Indistinguishability Obfuscation Without Multilinear Maps: $i\mathcal{O}$ from LWE, Bilinear Maps, and Weak Pseudorandomness

Prabhanjan Ananth
CSAIL, MIT
prabhanjan@csail.mit.edu

Aayush Jain
UCLA
aayushjain@cs.ucla.edu

Amit Sahai
UCLA
sahai@cs.ucla.edu

November 5, 2018

Abstract

The existence of secure indistinguishability obfuscators ($i\mathcal{O}$) has far-reaching implications, significantly expanding the scope of problems amenable to cryptographic study. All known approaches to constructing iO rely on *d-linear maps* which allow the encoding of elements from a large domain, evaluating degree d polynomials on them, and testing if the output is zero. While secure *bilinear maps* are well established in cryptographic literature, the security of candidates for $d > 2$ is poorly understood.

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

At the heart of our approach is the assumption that a new *weak* pseudorandom object exists, that we call a *perturbation resilient generator* (ΔRG). Informally, a ΔRG maps n integers to m integers, and has the property that for any sufficiently short vector $a \in \mathbb{Z}^m$, all efficient adversaries must fail to distinguish the distributions ΔRG(s) and (ΔRG(s)+a), with at least some probability that is inverse polynomial in the security parameter. ΔRGs have further implementability requirements; most notably they must be computable by a family of degree-3 polynomials over $\mathbb{Z}$. We use techniques building upon the Dense Model Theorem to deal with adversaries that have nontrivial but non-overwhelming distinguishing advantage. In particular, we obtain a new security amplification theorem for functional encryption.

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

- Subexponentially secure LWE
- Bilinear Maps
- $poly(\lambda)$-secure 3-block-local PRGs
- $(1 - 1/poly(\lambda))$-secure ΔRGs

Acknowledgements

We thank Dakshita Khurana for collaboration in the initial stages of this research, for contributing to the writeup, and for countless discussions and comments supporting this work and improving the write up. Eventually, the current set of authors had to reluctantly agree to Dakshita's repeated requests to not be listed in the set of authors, and be mentioned in these acknowledgements instead. We thank Boaz Barak, Sam Hopkins and Pravesh Kothari for insights and extremely helpful suggestions about how attacks based on the Sum of Squares paradigm could impact our new assumptions on perturbation-resilient generators.

Research supported in part from a DARPA/ARL SAFEWARE award, 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. Aayush Jain is also supported by a Google PhD fellowship award in Privacy and Security. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C- 0205. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, the U.S. Government or Google.

Contents

1 Introduction

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

The work of [GGH⁺13b] gave the first mathematical candidate $i\mathcal{O}$ construction, and since then several additional candidates have been proposed and studied [GGH13a, CLT13, GGH15, CLT15, Hal15, BR14, BGK⁺14, PST14, AGIS14], [BMSZ16, CHL⁺15, BWZ14, CGH⁺15, HJ15, BGH⁺15, Hal15, CLR15, MF15, MSZ16, DGG⁺16], as well as more recently [Lin16, LV16, AS17, LT17].

Constructing $i\mathcal{O}$. Securely building $i\mathcal{O}$ remains a central challenge in cryptography. In this work, we show how to utilize new techniques to securely build $i\mathcal{O}$. Most notably, we show new ways to leverage *bilinear maps* and tools building upon the *dense model theorem* [JP14, CCL18, RTTV08] in the context of constructing $i\mathcal{O}$. Using these new tools, we show how to securely construct $i\mathcal{O}$ *without using cryptographic multilinear maps* beyond bilinear maps. We now elaborate.

Graded Encodings. All known approaches for building $i\mathcal{O}$ crucially rely on the existence of a *graded encoding scheme* [GGH13a, CLT13, GGH15], which generalizes the notion of a cryptographic mulitilinear map [BS02]. In a degree-d graded encoding scheme, it is possible to compute encodings $[x]$ of values x, such that for any degree-d polynomial f with small coefficients, given only the encodings $[x]$, it is possible to efficiently test whether $f(x) \stackrel{?}{=} 0$. For $d = 2$, this corresponds to cryptographic *bilinear* maps [BF01], for which we know well-studied constructions based on the hardness present in elliptic curve groups that admit pairing operations.

However, the situation for $d > 2$ is much more problematic. While candidate constructions of such graded encoding schemes exist [GGH13a, CLT13, GGH15], their security is poorly understood due to several known explicit attacks on certain distributions of encoded values [CHL⁺15, BWZ14, CGH⁺15, HJ15, BGH⁺15, Hal15, CLR15, MF15, MSZ16].

Due to a recent line of work [Lin16, LV16, AS17, Lin17a, LT17], based additionally on the subexponential hardness of 3-blockwise-local PRGs and the Learning with Errors assumption (LWE), it is known that achieving security for $d = 3$ is already enough to construct $i\mathcal{O}$. Unfortunately, however, the security of candidate graded encodings supporting $d = 3$ seems no better understood than the general $d > 2$ case.

The state of our understanding strongly motivates the following central question:

Can we build $i\mathcal{O}$ without cryptographic multilinear maps?

Our Goals and Assumptions. Because subexponentially-secure[1] LWE and cryptographic bilinear maps have a long history of security, we consider using LWE or (generically secure) cryptographic bilinear maps as standard. This is in contrast with existing candidate multilinear maps, where both constructions [GGH13a, CLT13, GGH15] and standing security models [MSZ16] are complex and therefore difficult to understand and analyze.

Therefore, beyond using standard tools, our goal is to reduce the security of $i\mathcal{O}$ to problems that are simple to state, and where the underlying mathematics has a long history of study.

More specifically, we will show how to build $i\mathcal{O}$ from LWE, bilinear maps, and novel *weakly pseudorandom* objects that we call *perturbation-resilient generators* (ΔRG), that can be implemented with low degree polynomials over $\mathbb{Z}$. Informally speaking, a perturbation-resilient generator is a generator ΔRG such that the distributions $\Delta RG(s)$ and $(\Delta RG(s) + a)$ are *somewhat* hard to distinguish as long as the perturbation a is relatively small. We describe ΔRGs in more detail below in our technical overview, where we will also discuss why we conjecture that they exist (even in light of [BBKK17, LV17] and follow-up work [BHJ+18]).

Hardness of polynomials over the reals. As we elaborate in the technical overview in more detail, the security of our new perturbation-resilient generators crucially relies on the hardness of solving certain expanding systems of degree-3 polynomial equations over the reals. Indeed, solving systems of polynomials over the reals has been studied by mathematicians, scientists, and engineers for hundreds of years. This is precisely why we are taking this approach: we want to relate $i\mathcal{O}$ to simple-to-state problems related to areas of mathematics with long histories of study. Aside from that, our work also fundamentally diversifies the kinds of assumptions from which $i\mathcal{O}$ can be constructed.

A key innovation of our work is that we can work with perturbation-resilient generators where the security property only asks that efficient adversaries fail to distinguish between two distributions with at least some $1/poly(\lambda)$ probability – i.e. some fixed inverse polynomial in the security parameter. Thus, even if an efficient adversary correctly predicts whether a sample comes from the $\Delta RG(s)$ distribution or the $(\Delta RG(s) + a)$ distribution 99% of the time, our $i\mathcal{O}$ scheme will still be secure.

We stress that the new object (ΔRG) that we introduce is quite simple – indeed crucially it is implementable by degree-3 polynomials over $\mathbb{Z}$. In Section 5.2, we describe specific candidates for such ΔRGs suggested in follow-up work by [BHJ+18] that were inspired by the hardness of RANDOM 3-SAT. This simplicity stands in notable contrast to candidate multilinear maps. More generally, our work motivates the further cryptanalytic study of simple pseudorandom objects.

We will also only need to use similarly weakened[2] forms of 3-blockwise-local PRGs [LT17]. In followup work, our approach has been generalized to both remove the need for generic model security for bilinear maps and using c-block-local PRGs for any constant c [JS18, LM18], and to extend our results to perturbation-resilient generators implementable by polynomials of any *constant*-degree [JS18].

In particular, we obtain the following:

[1]For constructing $i\mathcal{O}$, we will always need to use subexponentially secure assumptions. For brevity in the introduction, we will omit mention of this except in theorem statements.

[2]There will be a tradeoff between how much we can weaken the indistinguishability requirements of the ΔRG and the 3-block-local PRG.

Theorem 1 (Informal)**.** *For every constants $c, \varepsilon > 0$, there is a construction of indistinguishability obfuscation for all polynomial-sized circuits from,*

- $\left(1 - \frac{1}{\lambda^c}\right)$*-secure perturbation-resilient generators of stretch $n^{1+\varepsilon}$ on seeds of length n (see Section 5), with security against sub-exponential size adversaries.*
- $\frac{1}{2\lambda^c}$*-secure three-block-local pseudorandom generators [LT17] of stretch $n^{1+\varepsilon}$ on seeds of length n, with security against sub-exponential size adversaries.*
- *Learning with errors secure against sub-exponential size adversaries.*
- *Assumptions on bilinear maps secure against sub-exponential size adversaries (that hold unconditionally in the generic bilinear map model).*

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

Along the way to proving the result above, we also obtain an amplification theorem for functional encryption:

Theorem 2 (Informal)**.** *Assuming there exists a constant $c > 0$ and there exists:*

- $(1 - 1/\lambda^c)-$*secure sublinear FE scheme for polynomial size circuits of depth λ.*
- *Learning with errors secure against sub-exponential size adversaries.*

There exists a sublinear secret key FE scheme for polynomial size circuits of depth λ with $\mathsf{negl}(\lambda)$*-security.*

Note that if we assume underlying FE scheme to be secure against subexponential size, then the resulting scheme satisfies subexponential security. Please refer Section 14 for a complete formulation. The amplification theorem above relies only on subexponential LWE, and no new assumptions.

The nature of our new assumptions for $i\mathcal{O}$. The key new assumption introduced by our work is the existence of perturbation-resilient generators implementable by degree-3 polynomials in a certain way. Here, we elaborate on what this assumption means, by breaking it up into two assumptions, each of which is simple to state:

Weak LWE with degree-3 leakage. This assumption says that there exists distributions χ over the integers and Q over families of multilinear degree-3 polynomials such that the following two distributions are weakly indistinguishable, meaning that no efficient adversary can correctly identify the distribution from which a sample arose with probability above $\frac{1}{2} + 1/\lambda$.

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

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

along with

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

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

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

along with

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

We can think of the polynomials $q_k(\boldsymbol{e}, \boldsymbol{y}, \boldsymbol{z})$ as "leaking" some information about the LWE errors e_i. The assumption above states that such leakage provides only a limited advantage to the adversary. Critically, the fact that there are $n^2 > n^{1+\epsilon}$ quadratic monomials involving just $\boldsymbol{y}$ and $\boldsymbol{z}$ above, which are not used in the LWE samples at all, is crucial to avoiding linearization attacks over $\mathbb{Z}_p$ in the spirit of Arora-Ge [AG11]. For more discussion of the security of the above assumption, see [BHJ+18].

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

Weak Perturbation-Resilience Over the Reals. The second assumption is that there exists polynomials that for the same parameters above the following two distributions are *very* weakly indistinguishable. By very weakly indistinguishability we mean that no efficient adversary can correctly identify the distribution from which a sample arose with probability beyond $1 - 1/\lambda$. Let $\delta_i \in \mathbb{Z}$ be such that $|\delta_i| < B(\lambda, n)$ for some polynomial B and $i \in [n^{1+\epsilon}]$:

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

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

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

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

The above assumptions are combined into a single assumption that is formally stated in Section 5.2.

2 Technical Overview

We begin with a very high-level overview of our techniques.

The story so far. Prior work, culminating in the most recent works of [AS17, Lin17a, LT17] showed us that the powerful primitive of indistinguishability obfuscation can be based on trilinear maps and (sub-exponential) 3-block-local pseudorandom generators. Importantly for us, these works also (implicitly) demonstrate that in order to achieve indistinguishability obfuscation, it suffices to construct (sub-exponentially secure) secret-key sublinear FE for cubic polynomials, satisfying semi-functional security. Unfortunately, these prior approaches necessarily relied on multilinear maps with degree at least 3 to build such a cubic FE scheme.

That is because intuitively such a cubic FE scheme guarantees a way to evaluate a cubic polynomial on encrypted inputs without revealing any information about the input except the evaluation of the polynomial. In other words, such a scheme provides a way to output the decryption of a degree-3 polynomial evaluated "homomorphically" on encoded inputs. However, we seek to accomplish this without the use of degree-3 maps.

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

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

A reader familiar with GSW will observe that this approach already runs into major hurdles. The first problem is that brute-force computing the message M also reveals the error e to a potential adversary, which is problematic when we try to invoke the semantic security of GSW. In fact, recent work shows how knowledge of such error can be used to build devastating attacks [Agr17]. We will crucially deal with this issue, but before we tackle this, let us first consider how we can force the adversary to obtain only inner products $\langle \mathsf{GSW.sk}, \mathsf{GSW.CT} \rangle$ where the messages correspond to cubic computations that the adversary is allowed to obtain.

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

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

Constructing Three-Restricted FE. Before getting to 3 restricted FE, let's first recap how secret key quadratic functional encryption schemes [AS17, Lin17a] work at a high level. Let's say that the encryptor wants to encrypt $\boldsymbol{y}, \boldsymbol{z} \in \mathbb{Z}_{\mathbf{p}}^n$. The master secret key consists of two secret random vectors $\boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathbb{Z}_{\mathbf{p}}^n$ that are used for enforcement of computations done on $\boldsymbol{y}$ and $\boldsymbol{z}$ respectively. The idea is that the encryptor encodes $\boldsymbol{y}$ and $\boldsymbol{\beta}$ using some randomness r, and similarly encodes $\boldsymbol{z}$ and $\boldsymbol{\gamma}$ together as well. These encodings are created using bilinear maps in one of the two base groups. These encodings are constructed so that the decryptor can compute an encoding of $[g(\boldsymbol{y}, \boldsymbol{z}) - r g(\boldsymbol{\beta}, \boldsymbol{\gamma})]_t$ in the target group for *any* quadratic function g. The function key for the given function f is constructed in such a manner that it allows the decryptor to compute the encoding

$[rf(\boldsymbol{\beta},\boldsymbol{\gamma})]_t$ in the target group. Thus the output $[f(\boldsymbol{y},\boldsymbol{z})]_t$ can be recovered in the exponent by computing the sum of $[rf(\boldsymbol{\beta},\boldsymbol{\gamma})]_t$ and $[f(\boldsymbol{y},\boldsymbol{z}) - rf(\boldsymbol{\beta},\boldsymbol{\gamma})]_t$ in the exponent. As long as $f(\boldsymbol{y},\boldsymbol{z})$ is polynomially small, this value can then be recovered efficiently.

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

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

Now observe that,

$$\begin{aligned}&\boldsymbol{x}[i] \cdot \big(\boldsymbol{y}[j]\boldsymbol{z}[k] - r\boldsymbol{\beta}[j]\boldsymbol{\gamma}[k]\big) + r(\boldsymbol{x}[i] - \boldsymbol{\alpha}[i]) \cdot \boldsymbol{\beta}[j]\boldsymbol{\gamma}[k] \\ =&\boldsymbol{x}[i]\boldsymbol{y}[j]\boldsymbol{z}[k] - r\boldsymbol{\alpha}[i]\boldsymbol{\beta}[j]\boldsymbol{\gamma}[k]\end{aligned}$$

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

In Section 11, we provide details of our 3-restricted FE; specifically, we define a notion of semi-functional security [AS17] (variant of function-hiding) associated with a three-restricted FE scheme. Once we have such a restricted FE, making the leap to cubic FE would require us to also keep the public encoding hidden. Therefore, it is not clear whether we have achieved anything meaningful yet.

Applying Three-Restricted FE. One way that we can hope to protect or hide the input that goes into the public component of the 3-restricted FE, is to let this component itself be a GSW-based fully homomorphic encryption of the input. We can then rely on 3-restricted FE to homomorphically evaluate the cubic function itself and obtain a GSW encryption of the output of cubic evaluation. Note, however, that releasing such a GSW encryption by itself is useless, because it does not allow even an honest evaluator to recover the output of cubic evaluation.

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

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

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

- How should we generate such noise? A natural idea is to rely a pseudorandom generator that can be computed via quadratic operations only. However, this is exactly the reason why previous approaches from the literature could not rely on bilinear maps – in fact, the recent works of [LV17, BBKK17] showed that such PRGs are quite difficult to construct. To overcome this problem, we introduce and rely on a very weak variant of a pseudorandom object, that instead of guaranteeing pseudorandomness, only guarantees perturbation resilience. Furthermore, we will implement this object with degree-3 polynomials. We will soon explain this object in more detail.

- For an honest evaluator to recover μ by iterating over all possible values of $\mathsf{out} = \mu\lfloor\frac{q}{2}\rfloor + e$, we crucially require the added noise be sampled from a polynomial-sized domain. But such noise appears to be insufficient for security, in particular, an adversary would have advantage at least $\frac{1}{poly(\lambda)}$ in distinguishing a message with added noise from a message without noise. Another key technical contribution of our work is to find a way to *amplify security*, via tools inspired by the dense model theorem. In the next two bullets, we describe these ideas in additional detail.

The challenge of constructing degree-3 pseudorandomness: a barrier at degree 2. As we've outlined above, we need a way to create pseudorandomness to (at least partially) hide noise values. The most straightforward way to do this would be to build a degree-2 pseudorandom generator (PRG) whose output is indistinguishable from some nice m-dimensional distribution, like a discrete gaussian. Intuitively, if such a degree-2 object existed, a bilinear map would be sufficient to implement it. However, the works of [BBKK17, LV17] showed that there are fundamental barriers to constructing such PRGs due to attacks arising from the Sum of Squares paradigm. Because we will propose a direction to overcome this barrier, we now review how these attacks work at a high level.

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

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

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

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

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

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

However, if the $z_i = r_i$, then the entries of the matrix M arise from a "random walk," and thus intuitively, the matrix M should behave a lot like a random matrix. However a random matrix has bounded eigenvalues, and thus we expect that there should not exist any x of bounded norm such that $x^\top M x$ is large. Indeed, this intuition can be made formal and gives rise to actual attacks on many degree-2 PRGs [BBKK17, LV17]. The attack above was generalized further in a followup work to this paper [BHJ+18], showing that several families of degree-2 pseudorandom objects cannot exist. While there are still potential caveats to known degree-2 attacks, we propose a different, more conservative, way forward:

Perturbation-Resilient Generators (ΔRG). We observe that even though the most natural way to "drown out" the GSW error term above is by adding some nice noise distribution, all we actually need is something we will call a perturbation-resilient generator (ΔRG): Informally speaking, we want that for every polynomial bound $B(\lambda)$, there should exist a low-degree[3] ΔRG using polynomially bounded seeds and coefficients, such that for any perturbation vector $a \in [-B, B]^m$, it should be true that all efficient adversaries must fail to distinguish between the distributions $\Delta RG(x)$ and $(\Delta RG(x) + a)$ with probability at least $1/poly(\lambda)$, which is a fixed inverse polynomial in the security parameter. We stress again that we are not seeking a ΔRG with standard negligible security, but only some low level of security. Indeed, even if an efficient adversary could distinguish between $\Delta RG(x)$ and $(\Delta RG(x) + a)$ with probability $1 - 1/poly(\lambda)$, but still fail to distinguish on at least $1/poly(\lambda)$ probability mass, our approach will succeed due to amplification (see below).

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

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

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

low-degree polynomials over the reals that are hard to solve. We refer the reader to [BHJ+18] for further discussion on this topic.

