_2 \rangle + e_i \mod p$.
  - Compute $\mathsf{S} = (\boldsymbol{s}_2, 1)^{\otimes \lceil \frac{d}{2} \rceil}$. In other words, S consists of all monomials generated from $\boldsymbol{s}_2$ of degree less than or equal to $\lceil \frac{d}{2} \rceil$.
  - Denote $\boldsymbol{b} = (b_1, ..., b_{n'})$
  - Parse $\mathsf{FE.pk} = (\{\boldsymbol{a}_i\}_{i \in [n']}, \mathsf{PHFE.pk})$. Compute $\mathsf{ct}_2 \leftarrow \mathsf{PHFE.Enc}(\mathsf{PHFE.pk}, (\boldsymbol{b}, (\boldsymbol{s}_1, \mathsf{S})))$. Here the public component of the ciphertext is $\boldsymbol{b}$.
  - Output $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$.
- Compute a secret key for circuit C as follows. Denote $C = (C_1, \ldots, C_\ell)$ where each C_i is the circuit computing i^{th} bit of the circuit.
  - For each $i \in [\ell]$, from the linear key generation structure of $\mathsf{1LGFEB}$ let crs_{C_i} denote the coefficient vector used to generate secret key for circuit C_i.
  - Compute $\mathsf{sk}_{C,i} \leftarrow \mathsf{PHFE.KeyGen}(\mathsf{PHFE.msk}, f_i)$ where f_i is the function in $\mathcal{F}_{O(n),d,p}$ described in the key generation procedure above.
  - Output $\mathsf{sk}_C = (\mathsf{sk}_{C_1}, \ldots \mathsf{sk}_{C_\ell})$
- Hand over to the adversary $(\mathsf{crs}, \mathsf{FE.pk}, \mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2), \mathsf{sk}_C = (\mathsf{sk}_{C,1}, \ldots, \mathsf{sk}_{C,\ell}))$.

In the next hybrid, we simulate PHFE ciphertext and the secret keys.
$\mathbf{Hybrid}_1$:

- The adversary on input outputs $m_0, m_1 \in \{0,1\}^n$ along with a circuit $C \in \mathcal{C}_{n,\lambda,\gamma}$ such that $C(m_0) = C(m_1)$.
- The challenger runs $\mathsf{PPGen}(1^\lambda, 1^n) \to \mathsf{crs}$. Note that crs has a modulus p.
- Sample $\boldsymbol{a}_i \leftarrow \mathbb{Z}_p^{n'^{0.5+\epsilon}}$ for $i \in [n']$.
- [**Change**] Run $\mathsf{PHFE}.\widetilde{\mathsf{Setup}}(\mathsf{crs}) \to (\mathsf{PHFE}.\widetilde{\mathsf{pk}}, \mathsf{PHFE}.\widetilde{\mathsf{msk}})$. Set $\mathsf{FE.pk} = (\{\boldsymbol{a}_i\}_{i \in [n']}, \mathsf{PHFE}.\widetilde{\mathsf{pk}})$ and $\mathsf{FE.msk} = \mathsf{PHFE}.\widetilde{\mathsf{msk}}$.
- Sample a bit $\mu \leftarrow \{0,1\}$. Compute the challenge ciphertext as follows. Run $\boldsymbol{s}_1 \leftarrow \mathsf{1LGFEB.Setup}(\mathsf{crs})$ and sample $\boldsymbol{s}_2 \leftarrow \mathbb{Z}_p^{n'^{0.5+\epsilon}}$. Then perform the following steps.
  - Compute $\mathsf{ct}_1 \leftarrow \mathsf{1LGFEB.Enc}(\boldsymbol{s}_1, m_\mu)$.
  - Sample $e_i \leftarrow \{0,1\}$ for $i \in [n']$. Then compute $b_i = \langle \boldsymbol{a}_i, \boldsymbol{s}_2 \rangle + e_i \mod p$.
  - Compute $\mathsf{S} = (\boldsymbol{s}_2, 1)^{\otimes \lceil \frac{d}{2} \rceil}$. In other words, S consists of all monomials generated from $\boldsymbol{s}_2$ of degree less than or equal to $\lceil \frac{d}{2} \rceil$.
  - Denote $\boldsymbol{b} = (b_1, ..., b_{n'})$
  - [**Change**] Parse $\mathsf{FE.pk} = (\{\boldsymbol{a}_i\}_{i \in [n']}, \mathsf{PHFE}.\widetilde{\mathsf{pk}})$. Compute $\mathsf{ct}_2 \leftarrow \mathsf{PHFE}.\widetilde{\mathsf{Enc}}(\mathsf{PHFE}.\widetilde{\mathsf{pk}}, \boldsymbol{b})$. Here the public component of the ciphertext is $\boldsymbol{b}$.
  - Output $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$.
- Compute a secret key for circuit C as follows. Denote $C = (C_1, \ldots, C_\ell)$ where each C_i is the circuit computing i^{th} bit of the circuit.
  - For each $i \in [\ell]$, from the linear key generation structure of $\mathsf{1LGFEB}$ let crs_{C_i} denote the coefficient vector used to generate secret key for circuit C_i.
  - [**Change**] Compute $\mathsf{sk}_{C,i} \leftarrow \mathsf{PHFE}.\widetilde{\mathsf{KeyGen}}(\mathsf{PHFE}.\widetilde{\mathsf{msk}}, f_i, \theta_i = f_i(\boldsymbol{b}, (\boldsymbol{s}_2, \mathsf{S}))$ where f_i is the function in $\mathcal{F}_{O(n),d,p}$ described in the key generation procedure above. Note that $\theta_i = \langle \mathsf{crs}_{C_i}, \boldsymbol{s}_1 \rangle + \sum_{j \in [t]} 2^{j-1} \cdot \mathsf{G}_{(i-1)\cdot t + j}(e_1, \ldots, e_n)$.
  - Output $\mathsf{sk}_C = (\mathsf{sk}_{C_1}, \ldots \mathsf{sk}_{C_\ell})$
- Hand over to the adversary $(\mathsf{crs}, \mathsf{FE.pk}, \mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2), \mathsf{sk}_C = (\mathsf{sk}_{C,1}, \ldots, \mathsf{sk}_{C,\ell}))$.

Lemma 6.2. *Assuming that the* PHFE *scheme satisfies simulation security, then, for any p.p.t adversary* $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_0) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_1)]| \leq \mathsf{negl}(\lambda)$.

Proof. The only difference between these two hybrids is that in $\mathbf{Hybrid}_1$, the public key pk, the ciphertext ct_2 and the PHFE function keys for f_i for all $i \in [\ell]$ are simulated, whereas, in $\mathbf{Hybrid}_0$ they were generated using the honest algorithms. Note that everything else in the hybrid can be simulated and the master secret key of the PHFE scheme is not in the view of the adversary. Thus, we can build a reduction to the security of the PHFE scheme where given an adversary $\mathcal{A}$ distinguishing these two hybrids with probability δ, the reduction can break the simulation security of PHFE with probability δ. ☐

In the next hybrid, we use the assumption and replace $\mathsf{G}_j(e_1, \ldots, e_n)$ with random bits.
$\mathbf{Hybrid}_2$:

- The adversary on input outputs $m_0, m_1 \in \{0,1\}^n$ along with a circuit $C \in \mathcal{C}_{n,\lambda,\gamma}$ such that $C(m_0) = C(m_1)$.
- The challenger runs $\mathsf{PPGen}(1^\lambda, 1^n) \to \mathsf{crs}$. Note that crs has a modulus p.
- Sample $\boldsymbol{a}_i \leftarrow \mathbb{Z}_p^{n'^{0.5+\epsilon}}$ for $i \in [n']$.
- Run $\mathsf{PHFE}.\widetilde{\mathsf{Setup}}(\mathsf{crs}) \to (\mathsf{PHFE}.\widetilde{\mathsf{pk}}, \mathsf{PHFE}.\widetilde{\mathsf{msk}})$. Set $\mathsf{FE.pk} = (\{\boldsymbol{a}_i\}_{i\in[n']}, \mathsf{PHFE}.\widetilde{\mathsf{pk}})$ and $\mathsf{FE.msk} = \mathsf{PHFE}.\widetilde{\mathsf{msk}}$.
- Sample a bit $\mu \leftarrow \{0,1\}$. Compute the challenge ciphertext as follows. Run $\boldsymbol{s}_1 \leftarrow \mathsf{1LGFEB.Setup}(\mathsf{crs})$ and sample $\boldsymbol{s}_2 \leftarrow \mathbb{Z}_p^{n'^{0.5+\epsilon}}$. Then perform the following steps.
  - Compute $\mathsf{ct}_1 \leftarrow \mathsf{1LGFEB.Enc}(\boldsymbol{s}_1, m_\mu)$.
  - Sample $e_i \leftarrow \{0,1\}$ for $i \in [n']$. Then compute $b_i = \langle \boldsymbol{a}_i, \boldsymbol{s}_2 \rangle + e_i \mod p$.
  - Compute $\mathsf{S} = (\boldsymbol{s}_2, 1)^{\otimes \lceil \frac{d}{2} \rceil}$. In other words, S consists of all monomials generated from $\boldsymbol{s}_2$ of degree less than or equal to $\lceil \frac{d}{2} \rceil$.
  - Denote $\boldsymbol{b} = (b_1, ..., b_{n'})$
  - [**Change**] Parse $\mathsf{FE.pk} = (\{\boldsymbol{a}_i\}_{i\in[n']}, \mathsf{PHFE}.\widetilde{\mathsf{pk}})$. Compute $\mathsf{ct}_2 \leftarrow \mathsf{PHFE}.\widetilde{\mathsf{Enc}}(\mathsf{PHFE}.\widetilde{\mathsf{pk}}, \boldsymbol{b})$. Here the public component of the ciphertext is $\boldsymbol{b}$.
  - Output $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$.
- Compute a secret key for circuit C as follows. Denote $C = (C_1, \ldots, C_\ell)$ where each C_i is the circuit computing i^{th} bit of the circuit.
  - For each $i \in [\ell]$, from the linear key generation structure of $\mathsf{1LGFEB}$ let crs_{C_i} denote the coefficient vector used to generate secret key for circuit C_i.
  - [**Change**] Compute $\mathsf{sk}_{C,i} \leftarrow \mathsf{PHFE}.\widetilde{\mathsf{KeyGen}}(\mathsf{PHFE}.\widetilde{\mathsf{msk}}, f_i, \widetilde{\theta}_i)$ where f_i is the function in $\mathcal{F}_{O(n),d,p}$ described in the key generation procedure above. Note that $\widetilde{\theta}_i = \langle \mathsf{crs}_{C_i}, \boldsymbol{s}_1 \rangle + r_i$ where $r_i \leftarrow [0, 2^t - 1]$.
  - Output $\mathsf{sk}_C = (\mathsf{sk}_{C_1}, \ldots \mathsf{sk}_{C_\ell})$
- Hand over to the adversary $(\mathsf{crs}, \mathsf{FE.pk}, \mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2), \mathsf{sk}_C = (\mathsf{sk}_{C,1}, \ldots, \mathsf{sk}_{C,\ell}))$.

Lemma 6.3. *Assuming that the pseudorandom generator* $\mathsf{G} : \{0,1\}^{n'} \to \{0,1\}^{m(n')}$ *scheme satisfies* $\mathsf{G\text{-}LWEleak}_{d,\epsilon,\rho}$ *security, then, for any p.p.t adversary* $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_1) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_2)]| \le \mathsf{negl}(\lambda)$.

Proof. The only difference between these two hybrids is that in $\mathbf{Hybrid}_1$, for all $i \in [\ell]$, θ_i is generated as $\theta_i = \langle \mathsf{crs}_{C_i}, \boldsymbol{s}_1 \rangle + \sum_{j\in[t]} 2^{j-1} \cdot \mathsf{G}_{(i-1)\cdot t+j}(e_1, \ldots, e_n)$. However, in $\mathbf{Hybrid}_2$, it is generated as $\widetilde{\theta}_i = \langle \mathsf{crs}_{C_i}, \boldsymbol{s}_1 \rangle + r_i$ where $r_i \leftarrow [0, 2^t - 1]$. Note, that in both the hybrids, the secret vector $\boldsymbol{s}_2$ is not in the view of the adversary. The claim follows from the assumption as the tuple $(\boldsymbol{b}, \mathsf{G}(e_1, ..., e_n))$ is computationally indistinguishable to $(\boldsymbol{b}, (u_1, \ldots, u_m))$ where $u_i \leftarrow \{0,1\}$ for all $i \in [n']$. Then, one can set $r_i = \sum_j 2^{j-1} \cdot u_{(i-1)\cdot t+j}$ for all $i \in [\ell]$ to perform the reduction. If the adversary can distinguish between these two hybrids with probability δ, then, the reduction can win the $\mathsf{G\text{-}LWEleak}_{d,\epsilon,p}$ security game with advantage δ. □

Finally, we replace the encryption ct_1 to an encryption of m_0. This hybrid is independent of the challenge bit μ.
Hybrid$_3$:

- The adversary on input outputs $m_0, m_1 \in \{0,1\}^n$ along with a circuit $C \in \mathcal{C}_{n,\lambda,\gamma}$ such that $C(m_0) = C(m_1)$.
- The challenger runs $\mathsf{PPGen}(1^\lambda, 1^n) \rightarrow \mathsf{crs}$. Note that crs has a modulus p.
- Sample $\boldsymbol{a}_i \leftarrow \mathbb{Z}_p^{n'^{0.5+\epsilon}}$ for $i \in [n']$.
- Run $\mathsf{PHFE}.\widetilde{\mathsf{Setup}}(\mathsf{crs}) \rightarrow (\mathsf{PHFE}.\widetilde{\mathsf{pk}}, \mathsf{PHFE}.\widetilde{\mathsf{msk}})$. Set $\mathsf{FE.pk} = (\{\boldsymbol{a}_i\}_{i\in[n']}, \mathsf{PHFE}.\widetilde{\mathsf{pk}})$ and $\mathsf{FE.msk} = \mathsf{PHFE}.\widetilde{\mathsf{msk}}$.
- Sample a bit $\mu \leftarrow \{0,1\}$. Compute the challenge ciphertext as follows. Run $\boldsymbol{s}_1 \leftarrow \mathsf{1LGFEB.Setup}(\mathsf{crs})$ and sample $\boldsymbol{s}_2 \leftarrow \mathbb{Z}_p^{n'^{0.5+\epsilon}}$. Then perform the following steps.
  - [**Change**] Compute $\mathsf{ct}_1 \leftarrow \mathsf{1LGFEB.Enc}(\boldsymbol{s}_1, m_0)$.
  - Sample $e_i \leftarrow \{0,1\}$ for $i \in [n']$. Then compute $b_i = \langle \boldsymbol{a}_i, \boldsymbol{s}_2 \rangle + e_i \mod p$.
  - Compute $\mathsf{S} = (\boldsymbol{s}_2, 1)^{\otimes \lceil \frac{d}{2} \rceil}$. In other words, S consists of all monomials generated from $\boldsymbol{s}_2$ of degree less than or equal to $\lceil \frac{d}{2} \rceil$.
  - Denote $\boldsymbol{b} = (b_1, ..., b_{n'})$
  - Parse $\mathsf{FE.pk} = (\{\boldsymbol{a}_i\}_{i\in[n']}, \mathsf{PHFE}.\widetilde{\mathsf{pk}})$. Compute $\mathsf{ct}_2 \leftarrow \mathsf{PHFE}.\widetilde{\mathsf{Enc}}(\mathsf{PHFE}.\widetilde{\mathsf{pk}}, \boldsymbol{b})$. Here the public component of the ciphertext is $\boldsymbol{b}$.
  - Output $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$.
- Compute a secret key for circuit C as follows. Denote $C = (C_1, \ldots, C_\ell)$ where each C_i is the circuit computing i^{th} bit of the circuit.
  - For each $i \in [\ell]$, from the linear key generation structure of $\mathsf{1LGFEB}$ let crs_{C_i} denote the coefficient vector used to generate secret key for circuit C_i.
  - Compute $\mathsf{sk}_{C,i} \leftarrow \mathsf{PHFE}.\widetilde{\mathsf{KeyGen}}(\mathsf{PHFE}.\widetilde{\mathsf{msk}}, f_i, \widetilde{\theta}_i)$ where f_i is the function in $\mathcal{F}_{O(n),d,p}$ described in the key generation procedure above. Note that $\widetilde{\theta}_i = \langle \mathsf{crs}_{C_i}, \boldsymbol{s}_1 \rangle + r_i$ where $r_i \leftarrow [0, 2^t - 1]$.
  - Output $\mathsf{sk}_C = (\mathsf{sk}_{C_1}, \ldots \mathsf{sk}_{C_\ell})$
- Hand over to the adversary $(\mathsf{crs}, \mathsf{FE.pk}, \mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2), \mathsf{sk}_C = (\mathsf{sk}_{C,1}, \ldots, \mathsf{sk}_{C,\ell}))$.

Lemma 6.4. *Assuming* $\mathsf{1LGFEB}$ *satisfies* $(1,\ell)-$*indistinguishability security, then, for any p.p.t adversary* $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_2) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_3)]| \leq \mathsf{negl}(\lambda)$.

Proof. The only difference between these two hybrids is how ct_1 is generated. In **Hybrid$_2$**, ct_1 is generated as $\mathsf{ct}_1 = \mathsf{1LGFEB.Enc}(\boldsymbol{s}_1, m_\mu)$ where as in **Hybrid$_3$**, $\mathsf{ct}_1 = \mathsf{1LGFEB.Enc}(\boldsymbol{s}_1, m_0)$. Now, in both the hybrids, the secret keys for functions C_i for $i \in [\ell]$ are geenrated as in honest algorithm of $\mathsf{1LGFEB}$. Also observe that for any $\mu \in \{0,1\}$, $C(m_0) = C(m_\mu)$ and the number of issued function keys are bounded by ℓ. Thus, the indistinguishability of these two hybrids can directly be reduced to the security of $\mathsf{1LGFEB}$ scheme. □

Remark 6.1 ($(\mathsf{size}, \frac{1}{\lambda})-$security from $(\mathsf{size}, \frac{1}{\lambda})$ secure $\Delta\mathsf{RG}$)**.** We now remark about how instead of using the pseudorandom generator G to get polynomially secure functional encryption scheme, we could have used a $\Delta\mathsf{RG}$ (proposed by [AJS18, AJL$^+$19, JLMS19] and defined in Section A) to obtain a $(\mathsf{size}, \frac{1}{\lambda})$ secure FE. A fully secure FE can be obtained by relying on the security amplification theorem in [AJS18]. The idea is that in the encryption algorithm we replace S with the private part of the $\Delta\mathsf{RG}$ seed, $\mathsf{Seed.Priv} = (\mathsf{Seed.Priv}(1), \mathsf{Seed.Priv}(2))$. Further, we replace LWE samples $(b_1, \ldots, b'_n)$ with the public part $\mathsf{Seed.Pub}$ of the $\Delta\mathsf{RG}$ seed. The function key remains the same except that it replaces the randomness generation function part with the part that computes $\Delta\mathsf{RG.Eval}(\mathsf{Seed})$. The parameter B to use for $\Delta\mathsf{RG}$ is the same as Bound_{smdg}. The proof is identical except that the hybrids invoking standard security of G will be replaced with the hybrid invoking the $\frac{1}{\lambda}$ security of the $\Delta\mathsf{RG}$.

Remark 6.2 (On Subexponential Security.)**.** In the security proof, we proved standard polynomial security of the scheme above. For obtaining $i\mathcal{O}$, we actually need the scheme to be subexponentially secure. This can be obtained if we assume PHFE, G, and $\mathsf{1LGFEB}$ are subexponentially secure. This can be obtained if we asssume SXDH, bilateral DLIN and G to be subexponentially secure and LWE holds against subexponential time adversaries.

Remark 6.3 (Secret-key FE.)**.** In order to build a public-key FE, we used a public-key PHFE that can be built using SXDH and Bilateral DLIN as in Section 8. However, if we cared only for a secret-key FE we could have used secret-key PHFE built in [JLMS19] from the SXDH Assumption.

6.1 Theorems for Indistinguishability Obfuscation

We obtain the following main result:

Theorem 6.2. *Assuming the following assumptions hold:*

- *SXDH and bilateral DLIN assumptions over bilinear maps.*
- *Learning with Error assumption.*
- *A pseudorandom generator G satisfying $\mathsf{G} - \mathsf{LWEleak}_{d,\epsilon,\rho}$ security (Can be instantiated using Assumption 5.1, Assumption 5.2 or Assumption 5.3) for some constants $d \geq 3$ and constants $\epsilon, \rho \in (0, 0.5)$.*

There exists a sublinearly efficient public-key Functional Encryption scheme for all polynomial sized circuits.

For secret-key FE we could have just used SXDH instead of two assumptions as described above. Since secret-key subexponentially secure FE implies $i\mathcal{O}$ [AJ15, BV15, KNT18], we obtain the following result:

Theorem 6.3. *Assuming the following assumptions hold:*

- *Subexponentially secure SXDH over bilinear maps.*
- *Learning with Error assumption against adversaries running in subexponential time.*
- *A subexponentially secure pseudorandom generator G satisfying $\mathsf{G} - \mathsf{LWEleak}_{d,\epsilon,\rho}$ security (instantiable using Assumption 5.1, or Assumption 5.2, or Assumption 5.3) for some constants $d \geq 3$ and constants $\epsilon, \rho \in (0, 0.5)$.*

There exists an $i\mathcal{O}$ scheme for all polynomial sized circuits.

Similarly, we can also obtain the following result assuming the existence of a perturbation resilient generator $\Delta\mathsf{RG}$ computable by constant degree polynomials. We write the result for obtaining both $i\mathcal{O}$ and FE.

Theorem 6.4. *Assuming the following assumptions hold:*

- *SXDH assumption over bilinear maps holds against adversaries of subexponential size.*
- *Learning with Error assumption against adversaries of subexponential size.*
- *A $(s, \frac{1}{\lambda})$ secure $\Delta\mathsf{RG}$ computable by constant degree polynomials where s is some subexponential function (See Section A for the definition.).*

Then, there exists a secure $i\mathcal{O}$ scheme for all circuits and a secret-key functional encryption scheme for all circuits.

7 Single Ciphertext Functional Encryption with Linear KeyGen from LWE

In this section, we construct a variant of secret key functional encryption satisfying the following specifications. We denote this primitive by $\mathsf{1LGFE}$.

- (Function Class $\mathcal{F}$.) The function class for $\mathsf{1LGFE}$ is $\mathcal{C}_{n,\lambda}$ which consists of all polynomial sized boolean circuits that output a single bit, takes as input n input bits and has depth bounded by λ. Here n is polynomially related to the security parameter.
- (Security.) Satisfies $(1, Q_{\mathsf{sk}})-$IND security as in Definition 4.4. That is, the number of ciphertexts is bounded by 1 and the number of secret keys are bounded by any desired polynomial Q_{sk}.
- (Efficiency.) Satisfies linear efficiency/compactness as in Definition 4.6. Further, the size of the ciphertext is independent of the polynomial Q_{sk}.
- Also admits Special Structure* defined in Definition 4.10.

To build this, we first show a scheme satisfying Special Structure (refer Definition 4.9) and then show that the scheme can be modified very slightly to satisfy Special Structure* as in Definition 4.10.

We will construct such a scheme relying on the GVW predicate encryption scheme [GVW15]. Below we recall some preliminaries from there and then we construct $\mathsf{1LGFE}$.

7.1 GVW Preliminaries

Predicate Encryption. Now we recall the definition of predicate encryption scheme. A predicate encryption is a functional encryption scheme as described in Section 4. There are following differences.

- Encryptor encrypts messages of the form (attr, m) where $\mathsf{attr} \in \{0,1\}^n$ and m is a bit.
- The circuit class is in $\mathcal{C}_{n+1,\lambda+1}$. Each circuit is of the form C_P, where P is a predicate in $\mathcal{C}_{n,\lambda}$. C_P on input (attr, m) outputs m if $P(\mathsf{attr}) = 1$ and 0 otherwise.

- Security definition allows adversary to ask for any number of functional keys corresponding to predicates $P_1, ..., P_\eta$ as long as $P_i(\mathsf{attr}_0) = P_i(\mathsf{attr}_1) = 0$ where (attr_0, m_0) and (attr_1, m_1) are the challenge messages. In such a setting the adversary needs to distinguish between encryption of (attr_0, m_0) from encryption of (attr_1, m_1).

For a complete definition refer [GVW15]. For our construction, we require some special properties from the predicate encryption scheme, such as efficiency, circuit homomorphism etc. All these properties are satisfied by the construction of [GVW15], and we recall them next. The text below will assume familiarity with some lattice preliminaries described in Section 3.2.

Properties of GVW Predicate Encryption Scheme. Let $n = \mathsf{poly}(\lambda)$ for any polynomial poly. We now describe various algorithms and associated properties of the GVW predicate encryption scheme. We denote the scheme by PE.

Setup. The setup algorithm takes as input security parameter λ and n and outputs a public key PK and a secret key SK. Namely, $\mathsf{Setup}(1^\lambda, 1^n) \to (\mathsf{PK}, \mathsf{SK})$

- As a part of PK is the modulus p_1. Length of p_1 is $O(\mathsf{poly}_1(\lambda))$. It also outputs dimensions $\dim_1 = \mathsf{poly}_2(\lambda)$ and $\dim_2 = \mathsf{poly}_3(\lambda)$ where these are some fixed polynomials. These are the dimensions of various matrices used in the scheme.
- PK consists of uniform matrices $\mathbf{B}_1, ..., \mathbf{B}_\ell, \mathbf{A}, \mathbf{D}$ where $\ell = n \cdot \mathsf{poly}(\lambda)$ for some polynomial poly. Each matrix is in $\mathbb{F}_{p_1}^{\dim_1 \times \dim_2}$. This is also the space of the gadget matrix $\mathbf{G}$.

Encryption. The encryption algorithm takes as input public key PK, attribute $\mathsf{attr} \in \{0,1\}^n$ and a message $m \in \{0,1\}$ and does the following. $\mathsf{Enc}(\mathsf{PK}, \mathsf{attr}, m) \to (\mathsf{ct}_1, \mathsf{ct}_2)$, Now we describe in more detail.

- The encryption algorithm first samples a secret vector $\boldsymbol{s}$ from $\chi^{\dim_1 \times 1}$. Here χ is LWE error distribution used by the scheme. Then, it encodes attr to output $\widehat{\mathsf{attr}} = (\widehat{\mathsf{attr}}_p, \widehat{\mathsf{attr}}_s) \in \mathbb{F}_{p_1}^\ell$.
- Now ct_1 is constructed as follows.
  - Compute $\boldsymbol{b}_i = \boldsymbol{s}^T(\mathbf{B}_i + \widehat{\mathsf{attr}}_i \mathbf{G}) + \mathbf{E}_i$ for $i \in [\ell]$. Here $\mathbf{E}_i \leftarrow \chi^{1 \times \dim_2}$.
  - Output $\mathsf{ct}_1 = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_\ell, \widehat{\mathsf{attr}}_p)$
- Now ct_2 is constructed as follows.
  - Compute $\boldsymbol{a} = \boldsymbol{s}^T \mathbf{A} + \mathbf{E}_1$. Here $\mathbf{E}_1 \leftarrow \chi^{1 \times \dim_2}$.
  - Compute $\boldsymbol{d} = \boldsymbol{s}^T \mathbf{D} + \mathbf{E}_2 + m\lfloor p_1/2 \rfloor [1, 0, ..., 0]$. Here $\mathbf{E}_2 \leftarrow \chi^{1 \times \dim_2}$.
  - Output $\mathsf{ct}_2 = (\boldsymbol{a}, \boldsymbol{d})$.
- By Enc_1 we denote the algorithm that takes as input PK and secret $\mathbf{s}$, attribute attr and outputs ct_1.
- Without loss of security we can assume $\boldsymbol{s}[1] = 1$ (first component of vector $\boldsymbol{s}$). This ensures that $\boldsymbol{v} = \boldsymbol{s}^T \mathbf{G}$ satisfies $\boldsymbol{v}[1] = 1$.
- In our construction, we will use Enc_1 algorithm instead of the encryption algorithm, thereby not computing ct_2 at all. This does not hamper security as we are just giving less information.

Evaluation. There are two algorithms: EvalPK and EvalCT. First we describe the $\mathsf{EvalPK}()$ algorithm. Formally, $\mathsf{EvalPK}(C, \mathbf{B}_1, ..., \mathbf{B}_\ell) \to \mathbf{B}_C$. On input $\mathbf{B}_1, \ldots, \mathbf{B}_\ell \in \mathbb{F}_{p_1}^{\dim_1 \times \dim_2}$ and $C \in \mathcal{C}_{n,\lambda}$ the algorithm deterministically outputs $\mathbf{B}_C \in \mathbb{F}_{p_1}^{\dim_1 \times \dim_2}$.
EvalCT is also a deterministic algorithm that takes as input $\widehat{\mathsf{attr}}_p, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_\ell$ and $C \in \mathcal{C}_{n,\lambda}$. Formally, $\mathsf{EvalCT}(\mathsf{PK}, C, \widehat{\mathsf{attr}}_p, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_\ell) \to \widehat{\boldsymbol{b}}_C$. Here, $\widehat{\boldsymbol{b}}_C$ has the following structure:

$$\widehat{\boldsymbol{b}}_C = \boldsymbol{s}^T(\mathbf{B}_C + (C(\mathsf{attr})\lfloor p_1/2 \rfloor + e_C)\mathbf{G}) + \mathbf{E}_C$$

Here $\|\mathbf{E}_C\|_\infty / p_1 < 2^{-\lambda^c}$ and $\|e_C\|/p_1 < 2^{-\lambda^c}$ for some constant $c > 0$. In fact $|e_C| < \mathsf{poly}(\lambda, n)$ for some polynomial.

Remark 7.1. The algorithms described above are already close enough to imply a construction of $\mathsf{1LGFE}$ where the encryption is simply Enc_1 above, the master secret key is $\boldsymbol{s}$ and the function key for any function C could just be computed as $\mathsf{sk}_C = \langle \boldsymbol{s}, \mathbf{B}_{C,1} \rangle + e$. Here e is chosen freshly from some bounded smudging distribution and $\mathbf{B}_{C,1}$ is the first column of $\mathbf{B}_C$. However, this leads to the decryption of ciphertext ct resulting in the following equation:

$$\mathsf{EvalCT}(\mathsf{PK}, C, \mathsf{ct}) - \mathsf{sk}_C = C(\mathsf{attr}) \cdot \lceil p_1/2 \rceil + e_C + \mathbf{E}_C[1] - e$$

Above, $\mathbf{E}_C[1]$ may not be polynomially bounded, and thus this does not fit in the requirements for an $\mathsf{1LGFE}$ scheme. To fix this issue, we introduce the following algorithm, which rounds the result of evaluating the ciphertext to another modulus p so that the rounded version of the error $\mathbf{E}'_C[1]$ also becomes polynomially bounded.

Rounding-Evaluation. We now describe a procedure of rounding evaluation, which can be done publicly. We denote this by $\mathsf{RoundEval}$. $\mathsf{RoundEval}$ takes as input $\mathsf{PK}, \mathsf{ct}_1 = (\widehat{\mathsf{attr}}_p, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_\ell)$, a circuit C, another modulus $p < p_1$.
More formally, $\mathsf{RoundEval}(\mathsf{PK}, C, \mathsf{ct}_1, p)$ does the following:

1. First run $\mathsf{EvalCT}(\mathsf{PK}, C, \mathsf{ct}_1) \to \widehat{\boldsymbol{b}}_C$.
2. Now compute $\widehat{\boldsymbol{b}}'_C = \lceil p/p_1 \cdot \widehat{\boldsymbol{b}}_C \rfloor$. Namely multiply $\widehat{\boldsymbol{b}}_C$ with p/p_1 over the reals and then take the nearest integer, component wise. $\widehat{\boldsymbol{b}}'_C$ is now a vector over $\mathbb{F}_p$.
3. Output $b'_C = \widehat{\boldsymbol{b}}'_C[1]$, the first element of vector $\widehat{\boldsymbol{b}}'_C$.

Now we observe the structure of b'_C. First observe $\widehat{\boldsymbol{b}}_C[1]$ has the following structure:

$$\widehat{\boldsymbol{b}}_C[1] = \boldsymbol{s}^T \cdot \boldsymbol{B}_{C,1} + (C(\mathsf{attr})\lfloor p_1/2 \rfloor + e_C) \cdot \boldsymbol{v}[1,1]) + \mathbf{E}_C[1].$$

Here $\boldsymbol{B}_{C,1}$ is the first column of $\mathbf{B}_C$ and $\boldsymbol{v} = \boldsymbol{s}^T \cdot \mathbf{G}$. Since $\boldsymbol{s}[1] = 1$, $\boldsymbol{v}[1] = 1$ the following holds:

$$\widehat{\boldsymbol{b}}_C[1] = \boldsymbol{s}^T \boldsymbol{B}_{C,1} + C(\mathsf{attr})\lfloor p_1/2 \rfloor + e_C + \mathbf{E}_C[1].$$

Let χ be a polynomially bounded distribution (bounded by $\mathsf{poly}_\chi(\lambda)$), then, we observe the following about b'_C relying on the theorems proven in [BGV12] (see lemma 1 of the paper).

Theorem 7.1. *Assuming:*

- $\widehat{\boldsymbol{b}}_C[1] = \boldsymbol{s}^T \boldsymbol{B}_{C,1} + C(\mathsf{attr})\lfloor p_1/2 \rfloor + e_C + \mathbf{E}_C[1]$ *where $\boldsymbol{s}$ is chosen from the distribution $\chi^{\dim_1}$*
- *χ is a polynomially bounded distribution, bounded by, $\mathsf{poly}_\chi(\lambda)$.*

Then $b'_C = \boldsymbol{s}^T \cdot \mathbf{B}'_{C,1} + C(\mathsf{attr})\lfloor p_1/2 \rfloor' + e'_C + \mathbf{E}'_C[1,1] + \mathsf{error}$. Here $\mathbf{B}'_{C,1}$ is the rounded version of $\mathbf{B}_{C,1}$, e' is a rounded version of e, $\lfloor p_1/2 \rfloor'$ is rounded version of $\lfloor p_1/2 \rfloor$ and $\mathbf{E}'_C[1]$ is rounded version of $\mathbf{E}_C[1,1]$. error is the rounding error satisfying $|\mathsf{error}| < \dim_1 \cdot \mathsf{poly}_\chi(\lambda) + 3$

7.2 Parameters.

Now we set parameters that will be relevant for our constructions. All these parameters can be realized using standard LWE assumption with subexponential approximation factors.

- $\dim_1$ and $\dim_2$ are chosen as in the [GVW15] predicate encryption scheme.
- Error distribution bound $\mathsf{poly}_\chi(\lambda)$ and the prime modulus p_1 are chosen that circuits of depth λ^3 can be evaluated. The bit length is therefore $\mathsf{poly}(\lambda)$ for some polynomial poly.
- Now we describe how p is chosen. The magnitude of p is so that the rounded evaluation error while computing circuits in $\mathcal{C}_{n,\lambda}$ is polynomially bounded. Namely, in the evaluation equation:

  $$\widehat{\boldsymbol{b}}_C[1] = \boldsymbol{s}^T \boldsymbol{B}_{C,1} + C(\mathsf{attr})\lfloor p_1/2 \rfloor + e_C + \mathbf{E}_C[1],$$

  In the construction of [GVW15] the evaluation error obtained by evaluating circuits of depth λ circuit satisfies,

  $$|\mathbf{E}_C[1]| \leq O((\dim_1 + \dim_2)^{\lambda^2} \cdot \mathsf{poly}_\chi(\lambda))$$

  Now p can be chosen so that:

  $$\left\lceil \frac{p \cdot (\dim_1 + \dim_2)^{\lambda^2} \cdot \mathsf{poly}_\chi(\lambda)}{p_1} \right\rceil = O(\mathsf{poly}(\lambda))$$

  for some polynomial poly.

  This can be achieved by setting:

  $$p = O\left(\frac{p_1}{(\dim_1 + \dim_2)^{\lambda^2}} \right).$$

- p is chosen as above to satisfy the equation above. Looking ahead, it will come from a bilinear map generation algorithm. It will be chosen to be a sufficiently large (subepxponential) prime satisfying the equation above.

Example Parameters:

$\log_2 p = \theta(\lambda^2)$	$\dim_1 = O(\lambda^4)$
$\dim_2 = O(\lambda^7)$	$\mathsf{poly}_\chi = \lambda^{20}$
$\log_2 p_1 = \theta(\lambda^3)$	

As shown in [GVW15], these parameters can be instantiated using LWE with subexponential approximation factors.

7.3 Construction of 1LGFE

With this the construction is really easy to follow. The construction can be found in Figure 7.3.

Remark 7.2. Observe that int the setup algorithm described in Figure 7.3, SK is just discarded. Also observe that a bilinear map is sampled here but the scheme below don't use it at all (except for the modulus p). In fact, all our schemes (including the ones that use the bilinear maps) described later will refer to the same PPGen algorithm.

Observe that correctness and syntactic properties are immediate due to the properties of the predicate encryption scheme. For completeness, we sketch these below.

$\underline{\mathsf{1LGFE.PPGen}(1^\lambda, 1^n):}$

- Run a bilinear map setup to generate a description of the bilinear map $\mathcal{PG} = (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, p, P_1, P_2, e)$. Here the order of the group p is set according to parameter instantiation as described in Section 7.2.
- Run $\mathsf{PE.Setup}(1^\lambda, 1^n) \to (p_1, \mathsf{PK}, \mathsf{SK})$. Parse $\mathsf{PK} = (\mathbf{B}_1, ..., \mathbf{B}_\ell)$. Set $\mathsf{crs} = (\mathcal{PG}, p, p_1, \mathsf{PK})$.

$\underline{\mathsf{1LGFE.Setup}(1^\lambda, 1^n, \mathsf{crs}):}$ Sample $\boldsymbol{s} \leftarrow \chi^{\dim_1 \times 1}$. Set $\mathsf{msk} = \boldsymbol{s}$.

$\underline{\mathsf{1LGFE.Enc}(\boldsymbol{s}, m):}$ Run $\mathsf{PE.Enc}_1(\mathsf{PK}, \boldsymbol{s}, m) \to \mathsf{ct}$. Output ct.

$\underline{\mathsf{1LGFE.KeyGen}(\boldsymbol{s}, C):}$ Compute $\mathsf{PE.EvalPK}(\mathsf{PK}, C) \to \mathbf{B}_C$. Let $\mathbf{B}_{C,1}$ be the first column of $\mathbf{B}_C$. Round this column to modulus p. Let this be denoted by $\mathbf{B}'_{C,1}$. Compute $\mathsf{sk}_C = \langle \mathbf{B}'_{C,1}, \boldsymbol{s} \rangle + e \mod p$ where e is uniformly chosen from $[-p/16, p/16]$.

$\underline{\mathsf{1LGFE.Dec}(\mathsf{sk}_C, \mathsf{ct}):}$ Compute $\mathsf{PE.RoundEval}(\mathsf{PK}, C, \mathsf{ct}, p) \to b'_C$. Compute $b'_C - \mathsf{sk}_C \mod p = y$. If $y \in [-p/4, p/4]$ output 0 otherwise output 1.

Figure 7: Construction of $\mathsf{1LGFE}$.

Correctness: If the setup, encryption and the key generation are done honestly, then from the properties of the PE scheme, the following happens. Let ct denote a ciphertext encrypting $m \in \{0,1\}^n$, and let sk_C be a function secret key for a circuit $C \in \mathcal{C}_{n,\lambda}$. Then, from the correctness of the PE scheme, $b'_C = \mathsf{PE.RoundEval}(\mathsf{pk}, C, \mathsf{ct}, p)$ has the following structure. $b'_C = \langle \boldsymbol{s}, \mathbf{B}'_{C,1} \rangle + C(m)\lceil p/2 \rceil + e'_C + \mathsf{error} \mod p$. If the parameters are chosen as prescribed in Section 7.2, then $e'_C + \mathsf{error}$ is bounded in absolute value by some polynomial Bound (See theorem 7.1). Now, $\mathsf{sk}_C = \langle \mathbf{B}'_{C,1}, \boldsymbol{s} \rangle + e \mod p$ where $e \in [-p/16, p/16]$. Further p is subexponentially large. Thus, in the final step, $y = b'_C - \mathsf{sk}_C \mod p = C(m)\lceil p/2 \rceil + e'_C + \mathsf{error} - e$. Because p is subexponentially large and $e'_C + \mathsf{error}$ is bounded by a polynomial bound Bound, if $C(m) = 0$ then $y \in [-p/4, p/4]$ and otherwise not.

Special Structure. Special structure is easier to justify. Observe that all three properties about the crs syntax, linear key generation and linear + round decryption can be verified by inspection.

We now describe the proof of security.

Theorem 7.2. *Assuming LWE assumption holds for the parameters described in Section 7.2, the construction* $\mathsf{1LGFE}$ *is a secure* $(1, \mathsf{poly}(\lambda))-$*indistinguishability secure secret key functional encryption scheme for any polynomial bound* poly.

Proof Overview: The security of this construction can be proven by a reduction to the security of the underlying PE scheme. In the first hybrid, the challenger encrypts m_b for a randomly chosen bit $b \leftarrow \{0,1\}$ and the keys are generated honestly as, $\mathsf{sk}_C = \langle \mathbf{B}'_{C,1}, \boldsymbol{s} \rangle + e \mod p$. In the next hybrid, we switch to generating sk_C as $b'_C - C(m_0)\lceil p/2 \rceil + e \mod p$ where b'_C is computed using $\mathsf{RoundEval}$ algorithm evaluated on the challenge ciphertext. As we show later these two hybrids are statistically close due to the smudging lemma 3.1. Finally, since the keys are simulated just from the ciphertext, in the last hybrid we switch the ciphertext to be an encryption of m_0. This change is indistinguishable due to the security of PE. This hybrid is independent of b.

$\mathbf{Hybrid}_0$:

- Adversary specifies $(m_0, m_1) \in \{0,1\}^n$ along with circuits $C_1, ..., C_Q \in \mathcal{C}_{n,\lambda}$ such that $C_i(m_0) = C_i(m_1)\ \forall i \in [Q]$.
- Challenger generates the $\mathsf{crs} = (p, p_1, \mathsf{PK}, \mathcal{PG})$ as in the algorithm.
- Sample $\boldsymbol{s} \leftarrow \chi^{\dim_1 \times 1}$. Set $\mathsf{msk} = \boldsymbol{s}$.
- Sample $b \leftarrow \{0,1\}$ and compute $\mathsf{ct} \leftarrow \mathsf{PE.Enc}_1(\mathsf{PK}, \boldsymbol{s}, m_b)$.
- Also for all $i \in [Q]$, compute $\mathsf{sk}_{C_i} \leftarrow \langle \mathbf{B}'_{C_i,1}, \boldsymbol{s} \rangle + e_i \mod p$ where $\mathbf{B}'_{C_i,1}$ is generated as in 1LGFE key generation algorithm by rounding the first column of $\mathsf{EvalPK}(\mathsf{PK}, C_i) \to \mathbf{B}_{C_i,1}$ and e_i is sampled uniformly from $[-p/16, p/16]$.
- Give to the adversary $\{\mathsf{crs}, \mathsf{ct}, \mathsf{sk}_{C_1}, ..., \mathsf{sk}_{C_Q}\}$

$\mathbf{Hybrid}_1$:

- Adversary specifies $(m_0, m_1) \in \{0,1\}^n$ along with circuits $C_1, ..., C_Q \in \mathcal{C}_{n,\lambda}$ such that $C_i(m_0) = C_i(m_1)\ \forall i \in [Q]$.
- Challenger generates the $\mathsf{crs} = (p, p_1, \mathsf{PK}, \mathcal{PG})$ as in the algorithm.
- Sample $\boldsymbol{s} \leftarrow \chi^{\dim_1 \times 1}$. Set $\mathsf{msk} = \boldsymbol{s}$.
- Sample $b \leftarrow \{0,1\}$ and compute $\mathsf{ct} \leftarrow \mathsf{PE.Enc}_1(\mathsf{pk}, \boldsymbol{s}, m_b)$.
- [**Change**] Also for all $i \in [Q]$, compute $\widetilde{\mathsf{sk}}_{C_i} \leftarrow \mathsf{PE.RoundEval}(\mathsf{PK}, C_i, \mathsf{ct}, p) - C_i(m_0)\lceil p/2 \rceil + e_i \mod p$ where e_i is sampled uniformly from $[-p/16, p/16]$.
- Give to the adversary $\{\mathsf{crs}, \mathsf{ct}, \widetilde{\mathsf{sk}}_{C_1}, ..., \widetilde{\mathsf{sk}}_{C_Q}\}$

The two hybrids above are statistically close.