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

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

Encrypting $\boldsymbol{e}$ as an LWE-style error. Instead, we sample a secret $\boldsymbol{s} \in \mathbb{Z}_{\mathbf{p}}^{\mathsf{d}}$ where d is some polynomial in the security parameter. We also sample vectors $\boldsymbol{a_i} \leftarrow \mathbb{Z}_{\mathbf{p}}^{\mathsf{d}}$ for $i \in [n]$. Then we compute $r_i = \langle \boldsymbol{a_i}, \boldsymbol{s} \rangle + e_i$. Let $\boldsymbol{w_i} = (\boldsymbol{a_i}, r_i)$ for $i \in [n]$. Thus we have encrypted $\boldsymbol{e}$ using the secret $\boldsymbol{s}$. Now to implement degree-3 randomness generator we consider the polynomial:

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

This polynomial can be re-written as:

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

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

For this approach to be secure, intuitively we want that $\boldsymbol{e}$ is sampled from an "error" distribution with respect to which the LWE assumption holds. (For simplicity, we can think of $\boldsymbol{y}$ and $\boldsymbol{z}$ also being sampled from such a distribution.) The security of our ΔRG would then rely on a variant of the LWE assumption. Experience teaches that one should be cautious when considering the security of variants of LWE, and our case is no exception. This variant was studied in a follow-up work of [BHJ+18], where several unsuccessful attacks were considered. We briefly review one of these now. The most common source of devastating attacks to LWE variants is linearization. However, a key barrier to such attacks in our setting is the fact that the LWE-based public values $\boldsymbol{w}_i$ contain *no information whatsoever* about $\boldsymbol{y}$ and $\boldsymbol{z}$. Thus, over $\mathbb{Z}_{\mathbf{p}}$, we would obtain a set of roughly $n^{1+\epsilon}$ quadratic equations in $\boldsymbol{y} \otimes \boldsymbol{s}$ and $\boldsymbol{z}$, but crucially with *large* coefficients in $\mathbb{Z}_{\mathbf{p}}$. These large coefficients would arise from the fact that r_i and $\boldsymbol{a}_i$ are large values. Such systems, called MQ systems, have been widely studied cryptanalytically and are widely believed to be hard to solve [Wol02, KS99] in general. We again refer the reader to [BHJ+18] for further discussion. Specific candidates for the degree-3 polynomials q_ℓ above, inspired by the hardness of RANDOM 3-SAT and suggested by [BHJ+18], are also given in Section 5.2.

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

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

How should we amplify security? An initial idea is to implement a direct-product type theorem for functional encryption. However, a simple XOR trick does not suffice: since we are trying to amplify security of a complex primitive like FE while retaining correctness, we will additionally need to rely on a special kind of secure computation. More precisely, we will use (subexponentially secure) n-out-of-n threshold fully homomorphic encryption (TFHE [MW16, BGG$^+$17]), that is known to exist based on LWE [Reg05]. Recall that such a threshold (public key) fully homomorphic encryption scheme allows to encrypt a ciphertext in such a way that all secret key holders can *partially* decrypt the ciphertext, and then can recover the plaintext by combining these partial decryptions. However, any coalition of secret key holders of size at most $n-1$ learns no information about the message.

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

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

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

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

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

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

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

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

Related Works. Concurrently and independently, the works of [LM18, Agr18] also considered the question of building iO without using multilinear maps. Later in revisions to both works [LM18, Agr18], both works used and built upon our work in order to make use of variants of our assumptions.

3 Reader's Guide

In Section 4, we recall some useful preliminaries for the rest of the paper. In Section 5, we define the notion of a perturbation resilient generator $\Delta\mathsf{RG}$. In Section 6, we define the notion of a tempered cubic encoding scheme. In Section 7, we define the notion of a three restricted FE scheme. In Section 8, we define the notion of semi-functional FE for degree-3 polynomials. In Section 9 we define the notion of semi-functional sublinear FE scheme for circuits.

We present our construction in following steps. In Section 5, we give candidate constructions of a perturbation resilient generator $\Delta\mathsf{RG}$. In Section 10.1, we construct a tempered cubic encoding scheme. In Section 11, we construct a three restricted FE scheme from bilinear maps. In Section 12, we construct the notion of a semi-functional FE scheme for cubic polynomials. In Section 13, we construct the notion of sublinear semi-functional FE scheme for circuits. In Section 14, we present our amplification theorem for semi functional FE for circuits. Finally in Section 15, we stitch all these results to construct $i\mathcal{O}$. We present a diagrammatic view of construction of $i\mathcal{O}$ in Figure 3.

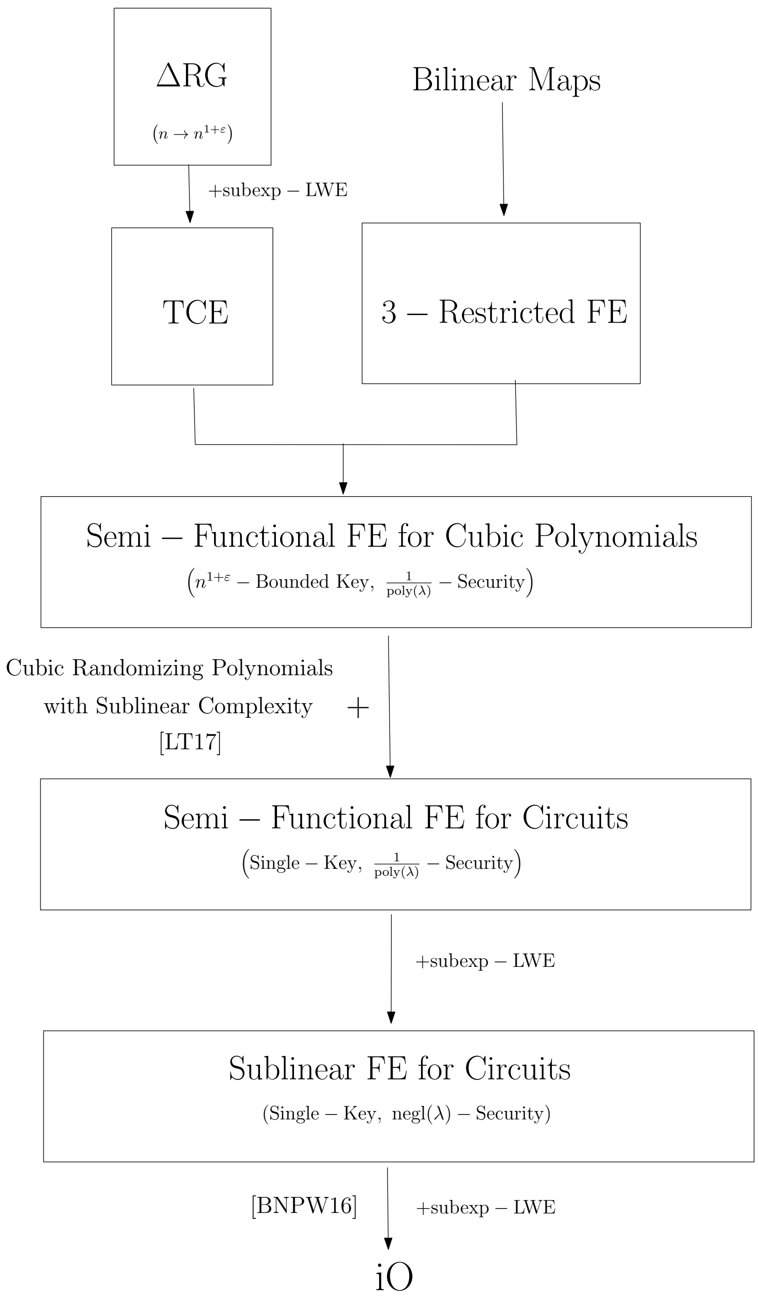

Figure 1: Steps involved in the construction of $i\mathcal{O}$.

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

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

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

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

4 Preliminaries

We denote the security parameter by λ. For a distribution X we denote by $x \leftarrow X$ the process of sampling a value x from the distribution X. Similarly, for a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling x from the uniform distribution over $\mathcal{X}$. 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 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 $(s, \epsilon)-$ indistinguishable if for every probabilistic polynomial time adversary $\mathcal{A}$ of size bounded by $O(s)$ it holds that: $\left|\Pr_{x\leftarrow\mathcal{X}_\lambda}[\mathcal{A}(1^\lambda, x) = 1] - \Pr_{y\leftarrow\mathcal{Y}_\lambda}[\mathcal{A}(1^\lambda, y) = 1]\right| \leq \epsilon$ for every sufficiently large $\lambda \in \mathbb{N}$. We drop the notation (s, ϵ) from (s, ϵ)-indistinguishable when s is polynomial and ϵ is negligible. By sub-exponential indistinguishability, we mean that there exists some constant $c > 0$, such that, for any adversary of polynomial size the distinguishing advantage is bounded by $\epsilon = O(2^{-\lambda^c})$.

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

Definition 1 (Distinguishing Gap)**.** *For any adversary $\mathcal{A}$ and two distributions $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$, define $\mathcal{A}$'s distinguishing gap in distinguishing these distributions to be* $|\Pr_{x\leftarrow\mathcal{X}_\lambda}[\mathcal{A}(1^\lambda, x) = 1] - \Pr_{y\leftarrow\mathcal{Y}_\lambda}[\mathcal{A}(1^\lambda, y) = 1]|$

Notions for Security Amplification. We define some preliminaries that will be useful for the security amplification theorem. We recall the definition of a measure.

Definition 2. *A measure is a function $\mathcal{M} : \{0,1\}^k \to [0,1]$. The size of a measure is $|\mathcal{M}| = \Sigma_{x\in\{0,1\}^k}\mathcal{M}(x)$. The density of a measure, $\mu(\mathcal{M}) = |\mathcal{M}|2^{-k}$*

Each measure $\mathcal{M}$ induces a probability distribution $\mathcal{D}_\mathcal{M}$.

Definition 3. *Let $\mathcal{M} : \{0,1\}^k \to [0,1]$ be a measure. The distribution defined by measure $\mathcal{M}$ (denoted by $\mathcal{D}_\mathcal{M}$) is a distribution over $\{0,1\}^k$, where for every $x \in \{0,1\}^k$, $\Pr_{X \leftarrow \mathcal{D}_\mathcal{M}}[X = x] = \mathcal{M}(x)/|\mathcal{M}|$.*

We will consider a scaled version $\mathcal{M}_c$ of a measure $\mathcal{M}$ for a constant $0 < c < 1$ defined as $\mathcal{M}_c = c\mathcal{M}$. Note that $\mathcal{M}_c$ induces the same distribution as $\mathcal{M}$.

4.1 Indistinguishability Obfuscation (iO)

The notion of indistinguishability obfuscation (iO), first conceived by Barak et al. [BGI+01], guarantees that the obfuscation of two circuits are computationally indistinguishable as long as they both are equivalent circuits, i.e., the output of both the circuits are the same on every input. Formally,

Definition 4 (Indistinguishability Obfuscator (iO) for Circuits)**.** *A uniform PPT algorithm $i\mathcal{O}$ is called an indistinguishability obfuscator for a circuit family $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$, where $\mathcal{C}_\lambda$ consists of circuits C of the form $C : \{0,1\}^n \to \{0,1\}$ with $n = n(\lambda)$, if the following holds:*

- **Completeness:** *For every $\lambda \in \mathbb{N}$, every $C \in \mathcal{C}_\lambda$, every input $x \in \{0,1\}^n$, we have that*

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

- **Indistinguishability:** *For any PPT distinguisher D, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds: for all sufficiently large $\lambda \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\lambda$ such that $C_0(x) = C_1(x)$ for all inputs $x \in \{0,1\}^n$ and $|C_0| = |C_1|$, we have:*

$$\Big| \mathsf{Pr}\left[D(\lambda, i\mathcal{O}(\lambda, C_0)) = 1\right] - \mathsf{Pr}[D(\lambda, i\mathcal{O}(\lambda, C_1)) = 1] \Big| \leq \mathsf{negl}(\lambda)$$

- **Polynomial Slowdown***: For every $\lambda \in \mathbb{N}$, every $C \in \mathcal{C}_\lambda$, we have that $|i\mathcal{O}(\lambda, C)| = \text{poly}(\lambda, C)$.*

4.2 Slotted Encodings

We define the notion of slotted encodings SE, a variant of composite order multilinear maps, introduced by [AS17]. Unlike [AS17], we consider a relaxation of slotted encodings associated with 5 slots and in the bilinear map setting. A slotted encoding scheme consists of the following algorithms:

- **Secret Key Generation,** $\mathsf{Gen}(1^\lambda)$: It outputs secret encoding key SEsp, a pairing function e along with a prime $\mathbf{p} > 2^\lambda$ and public parameters PP. We assume that e, PP and $\mathbf{p}$ are implicitly given to all the algorithms below.
- **Encoding,** $\mathsf{Encode}(\mathsf{SEsp}, a_1, .., a_4, a_5, l \in \{1,2\})$: In addition to secret key SEsp, it takes as input $a_1, .., a_4, a_5 \in \mathbb{Z}_\mathbf{p}$ and a level $l \in \{1,2\}$. It outputs an encoding $[a_1, a_2, a_3, a_4, a_5]_l$.
- **Multiply,** $e([a_1, a_2, a_3, a_4, a_5]_1, [b_1, b_2, b_3, b_4, b_5]_2) = [\Sigma_i a_i b_i]_T$. The pairing operation takes as input an encoding of $\boldsymbol{a}$ at level 1 and $\boldsymbol{b}$ at level 2 and it outputs an encoding of $\Sigma_i a_i b_i$ at level T. We require the set $G_T = \{[a]_T | a \in \mathbb{Z}_\mathbf{p}\}$ to form an additive group of order $\mathbf{p}$.

- **Addition at the top level T,** Given $[a]_T$ and $[b]_T$, the operation '$+$' computes $[a+b]_T = [a]_T + [b]_T$.

- **Encoding at level T,** Given $a \in \mathbb{Z}_\mathbf{p}$ and PP, $\mathsf{EncodeT}(\cdot)$ is an efficiently computable isomorphism that maps $a \in G$ to $[a]_T \in G_T$.

- **Zero test at all three levels $\mathsf{ZTest}(u, l)$:** The zero-test algorithm takes an element u at level $l \in \{1, 2, T\}$ and checks if $u = [0, 0, 0, 0, 0]_l$ if $l \in \{1, 2\}$. Otherwise it checks that $u = [0]_T$.

Remark 1. *The algorithms for addition and multiplication suggests what polynomials can be evaluated on the encodings. Given level 1 and level 2 encodings one can compute an encoding of a scaled inner product of the encoded element vectors at level T. At level 1 and level 2, we can only add encoded vectors component wise.*

Security: Since we prove security in the generic model, we require generic security from our slotted encodings at level 1 and 2 when SEsp is kept hidden from the adversary.

4.2.1 Generic Bilinear Group Model

We describe the generic bilinear group model [BBG05] tailored to the slotted asymmetric setting. This model is parameterized by slotted encodings SE, which encodes five dimensional vectors over a prime field $\mathbb{Z}_\mathbf{p}$ at level 1 and 2, and it encodes element from $\mathbb{Z}_\mathbf{p}$ at the target level T. The encodings are done over level 1, 2 and the target T. The multiplication operation computes encoding at level T. The (unbounded) adversary in this model has access to an oracle $\mathcal{O}$. Initially, the adversary is handed out *handles* (sampled uniformly at random) instead of being handed out actual encodings. A handle is an element in a ring $\mathbb{Z}$ of order $\mathbf{p}$. The oracle $\mathcal{O}$ maintains a list L consisting of tuples $(e, \mathbf{Y}[e], u)$, where e is the handle issued, $\mathbf{Y}[e]$ is the formal expression associated with e and e is associated with encoding at level $u \in \{1, 2, T\}$.

The adversary is allowed to submit the following types of queries to the oracle:

- *Addition/ Subtraction*: The adversary submits (e_1, u_1) and (e_2, u_2) along with the operation '+'(or '-') to the oracle where $u_1, u_2 \in \{1, 2, T\}$. If $u_1 \neq u_2$ or If there is no tuple associated with either e_1 or e_2, the oracle sends $\perp$ back to the adversary. Otherwise, it replies according to the following cases:
  - $u_1 \in \{1, 2\}$: In this case it locates $(e_1, p_{1,e_1}, p_{2,e_1}, p_{3,e_1}, p_{4,e_1}, p_{5,e_1}, u_1)$ and $(e_2, p_{1,e_2}, p_{2,e_2}, p_{3,e_2}, p_{4,e_2}, p_{5,e_2}, u_2)$. It creates a new handle e' (sampled uniformly at random from $\mathcal{R}$) and appends $(e', p_{1,e_1} + p_{1,e_2}, p_{2,e_1} + p_{2,e_2}, p_{3,e_1} + p_{3,e_2}, p_{4,e_1} + p_{4,e_2}, p_{5,e_1} + p_{5,e_2}, u_1)$ to the list (in case of subtractions the polynomials are subtracted). It outputs e' to the adversary.
  - $u_1 = u_2 = T$: In this case the adversary locates the tuples (e_1, p_{e_1}, u_1) and (e_2, p_{e_2}, u_2). It creates a new handle e' (sampled uniformly at random from $\mathcal{R}$) and appends $(e', p_{e_1} + p_{e_2}, u_1)$ (or $(e', p_{e_1} - p_{e_2}, u_1)$) to the list. The oracle sends e' to the adversary.

- *Multiplication*: The adversary submits (e_1, u_1) and (e_2, u_2) to the oracle. If there is no tuple associated with either e_1 or e_2, the oracle sends $\perp$ back to the adversary. If $u_1 = u_2$, $u_1 = T$ or $u_2 = T$, the oracle outputs $\perp$. Otherwise, it locates the tuples $(e_1, p_{1,e_1}, p_{2,e_1}, p_{3,e_1}, p_{4,e_1}, p_{5,e_1}, u_1)$

and $(e_2, p_{1,e_2}, p_{2,e_2}, p_{3,e_2}, p_{4,e_2}, p_{5,e_2}, u_2)$. It creates a new handle e' (sampled uniformly at random from $\mathcal{R}$) and appends $(e', \Sigma_{j\in[5]} p_{j,e_1} * p_{j,e_2}, T)$ to the list.

- *Zero Test*: The adversary submits element (e_1, u_1) to the oracle. If there is no tuple associated to e_1 it outputs $\perp$. Otherwise, if $u_1 = 1$ or $u_1 = 2$, it locates the tuples $(e_1, p_{1,e_1}, p_{2,e_1}, p_{3,e_1}, p_{4,e_1}, p_{5,e_1}, u_1)$. It outputs 1 if $p_{j,e_1} = 0$ for all $j \in [5]$ otherwise it outputs 0. If $u_1 = T$, it locates the tuples (e_1, p_{1,e_1}, u_1). It outputs 1 if $p_{1,e_1} = 0$, otherwise it outputs 0.

Inspired from [Fre10], in [AS17] it was shown how to construct degree-2 slotted encoding scheme in the bilinear generic group model. We remark here that the procedure given in [AS17], was instantiated for higher degrees using graded encoding schemes. However, it can be instantiated for degree two using bilinear maps. Thus, we have the following theorem.

Theorem 3 (Imported from [AS17])**.** *There exists a construction of degree 2 slotted encoding scheme in the generic bilinear group model.*

4.3 Threshold Leveled Fully Homomorphic Encryption

The following definition of threshold homomorphic encryption is adapted from [MW16, BGG+17]. A threshold homomorphic encryption scheme is a tuple of PPT algorithms $\mathsf{TFHE} = (\mathsf{TFHE.Setup}, \mathsf{TFHE.Enc}, \mathsf{TFHE.Eval}, \mathsf{TFHE.PartDec}, \mathsf{TFHE.FinDec})$ satisfying the following specifications:

- **Setup,** $\mathsf{Setup}(1^\lambda, 1^d, 1^n)$: It takes as input the security parameter λ, a circuit depth d, and the number of parties n. It outputs a public key fpk and secret key shares $\mathsf{fsk}_1, \ldots, \mathsf{fsk}_n$.