Lemma 7.1. *If $p = \Omega(2^{\lambda^c})$ for some constant $c > 0$, then, for any adversary $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_0) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_1) = 1]| < Q \cdot 2^{-\lambda^{c'}}$ for some constant $c' > 0$.*

Proof. The only difference between the two hybrids is how the function key sk_{C_i} are generated. Note that in $\mathbf{Hybrid}_0$ it is generated as $\mathsf{sk}_{C_i} = \langle \mathbf{B}'_{C_i,1}, \boldsymbol{s} \rangle + e_i \mod p$. On the other hand in $\mathbf{Hybrid}_1$, $\widetilde{\mathsf{sk}}_{C_i,1} \leftarrow \mathsf{PE.RoundEval}(\mathsf{PK}, C_i, \mathsf{ct}, p) - C(m_0)\lceil p/2 \rceil + e_i \mod p$. Now observe that:

$$\mathsf{PE.RoundEval}(\mathsf{PK}, C_i, \mathsf{ct}, p) = \langle \mathbf{B}'_{C_i,1}, \boldsymbol{s} \rangle + C(m_b)\lceil p/2 \rceil + e_{C_i} + \mathsf{error}$$

Above, both e_{C_i} and the rounding error error are bounded polynomially by some polynomial in the security parameter $\mathsf{poly}(\lambda, n)$. Also $C_i(m_0) = C_i(m_1)$. Thus,

$$\widetilde{\mathsf{sk}}_{C_i} = \langle \mathbf{B}'_{C_i,1}, \boldsymbol{s} \rangle + e_{C_i} + \mathsf{error} + e_i$$

Now note that since p is subeponentially large and e_i is chosen from $[-p/16, p/16]$ uniformly, the statistical distance between $e_{C_i} + \mathsf{error} + e_i$ and e_i is $o(2^{-\lambda^{c'}})$ for some constant $c' > 0$. This follows due to the smudging lemma 3.1. Thus, the claim holds.

□

Finally, we switch the encryption of m_b with an encryption of m_0.

Hybrid$_2$:

- Adversary specifies $(m_0, m_1) \in \{0,1\}^n$ along with circuits $C_1, ..., C_Q \in \mathcal{C}_{n,\lambda}$ such that $C_i(m_0) = C_i(m_1)\ \forall i \in [Q]$.
- Challenger generates the $\mathsf{crs} = (p, p_1, \mathsf{PK}, \mathcal{PG})$ as in the algorithm.
- Sample $\boldsymbol{s} \leftarrow \chi^{\dim_1 \times 1}$. Set $\mathsf{msk} = \boldsymbol{s}$.
- [**Change**] Compute $\mathsf{ct} \leftarrow \mathsf{PE.Enc}_1(\mathsf{PK}, \boldsymbol{s}, m_0)$.
- Also for all $i \in [Q]$, compute $\widetilde{\mathsf{sk}}_{C_i} \leftarrow \mathsf{PE.RoundEval}(\mathsf{PK}, C_i, \mathsf{ct}, p) - C_i(m_0)\lceil p/2 \rceil + e_i \mod p$ where e_i is sampled uniformly from $[-p/16, p/16]$.
- Give to the adversary $\{\mathsf{crs}, \mathsf{ct}, \widetilde{\mathsf{sk}}_{C_1}, ..., \widetilde{\mathsf{sk}}_{C_Q}\}$

Lemma 7.2. *If* PE *is a secure predicate encryption scheme, then for any p.p.t. adversary* $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_1) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_2) = 1]| < \mathsf{negl}(\lambda)$ *where* negl *is some negligible function.*

Proof. The only difference between the hybrids is how ct is generated. In $\mathbf{Hybrid}_1$ it is generated as an encryption of m_b, whereas in $\mathbf{Hybrid}_2$ it is generated as an encryption of m_0. Note that neither the secret key $\boldsymbol{s}$, nor the randomness in the ciphertext is used to simulate the function secret keys. Thus, the claim holds due to a straightforward reduction to the security of PE. □

7.4 1LGFE with Polynomially Bounded Decryption Error

The scheme described in the previous section suffers from an undesirable property. The property is that upon decryption the adversary learns a value of the form $y = C(m)\lceil p/2 \rceil + \widehat{e}$ where $\widehat{e}$ can be subexponentially large. This is essential for the security proof due to the smudging lemma (See theorem 3.1.). However, for our purposes we need this error to be polynomially bounded in the security parameter as well as the parameter n as this computation would be done in the exponent of a group element and then recovered by brute force.

Observe that $\widehat{e} = e_C + \mathsf{error} - e$ where $e_C + \mathsf{error}$ is already polynomially bounded and comes from the ciphertext. On the other hand, e is the smudging noise that comes from the function secret key sk_C. This e was required to be subexponentially large for our proof strategy to work in the previous section (mainly due to the smudging lemma).

What we show next is that even if the smudging noise is chosen from a polynomially bounded distribution, not all hope is lost. In fact, with a polynomially bounded smudging noise we can guarantee that the security holds as long as a bounded number of key queries are made. The exact trade off between the bound on the smudging noise and the number of key queries is discussed next.

Let Q_{sk} denote the number of queries we are interested in. Let Bound be the polynomial bound as decribed in Section 7.2 on the magnitude of the noise generated during ciphertext evaluation. The scheme described in the previous section ensured security by choosing the smudging noise e used in the function secret key to be subexponentially larger than Bound. Now we show that, if we sample uniformly this smudging noise from $[0, \mathsf{Bound}_{smdg}]$ for a sufficiently large but *polynomial* Bound_{smdg}, we can still ensure security as long as upto Q_{sk} function secret keys are given out. Note that this does not affect the efficiency of the ciphertext at all, as the bit length of the modulus is logarithmic in Q_{sk}. We will let $\mathsf{Bound}_{smdg} > 4 \cdot \lambda \cdot \mathsf{Bound} \cdot Q_{\mathsf{sk}}$ for the rest of the section below. We will denote this scheme by 1LGFEB.

$\underline{\mathsf{1LGFE.PPGen}(1^\lambda, 1^n):}$

- Run a bilinear map setup to generate a description of the bilinear map $\mathcal{PG} = (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, p, P_1, P_2, e)$. Here the order of the group p is set according to parameter instantiation as described in Section 7.2.
- Run $\mathsf{PE.Setup}(1^\lambda, 1^n) \to (p_1, \mathsf{PK}, \mathsf{SK})$. Parse $\mathsf{PK} = (\mathbf{B}_1, ..., \mathbf{B}_\ell)$. Set $\mathsf{crs} = (\mathcal{PG}, p, p_1, \mathsf{PK})$.

$\underline{\mathsf{1LGFE.Setup}(1^\lambda, 1^n, \mathsf{crs}):}$ Sample $\boldsymbol{s} \leftarrow \chi^{\dim_1 \times 1}$. Set $\mathsf{msk} = \boldsymbol{s}$.

$\underline{\mathsf{1LGFE.Enc}(\boldsymbol{s}, m)}$: Run $\mathsf{PE.Enc}_1(\mathsf{PK}, \boldsymbol{s}, m) \to \mathsf{ct}$. Output ct.

$\underline{\mathsf{1LGFE.KeyGen}(\boldsymbol{s}, C, Q_{\mathsf{sk}}):}$ Compute $\mathsf{PE.EvalPK}(\mathsf{PK}, C) \to \mathbf{B}_C$. Let $\mathbf{B}_{C,1}$ be the first column of $\mathbf{B}_C$. Round this column to modulus p. Let this be denoted by $\mathbf{B}'_{C,1}$. Compute $\mathsf{sk}_C = \langle \mathbf{B}'_{C,1}, \boldsymbol{s} \rangle + e \mod p$ where e is uniformly chosen from $[0, \mathsf{Bound}_{smdg}]$.

$\underline{\mathsf{1LGFE.Dec}(\mathsf{sk}_C, \mathsf{ct}):}$ Compute $\mathsf{PE.RoundEval}(\mathsf{PK}, C, \mathsf{ct}, p) \to b'_C$. Compute $b'_C - \mathsf{sk}_C \mod p = y$. If $y \in [-p/4, p/4]$ output 0 otherwise output 1.

Figure 8: Construction of 1LGFEB.

Constructing 1LGFEB: The construction is described in 1LGFEB.

Remark 7.3. Observe that the only change over the construction of 1LGFE is that the key generation procedure takes polynomially bounded noise to do the smudging.

Correctness and Special Structure*: As before, the correctness property is immediate and follows similarly like the correctnes of the 1LGFE scheme. Also, the scheme above satisfies special structure* (See Definition 4.10) since, like 1LGFE, it satisfies special structure, but in addition, the decryption noise is polynomially bounded. This is because it is bounded by $\mathsf{Bound}_{smdg} + \mathsf{Bound}$ in absolute value. Since both Bound_{smdg} and Bound are polynomially bounded, the claim holds.

Now we prove security.

Theorem 7.3. *Assuming LWE assumption holds for the parameters described in Section 7.2, the construction* 1LGFEB *is a secure* $(1, Q_{\mathsf{sk}})-$*indistinguishability secure secret key functional encryption scheme.*

Proof Overview: The security of this construction can be proven by a reduction to the security of the underlying PE scheme. However, this time we won't be able to use the smudging lemma (theorem 3.1). Instead, we would consider a non-uniform reduction and rely on rather a heavy hammer from hardness amplification literature. We use the following lemma from [JP14, CCL18b]. We recall the variant from [CCL18b].

Theorem 7.4 (Imported Theorem [CCL18b]). *Let* $k, t \in \mathbb{N}, \epsilon > 0$, *and* $\mathcal{C}_{leak}$ *be a family of distinguisher circuits from* $\{0,1\}^k \times \{0,1\}^t \to \{0,1\}$ *of size* $s(k)$. *Then, for every distribution* (X, Z) *over* $\{0,1\}^k \times \{0,1\}^t$, *there exists a simulator* $h : \{0,1\}^k \to \{0,1\}^t$ *such that:*

1. *h has size bounded by* $s' = O(s \cdot 2^t \epsilon^{-2})$

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

$$\left| \Pr_{(x,z)\leftarrow(X,Z)}[C(x,z) = 1] - \Pr_{x\leftarrow X,h}[C(x,h(x)) = 1] \right| \leq \epsilon$$

Here is how we prove the theorem. First we consider a mental experiment. Suppose we are given a tuple $\mathbf{T} = \{\delta_i + e_i\}_{i\in[Q_{\mathsf{sk}}]}$ where $|\delta_i| < \mathsf{Bound}$ and $e_i \leftarrow [0, \mathsf{Bound}_{smdg}]$. Looking ahead, each δ_i represents the error in the evaluated ciphertext computed during the $\mathsf{PE.RoundEval}$ algorithm (Namely, $e'_{C_i} + \mathsf{error}_i$). Observe that $\delta_1, ..., \delta_{Q_{\mathsf{sk}}}$ are some complex function of ct and the circuits $C_1, \ldots, C_{Q_{\mathsf{sk}}}$. The idea is that if the parameters are chosen appropriately, we replace this tuple $\mathbf{T}$ by one that is sampled "efficiently" using a non-uniform function h applied on the ciphertext ct. Here by efficient we mean a circuit that is larger than the adversary, but only *polynomially* larger. Due to this lemma above these hybrids are indistinguishable. Finally, we replace ct to be an encryption of m_0, thereby making the game independent of b. This step is also indistinguishable due to the security of PE. We now describe the proof in more detail.

Theorem 7.5. *Assuming LWE assumption holds against all polynomial time adversaries, then for any p.p.t. adversary* $\mathcal{A}$, *and any constant* $c > 0$, *and any large enough security parameter* λ

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

We now present hybrids where the first hybrid corresponds to the security game for $\mathsf{1LGFEB}$, where as the last hybrid is independent of bit b. We argue indistinguishability between all these hybrids thereby proving security.

Hybrid$_0$:

- Adversary specifies $(m_0, m_1) \in \{0,1\}^n$ along with circuits $C_1, ..., C_{Q_{\mathsf{sk}}} \in \mathcal{C}_{n,\lambda}$ such that $C_i(m_0) = C_i(m_1)\ \forall i \in [Q_{\mathsf{sk}}]$.
- Challenger generates the $\mathsf{crs} = (p, p_1, \mathsf{PK}, \mathcal{PG})$ as in the algorithm.
- Sample $\boldsymbol{s} \leftarrow \chi^{\dim_1 \times 1}$. Set $\mathsf{msk} = \boldsymbol{s}$.
- Sample $b \leftarrow \{0,1\}$ and compute $\mathsf{ct} \leftarrow \mathsf{PE.Enc}_1(\mathsf{PK}, \boldsymbol{s}, m_b)$.
- Also for all $i \in [Q_{\mathsf{sk}}]$, compute $\mathsf{sk}_{C_i} \leftarrow \langle \mathbf{B}'_{C_i,1}, \boldsymbol{s}\rangle + e_i \mod p$ where $\mathbf{B}'_{C_i,1}$ is generated as in $\mathsf{1LGFEB}$ key generation algorithm by rounding the first column of $\mathsf{EvalPK}(\mathsf{PK}, C_i) \rightarrow \mathbf{B}_{C_i,1}$ and e_i is sampled uniformly from $[0, \mathsf{Bound}_{smdg}]$.
- Give to the adversary $\{\mathsf{crs}, \mathsf{ct}, \mathsf{sk}_{C_1}, ..., \mathsf{sk}_{C_{Q_{\mathsf{sk}}}}\}$

The next hybrid is inefficient. We define a machine Mach:

$\mathsf{Mach}(\mathsf{Z}, \mathsf{ct}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0), \mathsf{crs})$

1. Compute $\boldsymbol{s}$ by opening up Z by brute-force.
2. Compute $v_i = \mathsf{PE.RoundEval}(\mathsf{PK}, C_i, \mathsf{ct}, p)$ for all $i \in [Q_{\mathsf{sk}}]$. Let $e'_{C_i} = v_i - C(m_0)\lceil p/2\rceil - \langle \mathbf{B}'_{C_i}, \boldsymbol{s}\rangle \mod p$.
3. Sample $e_i \leftarrow [0, \mathsf{Bound}_{smdg}]$ for $i \in [Q_{\mathsf{sk}}]$. Let $w_i = e_i - e'_{C_i}$.
4. Compute $\widetilde{\mathsf{sk}}_{C_i} = \mathsf{PE.RoundEval}(\mathsf{PK}, C_i, \mathsf{ct}, p) - C_i(m_0)\lceil p/2\rceil + w_i \mod p$.

5. Output $\{\widetilde{\mathsf{sk}}_{C_i}\}_{i \in [Q_{\mathsf{sk}_{C_i}}]}$.

Hybrid$_1$:

- Adversary specifies $(m_0, m_1) \in \{0,1\}^n$ along with circuits $C_1, ..., C_{Q_{\mathsf{sk}}} \in \mathcal{C}_{n,\lambda}$ such that $C_i(m_0) = C_i(m_1)\ \forall i \in [Q]$.
- Challenger generates the $\mathsf{crs} = (p, p_1, \mathsf{PK}, \mathcal{PG})$ as in the algorithm.
- [**Change**] Sample $\boldsymbol{s} \leftarrow \chi^{\dim_1 \times 1}$. Set $\mathsf{msk} = \boldsymbol{s}$. It also computes a perfectly binding commitment of the secret key $\mathsf{Z} = \mathsf{Com}(s)$.
- Sample $b \leftarrow \{0,1\}$ and compute $\mathsf{ct} \leftarrow \mathsf{PE.Enc}_1(\mathsf{pk}, \boldsymbol{s}, m_b)$.
- [**Change**] Generate $\widetilde{\mathsf{sk}}_{C_i}$ as follows. Run $\mathsf{Mach}(\mathsf{Z}, \mathsf{ct}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0), \mathsf{crs}) \rightarrow \{\widetilde{\mathsf{sk}}_{C_i}\}_{i \in [Q_{\mathsf{sk}}]}$.
- Give to the adversary $\{\mathsf{crs}, \mathsf{ct}, \widetilde{\mathsf{sk}}_{C_1}, ..., \widetilde{\mathsf{sk}}_{C_Q}\}$

Lemma 7.3. *If* Com *is perfectly binding, then for any adversary* $\mathcal{A}$*,* $|\Pr[\mathcal{A}(\mathbf{Hybrid}_0) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_1) = 1]| = 0$.

Proof. The difference between two hybrids is that in $\mathbf{Hybrid}_0$, the secret keys sk_{C_i} are generated as in the honest secret key generation algorithm. In $\mathbf{Hybrid}_1$, the function secret keys are generated by an inefficient algrorithm Mach, which first inverts the commitment Z to recover $\boldsymbol{s}$ first. Then it computes $\mathsf{PE.RoundEval}(\mathsf{PK}, C_i, \mathsf{ct}, p) = \langle \mathbf{B}'_{C_i,1}, \boldsymbol{s} \rangle + C_i(m_0)\lceil p/2 \rfloor + e'_{C_i}$. It first finds out e'_{C_i}. Then it subtracts from this, $e'_{C_i} - e_i$ where e_i is chosen at random from $[0, \mathsf{Bound}_{smdg}]$ along with $C_i(m_0)$. If $\boldsymbol{s}$ is recovered correctly, then,

$$\widetilde{\mathsf{sk}}_{C_i} = \langle \mathbf{B}'_{C_i,1}, \boldsymbol{s} \rangle + e_i \mod p$$

where $e_i \leftarrow [0, \mathsf{Bound}_{smdg}]$. This is identical to the distribution of sk_{C_i} in $\mathbf{Hybrid}_0$. □

The next hybrid relies on the following basic fact. Let $\delta \in [-\mathsf{Bound}, \mathsf{Bound}]$. Then, consider sampling $e \leftarrow [0, \mathsf{Bound}_{smdg}]$. If $\mathsf{Bound}_{smdg} > 2 \cdot \mathsf{Bound}$ then the distribution corresponding to $\mu = \delta + e$ is uniform over $[\delta, \mathsf{Bound}_{smdg} + \delta]$. Thus μ can equivalently be sampled by sampling uniformly from $(\mathsf{Bound}, \mathsf{Bound}_{smdg} - \mathsf{Bound})$ with probability $\alpha = \frac{\mathsf{Bound}_{smdg} - 2 \cdot \mathsf{Bound} - 1}{\mathsf{Bound}_{smdg} + 1}$ and with probability $1 - \alpha$, sampling uniformly from $[\delta, \mathsf{Bound}_{smdg} + \delta] \setminus (\mathsf{Bound}, \mathsf{Bound}_{smdg} - \mathsf{Bound})$.

$\mathsf{Mach}_1(\mathsf{Z}, \mathsf{ct}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0), \mathsf{crs})$

1. Compute $\boldsymbol{s}$ by opening up Z by brute-force.
2. Compute $v_i = \mathsf{PE.RoundEval}(\mathsf{PK}, C_i, \mathsf{ct}, p)$ for all $i \in [Q_{\mathsf{sk}}]$. Let $e'_{C_i} = v_i - C(m_0)\lceil p/2 \rfloor - \langle \mathbf{B}'_{C_i}, \boldsymbol{s} \rangle \mod p$.
3. Compute $\mathbf{L} \leftarrow \mathsf{Mach}_{inner}(\{e'_{C_i}\}_{i \in Q_{\mathsf{sk}}})$. For each $i \in [Q_{\mathsf{sk}}]$, if $(i, u_i) \in \mathbf{L}$, for some u_i, set $w_i = u_i$, else sample $w_i \leftarrow [\mathsf{Bound} + 1, \mathsf{Bound}_{smdg} - \mathsf{Bound} - 1]$.
4. Compute $\widetilde{\mathsf{sk}}_{C_i} = \mathsf{PE.RoundEval}(\mathsf{PK}, C_i, \mathsf{ct}, p) - C(m_0)\lceil p/2 \rfloor + w_i \mod p$.

5. Output $\{\widetilde{\mathsf{sk}}_{C_i}\}_{i\in[Q_{\mathsf{sk}_{C_i}}]}$.

Here, Mach_{inner} is implemented using the following algorithm:

$\mathsf{Mach}_{inner}(\{e'_{C_i}\}_{i\in[Q_{\mathsf{sk}}]})$

1. Maintain a list $\mathbf{L}$. Initialise it to be empty.
2. For each $i \in [Q_{\mathsf{sk}_{C_i}}]$, sample $e_i \leftarrow [0, \mathsf{Bound}_{smdg}]$. Compute $u_i = e_i - e'_{C_i}$. If $u_i \notin [\mathsf{Bound}+1, \mathsf{Bound}_{smdg} - \mathsf{Bound} - 1]$ append (i, u_i) in the list $\mathbf{L}$.

Hybrid$_2$:

- Adversary specifies $(m_0, m_1) \in \{0,1\}^n$ along with circuits $C_1, ..., C_{Q_{\mathsf{sk}}} \in \mathcal{C}_{n,\lambda}$ such that $C_i(m_0) = C_i(m_1)\ \forall i \in [Q]$.
- Challenger generates the $\mathsf{crs} = (p, p_1, \mathsf{PK}, \mathcal{PG})$ as in the algorithm.
- Sample $\boldsymbol{s} \leftarrow \chi^{\dim_1 \times 1}$. Set $\mathsf{msk} = \boldsymbol{s}$. It also computes a perfectly binding commitment of the secret key $\mathsf{Z} = \mathsf{Com}(s)$.
- Sample $b \leftarrow \{0,1\}$ and compute $\mathsf{ct} \leftarrow \mathsf{PE.Enc}_1(\mathsf{pk}, \boldsymbol{s}, m_b)$.
- [**Change**] Generate $\widetilde{\mathsf{sk}}_{C_i}$ as follows. Run $\mathsf{Mach}_1(\mathsf{Z}, \mathsf{ct}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0), \mathsf{crs}) \rightarrow \{\widetilde{\mathsf{sk}}_{C_i}\}_{i\in[Q_{\mathsf{sk}}]}$.
- Give to the adversary $\{\mathsf{crs}, \mathsf{ct}, \widetilde{\mathsf{sk}}_{C_1}, ..., \widetilde{\mathsf{sk}}_{C_Q}\}$

Lemma 7.4. *For any adversary $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_1) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_2) = 1]| = 0$.*

Proof. The difference between these two hybrids is how w_i is sampled. We prove a claim next, which will be useful to prove this hybrid.

Claim 7.1. *Fix a $\delta \in [-\mathsf{Bound}, \mathsf{Bound}]$. Consider the following two distributions:*
Distribution 1 :

- *Sample $e \leftarrow [0, \mathsf{Bound}_{smdg}]$.*
- *Output $\mu = e + \delta$.*

Distribution 2 :

- *Sample $e_1 \leftarrow [0, \mathsf{Bound}_{smdg}]$. If $\mu = \delta + e_1 \notin [\mathsf{Bound}+1, \mathsf{Bound}_{smdg} - \mathsf{Bound} - 1]$, output μ.*
- *Otherwise output $\mu \leftarrow [\mathsf{Bound}+1, \mathsf{Bound}_{smdg} - \mathsf{Bound} - 1]$*

These two distributions are identical.

Proof. The proof of this claim is straightforward. Consider distribution 1. For any $\mu \in [\delta, \mathsf{Bound}_{smdg} + \delta]$, the probability that the distribution samples μ is $\frac{1}{\mathsf{Bound}_{smdg}+1}$. For the second distribution, we consider two cases and compute probabilities.

- $\mu \in [\delta, \mathsf{Bound}_{smdg} + \delta] \setminus [\mathsf{Bound}+1, \mathsf{Bound}_{smdg} - \mathsf{Bound} - 1]$: This event happens only if μ shows up in Step 1 of the sampling algorithm. The probability of this happening is $\frac{1}{\mathsf{Bound}_{smdg}+1}$.

- $\mu \in [\mathsf{Bound}+1, \mathsf{Bound}_{smdg} - \mathsf{Bound} - 1]$: This event happens if in the first step, an element is sampled from $[\mathsf{Bound}+1, \mathsf{Bound}_{smdg} - \mathsf{Bound} - 1]$ and then μ is sampled in the second step. This happens with probability $\frac{\mathsf{Bound}_{smdg} - 2\cdot\mathsf{Bound} - 1}{\mathsf{Bound}_{smdg}+1} \cdot \frac{1}{B_{smdg} - 2\cdot\mathsf{Bound} - 1} = \frac{1}{\mathsf{Bound}_{smdg}+1}$. This step assumes $|\delta| \leq \mathsf{Bound}$.

Thus, these two distributions are identical. $\square$

With this claim at our disposal, we observe that the only difference in the hybrids $\mathbf{Hybrid}_1$ and $\mathbf{Hybrid}_2$ is how w_i is sampled for each i. Let e'_{C_i} be computed by Mach and Mach_1 respectively. In $\mathbf{Hybrid}_1$, w_i is generated as $e_i - e'_{C_i}$ where $e'_{C_i} \in [-\mathsf{Bound}, \mathsf{Bound}]$ and $e_i \leftarrow [0, \mathsf{Bound}_{smdg}]$. In $\mathbf{Hybrid}_2$, w_i is generated using sampler for distribution 2 by setting $\delta_i = -e'_{C_i}$ where the first step is computed by Mach_{inner} and the second by Mach_1. These two distributions are identical by the claim and hence the lemma holds. $\square$

The following hybrid is the same as the previous on except that the representation changes. In particular, $\mathsf{Mach}_{inner,1}$ remains the only inefficient algorithm.

```
$\mathsf{Mach}_2(\mathsf{Z}, \mathsf{ct}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0), \mathsf{crs})$
    1. Compute $\mathbf{L} \leftarrow \mathsf{Mach}_{inner}(\mathsf{Z}, \mathsf{crs}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0))$. For each $i \in [Q_{\mathsf{sk}}]$, if $(i, u_i) \in \mathbf{L}$, for some $u_i$, set $w_i = u_i$, else sample $w_i \leftarrow [-\mathsf{Bound}_{smdg} + \mathsf{Bound}, \mathsf{Bound}_{smdg} - \mathsf{Bound}]$.
    2. Compute $\widetilde{\mathsf{sk}}_{C_i} = \mathsf{PE.RoundEval}(\mathsf{PK}, C_i, \mathsf{ct}, p) - C_i(m_0)\lceil p/2 \rceil + w_i \mod p$.
    3. Output $\{\widetilde{\mathsf{sk}}_{C_i}\}_{i \in [Q_{\mathsf{sk}_{C_i}}]}$.
```

Here, $\mathsf{Mach}_{inner,1}$ is implemented using the following algorithm:

```
$\mathsf{Mach}_{inner,1}(\mathsf{Z}, \mathsf{crs}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0))$
    1. Compute $\boldsymbol{s}$ by opening up $\mathsf{Z}$ by brute-force.
    2. Compute $v_i = \mathsf{PE.RoundEval}(\mathsf{PK}, C_i, \mathsf{ct}, p)$ for all $i \in [Q_{\mathsf{sk}}]$. Let $e'_{C_i} = v_i - C_i(m_0)\lceil p/2 \rceil - \langle \mathbf{B}'_{C_i}, \boldsymbol{s} \rangle \mod p$.
    3. Maintain a list $\mathbf{L}$. Initialise it to be empty.
    4. For each $i \in [Q_{\mathsf{sk}_{C_i}}]$, sample $e_i \leftarrow [0, \mathsf{Bound}_{smdg}]$. Compute $u_i = e_i - e'_{C_i}$. If $u_i \notin [\mathsf{Bound}+1, \mathsf{Bound}_{smdg} - \mathsf{Bound} - 1]$ append $(i, u_i)$ in the list $\mathbf{L}$.
```

$\mathbf{Hybrid}_3$:

- Adversary specifies $(m_0, m_1) \in \{0,1\}^n$ along with circuits $C_1, ..., C_{Q_{\mathsf{sk}}} \in \mathcal{C}_{n,\lambda}$ such that $C_i(m_0) = C_i(m_1)\ \forall i \in [Q]$.
- Challenger generates the $\mathsf{crs} = (p, p_1, \mathsf{PK}, \mathcal{PG})$ as in the algorithm.
- Sample $\boldsymbol{s} \leftarrow \chi^{\dim_1 \times 1}$. Set $\mathsf{msk} = \boldsymbol{s}$. It also computes a perfectly binding commitment of the secret key $\mathsf{Z} = \mathsf{Com}(s)$.
- Sample $b \leftarrow \{0,1\}$ and compute $\mathsf{ct} \leftarrow \mathsf{PE.Enc}_1(\mathsf{pk}, \boldsymbol{s}, m_b)$.

- [**Change**] Generate $\widetilde{\mathsf{sk}}_{C_i}$ as follows. Run $\mathsf{Mach}_2(\mathsf{Z}, \mathsf{ct}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0), \mathsf{crs}) \to \{\widetilde{\mathsf{sk}}_{C_i}\}_{i \in [Q_{\mathsf{sk}}]}$.
- Give to the adversary $\{\mathsf{crs}, \mathsf{ct}, \widetilde{\mathsf{sk}}_{C_1}, ..., \widetilde{\mathsf{sk}}_{C_Q}\}$

Lemma 7.5. *For any adversary $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_2) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_3) = 1]| = 0$.*

Proof. The difference between the two hybrids is only the functionality of Mach_1 and Mach_2. Their functionality is identical except that the only inefficient step of breaking the commitment is done by Mach_1 in $\mathbf{Hybrid}_1$, where as it is done by $\mathsf{Mach}_{inner,1}$ as a soubroutine, by Mach_2 in $\mathbf{Hybrid}_2$. Thus the claim holds. □

In the next hybrid, we abort if the size of list $\mathbf{L}$ is more than $c^* = c + 1$. Consider the following machine.

```
$\mathsf{Mach}_3(\mathsf{Z}, \mathsf{ct}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0), \mathsf{crs})$
    1. Compute $\mathbf{L} \leftarrow \mathsf{Mach}_{inner,2}(\mathsf{Z}, \mathsf{crs}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0))$. If $\mathbf{L} = \perp$, output $\perp$, otherwise, for each $i \in [Q_{\mathsf{sk}}]$, if $(i, u_i) \in \mathbf{L}$, for some $u_i$, set $w_i = u_i$, else sample $w_i \leftarrow [\mathsf{Bound} + 1, \mathsf{Bound}_{smdg} - \mathsf{Bound} - 1]$.
    2. Compute $\widetilde{\mathsf{sk}}_{C_i} = \mathsf{PE.RoundEval}(\mathsf{PK}, C_i, \mathsf{ct}, p) - C_i(m_0)\lceil p/2 \rceil + w_i \mod p$.
    3. Output $\{\widetilde{\mathsf{sk}}_{C_i}\}_{i \in [Q_{\mathsf{sk}_{C_i}}]}$.
```

```
$\mathsf{Mach}_{inner,2}(\mathsf{Z}, \mathsf{crs}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0))$
    1. Compute $\boldsymbol{s}$ by opening up $\mathsf{Z}$ by brute-force.
    2. Compute $v_i = \mathsf{PE.RoundEval}(\mathsf{PK}, C_i, \mathsf{ct}, p)$ for all $i \in [Q_{\mathsf{sk}}]$. Let $e'_{C_i} = v_i - C_i(m_0)\lceil p/2 \rceil - \langle \mathbf{B}'_{C_i}, \boldsymbol{s} \rangle \mod p$.
    3. Maintain a list $\mathbf{L}$. Initialise it to be empty.
    4. For each $i \in [Q_{\mathsf{sk}_{C_i}}]$, sample $e_i \leftarrow [0, \mathsf{Bound}_{smdg}]$. Compute $u_i = e_i - e'_{C_i}$. If $u_i \notin [\mathsf{Bound} + 1, \mathsf{Bound}_{smdg} - \mathsf{Bound} - 1]$ append $(i, u_i)$ in the list $\mathbf{L}$. If $\mathbf{L}$ has more than $c^*$ tuples output $\perp$, otherwise output $\mathbf{L}$.
```

Hybrid$_4$:

- Adversary specifies $(m_0, m_1) \in \{0,1\}^n$ along with circuits $C_1, ..., C_{Q_{\mathsf{sk}}} \in \mathcal{C}_{n,\lambda}$ such that $C_i(m_0) = C_i(m_1)\ \forall i \in [Q]$.
- Challenger generates the $\mathsf{crs} = (p, p_1, \mathsf{PK}, \mathcal{PG})$ as in the algorithm.
- Sample $\boldsymbol{s} \leftarrow \chi^{\dim_1 \times 1}$. Set $\mathsf{msk} = \boldsymbol{s}$. It also computes a perfectly binding commitment of the secret key $\mathsf{Z} = \mathsf{Com}(s)$.
- Sample $b \leftarrow \{0,1\}$ and compute $\mathsf{ct} \leftarrow \mathsf{PE.Enc}_1(\mathsf{pk}, \boldsymbol{s}, m_b)$.
- [**Change**] Generate $\widetilde{\mathsf{sk}}_{C_i}$ as follows. Run $\mathsf{Mach}_3(\mathsf{Z}, \mathsf{ct}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0), \mathsf{crs})$. If the output is $\perp$, then abort, otherwise let the output be $\{\widetilde{\mathsf{sk}}_{C_i}\}_{i \in [Q_{\mathsf{sk}}]}$.

- Give to the adversary $\{\mathsf{crs}, \mathsf{ct}, \widetilde{\mathsf{sk}}_{C_1}, ..., \widetilde{\mathsf{sk}}_{C_Q}\}$

Lemma 7.6. *For any adversary $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_3) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_4) = 1]| < \lambda^{-c^*}$.*

Proof. The difference between two hybrids is that in $\mathbf{Hybrid}_4$, we abort if the list $\mathbf{L}$ has more than c^* elements. This probability is the same as the probability that out of $w_1, ..., w_{Q_{\mathsf{sk}}}$, more than c^* elements are sampled to be not in $[\mathsf{Bound}+1, \mathsf{Bound}_{smdg} - \mathsf{Bound} - 1]$. The probability of a single element not in $[\mathsf{Bound}+1, \mathsf{Bound}_{smdg} - \mathsf{Bound} - 1]$ is $1 - \frac{\mathsf{Bound}_{smdg} - 2\cdot\mathsf{Bound} - 1}{\mathsf{Bound}_{smdg}+1} = \frac{2\mathsf{Bound}+2}{\mathsf{Bound}_{smdg}+1}$. This probability is at most $Q_{\mathsf{sk}}^{c^*} \cdot (\frac{2\mathsf{Bound}+2}{\mathsf{Bound}_{smdg}+1})^{c^*}$ whenever c^* is an integer greater than 3. We used sterling approximation here: $\binom{n}{k} \leq (\frac{e\cdot n}{k})^k$ for any positive integers $n > k > 0$. Then, substitute $\mathsf{Bound}_{smdg} > 4 \cdot \lambda \cdot Q_{\mathsf{sk}} \cdot \mathsf{Bound}$. We obtain this probability:

$$\begin{aligned} Q_{\mathsf{sk}}^{c^*} \cdot (\frac{2\mathsf{Bound}+2}{\mathsf{Bound}_{smdg}+1})^{c^*} &\leq Q_{\mathsf{sk}}^{c^*} \cdot (\frac{4}{4 \cdot Q_{\mathsf{sk}}\lambda + \frac{1}{\mathsf{Bound}}})^{c^*} \\ &\leq \frac{1}{\lambda^{c^*}}. \end{aligned}$$

This conclude our proof.

□

In the next hybrid, we invoke the following theorem:

Theorem 7.6 (Imported Theorem [CCL18b])**.** *Let $k, t \in \mathbb{N}, \epsilon > 0$, and $\mathcal{C}_{leak}$ be a family of distinguisher circuits from $\{0,1\}^k \times \{0,1\}^t \to \{0,1\}$ of size $s(k)$. Then, for every distribution (X, Z) over $\{0,1\}^k \times \{0,1\}^t$, there exists a simulator $h : \{0,1\}^k \to \{0,1\}^t$ such that:*

1. *h has size bounded by $s' = O(s \cdot 2^t \epsilon^{-2})$*
2. *(X, Z) and $(X, h(Z))$ are indistinguishable by $\mathcal{C}_{leak}$. That is, for every $C \in \mathcal{C}_{leak}$,*

$$\left| \Pr_{(x,z)\leftarrow(X,Z)}[C(x,z) = 1] - \Pr_{x\leftarrow X,h}[C(x, h(x)) = 1] \right| \leq \epsilon$$

In the hybrid below, we will replace $\mathsf{Mach}_{inner,2}$ with an efficient circuit that is guaranteed to us by the lemma above. Note that in the previous hybrid, the output length of $\mathsf{Mach}_{inner,2}$ can be upper bounded by $\ell_h = c^* \cdot (\log_2 Q_{\mathsf{sk}} + \log_2(\mathsf{Bound}_{smdg} + \mathsf{Bound}) + 1) + 1$. Let $s_{\mathcal{A}}$ denote the size of the adversary $\mathcal{A}$. Set $\epsilon = \lambda^{-c-1}$. Thus, this means there exists a a circuit h of size $s_h = O((s_{\mathcal{A}} + \mathsf{poly}(\lambda, Q_{\mathsf{sk}})) \cdot (Q_{\mathsf{sk}} \cdot \mathsf{Bound}_{smdg} + \mathsf{Bound})^{c^*})$ that efficiently simulates $\mathsf{Mach}_{inner,2}$ and fools any adversary of size $s_{\mathcal{A}} + \mathsf{poly}(\lambda, Q_{\mathsf{sk}})$ for any fixed polynomial poly by advantage ϵ. Here is the new machine. We will set this polynomial poly below.

$\mathsf{Mach}_4(\mathsf{Z}, \mathsf{ct}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0), \mathsf{crs})$

1. Compute $\mathbf{L} \leftarrow h(\mathsf{Z}, \mathsf{crs}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0))$. If $\mathbf{L} = \perp$, output $\perp$, otherwise, for each $i \in [Q_{\mathsf{sk}}]$, if $(i, u_i) \in \mathbf{L}$, for some u_i, set $w_i = u_i$, else sample $w_i \leftarrow [\mathsf{Bound} + 1, \mathsf{Bound}_{smdg} - \mathsf{Bound} - 1]$.
2. Compute $\widetilde{\mathsf{sk}}_{C_i} = \mathsf{PE.RoundEval}(\mathsf{PK}, C_i, \mathsf{ct}, p) - C_i(m_0)\lceil p/2 \rceil + w_i \mod p$.
3. Output $\{\widetilde{\mathsf{sk}}_{C_i}\}_{i\in[Q_{\mathsf{sk}_{C_i}}]}$.

Hybrid$_5$:

- Adversary specifies $(m_0, m_1) \in \{0,1\}^n$ along with circuits $C_1, ..., C_{Q_{\mathsf{sk}}} \in \mathcal{C}_{n,\lambda}$ such that $C_i(m_0) = C_i(m_1)\ \forall i \in [Q]$.
- Challenger generates the $\mathsf{crs} = (p, p_1, \mathsf{PK}, \mathcal{PG})$ as in the algorithm.
- Sample $\boldsymbol{s} \leftarrow \chi^{\dim_1 \times 1}$. Set $\mathsf{msk} = \boldsymbol{s}$. It also computes a perfectly binding commitment of the secret key $\mathsf{Z} = \mathsf{Com}(s)$.
- Sample $b \leftarrow \{0,1\}$ and compute $\mathsf{ct} \leftarrow \mathsf{PE.Enc}_1(\mathsf{pk}, \boldsymbol{s}, m_b)$.
- [**Change**] Generate $\widetilde{\mathsf{sk}}_{C_i}$ as follows. Run $\mathsf{Mach}_4(\mathsf{Z}, \mathsf{ct}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0), \mathsf{crs})$. If the output is $\perp$, then abort, otherwise let the output be $\{\widetilde{\mathsf{sk}}_{C_i}\}_{i \in [Q_{\mathsf{sk}}]}$.
- Give to the adversary $\{\mathsf{crs}, \mathsf{ct}, \widetilde{\mathsf{sk}}_{C_1}, ..., \widetilde{\mathsf{sk}}_{C_Q}\}$

Lemma 7.7. *There exists an instantiation of the polynomial* poly *such that for any adversary* $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_4) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_5) = 1]| < \epsilon = \lambda^{-c-1}$.

Proof. To prove this, we choose $\epsilon = \lambda^{-c-1}$. Let $s_{\mathbf{Hybrid}}$ denote the size of the circuit used to run the process of the challenger in $\mathbf{Hybrid}_4$ except the $\mathsf{Mach}_{inner,2}$. We use the theorem above to construct an h, that fools an adversary of size $s_{\mathcal{A}} + s_{\mathbf{Hybrid}}$ with probability ϵ where $s_{\mathcal{A}}$ is the size of the adversary.

This can now be proven using a reduction to the leakage simulation lemma 7.6. The only difference between $\mathbf{Hybrid}_4$ and $\mathbf{Hybrid}_5$ is how $\mathbf{L}$ is generated. In $\mathbf{Hybrid}_4$, $\mathbf{L}$ is generated by running an inefficient machine $\mathsf{Mach}_{inner,2}$ on $\mathsf{Z}, \mathsf{crs}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0)$ where as in $\mathbf{Hybrid}_5$ it is generated by running h on the same input. We can build a reduction as follows. The challenger gets as input $\mathbf{L}$ which is either $\mathsf{Mach}_{inner,2}$ evaluated on $\mathsf{Z}, \mathsf{crs}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0)$ or h evaluated on the same input. Then $\mathbf{L}$ is used to simulate either $\mathbf{Hybrid}_5$ or $\mathbf{Hybrid}_4$ (depending on how $\mathbf{L}$ was computed). To do this, the reduction needs to run $\mathcal{A}$ and sample tuples as described in the hybrids. Note that the output length of both h and $\mathsf{Mach}_{inner,2}$ is bounded by ℓ. The hybrid can be simulated in time $s_{\mathbf{Hybrid}} = \mathsf{poly}(Q_{\mathsf{sk}}, \lambda)$. Finally if $\mathcal{A}$ guesses $\mathbf{Hybrid}_4$ then the reduction guesses that an inefficient machine was used to generate $\mathbf{L}$, otherwise it guesses that h was used to generate $\mathbf{L}$. The advantage of the reduction is the same as the advantage of $\mathcal{A}$ in distinguishing between hybrids. Finally, if h fools circuits of size $s_{\mathcal{A}} + s_{\mathbf{Hybrid}}$ with advantage ϵ, then the claim holds. $\square$

In the next hybrid, we replace Z with a commitment of 0 of appropriate length.

Hybrid$_6$:

- Adversary specifies $(m_0, m_1) \in \{0,1\}^n$ along with circuits $C_1, ..., C_{Q_{\mathsf{sk}}} \in \mathcal{C}_{n,\lambda}$ such that $C_i(m_0) = C_i(m_1)\ \forall i \in [Q]$.
- Challenger generates the $\mathsf{crs} = (p, p_1, \mathsf{PK}, \mathcal{PG})$ as in the algorithm.
- [**Change**] Sample $\boldsymbol{s} \leftarrow \chi^{\dim_1 \times 1}$. Set $\mathsf{msk} = \boldsymbol{s}$. It also computes a perfectly binding commitment of the secret key $\mathsf{Z} = \mathsf{Com}(0^{|\boldsymbol{s}|})$.
- Sample $b \leftarrow \{0,1\}$ and compute $\mathsf{ct} \leftarrow \mathsf{PE.Enc}_1(\mathsf{pk}, \boldsymbol{s}, m_b)$.

- Generate $\widetilde{\mathsf{sk}}_{C_i}$ as follows. Run $\mathsf{Mach}_4(\mathsf{Z}, \mathsf{ct}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0), \mathsf{crs})$. If the output is $\perp$, then abort, otherwise let the output be $\{\widetilde{\mathsf{sk}}_{C_i}\}_{i \in [Q_{\mathsf{sk}}]}$.
- Give to the adversary $\{\mathsf{crs}, \mathsf{ct}, \widetilde{\mathsf{sk}}_{C_1}, ..., \widetilde{\mathsf{sk}}_{C_Q}\}$

Lemma 7.8. *If* Com *is secure against adversaries of all polynomial sized circuits, then for any p.p.t. adversary* $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_5) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_6) = 1]| < \mathsf{negl}(\lambda)$ *for some negligible* negl.

Proof. The only difference between $\mathbf{Hybrid}_5$ and $\mathbf{Hybrid}_6$ is how Z is generated. In $\mathbf{Hybrid}_5$, it is generated as a commitment of $\boldsymbol{s}$, whereas, in $\mathbf{Hybrid}_6$ it is generated as a commitment of $0^{|\boldsymbol{s}|}$. One can build a reduction to the security of the commitment scheme as follows: the reduction either gets a commitment of $\boldsymbol{s}$ or a commitment of 0. The reduction interacts with the adversary as in $\mathbf{Hybrid}_6$ (or $\mathbf{Hybrid}_5$) using Z as this commitment. Note that the view of the adversary can be simulated by an algorithm of size $s_h + s_{\mathbf{Hybrid}}$ which is a polynomial in the security parameter. Finally if $\mathcal{A}$ guesses that it is in $\mathbf{Hybrid}_5$, the reduction guesses that Z as a commmitment of $\boldsymbol{s}$ otherwise it guesses it as a commitment of 0. Advantage of reduction is equal to the advantage of the adversary in distinguishing the commitment scheme. □

Finally, we replace ct to be an encryption of m_0.