- **Encryption,** $\mathsf{Enc}(\mathsf{fpk}, \mu)$: It takes as input a public key fpk and a single bit plaintext $\mu \in \{0, 1\}$ and outputs a ciphertext CT.

- **Evaluation,** $\mathsf{Eval}(C, \mathsf{CT}_1, \ldots, \mathsf{CT}_k)$: It takes as input a boolean circuit $C\colon \{0,1\}^k \to \{0,1\} \in \mathcal{C}_\lambda$ of depth $\leq d$ and ciphertexts $\mathsf{CT}_1, \ldots, \mathsf{CT}_k$ encrypted under the same public key. It outputs an evaluation ciphertext CT. We shall assume that the ciphertext also contains fpk.

- **Partial Decryption,** $p_i \leftarrow \mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{CT})$: It takes as input a secret key share fsk_i and a ciphertext CT. It outputs a partial decryption p_i related to the party i.

- **Final Decyrption,** $\mathsf{FinDec}(B)$: It is a deterministic algorithm that takes as input a set $B = \{p_i\}_{i\in[n]}$. It outputs a plaintext $\widehat{\mu} \in \{0, 1, \perp\}$.

Definition 5 (TFHE)**.** *A* TFHE *scheme is required to satisfy the following properties for all parameters* $(\mathsf{fpk}, \mathsf{fsk}_1, \ldots, \mathsf{fsk}_N) \leftarrow \mathsf{Setup}(1^\lambda, 1^d, 1^n)$*, any plaintexts* $\mu_1, \ldots, \mu_k \in \{0, 1\}$*, and any boolean circuit* $C\colon \{0,1\}^k \to \{0,1\} \in \mathcal{C}_\lambda$ *of depth* $\leq d$*.*

Correctness of Encryption. *Let* $\mathsf{CT} = \mathsf{Enc}(\mathsf{fpk}, \mu_1)$ *and* $B = \{\mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{CT})\}_{i\in[n]}$*. With all but negligible probability in* λ *over the coins of* Setup*,* Enc*, and* $\mathsf{PartDec}$*,* $\mathsf{FinDec}(B) = \mu_1$*.*

Correctness of Evaluation. *Let* $\mathsf{CT}_i = \mathsf{Enc}(\mathsf{fpk}, \mu_i)$ *for* $1 \leq i \leq k$*,* $\hat{\mathsf{CT}} = \mathsf{Eval}(C, \mathsf{CT}_1, \ldots, \mathsf{CT}_k)$*, and* $B = \{\mathsf{PartDec}(\mathsf{fsk}_i, \hat{\mathsf{CT}},)\}_{i\in[n]}$*. With all but negligible probability in* λ *over the coins of* Setup*,* Enc*, and* $\mathsf{PartDec}$*,* $\mathsf{FinDec}(B) = C(\mu_1, \ldots, \mu_k)$*.*

Compactness of Ciphertexts. *There exists a polynomial, poly, such that the size of circuit computing* $\mathsf{CT} \leftarrow \mathsf{TFHE.Enc}(\mathsf{fpk}, m)$ *is bounded by* $poly(\lambda, d, |m|)$*.*

Compactness of Partial Decryption Keys. *There exists a polynomial, poly, such that* $|\mathsf{fsk}_i| \leq poly(\lambda, d)$ *for any index* $i \in [n]$ *generated from the setup algorithm of* TFHE*.*

$(\mathsf{size}, \mathsf{adv})$**-Semantic Security of Encryption.** *Any adversary* $\mathcal{A}$ *of size* $O(\mathsf{size})$*, has advantage bounded by* adv *as a function of* λ *over the coins of all the algorithms in the following game:*

1. *Run* $\mathsf{Setup}(1^\lambda, 1^d, 1^n) \to (\mathsf{fpk}, \mathsf{fsk}_1, .., \mathsf{fsk}_n)$*. The adversary is given* fpk*.*
2. *The adversary outputs a set* $S \subset [n]$ *of size* $n-1$*.*
3. *The adversary receives* $\{\mathsf{fsk}_i\}_{i \in S}$ *along with* $\mathsf{Enc}(\mathsf{fpk}, b) \to \mathsf{CT}$ *for a random* $b \in \{0,1\}$*.*
4. *The adversary outputs* b' *and wins if* $b = b'$*.*

Simulation Security. *Let* $\mathsf{CT}_i = \mathsf{Enc}(\mathsf{fpk}, \mu_i)$ *for* $1 \leq i \leq k$*,* $\hat{\mathsf{CT}} = \mathsf{Eval}(C, \mathsf{CT}_1, \ldots, \mathsf{CT}_k)$*, and* $p_i = \mathsf{PartDec}(\mathsf{fsk}_i, \hat{\mathsf{CT}},)$ *for all* $i \in [n]$*. There exists a PPT algorithm* Sim *such that for any subset* S *of the form* $[n] \setminus i^*$*,* $\mathsf{Sim}(\hat{\mathsf{CT}}, \{\mathsf{fsk}\}_S, C(\mu_1, .., \mu_k)) \to p'_{i^*}$ *the following distributions are statistically close (with statistical distance* $2^{-\lambda^c}$ *for some constant* $c > 0$*):*

$$(p_i, \mathsf{fpk}, \mathsf{CT}_1, .., \mathsf{CT}_k, \{\mathsf{fsk}_i\}_{i \in [n]}) \approx (p'_{i^*}, \mathsf{fpk}, \mathsf{CT}_1, .., \mathsf{CT}_k, \{\mathsf{fsk}_i\}_{i \in [n]}).$$

4.4 Useful Lemmas for Security Amplification

We first import the following theorem from [MT10].

Theorem 4 (Imported Theorem [MT10])**.** *Let* $E : \{0,1\}^n \to \mathcal{X}$ *and* $F : \{0,1\}^m \to \mathcal{X}$ *be two functions, and let* $\epsilon, \gamma \in (0,1)$ *and* $s > 0$ *be given. If for all distinguishers* $\mathcal{A}$ *with size* s *we have*

$$|\Pr_{x \leftarrow \{0,1\}^n}[\mathcal{A}(E(x)) = 1] - \Pr_{y \leftarrow \{0,1\}^m}[\mathcal{A}(F(y)) = 1]| \leq \epsilon$$

Then there exist two measures $\mathcal{M}_0$ *(on* $\{0,1\}^n$*) and* $\mathcal{M}_1$ *(on* $\{0,1\}^n$*) that depend on* γ, s *such that:*

- $\mu(\mathcal{M}_b) \geq 1 - \epsilon$ *for* $b \in \{0,1\}$
- *For all distinguishers* $\mathcal{A}'$ *of size* $s' = \frac{s\gamma^2}{128(m+n+1)}$

$$|\Pr_{x \leftarrow \mathcal{D}_{\mathcal{M}_0}}[\mathcal{A}(E(x)) = 1] - \Pr_{y \leftarrow \mathcal{D}_{\mathcal{M}_1}}[\mathcal{A}(F(y)) = 1]| \leq \gamma$$

Now we describe a lemma from [Hol06], that shows that if we sample a set Set from any measure $\mathcal{M}$ by choosing each element i in the support with probability $\mathcal{M}(i)$, then no circuit of (some) bounded size can distinguish a sample x chosen randomly from the set Set from an element sampled from distribution given by $\mathcal{M}$. Formally,

Theorem 5 (Imported Theorem [Hol06].)**.** *Let $\mathcal{M}$ be any measure on $\{0,1\}^n$ of density $\mu(\mathcal{M}) \geq 1-\rho(n)$ Let $\gamma(n) \in (0,1/2)$ be any function. Then, for a random set* Set *chosen according to the measure $\mathcal{M}$ the following two holds with probability at least $1 - 2(2^{-2^n \gamma^2(1-\rho)^4/64})$:*

- $(1 - \frac{\gamma(1-\rho)}{4})(1-\rho)2^n \leq |\mathsf{Set}| \leq (1 + \frac{\gamma(1-\rho)}{4})(1-\rho)2^n$
- *For such a random set* Set*, for any distinguisher $\mathcal{A}$ with size $|\mathcal{A}| \leq 2^n(\frac{\gamma^2(1-\rho)^4}{64n})$ satisfying*

$$|\Pr_{x \leftarrow \mathsf{Set}}[\mathcal{A}(x)=1] - \Pr_{x \leftarrow \mathcal{D}_{\mathcal{M}}}[\mathcal{A}(x)=1]| \leq \gamma$$

We also import a theorem from [CCL18] that will be used by our security proofs. This lemma would be useful to simulate the randomness used to encrypt in an inefficient hybrid.

Theorem 6 (Imported Theorem [CCL18].)**.** *Let $n, \ell \in \mathbb{N}$, $\epsilon > 0$ and $\mathcal{C}_{leak}$ be a family of distinguisher circuits from $\{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}$ of size $s(n)$. Then, for every distribution (X,Z) over $\{0,1\}^n \times \{0,1\}^\ell$, there exists a simulator $h : \{0,1\}^n \to \{0,1\}^\ell$ such that:*

- *h has size bounded by $s' = O(s2^\ell \epsilon^{-2})$.*
- *(X,Z) and $(X,h(X))$ are indistinguishable by $\mathcal{C}_{leak}$. That is for every $C \in \mathcal{C}_{leak}$,*

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

4.5 LWE 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 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 $\mathbf{A} \in \mathbb{Z}_{\mathbf{p}}^{\mathsf{d}\times m}$, we define the "$\mathbf{p}$-ary" integer lattices:

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

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

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 6 (LWE)**.** *For an integer $\mathbf{p} = \mathbf{p}(\mathsf{d}) \geq 2$, and an error distribution $\chi = \chi(\mathsf{d})$ over $\mathbb{Z}_{\mathbf{p}}$, the* Learning With Errors problem *$\mathsf{LWE}_{\mathsf{d},m,\mathbf{p},\chi}$ is to distinguish between the following pairs of distributions (e.g. as given by a sampling oracle $\mathcal{O} \in \{\mathcal{O}_{\boldsymbol{s}}, \mathcal{O}_{\$}\}$):*

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

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

Gadget matrix. The gadget matrix described below is proposed in [MP12, AP14].

Definition 7. *Let $m = \mathsf{d} \cdot \lceil \log \mathbf{p} \rceil$, and define the gadget matrix $\mathbf{G} = \boldsymbol{g}_2 \otimes \mathbf{I}_{\mathsf{d}} \in \mathbb{Z}_{\mathbf{p}}^{\mathsf{d} \times m}$, where the vector $\boldsymbol{g}_2 = (1, 2, 4, ..., 2^{\lfloor \log \mathbf{p} \rfloor}) \in \mathbb{Z}_{\mathbf{p}}^{\lceil \log \mathbf{p} \rceil}$. We will also refer to this gadget matrix as "powers-of-two" matrix. We define the inverse function $\mathbf{G}^{-1} : \mathbb{Z}_{\mathbf{p}}^{\mathsf{d} \times m} \rightarrow \{0,1\}^{m \times m}$ which expands each entry $a \in \mathbb{Z}_{\mathbf{p}}$ of the input matrix into a column of size $\lceil \log \mathbf{p} \rceil$ consisting of the bits of binary representations. We have the property that for any matrix $\mathbf{A} \in \mathbb{Z}_{\mathbf{p}}^{\mathsf{d} \times m}$, it holds that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$.*

5 Perturbation-Resilient Generators

A perturbation-resilient generator, denoted by $\Delta\mathsf{RG}$, consists of the following algorithms:

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

The following properties are associated with a $\Delta\mathsf{RG}$ scheme.

Efficiency: The following conditions need to be satisfied.

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

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

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

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

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

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

5.1 ΔRG implementable by Three-Restricted FE

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

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

The security and efficiency requirements are same as before.

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

5.2 Candidate for 3ΔRG

We now describe our candidate for ΔRG implementable by a three-restricted FE scheme. All these candidates use a large enough prime modulus $\mathbf{p} = O(2^\lambda)$, which is the same as the modulus used by 3−restricted FE. Then, let χ be a distribution used to sample input elements over $\mathbb{Z}$. Let Q denote a polynomial sampler. Next we give candidate in terms of χ and Q but give concrete instantiations later.

- $\mathsf{Setup}(1^\lambda, 1^n, B) \to (\mathsf{PP}, \mathsf{Seed})$. Sample a secret $\boldsymbol{s} \leftarrow \mathbb{Z}_\mathbf{p}^{1\times \mathsf{d}}$ for $\mathsf{d} = poly(\lambda)$ such that $\mathsf{LWE}_{\mathsf{d},n\cdot d,\mathbf{p},\chi}$ holds. Here χ is a bounded distribution with bound $poly(\lambda)$ (see Section 4.5 for definitions). Let $\mathcal{Q}$ denote an efficiently samplable distribution of homogeneous degree 3 polynomials (instantiated later). Then proceed with $\mathsf{SetupSeed}$ as follows:
  1. Sample $\boldsymbol{a}_i \leftarrow \mathbb{Z}_\mathbf{p}^{1\times \mathsf{d}}$ for $i \in [n]$.
  2. Sample $e_i \leftarrow \chi$ for $i \in [n]$.
  3. Compute $r_i = \langle \boldsymbol{a}_i, \boldsymbol{s}\rangle + e_i \mod \mathbf{p}$ in $\mathbb{Z}_\mathbf{p}$ for $i \in [n]$.
  4. Define $\boldsymbol{w}_i = (\boldsymbol{a}_i, r_i)$ for $i \in [n]$.
  5. Set $\mathsf{Seed.pub}(i) = \boldsymbol{w}_i$ for $i \in [n]$.

6. Sample $y_i, z_i \leftarrow \chi$ for $i \in [n]$.
7. Set $\boldsymbol{t} = (-\boldsymbol{s}, 1)$. Note that $\langle \boldsymbol{w}_i, \boldsymbol{t} \rangle = e_i$ for $i \in [n]$.
8. Set $\boldsymbol{y}'_i = y_i \otimes \boldsymbol{t}$.
9. Set $\mathsf{Seed.priv}(1, j) = \boldsymbol{y}'_j$ for $j \in [n]$.
10. Set $\mathsf{Seed.priv}(2, k) = z_k$ for $k \in [n]$.

Now we describe SetupPoly. Fix $\eta = n^{1+\epsilon}$.

1. Write $\boldsymbol{e} = (e_1, \ldots, e_n)$ for $j \in [d]$, $\boldsymbol{y} = (y_1, \ldots, y_n)$ and $\boldsymbol{z} = (z_1, \ldots, z_n)$.
2. Sample polynomials q'_ℓ for $\ell \in [\eta]$ as follows.
3. $q'_\ell = \Sigma_{\boldsymbol{I}=(i,j,k)} c_{\boldsymbol{I}} e_i \cdot y_j \cdot z_k$ where coefficients $c_{\boldsymbol{I}}$ are bounded by $poly(\lambda)$. These polynomials are sampled according to $\mathcal{Q}$
4. Define q_i be a multilinear homogeneous degree 3 polynomial takes as input $\mathsf{Seed} = (\{\boldsymbol{w}_i\}_{i \in [n]}, \boldsymbol{y}'_1, \ldots, \boldsymbol{y}'_n, \boldsymbol{z})$. Then it computes each monomial $c_{\boldsymbol{I}} e_i y_j \cdot z_k$ as follows and then adds all the results (thus computes $q'_i(\boldsymbol{e}, \boldsymbol{y}, \boldsymbol{z})$):
   - Compute $c_{\boldsymbol{I}} \langle \boldsymbol{w}_i, \boldsymbol{t} \rangle \cdot y_j \cdot z_k$. This step requires $\boldsymbol{y}'_i = y_i \otimes \boldsymbol{t}$ to perform this computation.
5. Output $q_1, ..., q_\eta$.

- $\mathsf{Eval}(\mathsf{PP}, \mathsf{Seed}) \rightarrow (h_1, ..., h_\eta)$, evaluation algorithm output a vector $(h_1, ..., h_\eta) \in \mathbb{Z}^\eta$. Here for $i \in [\eta]$, $h_i = q_i(\mathsf{Seed})$ and η is the stretch of $3\Delta\mathsf{RG}$. Here q_i is a degree 3 homogenenous multilinear polynomial (defined above) which is degree 1 in public and 2 in private components of the seed.

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

Efficiency:

1. Note that Seed contains n LWE samples $\boldsymbol{w}_i$ for $i \in [n]$ of dimension d. Along with the samples, it contains elements $\boldsymbol{y}'_i = y_i \otimes \boldsymbol{t}$ for $i \in [n]$ and elements z_i for $i \in [n]$. Note that the size of these elements are bounded by $poly(\lambda)$ and is independent of n.

2. The values $h_i = q_i(\mathsf{Seed}) = \Sigma_{\boldsymbol{I}=(i,j,k)} c_{\boldsymbol{I}} e_i \cdot y_j \cdot z_k$. Since χ is a bounded distribution, bounded by $poly(\lambda, n)$, and coefficients $c_{\boldsymbol{I}}$ are also polynomially bounded, each $|h_i| < poly(\lambda, n)$ for $i \in [m]$.

Conjecture 1. *The above candidate is a secure perturbation-resilient generator implementable by three-restricted FE.*

5.2.1 Our Instantiations.

We now give various instantiations of Q. Let χ be the discrete gaussian distribution with 0 mean and standard deviation n. The following candidate is proposed by [BHJ+18] based on the investigation of the hardness of families of expanding polynomials over the reals. For any vector $\boldsymbol{v}$, denote by $\boldsymbol{v}[i]$, the i^{th} component of the vector.

Instantiation 1: 3SAT Based Candidate. Let $t = B^2\lambda$. Sample each polynomial q'_i for $i \in [\eta]$ as follows. $q'_i(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_t, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_t) = \Sigma_{j\in[t]} q'_{i,j}(\boldsymbol{x_j}, \boldsymbol{y_j}, \boldsymbol{z_j})$. Here $\boldsymbol{x}_j \in \chi^{d\times n}$ and $\boldsymbol{y}_j, \boldsymbol{z}_j \in \chi^n$ for $j \in [t]$. In other words, q'_i is a sum of t polynomials $q'_{i,j}$ over t disjoint set of variables.

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

1. Sample randomly inputs $\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^* \in \{0,1\}^n$.
2. To sample $q'_{i,j}$ do the following. Sample three indices randomly and independently $i_1, i_2, i_3 \leftarrow [n]$. Sample three signs $b_{1,i,j}, b_{2,i,j}, b_{3,i,j} \in \{0,1\}$ uniformly such that $b_{1,i,j} \oplus b_{2,i,j} \oplus b_{3,i,j} \oplus \boldsymbol{x}^*[i_1] \oplus \boldsymbol{y}^*[i_2] \oplus \boldsymbol{z}^*[i_3] = 1$.
3. Set $q'_{i,j}(\boldsymbol{x_j}, \boldsymbol{y_j}, \boldsymbol{z_j}) = 1 - (b_{1,i,j} \cdot \boldsymbol{x_j}[i_1] + (1 - b_{1,i,j}) \cdot (1 - \boldsymbol{x_j}[i_1])) \cdot (b_{2,i,j} \cdot \boldsymbol{y_j}[i_2] + (1 - b_{2,i,j}) \cdot (1 - \boldsymbol{y_j}[i_2])) \cdot ((b_{3,i,j} \cdot \boldsymbol{z_j}[i_3] + (1 - b_{3,i,j}) \cdot (1 - \boldsymbol{z_j}[i_3]))$

Remark: Note that any clause of the form $a_1 \vee a_2 \vee a_3$ can be written as $1-(1-a_1)(1-a_2)(1-a_3)$ over integers where a_1, a_2, a_3 are literals in first case and take values in $\{0,1\}$, and thus any random satisfiable 3SAT formula can be converted to polynomials in this manner.