$\mathbf{Hybrid}_7$:

- Adversary specifies $(m_0, m_1) \in \{0,1\}^n$ along with circuits $C_1, ..., C_{Q_{\mathsf{sk}}} \in \mathcal{C}_{n,\lambda}$ such that $C_i(m_0) = C_i(m_1)\ \forall i \in [Q]$.
- Challenger generates the $\mathsf{crs} = (p, p_1, \mathsf{PK}, \mathcal{PG})$ as in the algorithm.
- Sample $\boldsymbol{s} \leftarrow \chi^{\dim_1 \times 1}$. Set $\mathsf{msk} = \boldsymbol{s}$. It also computes a perfectly binding commitment of the secret key $\mathsf{Z} = \mathsf{Com}(0^{|\boldsymbol{s}|})$.
- [**Change**] Compute $\mathsf{ct} \leftarrow \mathsf{PE.Enc}_1(\mathsf{pk}, \boldsymbol{s}, m_0)$.
- Generate $\widetilde{\mathsf{sk}}_{C_i}$ as follows. Run $\mathsf{Mach}_4(\mathsf{Z}, \mathsf{ct}, C_1, ..., C_{Q_{\mathsf{sk}}}, C_1(m_0), ..., C_{Q_{\mathsf{sk}}}(m_0), \mathsf{crs})$. If the output is $\perp$, then abort, otherwise let the output be $\{\widetilde{\mathsf{sk}}_{C_i}\}_{i \in [Q_{\mathsf{sk}}]}$.
- Give to the adversary $\{\mathsf{crs}, \mathsf{ct}, \widetilde{\mathsf{sk}}_{C_1}, ..., \widetilde{\mathsf{sk}}_{C_Q}\}$

This hybrid is independent of b.

Lemma 7.9. *If* PE *is secure against adversaries of all polynomial sized circuits, then for any p.p.t. adversary* $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_6) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_7) = 1]| < \mathsf{negl}(\lambda)$ *for some negligible* negl.

Proof. The only difference between $\mathbf{Hybrid}_5$ and $\mathbf{Hybrid}_6$ is how ct is generated. In $\mathbf{Hybrid}_6$, it is generated as an encryption of m_b, whereas, in $\mathbf{Hybrid}_7$ it is generated as an encryption of m_0. One can build a reduction to the security of the PE scheme as follows: the reduction either gets an encryption of m_b or an encryption of m_0. The reduction interacts with the adversary as in $\mathbf{Hybrid}_6$ (or $\mathbf{Hybrid}_7$) using ct as this encryption. Note that the view of the adversary can be simulated by an algorithm of size $s_h + s_{\mathbf{Hybrid}}$ which is a polynomial in the security parameter. Finally if $\mathcal{A}$ guesses that it is in $\mathbf{Hybrid}_6$, the reduction guesses that ct as an encryption of m_b otherwise it guesses it as an encryption of m_0. Advantage of reduction is equal to the advantage of the adversary in the PE security game. □

Finishing up the Proof. Note that both the commitment scheme as well as the PE scheme can be instantiated from LWE. Combining all these lemmata above, the advantage of $\mathcal{A}$ in the security game is bounded by $\mathsf{negl}(\lambda) + \frac{4}{\lambda^{c+1}} < \lambda^{-c}$.

Remark 7.4 (On Subexponential Security)**.** Above, we prove polynomial security but we could have proved subexponential security by allowing the set of indices in $\mathbf{L}$ to be bounded by $\lambda^{O(1)}$ as opposed to a constant. This will mean that the size of h will also be subexponentially large. The argument can be made to go through relying on LWE secure against subexponential sized adversaries.

Thus,

Theorem 7.7. *Assuming subexponential time hardness of LWE with parameters in Section 7.2, the construction of* $\mathsf{1LGFEB}$ *above is subexponentially secure.*

8 Our $(\mathsf{NC}_1, \deg 2)$-PHFE from Pairings

In Fig.8.2 we present a Partially-Hiding FE (PHFE) for the functionality $\mathcal{F}^{\mathsf{phfe}}_{\mathcal{PG},n,\ell,w}$, parameterized by a pairing group $\mathcal{PG} = (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, p, P_1, P_2, e) \leftarrow \mathsf{PGGen}(1^\lambda)$ and integers $n, \ell, w = \mathsf{poly}(\lambda)$. Each function of $\mathcal{F}^{\mathsf{phfe}}_{\mathcal{PG},n,\ell,w}$ is represented by a tuple $(f^0, \ldots, f^{\ell+1})$ such that for all inputs $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in (\mathbb{Z}^n)^3$, it outputs $\left[f^0 \prod_{i=1}^{\ell} f^i(\boldsymbol{x}) f^{\ell+1}(\boldsymbol{y} \otimes \boldsymbol{z})\right]_T \in \mathbf{G}_T$, where $f^0 \in \mathbb{Z}_p^{1\times w}$, $\{f^i(\boldsymbol{x}) \in \mathbb{Z}_p^{w\times w}\}_{i\in[d]}$, $f^{\ell+1}(\boldsymbol{y} \otimes \boldsymbol{z}) \in \mathbb{Z}_p^w$, and all functions f^i for $i > 0$ are linear. That is, for all $i \in [d]$, $f^i : \mathbb{Z}^n \to \mathbb{Z}_p^{w\times w}$ is such that for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{Z}^n$, $f^i(\boldsymbol{x} + \boldsymbol{x}') = f^i(\boldsymbol{x}) + f^i(\boldsymbol{x}')$. Similarly, $f^{\ell+1} : \mathbb{Z}^{n^2} \to \mathbb{Z}_p^{w\times w}$ is such that for all $\boldsymbol{u}, \boldsymbol{u}' \in \mathbb{Z}^{n^2}$, $f^{\ell+1}(\boldsymbol{u} + \boldsymbol{u}') = f^{\ell+1}(\boldsymbol{u}) + f^{\ell+1}(\boldsymbol{u}')$. The computation is performed in the exponent of a generator of the cyclic group $\mathbf{G}_T$, of order p. This model of computation captures functions f of the form: $f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = w(g(\boldsymbol{x}), \boldsymbol{y} \otimes \boldsymbol{z})$, where w is a multilinear degree two polynomial (with degree one in $\boldsymbol{y} \otimes \boldsymbol{z}$) and g is a matrix branching program of width w and length ℓ over $\mathbb{Z}_p$. By Barrington's theorem, for sufficiently large $\ell, w, \log(p) = \mathsf{poly}(\lambda)$, it also contains the case when g is a Boolean NC_1 circuit ($\boldsymbol{x}$ being restricted to be a binary vector in this case). Note that to realize Boolean NC_1 circuits, we need each function f^i to be affine, which can be ensured by setting, say, $x_1 = 1$.

We give a modular construction of PHFE for the functionality $\mathcal{F}^{\mathsf{phfe}}_{\mathcal{PG},n,\ell,w}$ in Section 8.2 that builds upon inner-product FE, defined in Section 8.1. Our construction is linearly efficient as per Definition 4.6. That is, the ciphertext size is $|\mathsf{ct}| = n \cdot \mathsf{poly}(\lambda)$ for a fixed polynomial, where λ denotes the security parameter and n is dimension of the vectors being encrypted. As such, our PHFE can be used to build general purpose FE in Section 6. Finally, we build the concrete inner-product FE scheme that underlies our PHFE in Section 8.3. The security of all of our constructions rely on standard assumptions in pairing groups.

8.1 Ingredients: Inner-Product FE

For any dimension $\mathsf{dim} \in \mathbb{N}$ and pairing group $\mathcal{PG} = (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, p, P_1, P_2, e) \leftarrow \mathsf{PGGen}(1^\lambda)$ we define the functionality $\mathcal{F}^{\mathsf{ipfe}}_{\mathcal{PG},\mathsf{dim}} : \mathbf{G}_1^{\mathsf{dim}} \to \mathbf{G}_T$, where every function is described by a vector $[\boldsymbol{y}]_2 \in \mathbf{G}_2^{\mathsf{dim}}$, and on input $[\boldsymbol{x}]_1 \in \mathbf{G}_1^{\mathsf{dim}}$, outputs $[\boldsymbol{x}^\top \boldsymbol{y}]_T \in \mathbf{G}_T$. We define the functionality $\mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},\mathsf{dim}} : \mathbb{Z}^{\mathsf{dim}} \to \mathbf{G}_T$ similarly except the inputs $\boldsymbol{x}$ are in $\mathbb{Z}^{\mathsf{dim}}$ instead of $\mathbf{G}_1^{\mathsf{dim}}$. To build an FE for $\mathcal{F}^{\mathsf{phfe}}_{\mathcal{PG},n,\ell,w}$, we rely on a private-key IND-function-hiding FE $\widehat{\mathsf{IPFE}}$ for the functionality $\mathcal{F}^{\mathsf{ipfe}}_{\mathcal{PG},3}$ and an FE $\overline{\mathsf{IPFE}}$ for the functionality $\mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},n+1}$. We only require that the scheme $\overline{\mathsf{IPFE}}$ satisfies a simulation security that is slightly weaker than defined Definition 4.3, in the sense that the simulator generates the functional secret keys for a function $[\boldsymbol{y}]_2$ only knowing the output $[\boldsymbol{x}^\top \boldsymbol{y}]_2$ in $\mathbf{G}_2$ or $[\boldsymbol{x}^\top \boldsymbol{y}]_1$ in $\mathbf{G}_1$, as opposed to $\mathbf{G}_T$, where $\boldsymbol{x}$ denotes the challenge (see Definition 8.1).

Definition 8.1 (Weak simulation security). *wet* FE *be an FE scheme for the functionality* $\mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},\mathsf{dim}}$ *defined above, with dimension* $\mathsf{dim} \in \mathbb{N}$ *and pairing group* $\mathcal{PG} \leftarrow \mathsf{PGGen}(1^\lambda)$*. We say that* FE *is weakly simulation secure if for any PPT adversary* $\mathcal{A}$*, there exists a PPT simulator* $\mathcal{S} := (\widetilde{\mathsf{Setup}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{KeyGen}}_1, \widetilde{\mathsf{KeyGen}}_2)$ *such that:*

- *for all* $\boldsymbol{y} \in \mathbb{Z}^{\mathsf{dim}}$, $v \in \mathbb{Z}_p$, *the following are identically distributed:*

$$\widetilde{\mathsf{KeyGen}}_1(\widetilde{\mathsf{msk}}, [\boldsymbol{y}]_1, [v]_1) \quad \textit{and} \quad \widetilde{\mathsf{KeyGen}}_2(\widetilde{\mathsf{msk}}, [\boldsymbol{y}]_2, [v]_2),$$

  where $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{msk}}) \leftarrow \widetilde{\mathsf{Setup}}(1^\lambda, \mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},\mathsf{dim}})$.

- *For any security parameter* λ*, we have:*

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

  where the experiments are defined below.

```
$\underline{\mathsf{Real}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda)}$:
$[\boldsymbol{x}]_1 \leftarrow \mathcal{A}(1^\lambda)$
$(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}^{\mathsf{ipfe}}_{\mathcal{PG},\mathsf{dim}})$
$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, [\boldsymbol{x}]_1)$
$\alpha \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}(\cdot)}(\mathsf{ct}, \mathsf{pk})$
```

```
$\underline{\mathsf{Ideal}^{\mathsf{FE}}_{\mathcal{A},\mathcal{S}}(1^\lambda)}$:
$[\boldsymbol{x}]_1 \leftarrow \mathcal{A}(1^\lambda)$
$(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{msk}}) \leftarrow \widetilde{\mathsf{Setup}}(1^\lambda, \mathcal{F})$
$\mathsf{ct} \leftarrow \widetilde{\mathsf{Enc}}(\widetilde{\mathsf{msk}})$
$\alpha \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}(\cdot)}(\mathsf{ct}, \widetilde{\mathsf{pk}})$
```

In the real experiment, the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$*, when given as input* $[\boldsymbol{y}]_2 \in \mathbf{G}_2^{\mathsf{dim}}$*, returns* $\mathsf{KeyGen}(\mathsf{msk}, [\boldsymbol{y}]_2)$*. In the ideal experiment, the key generation oracle* $\mathcal{O}_{\mathsf{KeyGen}}$*, when given as input* $[\boldsymbol{y}]_2 \in \mathbf{G}_2^{\mathsf{dim}}$*, computes* $[\boldsymbol{x}^\top \boldsymbol{y}]_2$*, and returns* $\widetilde{\mathsf{KeyGen}}_2(\widetilde{\mathsf{msk}}, [\boldsymbol{y}]_2, [\boldsymbol{x}^\top \boldsymbol{y}]_2)$*. Note that this differs from Definition 4.3, where the algorithm* $\widetilde{\mathsf{KeyGen}}$ *gets as input* $[\boldsymbol{x}^\top \boldsymbol{y}]_T \in \mathbf{G}_T$*, not in* $\mathbf{G}_2$*.*

8.2 Modular Construction of the Partially-Hiding FE

In Fig.8.2 we present a modular construction of PHFE for the functionality $\mathcal{F}^{\mathsf{phfe}}_{\mathcal{PG},n,\ell,w}$, which relies on an IND-function-hiding FE for the functionality $\mathcal{F}^{\mathsf{ipfe}}_{\mathcal{PG},3}$ and weakly simulation-secure FE for the functionality $\mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},n+1}$. The simulation security of our PHFE relies on the security of the underlying building blocks and the SXDH assumption in $\mathcal{PG}$.

Linear efficiency:

By linear efficiency of $\widehat{\mathsf{IPFE}}$ for all $i, j \in [n]$, we have $|\mathsf{ct}_i|, |\mathsf{ct}'_j| = \mathsf{poly}(\lambda)$. By linear efficiency of $\overline{\mathsf{IPFE}}$, we have $|\overline{\mathsf{ct}}| = n \cdot \mathsf{poly}(\lambda)$. Overall, we have $|\mathsf{ct}| = n \cdot \mathsf{poly}(\lambda)$.

Correctness:

By correctness of $\widehat{\mathsf{IPFE}}$, for all $i, j \in [n]$, we have:

$$[\theta_{i,j}]_T = [y_i z_j + rs\boldsymbol{a}_i^\top \boldsymbol{b}_j]_T \quad \text{and} \quad [\theta]_T = f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) + rsf(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}),$$

$\underline{\mathsf{Setup}(1^\lambda, \mathcal{F}^{\mathsf{phfe}}_{\mathcal{PG},n,\ell,w}):}$

Given $\mathcal{PG} = (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, p, P_1, P_2, e) \leftarrow_{\mathrm{R}} \mathsf{GGen}(1^\lambda)$, it computes $(\overline{\mathsf{pk}}, \overline{\mathsf{msk}}) \leftarrow \overline{\mathsf{Setup}}(1^\lambda, \mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},n+1})$. For all $i, j \in [n]$: $\boldsymbol{a}_i, \boldsymbol{b}_j \leftarrow_{\mathrm{R}} \mathsf{DDH}$, for all $k \in [\ell]$, $\boldsymbol{u}_k \leftarrow_{\mathrm{R}} \mathbb{Z}_p^n$. Return $\mathsf{pk} := \big(\overline{\mathsf{pk}}, \{[\boldsymbol{a}_i]_1, [\boldsymbol{b}_j]_2\}_{i,j\in[n]}\big)$ and $\mathsf{msk} := \big(\overline{\mathsf{msk}}, \{\boldsymbol{a}_i, \boldsymbol{b}_j, \boldsymbol{u}_k\}_{i,j\in[n],k\in[\ell]}\big)$.

$\underline{\mathsf{Enc}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in (\mathbb{Z}_p)^3):}$

$r, s \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $(\widehat{\mathsf{pk}}, \widehat{\mathsf{msk}}) \leftarrow \widehat{\mathsf{Setup}}(1^\lambda, \mathcal{F}^{\mathsf{ipfe}}_{\mathcal{PG},3})$, $\overline{\mathsf{ct}} \leftarrow \overline{\mathsf{Enc}}\left(\overline{\mathsf{pk}}, \begin{pmatrix} rs\boldsymbol{x} \\ rs \end{pmatrix}\right)$. For all $i, j \in [n]$: $\mathsf{ct}_i \leftarrow \widehat{\mathsf{Enc}}\left(\widehat{\mathsf{msk}}, \begin{bmatrix} y_i \\ \boldsymbol{a}_i r \end{bmatrix}_1\right)$, $\mathsf{ct}'_j \leftarrow \widehat{\mathsf{KeyGen}}\left(\widehat{\mathsf{msk}}, \begin{bmatrix} z_j \\ \boldsymbol{b}_j s \end{bmatrix}_2\right)$. Return $\left(\overline{\mathsf{ct}}, \{\mathsf{ct}_i, \mathsf{ct}'_j\}_{i,j\in[n]}\right)$.

$\underline{\mathsf{KeyGen}\big(\mathsf{msk}, (f^0, \ldots, f^{\ell+1})\big):}$

For all $t \in [\ell]$, we write $[\mathbf{M}_t]_2 \in \mathbf{G}_2^{(n+1)\times w}$, the linear function such that for all $\begin{bmatrix} \boldsymbol{v} \\ \alpha \end{bmatrix}_1 \in \mathbf{G}_1^{n+1}$, $\left[\mathbf{M}_t^\top \begin{pmatrix} \boldsymbol{v} \\ \alpha \end{pmatrix}\right]_T = \left[\big(\alpha \cdot f^t(\boldsymbol{u}_t) - f^t(\boldsymbol{v})\big) \prod_{t<i\leq\ell} f^i(\boldsymbol{u}_i) f^{\ell+1}(\boldsymbol{a} \otimes \boldsymbol{b})\right]_T \in \mathbf{G}_T^w$, and $[\boldsymbol{m}_{\ell+1}]_2 \in \mathbf{G}_2^{n+1}$ the linear function such that for all $\begin{bmatrix} \boldsymbol{v} \\ \alpha \end{bmatrix}_1 \in \mathbf{G}_1^{n+1}$, $\left[\boldsymbol{m}_{\ell+1}^\top \begin{pmatrix} \boldsymbol{v} \\ \alpha \end{pmatrix}\right]_T = \left[\alpha \cdot f^0 \prod_{i\in[\ell]} f^i(\boldsymbol{u}_i) f^{\ell+1}(\boldsymbol{a} \otimes \boldsymbol{b})\right]_T \in \mathbf{G}_T$, where $\boldsymbol{a} \otimes \boldsymbol{b} = (\boldsymbol{a}_i^\top \boldsymbol{b}_j)_{i,j\in[n]} \in \mathbb{Z}^{n^2}$. For all $t \in [\ell]$, $\mathsf{sk}_t \leftarrow \overline{\mathsf{KeyGen}}(\overline{\mathsf{msk}}, [\mathbf{M}_t]_2)$, and $\mathsf{sk}_{\ell+1} \leftarrow \overline{\mathsf{KeyGen}}(\overline{\mathsf{msk}}, [\boldsymbol{m}_{\ell+1}]_2)$. Return $\{\mathsf{sk}_t\}_{t\in[\ell+1]}$.

$\underline{\mathsf{Dec}(\mathsf{ct}, \mathsf{sk}):}$

Parse $\mathsf{ct} = \left(\overline{\mathsf{ct}}, \{\mathsf{ct}_i, \mathsf{ct}'_j\}_{i,j\in[n]}\right)$ and $\mathsf{sk} = \{\mathsf{sk}_t\}_{t\in[\ell+1]}$. For all $i, j \in [n]$: $[\theta_{i,j}]_T \leftarrow \widehat{\mathsf{Dec}}(\mathsf{ct}_i, \mathsf{ct}'_j) \in \mathbf{G}_T$. $[\theta]_T = \left[f^0 \prod_{i\in[\ell]} f^i(\boldsymbol{x}) f^{\ell+1}(\theta_{i,j})_{i,j\in[n]}\right]_T \in \mathbf{G}_T$. For all $t \in [\ell]$, $[\boldsymbol{w}_t]_T \leftarrow \overline{\mathsf{Dec}}(\overline{\mathsf{ct}}, \mathsf{sk}_t) \in \mathbf{G}_T^w$, $[\theta_{\ell+1}]_T \leftarrow \overline{\mathsf{Dec}}(\overline{\mathsf{ct}}, \mathsf{sk}_{\ell+1}) \in \mathbf{G}_T$. Return $[\theta]_T + \left[\sum_{t\in[\ell]} f^0\big(\prod_{0<m<t} f^m(\boldsymbol{x})\big)\boldsymbol{w}_t\right]_T - [\theta_{\ell+1}]_T$.

Figure 9: This is PHFE, a simulation-secure FE scheme for the functionality $\mathcal{F}^{\mathsf{phfe}}_{\mathcal{PG},d,n,w}$. Here, $\widehat{\mathsf{IPFE}} := (\widehat{\mathsf{Setup}}, \widehat{\mathsf{Enc}}, \widehat{\mathsf{KeyGen}}, \widehat{\mathsf{Dec}})$ is an IND-function-hiding FE for the functionality $\mathcal{F}^{\mathsf{ipfe}}_{\mathcal{PG},3}$, and $\overline{\mathsf{IPFE}} := (\overline{\mathsf{Setup}}, \overline{\mathsf{Enc}}, \overline{\mathsf{KeyGen}}, \overline{\mathsf{Dec}})$ is a weakly simulation-secure FE for the functionality $\mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},n+1}$.

where $f(\boldsymbol{x},\boldsymbol{a},\boldsymbol{b}) = f^0 \prod_{i\in[\ell]} f^i(\boldsymbol{x}) f^{\ell+1}(\boldsymbol{a}\otimes\boldsymbol{b})$, with $\boldsymbol{a}\otimes\boldsymbol{b} = (\boldsymbol{a}_i^\top \boldsymbol{b}_j)_{i,j\in[n]} \in \mathbb{Z}^{n^2}$.