Instantiation 2: Density Variations.

1. One can set $q'_{i,j}$ to be a random sparse polynomial for all j.
2. One variation could be to instantiate $q'_{i,j}$ as a random dense polynomial for all j.
3. Finally one can even consider variants where $q'_{i,j}$ is dense for some values of j while it is sparse for the others.

6 Tempered Cubic Encoding

In this section, we describe the notion of a Tempered Cubic Encoding scheme (TCE for short). The encodings in this scheme are associated with a ring $\mathbb{Z}_{\mathbf{p}}$, for an integer $\mathbf{p} \in \mathbb{Z}^{\geq 0}$ that is fixed by the setup algorithm. The plaintext elements are sampled from the set $R \in \cap[-\delta, \delta]$ for some constant δ. TCE consists of the following polynomial time algorithms:

- **Setup,** $\mathsf{Setup}(1^\lambda, 1^n)$: On input security parameter λ, the number of inputs n, this algorithm outputs public parameters params.
- **Setup-Encode,** $\mathsf{SetupEnc}(\mathsf{params})$: On input params, this algorithm outputs secret encoding parameters sp.
- **Setup-Decode,** $\mathsf{SetupDec}(\mathsf{params})$: On input params, this algorithm outputs (public) decoding parameters $(q_1, ..., q_\eta)$ where $\eta = n^{1+\epsilon}$ described in the security definition.
- **Encode,** $\mathsf{Encode}(\mathsf{sp}, a, \mathsf{ind}, S)$: On input the secret parameter sp, a plain-text element $a \in R$, a set $S = \{i\}$ with $i \in \{1,2,3\}$ and an index $\mathsf{ind} \in [n]$, it outputs an encoding $[\mathbf{a}]_{\mathsf{ind},S}$ with respect to the set S and an index ind. Without loss of generality, this algorithm is deterministic as all the randomness can be chosen during $\mathsf{SetupEnc}$. This encoding satisfies two properties:

– The encoding $[\mathbf{a}]_{\mathsf{ind},S} = ([\mathbf{a}]_{\mathsf{ind},S}.\mathsf{pub}, [\mathbf{a}]_{\mathsf{ind},S}.\mathsf{priv}(1), [\mathbf{a}]_{\mathsf{ind},S}.\mathsf{priv}(2))$ consists of a public component $[\mathbf{a}]_{\mathsf{ind},S}.\mathsf{pub}$ and two private components $[\mathbf{a}]_{\mathsf{ind},S}.\mathsf{priv}(1)$ and $[\mathbf{a}]_{\mathsf{ind},S}.\mathsf{priv}(2)$.
– $[\mathbf{a}]_{\mathsf{ind},S}.\mathsf{pub}, [\mathbf{a}]_{\mathsf{ind},S}.\mathsf{priv}(1)$ and $[\mathbf{a}]_{\mathsf{ind},S}.\mathsf{priv}(2)$ are vectors over $\mathbb{Z}_{\mathbf{p}}$.

- **Decode,** $\mathsf{Decode}(q, f, \{[\mathbf{a_i}]_{i,1}\}_{i\in[n]}, \{[\mathbf{b_i}]_{i,2}\}_{i\in[n]}, \{[\mathbf{c_i}]_{i,3}\}_{i\in[n]})$: The decode algorithm takes as input a decoding parameter q, a polynomial $f = \Sigma_{i,j,k}\gamma_{i,j,k}a_ib_jc_k$ with $|\gamma_{i,j,k}| \leq \delta$. It also takes encodings $\{[\mathbf{a_i}]_{i,1}\}_{i\in[n]}$, $\{[\mathbf{b_i}]_{i,2}\}_{i\in[n]}$ and $\{[\mathbf{c_i}]_{i,3}\}_{i\in[3]}$. It outputs $\mathsf{leak} \in \mathbb{Z}_{\mathbf{p}}$.

Efficiency Properties: Consider the following experiment associated with any $n, \lambda \in \mathbb{N}$, any index $\mathsf{ind} \in [n]$, any level $\ell \in [3]$ and any plaintext $x \in [-\delta, \delta]$:

1. $\mathsf{Setup}(1^\lambda, 1^n) \to \mathsf{params}$,
2. $\mathsf{SetupEnc}(\mathsf{params}) \to \mathsf{sp}$,
3. $\mathsf{Encode}(\mathsf{sp}, x, \mathsf{ind}, \ell) \to [\mathbf{x}]_{\mathsf{ind},\ell}$.

Then we require the circuit size computing $[\mathbf{x}]_{\mathsf{ind},\ell}$ is less than $poly(\lambda, \log n)$ for some fixed polynomial $poly$.

$(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$-Multilinear polynomials. We define the notion of $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ cubic multilinear polynomials below.

Definition 8 ($(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$-Multilinear)**.** *Let $\mathbf{X} = (x_1, \ldots, x_n), \mathbf{Y} = (y_1, \ldots, y_n)$ and $\mathbf{Z} = (z_1, \ldots, z_n)$ be three sets of variables. A polynomial $p \in \mathbb{Z}_{\mathbf{p}}[x_1, \ldots, x_n, y_1, \ldots, y_n, z_1, \ldots, z_n]$ is $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$-multilinear if every term in the expansion of p is of the form $\tau_{ijk} \cdot x_iy_jz_k$, for some $i, j, k \in [n], \tau_{ijk} \in \mathbb{Z}_{\mathbf{p}}$.*

Cubic Evaluation and Correctness: Consider the following experiment associated with any $n, \lambda \in \mathbb{N}$, any index $\mathsf{ind} \in [n]$, any index $\mathsf{ind}_Q \in [\eta]$, any level $\ell \in [3]$, any polynomial $f = \Sigma_{i,j,k}\gamma_{i,j,k}a_ib_jc_k$ with $\gamma_{i,j,k} \in [-\delta, \delta]$ and any plaintexts $a_i, b_i, c_i \in [-\delta, \delta]$ for $i \in [n]$:

1. $\mathsf{Setup}(1^\lambda, 1^n) \to \mathsf{params}$
2. $\mathsf{SetupEnc}(\mathsf{params}) \to \mathsf{sp}$
3. $\mathsf{SetupDec}(\mathsf{params}) \to (q_1, ..., q_\eta)$
4. $\mathsf{Encode}(\mathsf{sp}, a, i, 1) \to [\mathbf{a}]_{i,1}$ for $i \in [n]$
5. $\mathsf{Encode}(\mathsf{sp}, b, i, 2) \to [\mathbf{b}]_{i,2}$ for $i \in [n]$
6. $\mathsf{Encode}(\mathsf{sp}, c, i, 3) \to [\mathbf{c}]_{i,3}$ for $i \in [n]$
7. Let $q = q_{\mathsf{ind}_Q}$
8. $\mathsf{Decode}(q, f, \{[\mathbf{a_i}]_{i,1}\}_{i\in[n]}, \{[\mathbf{b_i}]_{i,2}\}_{i\in[n]}, \{[\mathbf{c_i}]_{i,3}\}_{i\in[n]}) \to \mathsf{leak}$

Cubic Evaluation: We now describe cubic evaluation property. This property states that the $\mathsf{Decode}(q, f, \{[\mathbf{a_i}]_{i,1}\}_{i\in[n]}, \{[\mathbf{b_i}]_{i,2}\}_{i\in[n]}, \{[\mathbf{c_i}]_{i,3}\}_{i\in[n]})$ algorithm evaluates an efficiently computable cubic polynomial $\phi_{q,f}$ which depends on params, f, q, and which is a $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$-multilinear polynomial over $\mathbb{Z}_{\mathbf{p}}$ with:

- $\mathbf{X} = (\{[\mathbf{a_i}]_{i,1}.\mathsf{pub}, [\mathbf{b_i}]_{i,2}.\mathsf{pub}, [\mathbf{c_i}]_{i,3}.\mathsf{pub}\}_{i\in[n]})$
- $\mathbf{Y} = (\{[\mathbf{a_i}]_{i,1}.\mathsf{priv}(1), [\mathbf{b_i}]_{i,2}.\mathsf{priv}(1), [\mathbf{c_i}]_{i,3}.\mathsf{priv}(1)\}_{i\in[n]})$
- $\mathbf{Z} = (\{[\mathbf{a_i}]_{i,1}.\mathsf{priv}(2), [\mathbf{b_i}]_{i,2}.\mathsf{priv}(2), [\mathbf{c_i}]_{i,3}.\mathsf{priv}(2)\}_{i\in[n]})$

Correctness: We require that with overwhelming probability over the randomness of the algorithms:

- If $f(a_1,..,a_n,b_1,..,b_n,c_1,..,c_n) = 0$, $|\mathsf{leak}| < \mathsf{TCEbound}(\lambda, n)$ for some polynomial $\mathsf{TCEbound}$.
- Otherwise, $|\mathsf{leak}| > \mathsf{TCEbound}(\lambda, n)$.

6.1 Tempered Security

We present the definition of Tempered Security. Let $\mathcal{F}$ be a family of homogenous $(\mathbf{X},\mathbf{Y},\mathbf{Z})$-multilinear δ-bounded polynomials, for some sets of vectors $\mathbf{X},\mathbf{Y}$ and $\mathbf{Z}$ (where each vector is of size n). We define $\mathcal{S}_\eta$ to be a subset of η-sized product $\mathcal{F}\times\cdots\times\mathcal{F}$ (also, written as $\mathcal{F}^\eta$).

We first describe the experiments associated with tempered security property. The experiment is associated with a deterministic polynomial time algorithm Sim. It is also parameterised by $\mathsf{aux} = (1^\lambda, 1^n, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, f_1, ..., f_\eta)$. Each vector $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ is in $\mathbb{Z}^n$ and $f_1, ..., f_\eta \in \mathcal{S}_\eta$.

$\underline{\mathsf{Expt}_{\mathsf{aux}}(1^\lambda, 1^n, 0)}$:

1. Challenger performs $\mathsf{Setup}(1^\lambda, 1^n) \to \mathsf{params}$
2. The challenger samples $(q_1, ..., q_\eta) \leftarrow \mathsf{SetupDec}(\mathsf{params})$.
3. Challenger performs $\mathsf{SetupEnc}(\mathsf{params}) \to \mathsf{sp}$.
4. Now compute encodings as follows.
   - Compute the encodings, $[\mathbf{x_i}]_{i,1} \leftarrow \mathsf{Encode}(\mathsf{sp}, x_i, i, 1)$ for every $i \in [n]$.
   - Compute the encodings, $[\mathbf{y_i}]_{i,2} \leftarrow \mathsf{Encode}(\mathsf{sp}, y_i, i, 2)$ for every $i \in [n]$.
   - Compute the encodings, $[\mathbf{z_i}]_{i,3} \leftarrow \mathsf{Encode}(\mathsf{sp}, z_i, i, 3)$ for every $i \in [n]$.
5. Compute $\mathsf{leak}_j \leftarrow \mathsf{Decode}(q_j, f_j, \{[\mathbf{x_i}]_{i,1}\}_{i\in[n]}, \{[\mathbf{y_i}]_{i,2}\}_{i\in[n]}, \{[\mathbf{z_i}]_{i,3}\}_{i\in[n]})$ for $j \in [\eta]$.
6. Output the following:
   (a) Public components of the encodings, $\{[\mathbf{x_i}]_{i,1}.\mathsf{pub}, [\mathbf{y_i}]_{i,2}.\mathsf{pub}, [\mathbf{z_i}]_{i,3}.\mathsf{pub}\}_{i\in[n]}$.
   (b) Decoding parameters q_j for $j \in [\eta]$
   (c) Output of decodings, $\{\mathsf{leak}_j\}_{j\in[\eta]}$.

$\underline{\mathsf{Expt}_{\mathsf{aux}}(1^\lambda, 1^n, 1)}$:

1. Challenger performs $\mathsf{Setup}(1^\lambda, 1^n) \to \mathsf{params}$
2. The challenger samples $(q_1, ..., q_\eta) \leftarrow \mathsf{SetupDec}(\mathsf{params})$.
3. Challenger performs $\mathsf{SetupEnc}(\mathsf{params}) \to \mathsf{sp}$.

- Compute the encodings, $[\mathbf{x_i}]_{i,1} \leftarrow \mathsf{Encode}(\mathsf{sp}, 0, , i, 1)$ for every $i \in [n]$.
- Compute the encodings, $[\mathbf{y_i}]_{i,2} \leftarrow \mathsf{Encode}(\mathsf{sp}, 0, i, 2)$ for every $i \in [n]$.
- Compute the encodings, $[\mathbf{z_i}]_{i,3} \leftarrow \mathsf{Encode}(\mathsf{sp}, 0, i, 3)$ for every $i \in [n]$.

4. Compute the following for all $j \in [\eta]$:

$$\widehat{\mathsf{leak}_j} \leftarrow \mathsf{Sim}\left(q_j, f_j, \{[\mathbf{x_i}]_{i,1}\}_{i\in[n]}, \{[\mathbf{y_i}]_{i,2}\}_{i\in[n]}, \{[\mathbf{z_i}]_{i,3}\}_{i\in[n]}, f_j(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\right)$$

to obtain the simulated outputs.

5. Output the following:

   (a) Public components of the encodings, $\{[\mathbf{x_i}]_{i,1}.\mathsf{pub}, [\mathbf{y_i}]_{i,2}.\mathsf{pub}, [\mathbf{z_i}]_{i,3}.\mathsf{pub}\}_{i\in[n]}$.

   (b) Decoding parameters q_j for $j \in [\eta]$

   (c) Output of decodings, $\{\widehat{\mathsf{leak}_j}\}_{j\in[\eta]}$.

Definition 9 (Tempered Security)**.** *A tempered cubic encoding scheme* $\mathsf{TCE} = (\mathsf{Setup}, \mathsf{SetupEnc}, \mathsf{SetupDec}, \mathsf{Encode}, \mathsf{Decode})$ *associated with plaintext space* $\mathbb{Z} = [-\delta, \delta]$ *is said to satisfy* **Tempered security** *for polynomials (with coefficients over* $[-\delta, \delta]$*) if there exists an algorithm* Sim *so that following happens:*

$\exists c > 0$*, such that for all large enough security parameter* $\lambda \in \mathbb{N}$*, and polynomial* $n = n(\lambda)$ *and any* $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{Z}^n$*,* $(f_1, .., f_\eta) \in \mathcal{S}_\eta$ *and adversary* $\mathcal{A}$ *of size* 2^{λ^c}*,*

$$|\Pr[\mathcal{A}(\mathsf{Expt}_{\mathsf{aux}}(1^\lambda, 1^n, 0) = 1] - \Pr[\mathcal{A}(\mathsf{Expt}_{\mathsf{aux}}(1^\lambda, 1^n, 1)) = 1]| \le 1 - 1/\lambda + \mathsf{negl}(\lambda)$$

where $\mathsf{aux} = (1^\lambda, 1^n, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, f_1, ..., f_\eta)$ *and* $\mathsf{negl}(\lambda)$ *is some negligible function.*

Few remarks are in order:

Remark 3. *For the rest of the paper, we abbreviate tempered security as* $\mathcal{S}_\eta-$*tempered security to explicitly mention the function class* $\mathcal{S}_\eta$*. One can imagine* $\mathcal{S}_\eta$ *to be an arbitrary subset of* $\mathcal{F} \times \cdots \times \mathcal{F}$*. However, to pursue our approach, we will set* $\mathcal{S}_\eta$ *as the* $\eta-$*sized product of cubic polynomials in* $n(\lambda)$ *variables with the sum of absolute value of coefficients being bounded by some polynomial (in* λ*) independent of* n*. As described later, it turns out that this set contains the set of randomizing polynomials constructed by [LT17], and suffices to get iO.*

Remark 4. *(On distinguishing gap being* $1 - 1/\lambda$*) In the definition above and other definitions described in the paper, we require distinguishing gap of any adversary of some bounded size to be bounded by* $1 - 1/\lambda + \mathsf{negl}(\lambda)$*, however it actually suffices if it is bounded by* $1 - 1/poly(\lambda) + \mathsf{negl}(\lambda)$ *for any fixed polynomial poly. We do this for simplicity of description.*

Remark 5. *(On number of query polynomials) In the definition above, an implicit restriction on the number of polynomials (i.e.,* η *polynomials). Indeed, in the instantiation, we only support* $\eta = n^{1+\varepsilon}$ *for some* $0 < \epsilon < 0.5$*. This choice of parameters will suffice for our construction of iO. This* ε *will be set later.*

7 Three-restricted FE

In this section we describe the notion of a three-restricted functional encryption scheme (denoted by 3FE).

Function class of interest: Consider a set of functions $\mathcal{F}_{\mathsf{3FE}} = \mathcal{F}_{\mathsf{3FE},\lambda,\mathbf{p},n} = \{f : \{\mathbb{Z}_{\mathbf{p}}^n\}^3 \rightarrow \mathbb{Z}_{\mathbf{p}}\}$ where $\mathbb{Z}_{\mathbf{p}}$ is a finite field of order $\mathbf{p}(\lambda)$. Here n is seen as a function of λ. Each $f \in \mathcal{F}_{\lambda,\mathbf{p},n}$ takes as input three vectors $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ over $\mathbb{Z}_{\mathbf{p}}$ and computes a polynomial of the form $\Sigma c_{i,j,k} \cdot x_i y_j z_k$, where $c_{i,j,k}$ are coefficients from $\mathbb{Z}_{\mathbf{p}}$.

Syntax. Consider the set of functions $\mathcal{F}_{\mathsf{3FE},\lambda,\mathbf{p},n}$ as described above. A three-restricted functional encryption scheme 3FE for the class of functions $\mathcal{F}_{\mathsf{3FE}}$ (described above) consists of the following PPT algorithms:

- **Setup,** $\mathsf{Setup}(1^\lambda, 1^n)$: On input security parameter λ (and the number of inputs $n = poly(\lambda)$), it outputs the master secret key MSK.
- **Encryption,** $\mathsf{Enc}(\mathsf{MSK}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$: On input the encryption key MSK and input vectors $\boldsymbol{x} = (x_1, .., x_n), \boldsymbol{y} = (y_1, .., y_n)$ and $\boldsymbol{z} = (z_1, .., z_n)$ (all in $\mathbb{Z}_{\mathbf{p}}^n$) it outputs ciphertext CT. Here $\boldsymbol{x}$ is seen as a public attribute and $\boldsymbol{y}$ and $\boldsymbol{z}$ are thought of as private messages.
- **Key Generation,** $\mathsf{KeyGen}(\mathsf{MSK}, f)$: On input the master secret key MSK and a function $f \in \mathcal{F}_{\mathsf{3FE}}$, it outputs a functional key $sk[f]$.
- **Decryption,** $\mathsf{Dec}(sk[f], 1^B, \mathsf{CT})$: On input functional key $sk[f]$, a bound $B = poly(\lambda)$ and a ciphertext CT, it outputs the result *out*.

We define correctness property below.

B-Correctness. Consider any function $f \in \mathcal{F}_{\mathsf{3FE}}$ and any plaintext $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{Z}_{\mathbf{p}}$. Consider the following process:

- $sk[f] \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f)$.
- $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$
- If $f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in [-B, B]$, set $\theta = f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, otherwise set $\theta = \perp$.