By correctness of $\overline{\mathsf{IPFE}}$, for all $t\in[\ell]$, we have:

$$[\boldsymbol{w}_t]_T = \left[rs\big(f^t(\boldsymbol{u}_t) - f^t(\boldsymbol{x})\big) \prod_{t<i\leq\ell} f^i(\boldsymbol{u}_i) f^{\ell+1}(\boldsymbol{a}\otimes\boldsymbol{b}) \right]_T .$$

Besides, we have:

$$[\theta_{\ell+1}]_T = \left[rsf^0 \prod_{i\in[\ell]} f^i(\boldsymbol{u}_i) f^{\ell+1}(\boldsymbol{a}\otimes\boldsymbol{b}) \right]_T .$$

Thus, the telescoping sum is of the form:

$$\left[\sum_{t\in[\ell]} f^0 \prod_{0<i<t} f^i(\boldsymbol{x}) \boldsymbol{w}_t \right]_T = [\theta_{\ell+1} - rsf(\boldsymbol{x},\boldsymbol{a},\boldsymbol{b})]_T .$$

Consequently, we have:

$$[\theta]_T + \left[\sum_{t\in[\ell]} f^0 \prod_{0<i<t} f^i(\boldsymbol{x}) \boldsymbol{w}_t \right]_T - [\theta_{\ell+1}]_T = [f(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})]_T .$$

Theorem 8.1 (Simulation security)**.** *The scheme presented in Fig.8.2 is simulation secure (as defined in Definition 4.3), provided the underlying $\widehat{\mathsf{IPFE}}$ is indistinguishability function-hiding secure (as defined in Definition 4.5), and $\overline{\mathsf{IPFE}}$ is simulation secure as per Definition 8.1, which is implied by the notion given in Definition 4.3. Namely, for any PPT adversary $\mathcal{A}$, there exist PPT adversaries $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$ and $\mathcal{B}_4$ such that:*

$$\mathsf{adv}^{\mathsf{SIM}}_{\mathsf{PHFE},\mathcal{A}}(\lambda) \leq \mathsf{adv}^{\mathsf{weak\text{-}SIM}}_{\overline{\mathsf{IPFE}},\mathcal{B}_1}(\lambda) + (\ell+1)\cdot \mathsf{adv}^{\mathsf{DDH}}_{\mathbf{G}_2,\mathcal{B}_2}(\lambda) + 3\cdot \mathsf{adv}^{\mathsf{DDH}}_{\mathbf{G}_1,\mathcal{B}_3}(\lambda) + \mathsf{adv}^{\mathsf{IND\text{-}FH}}_{\widehat{\mathsf{IPFE}},\mathcal{B}_4}(\lambda) + \frac{2}{p}.$$

Proof. The proof proceeds using a series of hybrid games, described below. Let $\mathcal{A}$ be a PPT adversary against the simulation security of the scheme. For any game $\mathbf{Hybrid}_i$, we denote by $\mathsf{adv}_i := \Pr[1 \leftarrow \mathbf{Hybrid}_i(\mathcal{A})]$ the probability that $\mathbf{Hybrid}_i$ returns 1 when interacting with $\mathcal{A}$.

- $\underline{\mathbf{Hybrid}_0}$: is the real experiment as given in Definition 4.3.

- $\underline{\mathbf{Hybrid}_1}$: is the same as $\mathbf{Hybrid}_0$, except we replace the scheme $(\overline{\mathsf{Setup}},\overline{\mathsf{Enc}},\overline{\mathsf{KeyGen}})$ by its simulator $(\widetilde{\mathsf{Setup}},\widetilde{\mathsf{Enc}},\widetilde{\mathsf{KeyGen}_2})$. That is, we sample $(\widetilde{\mathsf{pk}},\widetilde{\mathsf{msk}}) \leftarrow \widetilde{\mathsf{Setup}}(1^\lambda,\mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},n+1})$, instead of $(\overline{\mathsf{pk}},\overline{\mathsf{msk}}) \leftarrow \overline{\mathsf{Setup}}(1^\lambda,\mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},n+1})$.

The challenge ciphertext is generated using $\overline{\mathsf{ct}} \leftarrow \widetilde{\mathsf{Enc}}\left(\widetilde{\mathsf{msk}}\right)$ instead of $\overline{\mathsf{ct}} \leftarrow \overline{\mathsf{Enc}}\left(\overline{\mathsf{pk}}, \begin{pmatrix} rs\boldsymbol{x} \\ rs \end{pmatrix}\right)$.

The functional secret keys are generated using, for all $t\in[\ell]$:

$$\mathsf{sk}_t \leftarrow \widetilde{\mathsf{KeyGen}_2}\left(\widetilde{\mathsf{msk}}, [\mathbf{M}_t]_2\,, \left[rs\big(f^t(\boldsymbol{u}_t) - f^t(\boldsymbol{x})\big) \prod_{t<i\leq\ell} f^i(\boldsymbol{u}_i) f^{\ell+1}(\boldsymbol{a}\otimes\boldsymbol{b}), \right]_2 \right)$$

and

$$\mathsf{sk}_{\ell+1} \leftarrow \widetilde{\mathsf{KeyGen}}_2\left(\widetilde{\mathsf{msk}}, [\boldsymbol{m}_{\ell+1}]_2, \left[rsf^0 \prod_{i\in[\ell]} f^i(\boldsymbol{u}_i) f^{\ell+1}(\boldsymbol{a}\otimes\boldsymbol{b})\right]_2\right),$$

where $\boldsymbol{a}\otimes\boldsymbol{b} = (\boldsymbol{a}_i^\top \boldsymbol{b}_j)_{i,j\in[n]} \in \mathbb{Z}^{n^2}$.

This transition is justified by the simulation security of $\overline{\mathsf{IPFE}}$. Namely, there is a PPT adversary $\mathcal{B}_0$ such that:

$$|\mathsf{adv}_0 - \mathsf{adv}_1| \leq \mathsf{adv}^{\mathsf{weak\text{-}SIM}}_{\overline{\mathsf{IPFE}},\mathcal{B}_0}(\lambda).$$

• $\underline{\textbf{Hybrid}_2}$: is the same as $\textbf{Hybrid}_1$, except we replace the vectors $\{\boldsymbol{u}_k\}_{k\in[\ell]}$ by $\{\boldsymbol{u}_k+\boldsymbol{x}\}_{k\in[\ell]}$. These values are identically distributed, since the vectors $\boldsymbol{u}_k$ are sampled uniformly over $\mathbb{Z}_p^n$, independently of the challenge $\boldsymbol{x}$, which is chosen beforehand. Consequently, the functional secret keys are now generated using, for all $t \in [\ell]$:

$$\mathsf{sk}_t \leftarrow \widetilde{\mathsf{KeyGen}}_2\left(\widetilde{\mathsf{msk}}, [\mathbf{M}_t]_2, \left[rsf^t(\boldsymbol{u}_t) \prod_{t<i\leq\ell} f^i(\boldsymbol{u}_i+\boldsymbol{x}) f^{\ell+1}(\boldsymbol{a}\otimes\boldsymbol{b}),\right]_2\right)$$

and

$$\mathsf{sk}_{\ell+1} \leftarrow \widetilde{\mathsf{KeyGen}}_2\left(\widetilde{\mathsf{msk}}, [\boldsymbol{m}_{\ell+1}]_2, \left[rsf^0 \prod_{i\in[\ell]} f^i(\boldsymbol{u}_i+\boldsymbol{x}) f^{\ell+1}(\boldsymbol{a}\otimes\boldsymbol{b})\right]_2\right).$$

Here, we use the fact that the functions f^i for all $i > 0$ are linear. We have:

$$\mathsf{adv}_1 = \mathsf{adv}_2.$$

• $\underline{\textbf{Hybrid}_3}$: is the same as $\textbf{Hybrid}_2$, except we replace the vectors $[s\boldsymbol{u}_k]_2$ by fresh $[\boldsymbol{s}_k]_2 \leftarrow_{\mathrm{R}} \mathbf{G}_2^n$ for all $k \in [\ell]$, using the DDH assumption in $\mathbf{G}_2$. Consequently, the functional secret keys are now generated using, for all $t \in [\ell]$:

$$\mathsf{sk}_t \leftarrow \widetilde{\mathsf{KeyGen}}_2\left(\widetilde{\mathsf{msk}}, [\mathbf{M}_t]_2, \left[rf^t(\boldsymbol{s}_t) \prod_{t<i\leq\ell} f^i(\boldsymbol{u}_i+\boldsymbol{x}) f^{\ell+1}(\boldsymbol{a}\otimes\boldsymbol{b}),\right]_2\right)$$

and

$$\mathsf{sk}_{\ell+1} \leftarrow \widetilde{\mathsf{KeyGen}}_2\left(\widetilde{\mathsf{msk}}, [\boldsymbol{m}_{\ell+1}]_2, [v]_2\right),$$

where

$$[v]_2 = \left[rsf(\boldsymbol{x},\boldsymbol{a},\boldsymbol{b}) + r\sum_{i\in[\ell]} \big(\prod_{j<i} f^j(\boldsymbol{x})\big) f^i(\boldsymbol{s}_i) \big(\prod_{j>i} f^j(\boldsymbol{u}_j+\boldsymbol{x})\big) f^{\ell+1}(\boldsymbol{a}\otimes\boldsymbol{b})\right]_2.$$

We proceed via a hybrid argument, switching the vector $[s\boldsymbol{u}_k]_2$ to uniformly random $[\boldsymbol{s}_k]_2 \leftarrow_{\mathrm{R}} \mathbb{Z}_p^n$ one index $k \in [\ell]$ at a time. That is, we define $\textbf{Hybrid}_{2.\rho}$ for all $\rho \in [0,d]$ as $\textbf{Hybrid}_2$, except the first ρ-th functional keys are computed as in $\textbf{Hybrid}_3$. For all $\rho \in [\ell]$, we show there exists a PPT adversary $\mathcal{B}_{2.\rho}$ such that $|\mathsf{adv}_{2.\rho-1} - \mathsf{adv}_{2.\rho}| \leq \mathsf{adv}^{\mathsf{DDH}}_{\mathbf{G}_2,\mathcal{B}_{2.\rho}}(\lambda)$.

The adversary $\mathcal{B}_{2.\rho}$ takes as input a tuple $([s]_2, [\boldsymbol{u}_\rho]_2, [\boldsymbol{s}_\rho]_2)$ where the value $[\boldsymbol{s}_\rho]_2$ is either of the form $[s\boldsymbol{u}_\rho]_2$ (case 1), or uniformly random over $\mathbf{G}_2^n$ (case 2). The adversary $\mathcal{B}_{2.\rho}$ samples $r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $\boldsymbol{a}_i, \boldsymbol{b}_j \leftarrow_{\mathrm{R}} \mathsf{DDH}$ for all $i,j \in [n]$, $\boldsymbol{u}_m \leftarrow_{\mathrm{R}} \mathbb{Z}_p^n$ for all $m \neq \rho$, $\boldsymbol{s}_t \leftarrow_{\mathrm{R}} \mathbb{Z}_p^n$ for all $t < \rho$,

$(\widetilde{\mathsf{msk}}, \widetilde{\mathsf{pk}}) \leftarrow \widetilde{\mathsf{Setup}}(1^\lambda, \mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},n+1})$, upon which it can simulate the view of the adversary $\mathcal{A}$. In case 1, $\mathcal{B}_{2.\rho}$ simulates $\mathbf{Hybrid}_{2.\rho-1}$ to $\mathcal{A}$, whereas it simulates $\mathbf{Hybrid}_{2.\rho}$ in case 2.

Putting everything together, we have the existence of a PPT adversary $\mathcal{B}_2$ such that:

$$|\mathsf{adv}_2 - \mathsf{adv}_3| \leq \ell \cdot \mathsf{adv}^{\mathsf{DDH}}_{\mathbf{G}_2,\mathcal{B}_2}(\lambda).$$

• **Hybrid$_4$**: is the same as **Hybrid$_3$**, except that we replace the values $[\boldsymbol{b}_j s]_2$ used for generating functional secret keys by fresh $[\boldsymbol{w}_j]_2 \leftarrow_{\mathrm{R}} \mathbf{G}_2^2$ for all $j \in [n]$, using the DDH assumption in $\mathbf{G}_2$.

Consequently, the challenge ciphertext now contains:

$$\mathsf{ct}'_j \leftarrow \widehat{\mathsf{KeyGen}}\left(\widehat{\mathsf{msk}}, \begin{bmatrix} z_j \\ -\boldsymbol{w}_j \end{bmatrix}_2\right).$$

Moreover, the functional secret keys are now generated using:

$$\mathsf{sk}_{\ell+1} \leftarrow \widetilde{\mathsf{KeyGen}}_2\left(\widetilde{\mathsf{msk}}, [\boldsymbol{m}_{\ell+1}]_2, [v]_2\right),$$

where

$$[v]_2 = \left[rf(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{w}) + r \sum_{i \in [\ell]} \Big(\prod_{j<i} f^j(\boldsymbol{x})\Big) f^i(\boldsymbol{s}_i) \big(\prod_{j>i} f^j(\boldsymbol{u}_j + \boldsymbol{x})\big) f^{\ell+1}(\boldsymbol{a} \otimes \boldsymbol{b}) \right]_2.$$

We show there exists a PPT adversary $\mathcal{B}_3$ such that:

$$|\mathsf{adv}_3 - \mathsf{adv}_4| \leq \mathsf{adv}^{\mathsf{DDH}}_{\mathbf{G}_2,\mathcal{B}_3}(\lambda).$$

The adversary $\mathcal{B}_1$ takes as input a tuple $([s]_2, \{[\boldsymbol{b}_j]_2, [\boldsymbol{w}_j]_2\}_{j \in [n]})$ where the values $[\boldsymbol{w}_j]_2$ are either of the form $[\boldsymbol{b}_j s]_2$ (case 1), or uniformly random over $\mathbf{G}_2^2$ (case 2). The adversary $\mathcal{B}_3$ samples $r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $(\widetilde{\mathsf{msk}}, \widetilde{\mathsf{pk}}) \leftarrow \widetilde{\mathsf{Setup}}(1^\lambda, \mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},n+1})$, $\boldsymbol{a}_i \leftarrow_{\mathrm{R}} \mathsf{DDH}$ for all $i \in [n]$, $\boldsymbol{u}_k, \boldsymbol{s}_k \leftarrow_{\mathrm{R}} \mathbb{Z}_p^n$ for all $k \in [\ell]$, upon which it can simulate the view of the adversary $\mathcal{A}$ straightforwardly. In case 1, it simulates **Hybrid$_3$** to $\mathcal{A}$, whereas it simulates **Hybrid$_4$** in case 2.

• **Hybrid$_5$**: is the same as **Hybrid$_4$**, except we use the key generation algorithm $\widetilde{\mathsf{KeyGen}}_1$, which takes inputs from $\mathbf{G}_1$ instead of $\widetilde{\mathsf{KeyGen}}_2$, which takes inputs from $\mathbf{G}_2$. Namely, the secret keys are now generated using, for all $t \in [\ell]$:

$$\mathsf{sk}_t \leftarrow \widetilde{\mathsf{KeyGen}}_1\left(\widetilde{\mathsf{msk}}, [\mathbf{M}_t]_1, \left[rf^t(\boldsymbol{s}_t) \prod_{t<i\leq\ell} f^i(\boldsymbol{u}_i + \boldsymbol{x}) f^{\ell+1}(\boldsymbol{a} \otimes \boldsymbol{b}), \right]_1\right)$$

and

$$\mathsf{sk}_{\ell+1} \leftarrow \widetilde{\mathsf{KeyGen}}_1\left(\widetilde{\mathsf{msk}}, [\boldsymbol{m}_{\ell+1}]_1, [v]_1\right),$$

where

$$[v]_1 = \left[rf(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{w}) + r \sum_{i \in [\ell]} \Big(\prod_{j<i} f^j(\boldsymbol{x})\Big) f^i(\boldsymbol{s}_i) \Big(\prod_{j>i} f^j(\boldsymbol{u}_j + \boldsymbol{x})\Big) f^{\ell+1}(\boldsymbol{a} \otimes \boldsymbol{b}) \right]_1.$$

By definition of the weak simulation security (cf Definition 8.1), the output of $\widetilde{\mathsf{KeyGen}}_1$ and $\widetilde{\mathsf{KeyGen}}_2$ are identically distributed, thus:

$$\mathsf{adv}_4 = \mathsf{adv}_5.$$

• <u>**Hybrid**$_6$</u>: is the same as **Hybrid**$_5$, except that we replace the values $[\boldsymbol{a}_i r]_1$ by fresh $[\boldsymbol{v}_i]_1 \leftarrow_{\mathrm{R}} \mathbf{G}_1^2$ for all $i \in [n]$, using the DDH assumption in $\mathbf{G}_1$. Consequently, the challenge ciphertext now contains:

$$\mathsf{ct}_i \leftarrow \widehat{\mathsf{KeyGen}}\left(\widehat{\mathsf{msk}}, \begin{bmatrix} y_i \\ \boldsymbol{v}_i \end{bmatrix}_1\right).$$

Moreover, the secret keys are now generated using, for all $t \in [\ell]$:

$$\mathsf{sk}_t \leftarrow \widetilde{\mathsf{KeyGen}}_1\left(\widetilde{\mathsf{msk}}, [\mathbf{M}_t]_1, \left[f^t(\boldsymbol{s}_t) \prod_{t<i\leq \ell} f^i(\boldsymbol{u}_i + \boldsymbol{x}) f^{\ell+1}(\boldsymbol{v} \otimes \boldsymbol{b}),\right]_1\right)$$

and

$$\mathsf{sk}_{\ell+1} \leftarrow \widetilde{\mathsf{KeyGen}}_1\left(\widetilde{\mathsf{msk}}, [\boldsymbol{m}_{\ell+1}]_1, [v]_1\right),$$

where

$$[v]_1 = \left[f(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{w}) + \sum_{i \in [\ell]} \Big(\prod_{j<i} f^j(\boldsymbol{x})\Big) f^i(\boldsymbol{s}_i) \Big(\prod_{j>i} f^j(\boldsymbol{u}_j + \boldsymbol{x})\Big) f^{\ell+1}(\boldsymbol{v} \otimes \boldsymbol{b})\right]_1.$$

We show there exists a PPT adversary $\mathcal{B}_5$ such that:

$$|\mathsf{adv}_5 - \mathsf{adv}_6| \leq \mathsf{adv}^{\mathsf{DDH}}_{\mathbf{G}_1, \mathcal{B}_5}(\lambda).$$

The adversary $\mathcal{B}_5$ takes as input a tuple $\big([r]_1, \{[\boldsymbol{a}_i]_1, [\boldsymbol{v}_i]_1\}_{i \in [n]}\big)$ where the values $[\boldsymbol{v}_i]_1$ are either of the form $[\boldsymbol{a}_i r]_1$ (case 1), or uniformly random over $\mathbf{G}_1^2$ (case 2). The adversary $\mathcal{B}_5$ samples $(\widetilde{\mathsf{msk}}, \widetilde{\mathsf{pk}}) \leftarrow \widetilde{\mathsf{Setup}}(1^\lambda, \mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG}, n+1})$, $\boldsymbol{b}_j \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $\boldsymbol{w}_j \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2$ for all $j \in [n]$, $\boldsymbol{u}_k, \boldsymbol{s}_k \leftarrow_{\mathrm{R}} \mathbb{Z}_p^n$ for all $k \in [\ell]$, upon which it can simulate the view of the adversary $\mathcal{A}$ straightforwardly. In case 1, it simulates **Hybrid**$_5$ to $\mathcal{A}$, whereas it simulates **Hybrid**$_6$ in case 2.

• <u>**Hybrid**$_7$</u>: is the same as **Hybrid**$_6$, except we replace the values $\{\boldsymbol{v}_i\}_{i \in [n]}$ by $\{\boldsymbol{v}_i + y_i \boldsymbol{h}\}_{i \in [n]}$, where $\boldsymbol{h} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2$. These values are identically distributed, since the $\boldsymbol{v}_i$ are sampled uniformly over $\mathbb{Z}_p^2$, independently of the challenge $\{y_i\}_{i \in [n]}$, which is chosen beforehand. Therefore, we have:

$$\mathsf{adv}_6 = \mathsf{adv}_7.$$

Consequently, the challenge ciphertext now contains:

$$\mathsf{ct}_i \leftarrow \widehat{\mathsf{KeyGen}}\left(\widehat{\mathsf{msk}}, \begin{bmatrix} y_i \\ \boldsymbol{v}_i + y_i \boldsymbol{h} \end{bmatrix}_1\right).$$

Moreover, the secret keys are now generated using for all $t \in [\ell]$:

$$\mathsf{sk}_t \leftarrow \widetilde{\mathsf{KeyGen}}_1\left(\widetilde{\mathsf{msk}}, [\mathbf{M}_t]_1, \left[f^t(\boldsymbol{s}_t) \prod_{t<i\leq \ell} f^i(\boldsymbol{u}_i + \boldsymbol{x}) f^{\ell+1}((\boldsymbol{v} + \boldsymbol{y} \otimes \boldsymbol{h}) \otimes \boldsymbol{b}),\right]_1\right),$$

and

$$\mathsf{sk}_{\ell+1} \leftarrow \widetilde{\mathsf{KeyGen}}_1\left(\widetilde{\mathsf{msk}}, [\boldsymbol{m}_{\ell+1}]_1, [v]_1\right),$$

where

$$[v]_1 = \left[f(\boldsymbol{x}, \boldsymbol{v} + \boldsymbol{y} \otimes \boldsymbol{h}, \boldsymbol{w}) + \sum_{i \in [\ell]} \Big(\prod_{j<i} f^j(\boldsymbol{x}) \Big) f^i(\boldsymbol{s}_i) \Big(\prod_{j>i} f^j(\boldsymbol{u}_j + \boldsymbol{x}) \Big) f^{\ell+1}(\boldsymbol{v} + \boldsymbol{y} \otimes \boldsymbol{h}) \otimes \boldsymbol{b}) \right]_1,$$

where $\boldsymbol{y} \otimes \boldsymbol{h} = (y_j \cdot \boldsymbol{h})_{j \in [n]} \in \mathbb{Z}^{2n}$, and $(\boldsymbol{v} + \boldsymbol{y} \otimes \boldsymbol{h}) \otimes \boldsymbol{b} = ((\boldsymbol{v}_i + y_i \boldsymbol{h})^\top \boldsymbol{b}_j)_{i,j \in [n]} \in \mathbb{Z}^{n^2}$.

• <u>**Hybrid$_8$**</u>: is the same as **Hybrid$_7$**, except that we replace the values $[\boldsymbol{v}_i + y_i \boldsymbol{h}]_1$ by $[\boldsymbol{d} r_i + y_i \boldsymbol{h}]_1$ with $\boldsymbol{d} \leftarrow \mathsf{DDH}$ and $r_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p$ for all $i \in [n]$, using the DDH assumption in $\mathbf{G}_1$. Consequently, the ciphertexts now contains:

$$\mathsf{ct}_i \leftarrow \widehat{\mathsf{KeyGen}} \left(\widehat{\mathsf{msk}}, \begin{bmatrix} y_i \\ \boldsymbol{d} r_i + y_i \boldsymbol{h} \end{bmatrix}_1 \right).$$

Moreover, the secret keys are now generated using for all $t \in [\ell]$:

$$\mathsf{sk}_t \leftarrow \widetilde{\mathsf{KeyGen}}_1 \left(\widetilde{\mathsf{msk}}, [\mathbf{M}_t]_1, \left[f^t(\boldsymbol{s}_t) \prod_{t < i \leq \ell} f^i(\boldsymbol{u}_i + \boldsymbol{x}) f^{\ell+1}((\boldsymbol{r} \otimes \boldsymbol{d} + \boldsymbol{y} \otimes \boldsymbol{h}) \otimes \boldsymbol{b}), \right]_1 \right),$$

and

$$\mathsf{sk}_{\ell+1} \leftarrow \widetilde{\mathsf{KeyGen}}_1 \left(\widetilde{\mathsf{msk}}, [\boldsymbol{m}_{\ell+1}]_1, [v]_1 \right),$$

where

$$[v]_1 = \left[f(\boldsymbol{x}, \boldsymbol{r} \otimes \boldsymbol{d} + \boldsymbol{y} \otimes \boldsymbol{h}, \boldsymbol{w}) + \sum_{i \in [\ell]} \Big(\prod_{j<i} f^j(\boldsymbol{x}) \Big) f^i(\boldsymbol{s}_i) \Big(\prod_{j>i} f^j(\boldsymbol{u}_j + \boldsymbol{x}) \Big) f^{\ell+1}(\boldsymbol{r} \otimes \boldsymbol{d} + \boldsymbol{y} \otimes \boldsymbol{h}) \otimes \boldsymbol{b}) \right]_1,$$

where $\boldsymbol{r} \otimes \boldsymbol{d} = (r_i \cdot \boldsymbol{d})_{i \in [n]} \in \mathbb{Z}^{2n}$, and $(\boldsymbol{r} \otimes \boldsymbol{d} + \boldsymbol{y} \otimes \boldsymbol{h}) \otimes \boldsymbol{b} = ((\boldsymbol{d} r_i + y_i \boldsymbol{v})^\top \boldsymbol{b}_j)_{i,j \in [n]} \in \mathbb{Z}^{n^2}$.