The following should hold:

$$\mathsf{Pr}\left[\mathsf{Dec}(sk[f], 1^B, \mathsf{CT}) = \theta\right] \geq 1 - \mathsf{negl}(\lambda),$$

for some negligible function negl.

Linear Efficiency: We require that for any message $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in \mathbb{Z}_{\mathbf{p}}^n$ the following happens:

- Let $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$.
- Compute $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.

The size of the circuit computing CT is less than $n \log_2 \mathbf{p} \cdot poly(\lambda)$. Here *poly* is some polynomial independent of n.

7.1 Semi-functional Security

We define the following auxiliary algorithms.

Semi-functional Key Generation, $\mathsf{sfKG}(\mathsf{MSK}, f, \theta)$: On input the master secret key MSK, function f and a value θ, it computes the semi-functional key $sk[f, \theta]$.

Semi-functional Encryption, $\mathsf{sfEnc}(\mathsf{MSK}, \boldsymbol{x}, 1^{|\boldsymbol{y}|}, 1^{|\boldsymbol{z}|})$: On input the master encryption key MSK, a public attribute $\boldsymbol{x}$ and length of messages $\boldsymbol{y}, \boldsymbol{z}$, it computes a semi-functional ciphertext $\mathsf{ct}_{\mathsf{sf}}$.

We define two security properties associated with the above two auxiliary algorithms. We will model the security definitions along the same lines as semi-functional FE.

Definition 10 (Indistinguishability of Semi-functional Ciphertexts)**.** *A three-restricted functional encryption scheme* $\mathsf{3FE}$ *for a class of functions* $\mathcal{F}_{\mathsf{3FE}} = \{\mathcal{F}_{\mathsf{3FE},\lambda,\mathbf{p},n}\}_{\lambda \in \mathbb{N}}$ *is said to satisfy* **indistinguishability of semi-functional ciphertexts property** *if there exists a constant* $c > 0$ *such that for sufficiently large* $\lambda \in \mathbb{N}$ *and any adversary* $\mathcal{A}$ *of size* 2^{λ^c}*, the probability that* $\mathcal{A}$ *succeeds in the following experiment is* $2^{-\lambda^c}$*.*

$\mathsf{Expt}(1^\lambda, \mathbf{b})$*:*

1. *$\mathcal{A}$ specifies the following:*
   - *Challenge message* $M^* = (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$*. Here each vector is in* $\mathbb{Z}_{\mathbf{p}}^n$*.*
   - *It can also specify additional messages* $\{M_k = (\boldsymbol{x}_k, \boldsymbol{y}_k, \boldsymbol{z}_k)\}_{k \in [q]}$ *Here each vector is in* $\mathbb{Z}_{\mathbf{p}}^n$*.*
   - *It also specifies functions* $f_1, \ldots, f_\eta$ *and hardwired values* $\theta_1, \ldots, \theta_\eta$*.*
2. *The challenger checks if* $\theta_k = f_k(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ *for every* $k \in [\eta]$*. If this check fails, the challenger aborts the experiment.*
3. *The challenger computes the following*
   - *Compute* $sk[f_k, \theta_k] \leftarrow \mathsf{sfKG}(\mathsf{MSK}, f_k, \theta_k)$*, for every* $k \in [\eta]$*.*
   - *If* $\mathbf{b} = 0$*, compute* $\mathsf{CT}^* \leftarrow \mathsf{sfEnc}(\mathsf{MSK}, \boldsymbol{x}, 1^{|\boldsymbol{y}|}, 1^{|\boldsymbol{z}|})$*. Else, compute* $\mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{MSK}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$*.*
   - $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{MSK}, M_i)$*, for every* $i \in [q]$*.*
4. *The challenger sends* $\left(\{\mathsf{CT}_i\}_{i \in [q]}, \mathsf{CT}^*, \{sk[f_k, \theta_k]\}_{k \in [\eta]}\right)$ *to* $\mathcal{A}$*.*
5. *The adversary outputs a bit* b'*.*

We say that the adversary $\mathcal{A}$ *succeeds in* $\mathsf{Expt}(1^\lambda, \mathbf{b})$ *with probability* ε *if it outputs* $b' = \mathbf{b}$ *with probability* $\frac{1}{2} + \varepsilon$*.*

We now define indistinguishability of semi-functional keys property.

Definition 11 (Indistinguishability of Semi-functional Keys)**.** *A three-restricted FE* $\mathsf{3FE}$ *for a class of functions* $\mathcal{F}_{\mathsf{3FE}} = \{\mathcal{F}_{\mathsf{3FE},\lambda,\mathbf{p},n}\}_{\lambda\in\mathbb{N}}$ *is said to satisfy* **indistinguishability of semi-functional keys property** *if there exists a constant* $c > 0$ *such that for all sufficiently large* λ*, any PPT adversary* $\mathcal{A}$ *of size* 2^{λ^c}*, the probability that* $\mathcal{A}$ *succeeds in the following experiment is* $2^{-\lambda^c}$*.*

$\mathsf{Expt}(1^\lambda, \mathbf{b})$*:*

1. $\mathcal{A}$ *specifies the following:*
   - *It can specify messages* $M_j = \{(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{z}_i)\}_{j\in[q]}$*. Here each vector is in* $\mathbb{Z}^n_{\mathbf{p}}$
   - *It specifies functions* $f_1, \ldots, f_\eta \in \mathcal{F}_{\mathsf{3FE}}$ *and hardwired values* $\theta_1, \ldots, \theta_\eta$*.*
2. *Challenger computes the following :*
   - *If* $\mathbf{b} = 0$*, compute* $sk[f_i]^* \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f_i)$ *for all* $i \in [\eta]$*. Otherwise, compute* $sk[f_i]^* \leftarrow \mathsf{sfKG}(\mathsf{MSK}, f_i, \theta_i)$ *for all* $i \in [\eta]$*.*
   - $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{MSK}, M_j)$*, for every* $j \in [q]$*.*
3. *Challenger then sends* $\left(\{\mathsf{CT}_i\}_{i\in[q]}, \{sk[f_i]^*\}_{i\in[\eta]}\right)$ *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *outputs* b'*.*

The success probability of $\mathcal{A}$ *is defined to be* ε *if* $\mathcal{A}$ *outputs* $b' = \mathbf{b}$ *with probability* $\frac{1}{2} + \varepsilon$*.*

If a three-restricted FE scheme satisfies both the above definitions, then it is said to satisfy semi-functional security.

Definition 12 (Semi-functional Security)**.** *Consider a three-restricted FE scheme* $\mathsf{3FE}$ *for a class of functions* $\mathcal{F}$*. We say that* $\mathsf{3FE}$ *satisfies* **semi-functional security** *if it satisfies indistinguishability of semi-functional ciphertexts property (Definition 10) and indistinguishability of semi-functional keys property (Definition 11).*

8 (Stateful) Semi-Functional Functional Encryption for Cubic Polynomials

In this section, we define the notion of Semi-Functional Functional Encryption (referred to as FE_3) for cubic polynomials. This is defined along the same lines as the definition of projective arithmetic functional encryption (PAFE), introduced by [AS17]. The main difference between our notion and PAFE is that, we allow for evaluation of arithmetic circuits over values from a bounded domain whereas PAFE allowed for evaluation of arithmetic circuits over large fields. Because of this, the decryption in [AS17] was expressed in two steps (Projective Decrypt and Recover), whereas the syntax of our decryption algorithm is the same as in a standard functional encryption scheme.

Function class of interest for FE_3**:** We consider functional encryption scheme for cubic homogenous polynomials over variables over integers $\mathbb{Z}$. Formally, consider a set of functions $\mathcal{F}_{\mathsf{FE}_3,\lambda,n} = \{f : [-\rho, \rho]^n \to \mathbb{Z}\}$ where ρ is some constant. Here n is interpreted as a function of λ. Each $f \in \mathcal{F}_{\mathsf{FE}_3,\lambda,n}$ takes as input $\boldsymbol{x} = (x_1, .., x_n) \in [-\rho, \rho]^n$ and computes a polynomial of the form

$\Sigma c_{i,j,k} x_i x_j x_k$ over $\mathbb{Z}$ (where some variables can repeat) and each coefficient $c_{i,j,k} \in [\rho, \rho]$ and sum of absolute values of the coefficients $\Sigma_{j,k}|c_{i,j,k}| < w(\lambda)$. Constructing functional encryption for homogenous polynomials suffice to construct functional encryption for all cubic polynomials. This is because we can always write any polynomial as a homogeneous polynomial in the same variables and an artificially introduced variable set to 1.

Syntax. Consider the set of functions $\mathcal{F}_{\mathsf{FE}_3} = \mathcal{F}_{\mathsf{FE}_3,\lambda,n}$ as described above. A semi-functional functional encryption scheme FE_3 for the class of functions $\mathcal{F}_{\mathsf{FE}_3}$ (described above) consists of the following PPT algorithms:

- **Setup,** $\mathsf{Setup}(1^\lambda, 1^n)$: On input security parameter λ and the length of the message 1^n, it outputs the master secret key MSK.
- **Encryption,** $\mathsf{Enc}(\mathsf{MSK}, \boldsymbol{x})$: On input the encryption key MSK and a vector of integers $\boldsymbol{x} = (x_1, .., x_n) \in [-\rho, \rho]^n$, it outputs ciphertext CT.
- **Key Generation,** $\mathsf{KeyGen}(\mathsf{MSK}, i, f)$: On input the master secret key MSK and an index $i \in [\eta]$ denoting the index of the function in $[\eta]$, function $f \in \mathcal{F}_{\mathsf{FE}_3}$, it outputs a functional key sk_f. Here, η denotes the number of key queries possible. Note that this algorithm is allowed to be stateful.
- **Decryption,** $\mathsf{Dec}(sk_f, \mathsf{CT})$: On input functional key sk_f and a ciphertext CT, it outputs the result *out*.

We define correctness property below.

Correctness. Consider any function $f \in \mathcal{F}_{\mathsf{FE}_3}$, any index $i \in [\eta]$ and any plaintext integer vector $\boldsymbol{x} \in [-\rho, \rho]^n$. Consider the following process:

- $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$
- $sk_f \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, i, f)$.
- $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, \boldsymbol{x})$

Let $\theta = 1$ if $f(\boldsymbol{x}) \neq 0$, $\theta = 0$ otherwise. The following should hold:

$$\Pr\left[\mathsf{Dec}(sk_f, \mathsf{CT}) = \theta\right] \geq 1 - \mathsf{negl}(\lambda),$$

for some negligible function negl.

Remark 6. *We consider a form of semi-functional functional encryption where the decryption algorithm only allows the decryptor to learn if the functional value $f(\boldsymbol{x})$ is 0 or not.*

Linear Efficiency: We require that for any message $\boldsymbol{x} \in [-\rho, \rho]^n$ the following holds:

- Let $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$.
- Compute $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, \boldsymbol{x})$.

The size of the circuit computing CT is less than $poly(\lambda, \log n)$. Here $poly$ is some fixed polynomial independent of n.

8.1 Semi-functional Security

We define the following auxiliary algorithms.

Semi-functional Key Generation, $\mathsf{sfKG}(\mathsf{MSK}, i, f, \theta)$: On input the master secret key MSK, function f, an index i and a value θ, it computes the semi-functional key $sk_{f,\theta}$.

Semi-functional Encryption, $\mathsf{sfEnc}(\mathsf{MSK}, 1^n)$: On input the master encryption key MSK, and the length 1^n, it computes a semi-functional ciphertext $\mathsf{ct}_{\mathsf{sf}}$.

We define two security properties associated with the above auxiliary algorithms.
We now define indistinguishability of semi-functional keys property.

Throughout the definition we denote by $\mathcal{S}_\eta$ a set of tuples of dimension η over $\mathcal{F}_{\mathsf{FE}_3}$. Thus $\mathcal{S}_\eta \subseteq \mathcal{F}^{\eta}_{\mathsf{FE}_3}$.

Definition 13 ($\mathcal{S}_\eta$-Bounded Indistinguishability of Semi-functional Keys)**.** *A Semi-Functional FE scheme for cubic polynomials* FE_3 *for a class of functions* $\mathcal{F}_{\mathsf{FE}_3} = \{\mathcal{F}_{\mathsf{FE}_3,\lambda,n}\}_{\lambda\in\mathbb{N}}$ *is said to satisfy* $\mathcal{S}_\eta-$**bounded indistinguishability of semi-functional keys property** *if there exists a constant* $c > 0$ *such that for any sufficiently large* $\lambda \in \mathbb{N}$ *and any adversary* $\mathcal{A}$ *of size* 2^{λ^c}*, the probability that* $\mathcal{A}$ *succeeds in the following experiment is* $2^{-\lambda^c}$*.*
$\mathsf{Expt}(1^\lambda, 1^n, \mathbf{b})$*:*

1. *$\mathcal{A}$ specifies the following:*
   - *It can specify messages $M_j = \{\boldsymbol{x}_i\}_{j\in[q]}$. Here each vector is in $[-\rho, \rho]^n$*
   - *It specifies function queries as follows:*
     - *It specifies $(f_1, \ldots, f_\eta) \in \mathcal{S}_\eta \subseteq \mathcal{F}^{\eta}_{\mathsf{FE}_3}$.*
     - *It specifies values $\theta_1, \ldots, \theta_\eta$.*
2. *The challenger computes the following:*
   - $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$
   - $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{MSK}, M_j)$*, for every $j \in [q]$.*
   - *If $\mathbf{b} = 0$, compute $sk^*_{f_i} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, i, f_i)$. Otherwise, compute $sk^*_{f_i} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, i, f_i, \theta_i)$ for all $i \in [\eta]$.*
3. *Challenger sends $\{\mathsf{CT}_i\}_{i\in q}$ and $\{sk^*_{f_i}\}_{i\in[\eta]}$ to $\mathcal{A}$:*
4. *$\mathcal{A}$ outputs b'.*

The success probability of $\mathcal{A}$ is defined to be ε if $\mathcal{A}$ outputs $b' = \mathbf{b}$ with probability $\frac{1}{2} + \varepsilon$.

Definition 14 ($\mathcal{S}_\eta$-Bounded Indistinguishability of Semi-functional Ciphertexts)**.** *For a semi-functional FE scheme* FE_3 *for a class of functions* $\mathcal{F}_{\mathsf{FE}_3} = \{\mathcal{F}_{\mathsf{FE}_3,\lambda,n}\}_{\lambda\in\mathbb{N}}$ *, the* $\mathcal{S}_\eta-$**bounded indistinguishability of semi-functional ciphertexts property** *is associated with two experiments. The experiments are parameterised with* $\mathsf{aux} = (1^\lambda, 1^n, \Gamma, M_i = \{(\boldsymbol{x}_i)\}_{i\in\Gamma}, M^* = (\boldsymbol{x}), f_1, .., f_\eta)$*.*
$\underline{\mathsf{Expt}_{\mathsf{aux}}(1^\lambda, 1^n, \mathbf{b})}$*:*

1. *The challenger sets $\theta_i = f_i(\boldsymbol{x})$ for $i \in [\eta]$. The challenger computes the following:*

2. *Compute* $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$.

3. *Compute* $sk_{f_k,\theta_k} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, k, f_k, \theta_k)$, *for every* $k \in [\eta]$.

4. $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{MSK}, M_i)$, *for every* $i \in \Gamma$.

5. *If* $\mathbf{b} = 0$, *compute* $\mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{MSK}, M^*)$.

6. *If* $\mathbf{b} = 1$ *compute* $\mathsf{CT}^* \leftarrow \mathsf{sfEnc}(\mathsf{MSK}, 1^n)$.

7. *Output the following:*

   (a) CT_i *for* $i \in \Gamma$ *and* CT^*.
   (b) sk_{f_k,θ_k} *for* $k \in [\eta]$
   (c) M^* *and* $\{M_i\}_{i\in\Gamma}$
   (d) $f_1, ..., f_\eta$

A semi-functional FE scheme FE_3 *associated with plaintext space* $\mathbb{Z} = [-\delta, \delta]$ *is said to satisfy* $\eta-$**indistinguishability of semi–functional ciphertexts property** *if the following happens:* $\exists c > 0$ *such that,* $\forall \lambda > \lambda_0$, *polynomial* $n = n(\lambda)$, *polynomial* Γ, *for any messages* $\{M_i\}_{i\in\Gamma} \in \mathbb{Z}^n$, $M^* \in \mathbb{Z}^n$, $(f_1, .., f_\eta) \in \mathcal{S}_\eta$ *and any adversary* $\mathcal{A}$ *of size* 2^{λ^c},

$$|\mathsf{Pr}[\mathcal{A}(\mathsf{Expt}_{\mathsf{aux}}(1^\lambda, 1^n, 0) = 1] - \mathsf{Pr}[\mathcal{A}(\mathsf{Expt}_{\mathsf{aux}}(1^\lambda, 1^n, 1)) = 1]| \leq 1 - 1/\lambda + \mathsf{negl}(\lambda)$$

where $\mathsf{aux} = (1^\lambda, 1^n, \Gamma, M_i = \{(\boldsymbol{x}_i)\}_{i\in\Gamma}, M^* = (\boldsymbol{x}), f_1, .., f_\eta)$

If a FE_3 scheme satisfies both the above definitions, then it is said to satisfy semi-functional security.

Definition 15 ($\mathcal{S}_\eta-$Bounded Semi-functional Security)**.** *Consider a semi-functional FE scheme for cubic polynomials* FE_3 *for a class of functions* $\mathcal{F}_{\mathsf{FE}_3}$. *We say that* FE_3 *satisfies* $\mathcal{S}_\eta-$**bounded semi-functional security** *if it satisfies* $\mathcal{S}_\eta-$*bounded indistinguishability of semi-functional ciphertexts property (Definition 14) and* $\mathcal{S}_\eta-$*bounded indistinguishability of semi-functional keys property (Definition 13).*

9 Semi-Functional Functional Encryption for Circuits

In this section, we define the notion of semi-functional FE (referred to as sFE) for circuits. This definition differs from the stateful semi-functional FE for cubic polynomials (defined in the previous section) in many ways:

- First, the key generation algorithm is defined to be stateful in the previous section, whereas in this section, this algorithm is stateless.
- The functional keys are associated with different classes of functions.
- The ciphertext complexity in the previous section was defined to be linear in the message size whereas in this section, the complexity is sublinear in the circuit size.
- The adversary is issued bounded number of keys in the previous section whereas the adversary is issued only one key associated with a multi-bit output circuit.

Syntax. A Semi-Functional secret-key functional encryption scheme for a message space $\chi = \{\chi_\lambda\}_{\lambda \in \mathbb{N}}$ and a function space $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ is a tuple of PPT algorithms with the following properties:

- **Setup,** $\mathsf{Setup}(1^\lambda)$: On input security parameter λ, it outputs the master secret key MSK.
- **Encryption,** $\mathsf{Enc}(\mathsf{MSK}, x)$: On input the encryption key MSK and a message $x \in \chi_\lambda$, it outputs ciphertext CT.
- **Key Generation,** $\mathsf{KeyGen}(\mathsf{MSK}, C)$: On input the master secret key MSK and a function $C \in \mathcal{C}_\lambda$, it outputs a functional key sk_C.
- **Decryption,** $\mathsf{Dec}(sk_C, \mathsf{CT})$: On input functional key sk_C and a ciphertext CT, it outputs the result out.

We define correctness property below.

Correctness. Consider any function $C \in \mathcal{C}_\lambda$ and any plaintext $x \in \chi_\lambda$. Consider the following process:

- $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$
- $sk_C \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, C)$.
- $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, x)$

The following should hold:

$$\mathsf{Pr}\left[\mathsf{Dec}(sk_C, \mathsf{CT}) = C(x)\right] \geq 1 - \mathsf{negl}(\lambda),$$

for some negligible function negl.

Sub-Linear Efficiency: We require that for any message $\boldsymbol{x} \in \chi_\lambda$ the following holds:

- Let $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$.
- Compute $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, x)$.

The size of circuit computing CT is less than $\ell^{1-\epsilon_C} \cdot poly(\lambda, |x|)$. $poly$ is some fixed polynomial, $\epsilon_C > 0$ is some constant, $|x|$ is the length of the message x and $\ell = max\{size(C)\}_{C \in \mathcal{C}_\lambda}$.
The notion described above suffices to construct iO. We define a more general notion below.

$\rho-$ **Efficiency:** We define another notion of efficiency where we require that for any message $\boldsymbol{x} \in \chi_\lambda$ the following holds:

- Let $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$.
- Compute $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, x)$.

The size of circuit computing CT is less than $\rho(\ell, |x|, \lambda)$. Here ρ is a polynomial in $\ell = max\{size(C)\}_{C \in \mathcal{C}_\lambda}$, $|x|$ and the security parameter. Note that if ρ is of the form $\ell^{1-\epsilon} poly(\lambda, |x|)$ for some $\epsilon > 0$, then $\rho-$efficiency is the same as sublinear efficiency.

Now we define the notion of $(\mathsf{adv}, \mathsf{size}_\mathcal{A})-$Semi-functional security. Here adv is a parameter denoting advantage of adversary and $\mathsf{size}_\mathcal{A}$ is the parameter denoting the size of the adversary.

9.1 Semi-functional Security

We define the following auxiliary algorithms.

Semi-functional Key Generation, $\mathsf{sfKG}(\mathsf{MSK}, C, \theta)$: On input the master secret key MSK, function $C \in \mathcal{C}_\lambda$ and a value θ, it computes the semi-functional key $sk_{C,\theta}$.

Semi-functional Encryption, $\mathsf{sfEnc}(\mathsf{MSK}, 1^\lambda)$: On input the master encryption key MSK, and the length 1^λ, it computes a semi-functional ciphertext $\mathsf{ct_{sf}}$.

We define two security properties associated with the above auxiliary algorithms.
We now define indistinguishability of semi-functional key property.

Definition 16 (Indistinguishability of Semi-functional Key)**.** *A Semi-Functional FE scheme for circuits* sFE *for a class of functions* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ *is said to satisfy* $(\mathsf{size}_\mathcal{A})$ **-indistinguishability of semi-functional key property** *if for sufficiently large* $\lambda \in \mathbb{N}$*, for any adversary* $\mathcal{A}$ *of size* $\mathsf{size}_\mathcal{A}$*, the probability that* $\mathcal{A}$ *succeeds in the following experiment bounded by* negl.

$\mathsf{Expt}(1^\lambda, \mathbf{b})$*:*

1. *$\mathcal{A}$ specifies the following:*
   - *It can specify messages $M_j = \{x_j\}_{j \in [q]}$ for any polynomial q. Here each $M_j \in \chi_\lambda$.*
   - *It specifies function queries as follows:*
     - *It specifies $C \in \mathcal{C}_\lambda$.*
     - *It specifies values θ in output space of C.*
2. *The challenger computes the following:*
   - $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$
   - $\mathsf{CT}_j \leftarrow \mathsf{Enc}(\mathsf{MSK}, M_j)$*, for every $j \in [q]$.*
   - *If $\mathbf{b} = 0$, compute $sk^*_C \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, C)$. Otherwise, compute $sk^*_C \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C, \theta_i)$.*
3. *Challenger sends $\{\mathsf{CT}_i\}_{i \in q}$ and $\{sk^*_C\}$ to $\mathcal{A}$:*
4. *$\mathcal{A}$ outputs b'.*

The success probability of $\mathcal{A}$ is defined to be ε if $\mathcal{A}$ outputs $b' = \mathbf{b}$ with probability $\frac{1}{2} + \varepsilon$.

Definition 17 (Indistinguishability of Semi-functional Ciphertexts)**.** *For a semi-functional FE scheme* sFE *for a class of functions* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ *, the* $(\mathsf{adv}, \mathsf{size}_\mathcal{A})-$ **indistinguishability of semi-functional ciphertexts property** *is associated with two experiments. The experiments are parameterised with* $\mathsf{aux} = (1^\lambda, \Gamma, M_i = \{x_i\}_{i \in \Gamma}, M^* = x, C)$
$\underline{\mathsf{Expt}_{\mathsf{aux}}(1^\lambda, \mathbf{b})}$*:*

1. *The challenger sets $\theta = C(x)$. The challenger computes the following:*
2. *Compute* $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$.

3. *Compute* $sk_{C,\theta} \leftarrow \mathsf{sfKG}(\mathsf{MSK}, C, \theta)$.

4. $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{MSK}, M_i)$, *for every* $i \in [\Gamma]$.

5. *If* $\mathbf{b} = 0$, *compute* $\mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{MSK}, M^*)$.

6. *If* $\mathbf{b} = 1$ *compute* $\mathsf{CT}^* \leftarrow \mathsf{sfEnc}(\mathsf{MSK}, 1^\lambda)$.

7. *Output the following:*

   (a) CT_i *for* $i \in \Gamma$ *and* CT^*.
   (b) $sk_{C,\theta}$.
   (c) M^* *and* $\{M_i\}_{i\in\Gamma}$
   (d) C

A semi-functional FE scheme sFE *associated with plaintext space* χ *is said to satisfy* ($\mathsf{size}_\mathcal{A}$, adv)*-***indistinguishability of semi-functional ciphertexts property** *if the following happens:* $\forall \lambda > \lambda_0$*, any polynomial* Γ*, messages* $\{M_i\}_{i\in\Gamma} \in \chi_\lambda$*,* $M^* \in \chi_\lambda$ *,* $C \in \mathcal{C}_\lambda$ *and any adversary* $\mathcal{A}$ *of size* $\mathsf{size}_\mathcal{A}$*:*

$$|\Pr[\mathcal{A}(\mathsf{Expt}_{\mathsf{aux}}(1^\lambda, 0) = 1] - \Pr[\mathcal{A}(\mathsf{Expt}_{\mathsf{aux}}(1^\lambda, 1)) = 1]| \leq \mathsf{adv}$$

where $\mathsf{aux} = (1^\lambda, \Gamma, M_i = \{x_i\}_{i\in\Gamma}, M^* = x, C)$

Definition 18 (Semi-functional Security)**.** *Consider a semi-functional FE scheme* sFE *for a class of circuits* $\mathcal{C}_{n,s}$*. We say that* sFE *satisfies* $(\mathsf{adv}, \mathsf{size}_\mathcal{A})-$**semi-functional security** *if it satisfies* $(\mathsf{adv}, \mathsf{size}_\mathcal{A})-$*indistinguishability of semi-functional ciphertexts property (Definition 17) and* $\mathsf{size}_\mathcal{A}-$*indistinguishability of semi-functional key property (Definition 16).*

Remark 7. *Note that if we do not specify* $\mathsf{adv}, \mathsf{size}_\mathcal{A}$ *we will assume* $\mathsf{adv} = 1 - 1/\lambda + \mathsf{negl}(\lambda)$ *and* $\mathsf{size}_\mathcal{A}$ *to be* 2^{λ^c} *for some constant* $c > 0$.

Now, we rephrase the above definition of indistinguishability of semi-functional ciphertext security by using theorem 8.

Theorem 7. *Fix* $1^\lambda, 1^n, \Gamma, \{M_i\}, M^*, C$ *as above. Define two functions* E_b *for* $b \in \{0, 1\}$*, that takes as input* $\{0, 1\}^{\ell_b}$*. Here* ℓ_b *is the length of randomness required to compute the following. The functions do the following.*

Consider the following process:

1. *Compute* $\mathsf{MSK} \leftarrow \mathsf{sFE.Setup}(1^\lambda)$.

2. *Compute* $\mathsf{CT}_i \leftarrow \mathsf{sFE.Enc}(\mathsf{MSK}, M_i)$ *for* $i \in [\Gamma]$.

3. *Set* $\theta = C(M^*)$*. Compute* $sk_C \leftarrow \mathsf{sFE.sfKG}(\mathsf{MSK}, C, \theta)$.

4. *If* $b = 0$*, compute* $\mathsf{CT}^* = \mathsf{sFE.Enc}(\mathsf{MSK}, M^*)$ *and if* $b = 1$*, compute* $\mathsf{CT}^* = \mathsf{sFE.sfEnc}(\mathsf{MSK}, 1^\lambda)$.

5. *For* $b \in \{0, 1\}$*,* E_b *on input* $r \in \{0, 1\}^{\ell_b}$ *outputs* $\{\mathsf{CT}_i\}_{i\in\Gamma}, sk_C, \mathsf{CT}^*$.

If sFE *satisfies* $(\mathsf{size}_{\mathcal{A}}, \mathsf{adv})-$*indistinguishability of semi-functional ciphertexts property, then, there exists two computable (not necessarily efficient) measures* $\mathcal{M}_0$ *and* $\mathcal{M}_1$ *(*$\mathcal{M}_b$ *defined over* $\{0,1\}^{\ell_b}$ *for* $b \in \{0,1\}$*) of density exactly* $1-\mathsf{adv}/2$ *such that, for all circuits* $\mathcal{A}$ *of size* $\mathsf{size}'_{\mathcal{A}} > \mathsf{size}_{\mathcal{A}}\mathsf{adv}'^2/128(\ell_0+\ell_1+1)$*,*

$$|\Pr_{u \leftarrow \mathcal{D}_{\mathcal{M}_0}}[\mathcal{A}(E_0(u)) = 1] - \Pr_{v \leftarrow \mathcal{D}_{\mathcal{M}_1}}[\mathcal{A}(E(v)) = 1]| < \mathsf{adv}'$$

Here both measures may depend on $(\{M_i\}_{i\in\Gamma}, C, M^*)$

Proof. We invoke theorem 8 to prove this. We recall the theorem below:

Theorem 8 (Imported Theorem [MT10])**.** *Let* $E : \{0,1\}^n \to \mathcal{X}$ *and* $F : \{0,1\}^m \to \mathcal{X}$ *be two functions, and let* $\epsilon, \gamma \in (0,1)$ *and* $s > 0$ *be given. If for all distinguishers* $\mathcal{A}$ *with size* s *we have*

$$|\Pr_{x \leftarrow \{0,1\}^n}[\mathcal{A}(E(x)) = 1] - \Pr_{y \leftarrow \{0,1\}^m}[\mathcal{A}(F(y)) = 1]| \leq \epsilon$$

Then there exist two measures $\mathcal{M}_0$ *(on* $\{0,1\}^n$*) and* $\mathcal{M}_1$ *(on* $\{0,1\}^n$*) that depend on* γ, s *such that:*

- $\mu(\mathcal{M}_b) \geq 1 - \epsilon$ *for* $b \in \{0,1\}$
- *For all distinguishers* $\mathcal{A}'$ *of size* $s' = \frac{s\gamma^2}{128(m+n+1)}$

$$|\Pr_{x \leftarrow \mathcal{D}_{\mathcal{M}_0}}[\mathcal{A}(E(x)) = 1] - \Pr_{y \leftarrow \mathcal{D}_{\mathcal{M}_1}}[\mathcal{A}(F(y)) = 1]| \leq \gamma$$

Due to security of sFE, we know that for any adversary $\mathcal{A}$ of size $s = \mathsf{size}_{\mathcal{A}}$,

$$|\Pr_{u \leftarrow \{0,1\}^{\ell_0}}[\mathcal{A}(E_0(u)) = 1] - \Pr_{v \leftarrow \{0,1\}^{\ell_1}}[\mathcal{A}(E(v)) = 1]| < \mathsf{adv}$$

Thus, there exists two measures $\mathcal{M}'_0$ (on $\{0,1\}^{\ell_0}$) and $\mathcal{M}'_1$ (on $\{0,1\}^{\ell_1}$) with density at least $1-\mathsf{adv}$ such that for all adversaries $\mathcal{A}'$ of size $\mathsf{size}'_{\mathcal{A}} = \mathsf{size}_{\mathcal{A}}\mathsf{adv}'^2/128(\ell_0+\ell_1+1)$,

$$|\Pr_{u \leftarrow \mathcal{D}_{\mathcal{M}'_0}}[\mathcal{A}(E_0(u)) = 1] - \Pr_{v \leftarrow \mathcal{D}_{\mathcal{M}'_1}}[\mathcal{A}(E(v)) = 1]| < \mathsf{adv}'$$

Now define $\mathcal{M}_b = (\frac{1-\mathsf{adv}/2}{\mu(\mathcal{M}'_b)})\mathcal{M}'_b$ for $b \in \{0,1\}$. Note that the constants $\frac{1-\mathsf{adv}/2}{\mu(\mathcal{M}'_b)} < 1$ as the density $\mu(\mathcal{M}'_b) \geq 1 - \mathsf{adv}$. Thus, these measures can be scaled so that their density is exactly $1 - \mathsf{adv}/2$. Since, $\mathcal{M}_b$ induce the same distribution as $\mathcal{M}'_b$ for $b \in \{0,1\}$, the claim holds.

□

10 Step 1: Instantiating TCE

The main building block in the construction of a TCE scheme is a perturbation-resilient generator. We show how to combine a perturbation-resilient generator with techniques from [GSW13] to obtain our TCE candidate. Now we describe some preliminaries.

10.1 Construction of TCE

This scheme will be parameterized by d, $\mathbf{p}$, η, $\mathcal{S}_\eta$, $\mathsf{TCEbound}$, B_3 and B_4.

Non-commutative Product Lemma. We state a non-commutative product lemma that will be useful for our construction. In particular, the function F_{ncp} described in the below lemma will be used in the decode algorithm.

We define the tensor of a vector $\boldsymbol{a} \in \mathbb{Z}_q^{1\times \mathsf{d}}$ and a matrix $\mathbf{V} \in \mathbb{Z}_q^{m\times m}$ to be a $1 \times \mathsf{d}m^2$-dimensional vector with the $((i-1)m + (j-1)m + k)^{th}$ entry, for every $i \in [m], j \in [m], k \in [\mathsf{d}]$, in $\boldsymbol{a} \otimes \mathbf{V}$ to be $v_{ij} \cdot a_k$, where v_{ij} is the $(i,j)^{th}$ element of $\mathbf{V}$ and a_k is the k^{th} element of $\boldsymbol{a}$.

Lemma 1 (Non-commutative Product Lemma)**.** *Suppose we have a vector $\boldsymbol{a} \in \mathbb{Z}_q^{1\times \mathsf{d}}$, matrices $\mathbf{U} \in \mathbb{Z}_q^{\mathsf{d}\times m}, \mathbf{V} \in \mathbb{Z}_q^{m\times m}$. There is a function $\mathsf{F}_{ncp} : \mathbb{Z}_q^{1\times \mathsf{d}m^2} \times \mathbb{Z}_q^{\mathsf{d}\times m} \to \mathbb{Z}_q^{1\times m}$ that given $\boldsymbol{a} \otimes \mathbf{V}$ and $\mathbf{U}$, computes $\boldsymbol{a}\mathbf{U}\mathbf{V}$. That is, $\mathsf{F}_{ncp}(\boldsymbol{a} \otimes \mathbf{V}, \mathbf{U})$ outputs $\boldsymbol{a}\mathbf{U}\mathbf{V}$. Moreover, $\mathsf{F}_{ncp}(\boldsymbol{a} \otimes \mathbf{V}, \mathbf{U}) = (\mathsf{q}_1(\boldsymbol{a} \otimes \mathbf{V}, \mathbf{U}), \ldots, \mathsf{q}_m(\boldsymbol{a} \otimes \mathbf{V}, \mathbf{U}))$, where q_i can be expressed as a quadratic polynomial (over $\mathbb{Z}_q$) with every term being a product of an element in $\boldsymbol{a} \otimes \mathbf{V}$ and an element in $\mathbf{U}$.*

Proof. Let $\boldsymbol{a} = [a_1 \cdots a_\mathsf{d}]$. The $(i,j)^{th}$ element in $\mathbf{U}$ is denoted by $u_{i,j}$, for every $i \in [\mathsf{d}], j \in [m]$. The $(i,j)^{th}$ element in $\mathbf{V}$ is denoted by $v_{i,j}$.

Observe that the i^{th} element, for every $i \in [m]$, in $\boldsymbol{a}\mathbf{U}$ is denoted by $\sum_{j=1}^m a_j u_{ij}$. The i^{th} element in $\boldsymbol{a}\mathbf{U}\mathbf{V}$, for $i \in [m]$, is denoted by $\sum_{k=1}^m (\sum_{j=1}^{\mathsf{d}} a_j u_{kj}) \cdot v_{ik}$. The expression $\sum_{k=1}^m (\sum_{j=1}^n a_j u_{kj}) \cdot v_{ik}$ can be rewritten as, $\sum_{k=1}^m \sum_{j=1}^{\mathsf{d}} (a_j v_{ik}) \cdot u_{kj}$. Recall that $\boldsymbol{a} \otimes \mathbf{V}$ is a vector consisting of $a_j v_{ik}$, for every $i \in [\mathsf{d}], j \in [m], k \in [m]$. Thus, $\sum_{k=1}^m \sum_{j=1}^{\mathsf{d}} (a_j v_{ik}) \cdot u_{kj}$ is a quadratic polynomial, denoted by q_i, with every term being a product of an element in $\boldsymbol{a}\otimes\mathbf{V}$ and an element in $\mathbf{U}$. Thus, $\mathsf{q}_i(\boldsymbol{a}\otimes\mathbf{V}, \mathbf{U})$ computes the i^{th} element in $\boldsymbol{a}\mathbf{U}\mathbf{V}$, for $i \in [m]$. This completes the proof. ☐

Construction. We describe the scheme TCE below. This scheme will be parameterized by d, $\mathbf{p}$, η, $\mathcal{S}_\eta$, $\mathsf{TCEbound}$, B_3 and B_4.