We show there exists a PPT adversary $\mathcal{B}_7$ such that:

$$|\mathsf{adv}_7 - \mathsf{adv}_8| \leq \mathsf{adv}^{\mathsf{DDH}}_{\mathbf{G}_1, \mathcal{B}_7}(\lambda).$$

The adversary $\mathcal{B}_7$ takes as input a tuple $([\boldsymbol{d}]_1, \{[\boldsymbol{v}_i]_1\}_{i \in [n]})$ where the values $[\boldsymbol{v}_i]_1$ are either of the form $[\boldsymbol{d} r_i]_1$ (case 1), or uniformly random over $\mathbf{G}_1^2$ (case 2). The adversary $\mathcal{B}_7$ samples $\boldsymbol{h} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2$, $(\widetilde{\mathsf{msk}}, \widetilde{\mathsf{pk}}) \leftarrow \widetilde{\mathsf{Setup}}(1^\lambda, \mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG}, n+1})$, $\boldsymbol{a}_i, \boldsymbol{b}_j \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $\boldsymbol{w}_j \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2$ for all $i, j \in [n]$, $\boldsymbol{u}_k, \boldsymbol{s}_k \leftarrow_{\mathrm{R}} \mathbb{Z}_p^n$ for all $k \in [\ell]$, upon which it can simulate the view of the adversary $\mathcal{A}$ straightforwardly. In case 1, it simulates **Hybrid$_8$** to $\mathcal{A}$, whereas it simulates **Hybrid$_7$** in case 2.

• <u>**Hybrid$_9$**</u>: is the same as **Hybrid$_8$**, except 1) we change the distribution of $\boldsymbol{h}$ from uniformly random over $\mathbb{Z}_p^2$ to uniformly random over $\mathbb{Z}_p^2 \setminus \mathsf{Span}(\boldsymbol{d})$, which only induces a statistical change of $1/p$, given $\mathsf{Span}(\boldsymbol{d})$ is of size at most p; 2) we replace the values $\{\boldsymbol{w}_j\}_{j \in [n]}$ by $\{\boldsymbol{w}_j + z_i \boldsymbol{d}^\perp\}_{j \in [n]}$, where $\boldsymbol{d}^\perp \in \mathbb{Z}_p^2$ is such that $\boldsymbol{d}^\top \boldsymbol{d}^\perp = 0$ and $\boldsymbol{h}^\top \boldsymbol{d}^\perp = 1$ (note that such a vector exists as long as $\boldsymbol{h} \notin \mathsf{Span}(\boldsymbol{d})$). These values are identically distributed, since the $\boldsymbol{w}_j$ are sampled uniformly over $\mathbb{Z}_p^2$, independently of the challenge $\{z_j\}_{j \in [n]}$, which is chosen beforehand. Therefore, we have:

$$|\mathsf{adv}_8 - \mathsf{adv}_9| \leq \frac{1}{p}.$$

Consequently, the ciphertexts now contains:

$$\mathsf{ct}'_j \leftarrow \widehat{\mathsf{KeyGen}}\left(\widehat{\mathsf{msk}}, \begin{bmatrix} z_j \\ -\boldsymbol{w}_j - z_j\boldsymbol{d}^\perp \end{bmatrix}_1\right).$$

Moreover, the secret keys are now generated using:

$$\mathsf{sk}_{\ell+1} \leftarrow \widetilde{\mathsf{KeyGen}}_1\left(\widetilde{\mathsf{msk}}, [\boldsymbol{m}_{\ell+1}]_1, [v]_1\right),$$

where

$$[v]_1 = \Big[f(\boldsymbol{x}, \boldsymbol{r}\otimes\boldsymbol{d} + \boldsymbol{y}\otimes\boldsymbol{h}, \boldsymbol{w}) + f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) + \sum_{i\in[\ell]}\Big(\prod_{j<i} f^j(\boldsymbol{x})\Big) f^i(\boldsymbol{s}_i)\Big(\prod_{j>i} f^j(\boldsymbol{u}_j + \boldsymbol{x})\Big) f^{\ell+1}\big((\boldsymbol{r}\otimes\boldsymbol{d} + \boldsymbol{y}\otimes\boldsymbol{h})\otimes\boldsymbol{b})\big)\Big]_1.$$

• **Hybrid$_{10}$**: is the same as **Hybrid$_9$**, except the challenge ciphertext contains:

$$\mathsf{ct}_i \leftarrow \widehat{\mathsf{Enc}}\left(\widehat{\mathsf{msk}}, \begin{bmatrix} 0 \\ \boldsymbol{d}r_i + y_i\boldsymbol{h} \end{bmatrix}_1\right), \ \mathsf{ct}'_j \leftarrow \widehat{\mathsf{KeyGen}}\left(\widehat{\mathsf{msk}}, \begin{bmatrix} 0 \\ -\boldsymbol{w}_j \end{bmatrix}_1\right)$$

instead of

$$\mathsf{ct}_i \leftarrow \widehat{\mathsf{Enc}}\left(\widehat{\mathsf{msk}}, \begin{bmatrix} y_i \\ \boldsymbol{d}r_i + y_i\boldsymbol{h} \end{bmatrix}_1\right), \ \mathsf{ct}'_j \leftarrow \widehat{\mathsf{KeyGen}}\left(\widehat{\mathsf{msk}}, \begin{bmatrix} z_j \\ -\boldsymbol{w}_j - z_j\boldsymbol{d}^\perp \end{bmatrix}_1\right).$$

This transition is justified by the function-hiding IND security of $\widehat{\mathsf{IPFE}}$, which can be used since for all $i, j \in [n]$, we have $\binom{y_i}{\boldsymbol{d}r_i+y_i\boldsymbol{h}}^\top\binom{z_j}{-\boldsymbol{w}_j-z_j\boldsymbol{d}^\perp} = \binom{0}{\boldsymbol{d}r_i+y_i\boldsymbol{h}}^\top\binom{0}{-\boldsymbol{w}_j}$. The equality uses the fact that $\boldsymbol{d}^\top\boldsymbol{d}^\perp = 0$ and $\boldsymbol{h}^\top\boldsymbol{d}^\perp = 1$.

There exists a PPT adversary $\mathcal{B}_9$ such that:

$$|\mathsf{adv}_9 - \mathsf{adv}_{10}| \leq \mathsf{adv}^{\mathsf{IND\text{-}FH}}_{\widehat{\mathsf{IPFE}},\mathcal{B}_9}(\lambda).$$

The adversary $\mathcal{B}_9$ first samples $\boldsymbol{d} \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $\boldsymbol{h} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2 \setminus \mathsf{Span}(\boldsymbol{d})$, $\boldsymbol{d}^\perp \in \mathbb{Z}_p^2$ such that $\boldsymbol{d}^\top\boldsymbol{d}^\perp = 0$ and $\boldsymbol{h}^\top\boldsymbol{d}^\perp = 1$, $(\widetilde{\mathsf{msk}}, \widetilde{\mathsf{pk}}) \leftarrow \widetilde{\mathsf{Setup}}(1^\lambda, \mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},n+1})$, $r_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $\boldsymbol{a}_i, \boldsymbol{b}_j \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $\boldsymbol{w}_j \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2$ for all $i, j \in [n]$, $\boldsymbol{u}_k, \boldsymbol{s}_k \leftarrow_{\mathrm{R}} \mathbb{Z}_p^n$ for all $k \in [\ell]$. It sends the challenge

$$\left\{\begin{bmatrix} y_i \\ \boldsymbol{d}r_i + y_i\boldsymbol{h} \end{bmatrix}_1, \begin{bmatrix} 0 \\ \boldsymbol{d}r_i + y_i\boldsymbol{h} \end{bmatrix}_1\right\}_{i\in[n]}, \left\{\begin{bmatrix} z_j \\ -\boldsymbol{w}_j - z_j\boldsymbol{d}^\perp \end{bmatrix}_1, \begin{bmatrix} 0 \\ -\boldsymbol{w}_j \end{bmatrix}_1\right\}_{j\in[n]}$$

to its own experiment, upon which it receives $\{\mathsf{ct}_i\}_{i\in[n]}$, encryptions of the left or right challenges; together with $\{\mathsf{ct}'_j\}_{j\in[n]}$, functional secret keys associated with the left or right challenges. In the left case, $\mathcal{B}_9$ simulates **Hybrid$_9$** to $\mathcal{A}$, whereas it simulates **Hybrid$_{10}$** in the right case.

• **Hybrid$_{11}$**: is the same as **Hybrid$_{10}$**, except 1) we change the distribution of $\boldsymbol{h}$ from uniformly random over $\mathbb{Z}_p^2 \setminus \mathsf{Span}(\boldsymbol{d})$ to uniformly random over $\mathbb{Z}_p^2$; this introduces a statistical distance of $1/p$ since the size of $\mathsf{Span}(\boldsymbol{d})$ is at most p; 2) we replace the values $\{[\boldsymbol{d}r_i + y_i\boldsymbol{h}]_1\}_{i\in[n]}$ by $\{[\boldsymbol{v}_i + y_i\boldsymbol{h}]_1\}_{i\in[n]}$, where $\boldsymbol{v}_i \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2$ for all $i \in [n]$, using the DDH assumption in $\mathbf{G}_1$. This transition is the reverse to the transition from **Hybrid$_5$** to **Hybrid$_6$**.

Consequently, the challenge ciphertext now contains:

$$\mathsf{ct}_i \leftarrow \widehat{\mathsf{Enc}}\left(\widehat{\mathsf{msk}}, \begin{bmatrix} 0 \\ \boldsymbol{v}_i + y_i \boldsymbol{h} \end{bmatrix}_1\right),$$

and the secret keys are now generated using, for all $t \in [\ell]$:

$$\mathsf{sk}_t \leftarrow \widetilde{\mathsf{KeyGen}}_1\left(\widetilde{\mathsf{msk}}, [\mathbf{M}_t]_1, \left[f^t(\boldsymbol{s}_t) \prod_{t<i\leq \ell} f^i(\boldsymbol{u}_i + \boldsymbol{x}) f^{\ell+1}((\boldsymbol{v} + \boldsymbol{y} \otimes \boldsymbol{h}) \otimes \boldsymbol{b}),\right]_1\right),$$

and

$$\mathsf{sk}_{\ell+1} \leftarrow \widetilde{\mathsf{KeyGen}}_1\left(\widetilde{\mathsf{msk}}, [\boldsymbol{m}_{\ell+1}]_1, [v]_1\right),$$

where

$$[v]_1 = \Big[f(\boldsymbol{x}, \boldsymbol{v} + \boldsymbol{y} \otimes \boldsymbol{h}, \boldsymbol{w}) + f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) + \sum_{i \in [\ell]} \Big(\prod_{j<i} f^j(\boldsymbol{x})\Big) f^i(\boldsymbol{s}_i) \Big(\prod_{j>i} f^j(\boldsymbol{u}_j + \boldsymbol{x})\Big) f^{\ell+1}\big((\boldsymbol{v} + \boldsymbol{y} \otimes \boldsymbol{h}) \otimes \boldsymbol{b}\big)\Big]_1.$$

We show there exists a PPT adversary $\mathcal{B}_{10}$ such that:

$$|\mathsf{adv}_{10} - \mathsf{adv}_{11}| \leq \mathsf{adv}^{\mathsf{DDH}}_{\mathbf{G}_1, \mathcal{B}_{10}}(\lambda) + \frac{1}{p}.$$

The adversary $\mathcal{B}_{10}$ takes as input a tuple $([\boldsymbol{d}]_1, \{[\boldsymbol{v}_i]_1\}_{i \in [n]})$ where the vectors $[\boldsymbol{v}_i]_1$ are either of the form $[\boldsymbol{d} r_i]_1$ (case 1), or uniformly random over $\mathbf{G}_1^2$ (case 2). The adversary $\mathcal{B}_{10}$ samples $\boldsymbol{h} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2$, $(\widetilde{\mathsf{msk}}, \widetilde{\mathsf{pk}}) \leftarrow \widetilde{\mathsf{Setup}}(1^\lambda, \mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG}, n+1})$, $\boldsymbol{a}_i, \boldsymbol{b}_j \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $\boldsymbol{w}_j \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2$ for all $i, j \in [n]$, $\boldsymbol{u}_k, \boldsymbol{s}_k \leftarrow_{\mathrm{R}} \mathbb{Z}_p^n$ and for all $k \in [\ell]$, upon which it can simulate the view of the adversary $\mathcal{A}$ straightforwardly. In case 1, it simulates $\mathbf{Hybrid}_{11}$ to $\mathcal{A}$, whereas it simulates $\mathbf{Hybrid}_{10}$ in case 2.

• <u>$\mathbf{Hybrid}_{12}$</u>: is the same as $\mathbf{Hybrid}_{11}$, except we replace the values $\{\boldsymbol{v}_i + y_i \boldsymbol{h}\}_{i \in [n]}$ by $\{\boldsymbol{v}_i\}_{i \in [n]}$. These values are identically distributed, since the $\boldsymbol{v}_i$ are sampled uniformly over $\mathbb{Z}_p^2$, independently of the challenge $\{y_i\}_{i \in [n]}$, which is chosen beforehand. Therefore, we have:

$$\mathsf{adv}_{11} = \mathsf{adv}_{12}.$$

This transition is the reverse of the transition from $\mathbf{Hybrid}_6$ to $\mathbf{Hybrid}_7$. The secret keys are now generated using, for all $t \in [\ell]$:

$$\mathsf{sk}_t \leftarrow \widetilde{\mathsf{KeyGen}}_1\left(\widetilde{\mathsf{msk}}, [\mathbf{M}_t]_1, \left[f^t(\boldsymbol{s}_t) \prod_{t<i\leq \ell} f^i(\boldsymbol{u}_i + \boldsymbol{x}) f^{\ell+1}(\boldsymbol{v} \otimes \boldsymbol{b}),\right]_1\right),$$

and

$$\mathsf{sk}_{\ell+1} \leftarrow \widetilde{\mathsf{KeyGen}}_1\left(\widetilde{\mathsf{msk}}, [\boldsymbol{m}_{\ell+1}]_1, [v]_1\right),$$

where

$$[v]_1 = \left[f(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{w}) + f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) + \sum_{i \in [\ell]} \Big(\prod_{j<i} f^j(\boldsymbol{x})\Big) f^i(\boldsymbol{s}_i) \Big(\prod_{j>i} f^j(\boldsymbol{u}_j + \boldsymbol{x})\Big) f^{\ell+1}(\boldsymbol{v} \otimes \boldsymbol{b})\right]_1.$$

In $\mathbf{Hybrid}_{13}$, the challenge ciphertext $\left(\overline{\mathsf{ct}}, \{\mathsf{ct}_i, \mathsf{ct}'_j\}_{i,j\in[n]}\right)$ is as follows. $\overline{\mathsf{ct}} \leftarrow \widetilde{\mathsf{Enc}}\left(\widetilde{\mathsf{msk}}\right)$. For all $i, j \in [n]$: $\mathsf{ct}_i \leftarrow \widehat{\mathsf{Enc}}\left(\widehat{\mathsf{msk}}, \begin{bmatrix} 0 \\ \boldsymbol{v}_i \end{bmatrix}_1\right)$, $\mathsf{ct}'_j \leftarrow \widehat{\mathsf{KeyGen}}\left(\widehat{\mathsf{msk}}, \begin{bmatrix} 0 \\ -\boldsymbol{w}_j \end{bmatrix}_2\right)$, $\overline{\mathsf{ct}} \leftarrow \widetilde{\mathsf{Enc}}\left(\widetilde{\mathsf{msk}}\right)$.

This exactly corresponds to the experiment $\mathsf{Ideal}^{\mathsf{FE}}_{\mathcal{A},\mathcal{S}}(1^\lambda)$ for the simulator $\mathcal{S} = (\widetilde{\mathsf{Setup}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{KeyGen}})$ defined in Fig.8.2.

Summing up, we have PPT adversaries $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$ and $\mathcal{B}_4$ such that:

$$\mathsf{adv}^{\mathsf{SIM}}_{\mathsf{PHFE},\mathcal{A}}(\lambda) \leq \mathsf{adv}^{\mathsf{weak\text{-}SIM}}_{\overline{\mathsf{IPFE}},\mathcal{B}_1}(\lambda) + (\ell+1)\cdot \mathsf{adv}^{\mathsf{DDH}}_{\mathbf{G}_2,\mathcal{B}_2}(\lambda) + 3\cdot \mathsf{adv}^{\mathsf{DDH}}_{\mathbf{G}_1,\mathcal{B}_3}(\lambda) + \mathsf{adv}^{\mathsf{IND\text{-}FH}}_{\widehat{\mathsf{IPFE}},\mathcal{B}_4}(\lambda) + \frac{2}{p}.$$

$\square$

$\underline{\widetilde{\mathsf{Setup}}(1^\lambda, \mathcal{F}^{\mathsf{phfe}}_{\mathcal{PG},n,\ell,w})\text{:}}$

$\mathcal{PG} = (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, p, P_1, P_2, e) \leftarrow_{\mathrm{R}} \mathsf{PGGen}(1^\lambda)$, $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{msk}}) \leftarrow \widetilde{\mathsf{Setup}}(1^\lambda, \mathcal{F}_{\mathsf{IPFE},\mathcal{PG},n+1})$. For all $i, j \in [n]$: $\boldsymbol{a}_i, \boldsymbol{b}_j \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $\boldsymbol{v}_i, \boldsymbol{w}_j \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2$. For all $k \in [\ell]$: $\boldsymbol{u}_k \leftarrow_{\mathrm{R}} \mathbb{Z}_p^n$. $\widetilde{\mathsf{pk}} := \left(\widetilde{\mathsf{pk}}, \{[\boldsymbol{a}_i]_1, [\boldsymbol{b}_j]_2\}_{i,j\in[n]}\right)$, $\widetilde{\mathsf{msk}} := \left(\widetilde{\mathsf{msk}}, \{\boldsymbol{a}_i, \boldsymbol{b}_j, \boldsymbol{v}_i, \boldsymbol{u}_k\}_{i,j\in[n],k\in[\ell]}\right)$. Return $\widetilde{\mathsf{pk}}, \widetilde{\mathsf{msk}}$.

$\underline{\widetilde{\mathsf{Enc}}(\widetilde{\mathsf{msk}})\text{:}}$

$(\widehat{\mathsf{pk}}, \widehat{\mathsf{msk}}) \leftarrow \widehat{\mathsf{Setup}}(1^\lambda, \mathcal{F}^{\mathsf{ipfe}}_{\mathcal{PG},3})$, $\overline{\mathsf{ct}} \leftarrow \widetilde{\mathsf{Enc}}\left(\widetilde{\mathsf{msk}}\right)$. For all $i, j \in [n]$: $\mathsf{ct}_i \leftarrow \widehat{\mathsf{Enc}}\left(\widehat{\mathsf{msk}}, \begin{bmatrix} 0 \\ \boldsymbol{v}_i \end{bmatrix}_1\right)$, $\mathsf{ct}'_j \leftarrow \widehat{\mathsf{KeyGen}}\left(\widehat{\mathsf{msk}}, \begin{bmatrix} 0 \\ -\boldsymbol{w}_j \end{bmatrix}_2\right)$. Return $\left(\overline{\mathsf{ct}}, \{\mathsf{ct}_i, \mathsf{ct}'_j\}_{i,j\in[n]}\right)$.

$\underline{\widetilde{\mathsf{KeyGen}}(\widetilde{\mathsf{msk}}, (f^0, \ldots, f^{\ell+1}), f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}), \boldsymbol{x})\text{:}}$

For all $t \in [\ell]$, $\mathsf{sk}_t \leftarrow \widetilde{\mathsf{KeyGen}}_1\left(\widetilde{\mathsf{msk}}, [\mathbf{M}_t]_1, \left[f^t(\boldsymbol{s}_t)\prod_{t<i\leq\ell} f^i(\boldsymbol{u}_i + \boldsymbol{x}) f^{\ell+1}(\boldsymbol{v}\otimes\boldsymbol{b}),\right]_1\right)$, $\mathsf{sk}_{\ell+1} \leftarrow \widetilde{\mathsf{KeyGen}}_1\left(\widetilde{\mathsf{msk}}, [\boldsymbol{m}_{\ell+1}]_1, [v]_1\right)$, where

$$[v]_1 = \left[f(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{w}) + f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) + \sum_{i\in[\ell]}\Big(\prod_{j<i} f^j(\boldsymbol{x})\Big) f^i(\boldsymbol{s}_i)\Big(\prod_{j>i} f^j(\boldsymbol{u}_j + \boldsymbol{x})\Big) f^{\ell+1}(\boldsymbol{v}\otimes\boldsymbol{b})\right]_1.$$

Return $\{\mathsf{sk}_t\}_{t\in[\ell+1]}$.

Figure 10: Simulator for the FE scheme depicted in Fig.8.2 for the functionality $\mathcal{F}^{\mathsf{phfe}}_{\mathcal{PG},n,\ell,w}$.

8.3 Constructing Inner-Product FE

Here, we build a public-key FE inner products, that is, the functionality $\mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},\mathsf{dim}}$ for some pairing group $\mathcal{PG} \leftarrow \mathsf{PGGen}(1^\lambda)$ and dimension $\mathsf{dim} \in \mathbb{N}$. Our scheme is presented in Fig.8.3.