- **Setup,** $\mathsf{Setup}(1^\lambda, 1^n)$: On input security parameter λ, input length bound n, it sets $\mathsf{params} = (1^\lambda, 1^n, \mathbf{p}, B)$. Here $\mathbf{p}$ is the modulus, which is also the modulus of the three restricted FE scheme.
- **SetupEncode,** $\mathsf{SetupEnc}(\mathsf{params})$: On input $\mathsf{params} = (1^\lambda, 1^n, \mathbf{p}, B)$, compute the following:
  1. Sample $\boldsymbol{t} \xleftarrow{\$} \mathbb{Z}_\mathbf{p}^{\mathsf{d}\times 1}$ and $\mathbf{C} \xleftarrow{\$} \mathbb{Z}_\mathbf{p}^{\mathsf{d}\times m}$.
  2. Set $\boldsymbol{b} = \mathbf{C}^\mathbf{T}\boldsymbol{t} + \mathbf{e}^\mathbf{T}$, where $\mathbf{e} \leftarrow \chi^{1\times m}$ with $||e||_\infty \leq B_3$.
  3. Set $\mathbf{A} = [\mathbf{C}^\mathbf{T}||\boldsymbol{b}]^\mathbf{T}$ in $\mathbb{Z}_\mathbf{p}^{(\mathsf{d}+1)\times m}$.
  4. Also set $\boldsymbol{s} = (\boldsymbol{t}^\mathbf{T}, -1)$ in $\mathbb{Z}_\mathbf{p}^{1\times(\mathsf{d}+1)}$
  5. Sample $\mathsf{Seed} \leftarrow 3\Delta\mathsf{RG.SetupSeed}(1^\lambda, 1^n, B)$. Without loss of generality assume that $\mathsf{Seed} = (\mathsf{Seed.pub}, \mathsf{Seed.priv}(1), \mathsf{Seed.priv}(2))$. Here $\mathsf{Seed.pub} = (\mathsf{Seed}_{\mathsf{pub},1}, ..., \mathsf{Seed}_{\mathsf{pub},n})$ and $\mathsf{Seed.priv}(i) = (\mathsf{Seed}_{\mathsf{priv}(i),1}, ..., \mathsf{Seed}_{\mathsf{priv}(i),n})$ for $i \in [1,2]$ are vectors in $\mathbb{Z}_\mathbf{p}^n$.
  6. Output $\mathsf{sp} = (\boldsymbol{s}, \mathbf{A}, \mathsf{Seed})$

- **Encode,** $\mathsf{Encode}(\mathsf{sp}, x, \mathsf{ind}, \ell)$: On input $\mathsf{sp} = (\boldsymbol{s}, \mathbf{A}, \mathsf{Seed})$, plaintext $x \in [-\rho, \rho]$, index ind and level $\ell \in [3]$, proceed according to the three cases:

  Sample uniformly $\mathbf{R}_{\mathsf{ind},\ell} \xleftarrow{\$} \{0,1\}^{m\times m}$. Let $\mathbf{G} \in \mathbb{Z}_{\mathbf{p}}^{(\mathsf{d}+1)\times m}$ denote the gadget matrix and let its inverse function be $\mathbf{G}^{-1}(\cdot)$, as given in Definition 7. Set $\phi \in \mathbb{Z}_{\mathbf{p}}$ such that $\phi \boldsymbol{s}\mathbf{G}\mathbf{e}_m = \lfloor \frac{\mathbf{p}}{2n^3 B_4 \rho^3} \rfloor$, where $\mathbf{e}_m$ is an indicator vector of dimension m with the m^{th} position containing 1 and the rest of the elements are zero. Compute $([\mathbf{x}]_{\mathsf{ind},\ell}.\mathsf{pub}, [\mathbf{x}]_{\mathsf{ind},\ell}.\mathsf{priv}(1), [\mathbf{x}]_{\mathsf{ind},\ell}.\mathsf{priv}(2))$ according to Figure 2.

	$[\mathbf{x}]_{\mathsf{ind},\ell}.\mathsf{pub}$	$[\mathbf{x}]_{\mathsf{ind},\ell}.\mathsf{priv}(1)$	$[\mathbf{x}]_{\mathsf{ind},\ell}.\mathsf{priv}(2)$
$\ell = 1$	$(\mathbf{A}\mathbf{R}_{\mathsf{ind},\ell} + x\phi\mathbf{G},\ \mathsf{Seed}_{\mathsf{pub},\mathsf{ind}})$	$(x\phi,\ \mathsf{Seed}_{\mathsf{priv}(1),\mathsf{ind}})$	$(1,\ \mathsf{Seed}_{\mathsf{priv}(2),\mathsf{ind}})$
$\ell = 2$	$\mathbf{A}\mathbf{R}_{\mathsf{ind},\ell} + x\phi^{-1}\mathbf{G}$	$\mathbf{G}^{-1}(-\mathbf{A}\mathbf{R}_{\mathsf{ind},\ell})$	$x\phi^{-1}\boldsymbol{s}$
$\ell = 3$	$\mathbf{A}\mathbf{R}_{\mathsf{ind},\ell} + x\phi\mathbf{G}$	1	$\boldsymbol{s} \otimes \mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_{\mathsf{ind},\ell})$

Figure 2: Public and private encodings provided at all the three levels.

  We also assume that all these public and private parts of the encodings are padded appropriately with string consisting of zeroes such that their lengths are same. This length is equal to $\ell_{enc} = (\mathsf{d}+1) \times m \times m \log \mathbf{p}$, which is computed from the length of $[\mathbf{x}]_{\mathsf{ind},\ell}.\mathsf{priv}(2)$.

  Output $([\mathbf{x}]_{\mathsf{ind},\ell}.\mathsf{pub}, [\mathbf{x}]_{\mathsf{ind},\ell}.\mathsf{priv}(1), [\mathbf{x}]_{\mathsf{ind},\ell}.\mathsf{priv}(2))$.

- **Setup-Decode,** $\mathsf{SetupDec}(\mathsf{params})$: On input $\mathsf{params} = (1^\lambda, 1^n, \mathbf{p}, B)$, compute $3\Delta\mathsf{RG}.\mathsf{SetupPoly}(1^\lambda, 1^n, B)$ to obtain the polynomials $q_1, ..., q_\eta$.

- **Decode,** $\mathsf{Decode}(q, f, \{U_i\}_{i\in[n]}, \{V_j\}_{j\in[n]}, \{W_k\}_{k\in[n]})$: Let $f = \Sigma_{i,j,k\in[n]}\ \gamma_{i,j,k} x_i y_j z_k \in \mathcal{S}_\eta$. $\{U_i\}_{i\in[n]}$ are encodings computed with respect to first level, $\{V_j\}_{j\in[n]}$ are encodings with respect to second level and $\{W_k\}_{k\in[n]}$ are encodings computed with respect to the third level. Parse U_i as follows: $U_i.\mathsf{pub} = (Q_{i,\mathsf{pub}}, \mathsf{Seed}_{\mathsf{pub},i})$, $U_i.\mathsf{priv}(1) = (Q_{i,\mathsf{priv}(1)}, \mathsf{Seed}_{\mathsf{priv}(1),i})$ and $U_i.\mathsf{priv}(2) = (Q_{i,\mathsf{priv}(2)}, \mathsf{Seed}_{\mathsf{priv}(2),i})$. Consider the following operations:

  - **Computing a monomial**: for every monomial of the form $x_i y_j z_k$, compute the following polynomial,

  $$Z_{ijk} = \left(Q_{i,\mathsf{priv}(1)} \times V_j.\mathsf{priv}(2) \times W_k.\mathsf{pub}\right) + \mathsf{F}_{ncp}\left(W_k.\mathsf{priv}(2),\ Q_{i,\mathsf{pub}} \times V_j.\mathsf{priv}(1) - V_j.\mathsf{pub} \times Q_{i,\mathsf{priv}(1)}\right),$$

    where F_{ncp} is the function guaranteed by Lemma 1.

  Output $\left|\left(\sum_{i,j,k} \gamma_{i,j,k} Z_{ijk}\right)\mathbf{e}_m + q(\mathsf{Seed})\right|$. Note that q is a multilinear cubic polynomial.

Setting of Parameters.

- $d = \lambda^{c_1}$ for some constant $c_1 > 0$
- Let $\mathbf{p} = O(2^{\lambda^{c_2}})$ be a prime and $m = d\lceil \log \mathbf{p} \rceil$ for some constant $c_2 > 0$ such that $\mathsf{LWE}_{\mathsf{d},m,\mathbf{p}\chi}$ holds for a distribution χ bounded by a polynomial $B_3(\lambda)$.
- Let $\eta = \ell = n^{1+\epsilon}$ be the stretch of $3\Delta\mathsf{RG}$ for some constant $\epsilon > 0$.

- $\mathcal{S}_\eta$: We set $\mathcal{S}_\eta$ to be $(\mathcal{F}_{n,B_4})^\eta$. Here, $\mathcal{F}_{n,B_4}$ is the set of homogenous cubic polynomials with sum of absolute value of coefficients in $[-B_4, B_4]$ for some polynomial $B_4(\lambda)$. This choice turns out to be sufficient to construct iO. Looking ahead, these polynomials will come from the set of degree three randomizing polynomials [LT17], which satisfy this property.

- B: The bound B is set to be $m^3 B_3 B_4$. This is computed as the maximum norm on the encodings for any function $f \in \mathcal{F}_{n,B_4}$ before smudging with $3\Delta\mathsf{RG}$ values.

- $\mathsf{TCEbound}$: $\mathsf{TCEbound}$ is the maximum norm of the decoded value for any function $f \in \mathcal{F}_{n,B_4}$ which evaluates to 0. We set $\mathsf{TCEbound}$ to be $n^3 m^3 B_3 + poly(\lambda, n)$, where $poly(\lambda, n)$ is the bound on the output of $3\Delta\mathsf{RG}$.

We now prove the following properties.

Correctness. First, we prove correctness of homomorphic evaluation with respect to a cubic monomial. Then, we show how to generalize this to homomorphic evaluation of cubic polynomials.

Consider plaintexts $x_i, x_j, x_k \in [-\rho, \rho]$, indices $i, j, k \in [n]$. Generate $\mathsf{Setup}(1^\lambda, 1^n)$ to obtain $\mathsf{params} = (1^\lambda, 1^n, \mathbf{p}, B)$. Generate $\mathsf{SetupEnc}(\mathsf{params})$ to obtain $\mathsf{sp} = (\boldsymbol{s}, \mathbf{A}, \mathsf{Seed})$. Compute the following three encodings:

- $U_i \leftarrow \mathsf{Encode}(\mathsf{sp}, x_i, i, 1)$
- $V_j \leftarrow \mathsf{Encode}(\mathsf{sp}, x_j, j, 2)$
- $W_k \leftarrow \mathsf{Encode}(\mathsf{sp}, x_k, k, 3)$

Parse U_i as follows: $U_i.\mathsf{pub} = (Q_{i,\mathsf{pub}}, \mathsf{Seed}_{\mathsf{pub},i})$, $U_i.\mathsf{priv}(1) = (Q_{i,\mathsf{priv}(1)}, \mathsf{Seed}_{\mathsf{priv}(1),i})$ and $U_i.\mathsf{priv}(2) = (Q_{i,\mathsf{priv}(2)}, \mathsf{Seed}_{\mathsf{priv}(2),i})$. Perform the following operations.

- **Computing** $Int_1 = Q_{i,\mathsf{priv}(1)} \times V_j.\mathsf{priv}(2) \times W_k.\mathsf{pub}$:

$$\begin{aligned} Int_1 &= x_i\phi \cdot x_j\phi^{-1}\boldsymbol{s} \cdot (\mathbf{A}\mathbf{R}_k + x_k\phi\mathbf{G}) \\ &= x_i x_j \boldsymbol{s}\mathbf{A}\mathbf{R}_k + x_i x_j x_k \phi \boldsymbol{s}\mathbf{G} \end{aligned}$$

- **Computing** $Int_2 = (Q_{i,\mathsf{pub}} \times V_j.\mathsf{priv}(1) - V_j.\mathsf{pub} \times Q_{i,\mathsf{priv}(1)})$:

$$\begin{aligned} Int_2 &= ((\mathbf{A}\mathbf{R}_i + x_i\phi\mathbf{G}) \times \mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_j) - (\mathbf{A}\mathbf{R}_j + x_j\phi^{-1}\mathbf{G}) \times x_i\phi \\ &= \mathbf{A}\mathbf{R}_i\mathbf{G}^{-1}(\mathbf{A}\cdot\mathbf{R}_j) + x_i\phi\mathbf{A}\mathbf{R}_j - x_i\phi\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_j) - x_i x_j\mathbf{G} \\ &= \mathbf{A}\mathbf{R}_i\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_j) - x_i x_j\mathbf{G} \end{aligned}$$

- **Computing** $Int_3 = \mathsf{F}_{ncp}(W_k.\mathsf{priv}(2), Int_2)$: Recall that $W_k = \boldsymbol{s} \otimes \mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_k)$. From Lemma 1, we have

$$\begin{aligned} Int_3 &= \mathsf{F}_{ncp}(W_k.\mathsf{priv}(2), Int_2) \\ &= \boldsymbol{s} \times \big(\mathbf{A}\mathbf{R}_i\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_j) - x_i x_j\mathbf{G}\big) \times \mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_k) \\ &= \boldsymbol{s}\mathbf{A}\mathbf{R}_i\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_j)\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_k) - x_i x_j \boldsymbol{s}\mathbf{A}\mathbf{R}_k \end{aligned}$$

- **Computing** $Int = Int_1 + Int_3$:

$$Int_{ijk} = \boldsymbol{s}\mathbf{A}\mathbf{R}_i\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_j)\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_k) + x_i x_j x_k \phi \boldsymbol{s}\mathbf{G}$$

We calculate $|Int_{ijk} \times \mathbf{e}_m|$ below.

$$\begin{aligned}
|Int_{ijk} \times \mathbf{e}_m| &= |\left(\boldsymbol{s}\mathbf{A}\mathbf{R}_i\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_j)\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_k) + x_i x_j x_k \phi \boldsymbol{s}\mathbf{G}\right)\mathbf{e}_m| \\
&\leq |\left(\boldsymbol{s}\mathbf{A}\mathbf{R}_i\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_j)\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_k)\right)\mathbf{e}_m| + |x_i x_j x_k \phi \boldsymbol{s}\mathbf{G}\mathbf{e}_m| \\
&= |\left(\boldsymbol{s}\mathbf{A}\mathbf{R}_i\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_j)\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_k)\right)\mathbf{e}_m| + x_i x_j x_k \left\lfloor \frac{\mathbf{p}}{2n^3 B_4 \rho^3} \right\rfloor \\
&\leq m^3 ||\boldsymbol{s}\mathbf{A}||_\infty \cdot ||\mathbf{R}_i||_\infty \cdot ||\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_j)||_\infty \cdot ||\mathbf{G}^{-1}(\mathbf{A}\mathbf{R}_k)||_\infty + x_i x_j x_k \left\lfloor \frac{\mathbf{p}}{2n^3 B_4 \rho^3} \right\rfloor \\
&\leq m^3 B_3 + x_i x_j x_k \left\lfloor \frac{\mathbf{p}}{2n^3 B_4 \rho^3} \right\rfloor
\end{aligned}$$

We now prove the correctness of evaluation of a polynomial $f(x_1, \ldots, x_n) = \sum_{i,j,k \in [n]} \gamma_{i,j,k} x_i x_j x_k$. We have,

$$\mathsf{Decode}(q, f, \{U_i\}_{i \in [n]}, \{V_j\}_{j \in [n]}, \{W_k\}_{k \in [n]}) = \sum_{i,j,k \in [n]} \gamma_{i,j,k} Int_{ijk} \times \mathbf{e}_m + q(\mathsf{Seed})$$

There are two cases:

- **Case** $f(x_1, \ldots, x_n) = 0$:

$$\begin{aligned}
\left|\mathsf{Decode}(q, f, \{U_i\}_{i \in [n]}, \{V_j\}_{j \in [n]}, \{W_k\}_{k \in [n]})\right| &\leq \left|\sum_{i,j,k \in [n]} \gamma_{i,j,k} Int_{ijk} \times \mathbf{e}_m\right| + |q(\mathsf{Seed})| \\
&\leq \sum_{i,j,k \in [n]} \gamma_{i,j,k} m^3 B_3 + \left(\sum_{i,j,k \in [n]} \gamma_{i,j,k} x_i x_j x_k\right) \cdot \left\lfloor \frac{\mathbf{p}}{2n^3 B_4 \rho^3} \right\rfloor \\
&\quad + q(\mathsf{Seed}) \\
&= \sum_{i,j,k \in [n]} \gamma_{i,j,k} m^3 B_3 + q(\mathsf{Seed}) \\
&\leq n^3 m^3 B_3 + poly(\lambda, n)
\end{aligned}$$

  Note that the last inequality follows from the efficiency property of $3\Delta\mathsf{RG}$.

- **Case** $f(x_1, \ldots, x_n) = 1$:

$$\begin{aligned}
\mathsf{Decode}(q, f, \{U_i\}_{i \in [n]}, \{V_j\}_{j \in [n]}, \{W_k\}_{k \in [n]}) &\leq \sum_{i,j,k \in [n]} \gamma_{i,j,k} m^3 B_3 + \left(\sum_{i,j,k \in [n]} \gamma_{i,j,k} x_i x_j x_k\right) \cdot \left\lfloor \frac{\mathbf{p}}{2n^3 B_4 \rho^3} \right\rfloor \\
&\quad + q(\mathsf{Seed}) \\
&\leq n^3 m^3 B_3 + \left\lfloor \frac{\mathbf{p}}{2} \right\rfloor + poly(\lambda, n)
\end{aligned}$$

Also,

$$\begin{aligned}
\mathsf{Decode}(q, f, \{U_i\}_{i\in[n]}, \{V_j\}_{j\in[n]}, \{W_k\}_{k\in[n]}) &\geq \left(\sum_{i,j,k\in[n]} \gamma_{i,j,k} x_i x_j x_k\right) \cdot \left\lfloor \frac{\mathbf{p}}{2n^3 B_4 \rho^3} \right\rfloor \\
&\geq \left\lfloor \frac{\mathbf{p}}{2n^3 B_4 \rho^3} \right\rfloor \\
&\geq n^3 m^3 B_3 + poly(\lambda, n)
\end{aligned}$$

The last inequality holds because $\mathbf{p} = O(2^{\lambda^{c_2}})$, for some constant c_2, and the parameters n, m, B_3 are polynomial in λ.

Cubic Evaluation Property. The cubic evaluation property can be observed from the description of Decode.

Security. We prove security below.

Theorem 9. *The above scheme satisfies tempered security assuming that* $3\Delta\mathsf{RG}$ *is a secure perturbation-resilient generator implementable by a three restricted FE scheme and learning with errors.*

Proof. We first describe the simulator associated with the above scheme.

$\underline{\mathsf{Sim}(q_j, f_j, \{[\mathbf{x_i}]_{i,1}\}_{i\in[n]}, \{[\mathbf{y_i}]_{i,2}\}_{i\in[n]}, \{[\mathbf{z_i}]_{i,3}\}_{i\in[n]}, f_j(\mathbf{x},\mathbf{y},\mathbf{z}))}$: On input polynomial q_j, function f_j associated with index $j \in [\eta]$, encodings $\{[\mathbf{x_i}]_{i,1}\}_{i\in[n]}$, $\{[\mathbf{y_i}]_{i,2}\}_{i\in[n]}$, $\{[\mathbf{z_i}]_{i,3}\}_{i\in[n]}$ and output $f_j(\mathbf{x},\mathbf{y},\mathbf{z})$,

- Parse $[\mathbf{x_i}]_{i,1}.\mathsf{pub} = (Q_{i,\mathsf{pub}}, \mathsf{Seed}_{\mathsf{pub},i})$ and $[\mathbf{x_i}]_{i,1}.\mathsf{priv}(1) = (Q_{i,\mathsf{priv}(1)}, \mathsf{Seed}_{\mathsf{priv}(1),i})$ and $[\mathbf{x_i}]_{i,1}.\mathsf{priv}(2) = (Q_{i,\mathsf{priv}(2)}, \mathsf{Seed}_{\mathsf{priv}(2),i})$.
- Compute $(e'_1, \ldots, e'_\eta) \leftarrow (q_1(\mathsf{Seed}), ..., q_\eta(\mathsf{Seed}))$.
- Set $\widehat{\mathsf{leak}_j} \leftarrow e'_j + f_j(\mathbf{x},\mathbf{y},\mathbf{z}) \cdot \left\lfloor \frac{\mathbf{p}}{2n^3 B_4 \rho^3} \right\rfloor$.
- Output $\widehat{\mathsf{leak}_j}$.

We describe the hybrids below. Let $\mathsf{aux} = (1^\lambda, 1^n, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, f_1, ..., f_\eta)$. Each vector $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ is in $\mathbb{Z}^n$.