It builds upon the inner-product FE from [ALS16], that relies on the DDH assumption in pairing-free cyclic groups. We instead use a pairing group $\mathcal{PG} = (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, p, P_1, P_2, e)$, where the ciphertexts will consist of group elements in $\mathbf{G}_1$, and the ALS functional secret key are embedded in $\mathbf{G}_2$, instead of $\mathbb{Z}_p$. Decryption now yields the inner product in $\mathbf{G}_T$.

This simple modification of ALS scheme already satisfies a simulation-security where the simulator needs to know the values $[\boldsymbol{x}^\top \boldsymbol{y}]_2 \in \mathbf{G}_2$ and $[\boldsymbol{y}]_2 \in \mathbf{G}_2^{\mathsf{dim}}$ in order to simulate the challenge ciphertext that encrypts $[\boldsymbol{x}]_1 \in \mathbf{G}_1^{\mathsf{dim}}$ and the functional secret key associated to $[\boldsymbol{y}]_2 \in \mathbf{G}_2^{\mathsf{dim}}$. This security property is inherited from the ALS scheme, which was proven simulation-secure in [Wee17] (see also [AGRW17, Appendix A]). Note that this is weaker than the standard simulation security notion, given in Definition 4.3, where the simulator gets the output of the function, which in this case, is $[\boldsymbol{x}^\top \boldsymbol{y}]_T \in \mathbf{G}_T$, not $[\boldsymbol{x}^\top \boldsymbol{y}]_2$.

For our purposes, we want it to be possible for the simulator to choose whether it simulates the adversary's view from the values $[\boldsymbol{x}^\top \boldsymbol{y}]_2$, $[\boldsymbol{y}]_2$ or $[\boldsymbol{x}^\top \boldsymbol{y}]_1$, $[\boldsymbol{y}]_1$. We achieve this by giving two copies of the encryption, one in $\mathbf{G}_1$, one $\mathbf{G}_2$, and splitting each functional secret key in two additive secret shares summing up to the actual key, one in $\mathbf{G}_2$ and one in $\mathbf{G}_1$. This simulation security relies on the fact that it is possible to produce both of these shares knowing the secret either in $\mathbf{G}_1$ or $\mathbf{G}_2$.

$\underline{\mathsf{Setup}(1^\lambda, \mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},\mathsf{dim}})\text{:}}$

Given $\mathcal{PG} = (\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, p, P_1, P_2, e) \leftarrow_{\mathrm{R}} \mathsf{PGGen}(1^\lambda)$, it computes $\boldsymbol{a} \leftarrow_{\mathrm{R}} \mathsf{DDH}$, $\mathbf{W} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{\mathsf{dim}\times 2}$,
Return $\mathsf{pk} := \{[\boldsymbol{a}]_s, [\mathbf{W}\boldsymbol{a}]_s\}_{s\in[1,2]}$ and $\mathsf{msk} = \mathbf{W}$.

$\underline{\mathsf{Enc}(\mathsf{pk}, \boldsymbol{x} \in \mathbb{Z}^{\mathsf{dim}})\text{:}}$

$r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $\boldsymbol{c} = \begin{pmatrix} \boldsymbol{a}r \\ \boldsymbol{x} + \mathbf{W}\boldsymbol{a}r \end{pmatrix}$. Return $([\boldsymbol{c}]_1, [\boldsymbol{c}]_2)$.

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \boldsymbol{y} \in \mathbb{Z}^{\mathsf{dim}})\text{:}}$

$\boldsymbol{u} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{2+\mathsf{dim}}$, $\boldsymbol{k} = \begin{pmatrix} -\mathbf{W}^\top \boldsymbol{y} \\ \boldsymbol{y} \end{pmatrix}$. Return $([\boldsymbol{u}]_1, [\boldsymbol{k} - \boldsymbol{u}]_2)$.

$\underline{\mathsf{Dec}(\mathsf{ct}, \mathsf{sk})\text{:}}$

Parse $\mathsf{ct} = ([\boldsymbol{c}_1]_1, [\boldsymbol{c}_2]_2)$ and $\mathsf{sk} = ([\boldsymbol{k}_1]_1, [\boldsymbol{k}_2]_2)$. Return $[\boldsymbol{c}_1^\top \boldsymbol{k}_1 + \boldsymbol{c}_2^\top \boldsymbol{k}_2]_T$.

Figure 11: This is IPFE, an FE scheme for the functionality $\mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},\mathsf{dim}}$, with weak-simulation security.

Linear efficiency.

The encryption of any $\boldsymbol{x} \in \mathbb{Z}^{\mathsf{dim}}$ comprises $\mathsf{dim} + 2$ group elements from $\mathbf{G}_1$ and $\mathsf{dim} + 2$ group elements from $\mathbf{G}_2$, each of which is $\mathsf{poly}(\lambda)$ bits.

Correctness.

For any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}^{\mathsf{dim}}$:

$$[\boldsymbol{c}_1^\top \boldsymbol{k}_1 + \boldsymbol{c}_2^\top \boldsymbol{k}_2]_T = [\boldsymbol{c}^\top \boldsymbol{k}]_T = [\boldsymbol{x}^\top \boldsymbol{y}]_T.$$

Theorem 8.2 (Weak-simulation security)**.** *The scheme presented in Fig.8.3 is weakly-simulation secure (as per Definition 8.1) assuming the bilateral DLIN assumption. Namely, for any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that:*

$$\mathsf{adv}^{\mathsf{weak\text{-}SIM}}_{\mathsf{IPFE},\mathcal{A}}(\lambda) \leq \mathsf{adv}^{\mathsf{DLIN}}_{\mathcal{PG},\mathcal{B}}(\lambda) + \frac{1}{p}.$$

Proof. The proof proceeds using a series of hybrid games, described below. Let $\mathcal{A}$ be a PPT adversary against the weak simulation security of the scheme. For any game $\mathbf{Hybrid}_i$, we denote by $\mathsf{adv}_i := \Pr[1 \leftarrow \mathbf{Hybrid}_i(\mathcal{A})]$ the probability that $\mathbf{Hybrid}_i$ returns 1 when interacting with $\mathcal{A}$.

• $\underline{\mathbf{Hybrid}_0}$: is the real experiment as given in Definition 8.1.

• $\underline{\mathbf{Hybrid}_1}$: is the same as $\mathbf{Hybrid}_0$, except the challenge ciphertext is computed using $\boldsymbol{c} = \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{x} + \mathbf{W}\boldsymbol{u} \end{pmatrix}$ with $\boldsymbol{u} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2$ instead of $\boldsymbol{c} = \begin{pmatrix} \boldsymbol{a}r \\ \boldsymbol{x} + \mathbf{W}\boldsymbol{a}r \end{pmatrix}$ with $r \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, using the bilateral DLIN assumption. We show there exists a PPT adversary $\mathcal{B}$ such that:

$$|\mathsf{adv}_0 - \mathsf{adv}_1| \leq \mathsf{adv}^{\mathsf{DLIN}}_{\mathcal{PG},\mathcal{B}}(\lambda).$$

The adversary $\mathcal{B}$ takes as input a tuple $([\mathbf{A}]_s, [\boldsymbol{z}]_s)_{s\in[1,2]}$, where the vectors $[\boldsymbol{z}]_s$ are of the form $[\mathbf{A}\boldsymbol{r}]_s$ with $\boldsymbol{r} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^2$ (case 1) or uniformly random over $\mathbf{G}_s^2$ (case 2). The adversary $\mathcal{B}$ samples $\mathbf{W} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{\mathsf{dim}\times 3}$, upon which it can simulate the view of the adversary $\mathcal{A}$ straightforwardly. In case 1, it simulate $\mathbf{Hybrid}_0$, whereas it simulates $\mathbf{Hybrid}_1$ in case 2.

• $\underline{\mathbf{Hybrid}_2}$: is the same as $\mathbf{Hybrid}_1$, except the challenge ciphertext is computed using $\boldsymbol{u} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3 \setminus \mathsf{Span}(\mathbf{A})$ instead of $\boldsymbol{u} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3$. This only induces a statistical change of $1/p$ since the size of $\mathsf{Span}(\mathbf{A})$ is at most p^2. Thus:

$$|\mathsf{adv}_1 - \mathsf{adv}_2| \leq \frac{1}{p}.$$

• $\underline{\mathbf{Hybrid}_3}$: is the same as $\mathbf{Hybrid}_2$, except the challenge ciphertext is computed using:

$$\boldsymbol{c} = \begin{pmatrix} \boldsymbol{u} \\ \mathbf{W}\boldsymbol{u} \end{pmatrix},$$

where $\boldsymbol{u} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3 \setminus \mathsf{Span}(\mathbf{A})$. Besides, the functional keys are computed using:

$$\boldsymbol{k} = \begin{pmatrix} \boldsymbol{x}^\top \boldsymbol{y} - \mathbf{W}^\top \boldsymbol{y} \\ \boldsymbol{y} \end{pmatrix}.$$

We show that these two games are identically distributed, using the fact that for any $\boldsymbol{x} \in \mathbb{Z}^{\mathsf{dim}}$ and $\boldsymbol{a}^\perp \in \mathbb{Z}_p^3$, the following are identically distributed:

$$\mathbf{W} \ \text{ and } \ \mathbf{W} - \boldsymbol{x}(\boldsymbol{a}^\perp)^\top,$$

with $\mathbf{W} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{\mathsf{dim}\times 3}$. We use that fact with $\boldsymbol{x}$ the challenge chosen by the adversary, which is chosen beforehand, and therefore, independently of $\mathsf{msk} = \mathbf{W}$; and $\boldsymbol{a}^\perp \in \mathbb{Z}_p^3$ such that $\mathbf{A}^\top \boldsymbol{a}^\perp = \mathbf{0}$ and $\boldsymbol{u}^\top \boldsymbol{a}^\perp = 1$. Note that such a vector exists since $\boldsymbol{u} \notin \mathsf{Span}(\mathbf{A})$. The leftmost distribution corresponds to $\mathbf{Hybrid}_2$, whereas the rightmost distribution corresponds to $\mathbf{Hybrid}_3$. Thus:

$$\mathsf{adv}_2 = \mathsf{adv}_3.$$

It is clear hat $\mathbf{Hybrid}_3$ corresponds to $\mathsf{Ideal}^{\mathsf{IPFE}}_{\mathcal{A},\mathcal{S}}(1^\lambda)$ with the simulator $\mathcal{S}$ described in Fig.8.3. Consequently, we have:

$$\mathsf{adv}^{\mathsf{weak\text{-}SIM}}_{\mathsf{IPFE},\mathcal{A}}(\lambda) \leq \mathsf{adv}^{\mathsf{DLIN}}_{\mathcal{PG},\mathcal{B}}(\lambda) + \frac{1}{p}.$$

□

$\widetilde{\mathsf{Setup}}(1^\lambda, \mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},\mathsf{dim}})$:

$\mathbf{A} \leftarrow_{\mathrm{R}} \mathsf{DLIN}$, $\mathbf{W} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{\mathsf{dim}\times 3}$, $\boldsymbol{u} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^3 \setminus \mathsf{Span}(\mathbf{A})$, $\boldsymbol{a}^\perp \in \mathbb{Z}_p^3$ such that $\mathbf{A}^\top \boldsymbol{a}^\perp = \mathbf{0}$ and $\boldsymbol{u}^\top \boldsymbol{a}^\perp = 1$.
Return $\widetilde{\mathsf{pk}} = \{[\mathbf{A}]_s, [\mathbf{W}\mathbf{A}]_s\}_{s\in[1,2]}\}$ and $\widetilde{\mathsf{msk}} = (\mathbf{W}, \boldsymbol{u}, \boldsymbol{a}^\perp)$.

$\widetilde{\mathsf{Enc}}(\widetilde{\mathsf{msk}})$:

$\boldsymbol{c} = \begin{pmatrix} \boldsymbol{u} \\ \mathbf{W}\boldsymbol{u} \end{pmatrix}$. Return $([\boldsymbol{c}]_1, [\boldsymbol{c}]_2)$.

For all $s \in [1,2]$, $\widetilde{\mathsf{KeyGen}}_s(\widetilde{\mathsf{msk}}, [\boldsymbol{x}^\top \boldsymbol{y}]_s, [\boldsymbol{y}]_s)$:

Return $\begin{bmatrix} \boldsymbol{x}^\top \boldsymbol{y} \cdot \boldsymbol{a}^\perp - \mathbf{W}^\top \boldsymbol{y} \\ \boldsymbol{y} \end{bmatrix}_s$.

Figure 12: Simulator for the FE scheme from Fig.8.3 for the functionality $\mathcal{F}^{\mathsf{ipfe}'}_{\mathcal{PG},\mathsf{dim}}$.

9 Acknowledgements

Romian Gay is partially supported by NSF Award SATC-1704788 and in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via 2019-19-020700006. This work was partially done while the author was working at UC Berkeley.

Aayush Jain was partially supported by grants listed under Amit Sahai and a Google PhD fellowship in the area of security and privacy (2018). This work was partly carried out during a research visit to Simons insititute of theoretical computer science conducted with support from DIMACS in association with its Special Focus on Cryptography and partly while the author was at NTT Research.

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.

This work was partly carried out when the last three authors were at the Simons Institute Program on Lattices 2020.

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, SIEVE, the National Science Foundation, NTT Research, Google, or the U.S. Government.

10 References

[ABDP15] 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.

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

[ABSV15] 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.

[ACF$^+$15] 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.

[ACGU20] 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`.

[AG11] 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.

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

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

[AGRW17] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 601–626. Springer, Heidelberg, April / May 2017.

[AJ15] 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.

[AJL$^+$12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012.

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

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

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

[ALS16] Shweta Agrawal, Benoˆ

tLibert, andDamienStehlé.Fullysecurefunctionalencryptionforinnerproducts, fromstandardassu
*−362.Springer, Heidelberg, August*2016.

[AP20a] [2] 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.

[AP20b] Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, LNCS, pages 110–140. Springer, Heidelberg, May 2020.

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

[AR17a] Shweta Agrawal and Alon Rosen. Functional encryption for bounded collusions, revisited. In *TCC*, pages 173–205, 2017.

[AR17b] 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.

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

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

[BBKK18] Boaz Barak, Zvika Brakerski, Ilan Komargodski, and Pravesh K. Kothari. Limits on low-degree pseudorandom generators (or: Sum-of-squares meets program obfuscation). In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 649–679. Springer, Heidelberg, April / May 2018.

[BCFG17] Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 67–98. Springer, Heidelberg, August 2017.

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

[BF01] 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.

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

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

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

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

[BGPW16] 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.

[BGV12] 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.

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

[BIJ$^+$20a] 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.

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

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

[BNPW16a] 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.

[BNPW16b] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. Cryptology ePrint Archive, Report 2016/558, 2016. `http://eprint.iacr.org/2016/558`.

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

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

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

[BSW11] 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.

[BV11] 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.

[BV15] 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.

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

[CCL18a] Yi-Hsiu Chen, Kai-Min Chung, and Jyun-Jie Liao. On the complexity of simulating auxiliary input. In *EUROCRYPT*, Cham, 2018.

[CCL18b] Yi-Hsiu Chen, Kai-Min Chung, and Jyun-Jie Liao. On the complexity of simulating auxiliary input. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 371–390. Springer, Heidelberg, April / May 2018.

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

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

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

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

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

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

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

[CSA20] 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`.

[CTA19] 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.

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

[Gay20] 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.

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

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

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

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

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

[GKP$^+$13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 555–564. ACM Press, June 2013.

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

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

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

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

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

[GVW12a] 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.

[GVW12b] 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.

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

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

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

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

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

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

[JLS19] Aayush Jain, Huijia Lin, and Amit Sahai. Simplifying constructions and assumptions for io. Technical report, Cryptology ePrint Archive, Report 2019/1252, 2019. https://eprint.iacr.org/2019/1252, 2019.

[Jou00] 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.

[JP14] Dimitar Jetchev and Krzysztof Pietrzak. How to fake auxiliary input. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 566–590. Springer, Heidelberg, February 2014.

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

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

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

[Las01] Jean B. Lasserre. New positive semidefinite relaxations for nonconvex quadratic programs. In *Advances in convex analysis and global optimization (Pythagorion, 2000)*, volume 54 of *Nonconvex Optim. Appl.*, pages 319–331. Kluwer Acad. Publ., Dordrecht, 2001.

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

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

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

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

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

[LV17a] Alex Lombardi and Vinod Vaikuntanathan. Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 119–137. Springer, Heidelberg, November 2017.

[LV17b] Alex Lombardi and Vinod Vaikuntanathan. Minimizing the complexity of goldreich's pseudorandom generator. *IACR Cryptology ePrint Archive*, 2017:277, 2017.

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

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012.

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

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

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

[MZ18] Fermi Ma and Mark Zhandry. The MMap strikes back: Obfuscation and new multilinear maps immune to CLT13 zeroizing attacks. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 513–543. Springer, Heidelberg, November 2018.

[Nes00] Yurii Nesterov. Squared functional systems and optimization problems. In *High performance optimization*, volume 33 of *Appl. Optim.*, pages 405–440. Kluwer Acad. Publ., Dordrecht, 2000.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. *IACR Cryptology ePrint Archive*, 2010:556, 2010.

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

[Par00] Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization.* PhD thesis, Citeseer, 2000.

[PST14a] 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.

[PST14b] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 500–517, 2014.

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

[Sch94] Claus-Peter Schnorr. Block reduced lattice bases and successive minima. *Comb. Probab. Comput.*, 3:507–522, 1994.

[Sho87] N. Z. Shor. Quadratic optimization problems. *Izv. Akad. Nauk SSSR Tekhn. Kibernet.*, (1):128–139, 222, 1987.

[SS10a] 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.

[SS10b] 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.

[Ste] Damien Stehlé. Slides: The lwe problem from lattices to cryptography. *https://summerschool-croatia.cs.ru.nl/2015/Lattice-based%20crypto.pdf.*

[SW05] 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.

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

[Wee17] Hoeteck Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 206–233. Springer, Heidelberg, November 2017.

A Perturbation Resilient Generators

Now we describe the notion of a Perturbation Resilient Generator ($\Delta\mathsf{RG}$ for short), proposed by [AJS18, AJL+19, JLMS19]. A $\Delta\mathsf{RG}$ consists of the following algorithms:

- $\mathsf{Setup}(1^\lambda, 1^n, B) \to (\mathsf{pp}, \mathsf{Seed})$. The setup algorithm takes as input a security parameter λ, the length parameter 1^n and a polynomial $B = B(\lambda)$ and outputs a seed $\mathsf{Seed} \in \{0,1\}^*$ and public parameters pp.
- $\mathsf{Eval}(\mathsf{pp}, \mathsf{Seed}) \to (h_1, ..., h_\ell)$, evaluation algorithm output a vector $(h_1, ..., h_\ell) \in \mathbb{Z}^\ell$. Here ℓ is the stretch of $\Delta\mathsf{RG}$.

We have following properties of in a $\Delta\mathsf{RG}$ scheme.

Efficiency: We require for $\mathsf{Setup}(1^\lambda, 1^n, B) \to (\mathsf{pp}, \mathsf{Seed})$ and $\mathsf{Eval}(\mathsf{pp}, \mathsf{Seed}) \to (h_1, ..., h_\ell)$,

- $|\mathsf{Seed}| = n \cdot \mathsf{poly}(\lambda)$ for some polynomial poly independent of n. The size of Seed is linear in n.
- For all $i \in [\ell]$, $|h_i| < \mathsf{poly}(\lambda, n)$. The norm of each output component h_i in $\mathbb{Z}$ is bounded by some polynomial in λ and n.

$(s, \mathsf{adv})-$**Perturbation Resilience:** We require that for large enough security parameter λ, for every polynomial B, there exists a large enough polynomial $n_B(\lambda)$ such that for any $n > n_B$, we have that for any distinguisher D of size s and any $(a_1, .., a_\ell) \in [-B, B]^\ell$

$$|\Pr[D(x \xleftarrow{\$} \mathcal{D}_1) = 1] - \Pr[D(x \xleftarrow{\$} \mathcal{D}_2) = 1]| < \mathsf{adv}$$

Here $\mathcal{D}_1$ and $\mathcal{D}_2$ are defined below:

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

Now we describe the notion of Perturbation Resilient Generator implementable in a function class $\mathcal{F}$ ($\mathcal{F}$-$\Delta\mathsf{RG}$ for short.)

$\Delta\mathsf{RG}$ **implementable in $\mathcal{F}$.** A $\Delta\mathsf{RG}$ scheme implementable in function class $\mathcal{F} = \{\mathcal{F}_n\}_{n\in\mathbb{N}}$ ($\mathcal{F}$-$\Delta\mathsf{RG}$ for short) is a perturbation resilient generator with additional properties. We describe syntax again for a complete specification.

- $\mathsf{Setup}(1^\lambda, 1^n, B) \rightarrow (\mathsf{pp}, \mathsf{Seed})$. The setup algorithm takes as input a security parameter λ, the length parameter 1^n and a polynomial $B = B(\lambda)$ and outputs a seed Seed and public parameters pp. Here, $\mathsf{Seed} = (\mathsf{Seed.pub}, \mathsf{Seed.priv}(1), \mathsf{Seed.priv}(2))$ is a vector on $\mathbb{F}_p$. Also, $\mathsf{pp} = (\mathsf{Seed.pub}(1), q_1, .., q_\ell)$. We require syntactically there exists two algorithms $\mathsf{SetupSeed}$ and $\mathsf{SetupPoly}$ such that Setup can be decomposed follows:
  1. $\mathsf{SetupSeed}(1^\lambda, 1^n, B) \rightarrow \mathsf{Seed}$. The $\mathsf{SetupSeed}$ algorithm outputs the seed.
  2. $\mathsf{SetupPoly}(1^\lambda, 1^n, B) \rightarrow q_1, ..., q_\ell$. The $\mathsf{SetupPoly}$ algorithm outputs $q_1, .., q_\ell$.
- $\mathsf{Eval}(\mathsf{pp}, \mathsf{Seed}) \rightarrow (h_1, ..., h_\ell)$, evaluation algorithm output a vector $(h_1, ..., h_\ell) \in \mathbb{Z}^\ell$. Here for $i \in [\ell]$, $h_i = q_i(\mathsf{Seed})$ and ℓ is the stretch of $\mathcal{F}$-$\Delta\mathsf{RG}$. Here each q_i is in $\mathcal{F}_n$.

The security and efficiency requirements are same as before.
Remark: Few remarks are in order,

1. To construct $i\mathcal{O}$ we need the stretch of $\mathcal{F}$-$\Delta\mathsf{RG}$ to be equal to $\ell = n^{1+\epsilon}$ for some constant $\epsilon > 0$.
2. Looking ahead, we will use a $\mathcal{F}$-$\Delta\mathsf{RG}$ for a function class $\mathcal{F}$, that is also the function class for a PHFE scheme.

We refer a reader for assumptions under which we can build $\Delta\mathsf{RG}$ to [JLMS19].