$\mathbf{Hybrid}_1$: This corresponds to the real experiment. In particular, the output of this hybrid is:

1. Challenger performs $\mathsf{Setup}(1^\lambda, 1^n) \to \mathsf{params}$
2. The challenger samples $(q_1, ..., q_\eta) \leftarrow \mathsf{SetupDec}(\mathsf{params})$.
3. Challenger performs $\mathsf{SetupEnc}(\mathsf{params}) \to \mathsf{sp}$.
4. Now compute encodings as follows.
   - Compute the encodings, $[\mathbf{x_i}]_{i,1} \leftarrow \mathsf{Encode}(\mathsf{sp}, x_i, i, 1)$ for every $i \in [n]$.
   - Compute the encodings, $[\mathbf{y_i}]_{i,2} \leftarrow \mathsf{Encode}(\mathsf{sp}, y_i, i, 2)$ for every $i \in [n]$.

- Compute the encodings, $[\mathbf{z_i}]_{i,3} \leftarrow \mathsf{Encode}(\mathsf{sp}, z_i, i, 3)$ for every $i \in [n]$.

5. Compute $\mathsf{leak}_j \leftarrow \mathsf{Decode}(q_j, f_j, \{[\mathbf{x_i}]_{i,1}\}_{i\in[n]}, \{[\mathbf{y_i}]_{i,2}\}_{i\in[n]}, \{[\mathbf{z_i}]_{i,3}\}_{i\in[n]})$ for $j \in [\eta]$.

6. Output the following:

    (a) Public components of the encodings, $\{[\mathbf{x_i}]_{i,1}.\mathsf{pub}, [\mathbf{y_i}]_{i,2}.\mathsf{pub}, [\mathbf{z_i}]_{i,3}.\mathsf{pub}\}_{i\in[n]}$.

    (b) Decoding Parameters, $\{q_j\}_{j\in[\eta]}$.

    (c) Output of decodings, $\{\mathsf{leak}_j\}_{j\in[\eta]}$.

Hybrid$_2$: In this hybrid, the leakage output by decode is instead generated by the simulator.

1. Challenger performs $\mathsf{Setup}(1^\lambda, 1^n) \to \mathsf{params}$

2. The challenger samples $(q_1, ..., q_\eta) \leftarrow \mathsf{SetupDec}(\mathsf{params})$.

3. Challenger performs $\mathsf{SetupEnc}(\mathsf{params}) \to \mathsf{sp}$.

4. Now compute encodings as follows.

    - Compute the encodings, $[\mathbf{x_i}]_{i,1} \leftarrow \mathsf{Encode}(\mathsf{sp}, x_i, i, 1)$ for every $i \in [n]$.
    - Compute the encodings, $[\mathbf{y_i}]_{i,2} \leftarrow \mathsf{Encode}(\mathsf{sp}, y_i, i, 2)$ for every $i \in [n]$.
    - Compute the encodings, $[\mathbf{z_i}]_{i,3} \leftarrow \mathsf{Encode}(\mathsf{sp}, z_i, i, 3)$ for every $i \in [n]$.

5. Compute $\{\widehat{\mathsf{leak}_j}\}_{j\in[\eta]} \leftarrow \mathsf{Sim}\left(q_j, f_j, \{[\mathbf{x_i}]_{i,1}\}_{i\in[n]}, \{[\mathbf{y_i}]_{i,2}\}_{i\in[n]}, \{[\mathbf{z_i}]_{i,3}\}_{i\in[n]}, f_j(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\right)$.

6. Output the following:

    (a) Public components of the encodings, $\{[\mathbf{x_{b,i}}]_{i,1}.\mathsf{pub}, [\mathbf{y_{b,i}}]_{i,2}.\mathsf{pub}, [\mathbf{z_{b,i}}]_{i,3}.\mathsf{pub}\}_{i\in[n]}$.

    (b) Decoding parameters, $\{q_j\}_{j\in[\eta]}$.

    (c) Output of decodings, $\{\widehat{\mathsf{leak}_j}\}_{j\in[\eta]}$.

Claim 1. *Assuming that* $3\Delta\mathsf{RG}$ *is* $(1 - 1/\lambda)$*-secure against any adversary* $\mathcal{A}$ *of size at most* 2^λ*,* $|\Pr[\mathcal{A}(\mathbf{Hybrid}_1) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_2)]| \leq 1 - 1/\lambda$

Proof. The only difference between $\mathbf{Hybrid}_1$ and $\mathbf{Hybrid}_2$ is in how the η number of leakages are generated. In $\mathbf{Hybrid}_1$, the j^{th} leakage is of the form $q_j(\cdot) + a_j$. Note that $a_j = e_{j,fhe} + f_j(\mathbf{x}, \mathbf{y}, \mathbf{z}) \cdot \left\lfloor \frac{\mathbf{p}}{2n^3 B_4 \rho^3} \right\rfloor$, where $e_{j,fhe}$ is some value in the range $[-B, B]$. In $\mathbf{Hybrid}_2$, the j^{th} leakage is of the form $\widehat{\mathsf{leak}}_j = q_j(\cdot) + f_j(\mathbf{x}, \mathbf{y}, \mathbf{z}) \cdot \left\lfloor \frac{\mathbf{p}}{2n^3 B_4 \rho^3} \right\rfloor$.

Suppose the output distributions of $\mathbf{Hybrid}_1$ and $\mathbf{Hybrid}_2$ are computationally distinguishable with probability greater than $1 - 1/\lambda + \mathsf{negl}(\lambda)$, we can design an attacker that breaks the $\Delta\mathsf{RG}$ assumption as follows. This attacker first generates $(e_{1,fhe}, \ldots, e_{\eta,fhe})$: this is performed by first generating the TCE encodings and then computing $(e_{1,fhe}, \ldots, e_{\eta,fhe})$ as a function of these encodings. The attacker submits this tuple to the challenger of the $3\Delta\mathsf{RG}$. The challenger returns the polynomials $(q_1, \ldots, q_\eta)$ along with $(\theta_1, \ldots, \theta_\eta)$. The attacker then submits the TCE encodings along with $(q_1, \ldots, q_\eta)$ and $(\theta_1 + f_1(\mathbf{x}, \mathbf{y}, \mathbf{z}) \cdot \left\lfloor \frac{\mathbf{p}}{2n^3 B_4 \rho^3} \right\rfloor, \ldots, \theta_\eta + f_1(\mathbf{x}, \mathbf{y}, \mathbf{z}) \cdot \left\lfloor \frac{\mathbf{p}}{2n^3 B_4 \rho^3} \right\rfloor)$ to the distinguisher (who distinguishes $\mathbf{Hybrid}_1$ and $\mathbf{Hybrid}_2$). The output of the attacker is the same as the output of the distinguisher. Thus, if the distinguisher distinguishes with probability ε then the attacker breaks $3\Delta\mathsf{RG}$ with probability ε. □

$\mathbf{Hybrid}_3$: In this hybrid, generate the encodings as encodings of zeroes. In particular, execute the following operations.

1. Challenger performs $\mathsf{Setup}(1^\lambda, 1^n) \rightarrow \mathsf{params}$
2. The challenger samples $(q_1, ..., q_\eta) \leftarrow \mathsf{SetupDec}(\mathsf{params})$.
3. Challenger performs $\mathsf{SetupEnc}(\mathsf{params}) \rightarrow \mathsf{sp}$.
   - Compute the encodings, $[\mathbf{x_i}]_{i,1} \leftarrow \mathsf{Encode}(\mathsf{sp}, 0, i, 1)$ for every $i \in [n]$.
   - Compute the encodings, $[\mathbf{y_i}]_{i,2} \leftarrow \mathsf{Encode}(\mathsf{sp}, 0, i, 2)$ for every $i \in [n]$.
   - Compute the encodings, $[\mathbf{z_i}]_{i,3} \leftarrow \mathsf{Encode}(\mathsf{sp}, 0, i, 3)$ for every $i \in [n]$.
4. Compute $\{\widehat{\mathsf{leak}_j}\}_{j \in [\eta]} \leftarrow \mathsf{Sim}\left(q_j, f_j, \{[\mathbf{x_i}]_{i,1}\}_{i \in [n]}, \{[\mathbf{y_i}]_{i,2}\}_{i \in [n]}, \{[\mathbf{z_i}]_{i,3}\}_{i \in [n]}, f_j(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\right)$ to obtain the simulated outputs.
5. Output the following:
   (a) Public components of the encodings, $\{[\mathbf{x_{b,i}}]_{i,1}.\mathsf{pub}, [\mathbf{y_{b,i}}]_{i,2}.\mathsf{pub}, [\mathbf{z_{b,i}}]_{i,3}.\mathsf{pub}\}_{i \in [n]}$.
   (b) Decoding parameters, $\{q_j\}_{j \in [\eta]}$.
   (c) Output of decodings, $\{\widehat{\mathsf{leak}_j}\}_{j \in [\eta]}$.

Claim 2. *Suppose the learning with errors assumption is true, then for any adversary $\mathcal{A}$ of size 2^λ, it holds that $|\Pr[\mathcal{A}(\mathbf{Hybrid}_2) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_3) = 1]| \leq 2^{-\lambda}$.*

Proof. We show the indistinguishability of $\mathbf{Hybrid}_2$ and $\mathbf{Hybrid}_3$ by considering the following sub-hybrids.

$\mathbf{Hybrid}_{2.1}$: The only change between hybrids $\mathbf{Hybrid}_2$ and $\mathbf{Hybrid}_{2.1}$ are in the generation of $\boldsymbol{b}$. In this hybrid, generate $\boldsymbol{b} \xleftarrow{\$} \mathbb{Z}_{\mathbf{p}}^{m \times 1}$.

The indistinguishability of hybrids $\mathbf{Hybrid}_2$ and $\mathbf{Hybrid}_{2.1}$ follow from the learning with errors assumption.

$\mathbf{Hybrid}_{2.2}$: The only change between $\mathbf{Hybrid}_{2.1}$ and $\mathbf{Hybrid}_{2.2}$ is in the generation of the public parts of the encodings. Specifically, for every $i \in [n], \ell \in \{2, 3\}$, generate the public part of the encoding of $x_{i,\ell}$ as $[\mathbf{x_i}]_{i,\ell}.\mathsf{pub} = \mathbf{U}_{i,\ell} + x_{i,\ell}\mathbf{G}$, where $\mathbf{U}_{i,\ell} \xleftarrow{\$} \mathbb{Z}_{\mathbf{p}}^{(\mathsf{d}+1) \times m}$. For $\ell = 1$, compute $Q_{i.\mathsf{pub}} = \mathbf{U}_{i,\ell} + x_{i,\ell}\mathbf{G}$, where $\mathbf{U}_{i,\ell} \xleftarrow{\$} \mathbb{Z}_{\mathbf{p}}^{(\mathsf{d}+1) \times m}$, and additionally generate $\mathsf{Seed}_{\mathsf{pub},i}$.

The statistical indistinguishability of $\mathbf{Hybrid}_{2.1}$ and $\mathbf{Hybrid}_{2.2}$ follows from the extended left-over hash lemma.

Hybrid$_{2.3}$: The only change between **Hybrid**$_{2.2}$ and **Hybrid**$_{2.3}$ is in the generation of the public parts of the encodings. Specifically, for every $i \in [n], \ell \in \{2,3\}$, generate the public part of the encoding of $x_{i,\ell}$ as $[\mathbf{x}_{\mathbf{i},\ell}]_{i,\ell}.\mathsf{pub} = \mathbf{U}_{i,\ell} + 0 \cdot \mathbf{G}$. For $\ell = 1$, compute $Q_{i.\mathsf{pub}} = \mathbf{U}_{i,\ell} + 0 \cdot \mathbf{G}$, where $\mathbf{U}_{i,\ell} \xleftarrow{\$} \mathbb{Z}_{\mathbf{p}}^{(\mathsf{d}+1)\times m}$, and additionally generate $\mathsf{Seed}_{\mathsf{pub},i}$.

The output distributions of **Hybrid**$_{2.2}$ and **Hybrid**$_{2.3}$ are identical.

Hybrid$_{2.4}$: The only change between **Hybrid**$_{2.2}$ and **Hybrid**$_{2.3}$ is in the generation of the public parts of the encodings. Specifically, for every $i \in [n], \ell \in \{2,3\}$, generate the public part of the encoding of $x_{i,\ell}$ as $[\mathbf{x_i}]_{i,\ell}.\mathsf{pub} = \mathbf{A}\mathbf{R}_{i,\ell} + 0 \cdot \mathbf{G}$. For $\ell = 1$, compute $Q_{i,\mathsf{pub}} = \mathbf{A}\mathbf{R}_{i,\ell} + 0 \cdot \mathbf{G}$, and additionally generate $\mathsf{Seed}_{\mathsf{pub},i}$. Here $\mathbf{A} = [\mathbf{C}^{\mathbf{T}} || \boldsymbol{b}^{\mathbf{T}}]$, where (i) $\mathbf{C} \xleftarrow{\$} \mathbb{Z}_{\mathbf{p}}^{\mathsf{d}\times m}$ and, (ii) $\boldsymbol{b} \xleftarrow{\$} \mathbb{Z}_{\mathbf{p}}^{m\times 1}$.

The statistical indistinguishability of the output distributions of **Hybrid**$_{2.3}$ and **Hybrid**$_{2.4}$ follows from the extended leftover hash lemma.

Finally, learning with errors assumption implies that the output distributions of **Hybrid**$_{2.4}$ and **Hybrid**$_3$ are computationally indistinguishable. This concludes the proof.

□

□

11 Step 2: Construction of Three-Restricted FE from Bilinear Maps

We construct a three-restricted FE scheme 3FE for the class of functions $\mathcal{F}_{\mathsf{3FE}} = \{\mathcal{F}_{\mathsf{3FE},\lambda,\mathbf{p},n}\}_{\lambda \in [\mathbb{N}]}$ (recalled below). We later show that 3FE satisfies semi-functional security property. The tool to construct this primitive is a 5-slotted encodings scheme, introduced by [AS17], of degree 2. We use additive notation to indicate the group operation. Each slot will correspond to a group of order $\mathbf{p}$. We recall this definition in Section 4.2. The abstraction of this scheme is similar to bilinear maps of composite order.

- Recall function class of interest. $\mathcal{F}_{\mathsf{3FE}}$ consists of all functions $\mathcal{F}_{\mathsf{3FE},\lambda,\mathbf{p},n} = \{f : \{\mathbb{Z}_{\mathbf{p}}^n\}^3 \to \mathbb{Z}_{\mathbf{p}}\}$ where $\mathbb{Z}_{\mathbf{p}}$ is a finite field of order $\mathbf{p}(\lambda)$. Here n is seen as a function of λ. Each $f \in \mathcal{F}_{\mathsf{3FE},\lambda,\mathbf{p},n}$ takes as input three vectors $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ over $\mathbb{Z}_{\mathbf{p}}$ and computes a polynomial of the form $\Sigma c_{i,j,k} x_i y_j z_k$ over $\mathbb{Z}_{\mathbf{p}}$, where the coefficients are specified by the function f.

We describe the construction below. We assume that n is known to the algorithms implicitly.

$\underline{\mathsf{Setup}(1^\lambda, 1^n)}$: On input security parameter λ,

- Sample $\alpha_i, \beta_i, \gamma_i \leftarrow \mathbb{Z}_{\mathbf{p}}$ for all $i \in [n]$. Denote $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_n)$, $\boldsymbol{\beta} = (\beta_1, ..., \beta_n)$ and $\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_n)$.
- Compute $\mathbf{k}_{j,k} = [0, 0, \beta_j \cdot \gamma_k, 0, 0]_2$, for every $i, j \in [n]$.
- This algorithm also does setup for a slotted encoding scheme. For simplicity of notation, we assume that the encoding key and public parameters of this scheme are implicitly known to the encoding algorithm and public parameters are known to the evaluation algorithms. We also assume that the slotted encoding encodes elements in $\mathbb{Z}_{\mathbf{p}}$.

Set the master secret key to be $\mathsf{MSK} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \{\mathbf{k}_{j,k}\}_{j,k \in [n]})$.

KeyGen(MSK, f): On input the master secret key MSK and function f,

- Compute $\mathbf{k}_f = [0, 0, 0, V, 0]_2$. We compute V as follows: set $V = f(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$.

Output the resulting functional key $sk[f] = (\{\mathbf{k}_{j,k}\}_{j \in [n], k \in [n]}, \mathbf{k}_f)$.
Note: the description of the function f is implicit in the description of $\mathbf{k}_f$.

Enc$(\mathsf{MSK}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$: The input message $M = (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ consists of a public attribute $\boldsymbol{x}$ and private vectors $\boldsymbol{y}, \boldsymbol{z}$. Denote by x_i to be the i^{th} component in the vector of $\boldsymbol{x}$ (and likewise for $\boldsymbol{y}$ and $\boldsymbol{z}$). Perform the following operations:

- Sample $r \in \mathbb{Z}_{\mathbf{p}}$.
- Compute $\mathsf{CT}_{2,j} = [y_j, r \cdot \beta_j, 0, 0, 0]_1$, for every $j \in [n]$.
- Compute $\mathsf{CT}_{3,k} = [z_k, -r \cdot \gamma_k, 0, 0, 0]_2$, for every $k \in [n]$.
- Compute $\mathsf{CT}_{1,i} = [0, 0, (x_i - \alpha_i) \cdot r^2, 0, 0]_1$, for every $i \in [n]$.
- Compute $\mathsf{CT}_0 = [0, 0, 0, r^2, 0]_1$.

Output the ciphertext $\mathsf{CT} = (\boldsymbol{x}, \{\mathsf{CT}_{2,i}\}_{i \in [n]}, \{\mathsf{CT}_{3,i}\}_{i \in [n]}, \{\mathsf{CT}_i^x\}_{i \in [n]}, \mathsf{CT}_0)$.

Dec$(sk[f], 1^B, \mathsf{CT})$: On input the functional key $sk[f]$ and a ciphertext CT, perform the following: Parse the ciphertext as $\mathsf{CT} = (\boldsymbol{x}, \{\mathsf{CT}_{2,j}\}_{j \in [n]}, \{\mathsf{CT}_{3,k}\}_{k \in [n]}, \{\mathsf{CT}_{1,i}\}_{i \in [n]}, \mathsf{CT}_0)$ and the functional key as $sk[f] = (\{\mathbf{k}_{i,j}\}_{i \in [n], j \in [n]}, \mathbf{k}_f)$.

- For all $i, j, k \in [n]$, first compute $e(\mathsf{CT}_{2,j}, \mathsf{CT}_{3,k})$ to obtain $\widehat{\mathsf{CT}_{j,k}}$ and then compute $\widehat{\mathsf{CT}_{i,j,k}} = \widehat{\mathsf{CT}_{j,k}}^{x_i}$. Note that $\widehat{\mathsf{CT}_{j,k}}^{x_i}$ is equal to $[x_i y_j z_k - r^2 x_i \beta_j \gamma_k]_T$.
- For all $i, j, k \in [n]$, compute $e(\mathsf{CT}_{1,i}, \mathbf{k}_{j,k})$ to obtain $\widehat{\mathsf{CT}^x_{i,j,k}}$. Note that $\widehat{\mathsf{CT}^x_{i,j,k}}$ is equal to $[r^2(x_i - \alpha_i)\beta_j \gamma_k]_T$.
- For all $i, j, k \in [n]$, compute $\mathsf{ans}_{i,j,k} = \widehat{\mathsf{CT}_{i,j,k}} \cdot \widehat{\mathsf{CT}^x_{i,j,k}}$. Note that $\mathsf{ans}_{i,j,k} = [x_i y_j z_k - r^2 \alpha_i \beta_j \gamma_k]_T$
- Let f be represented as a polynomial $g = \Sigma_{i,j,k} c_{i,j,k} x_i y_j z_k$ where each $c_{i,j,k} \in \mathbb{Z}_{\mathbf{p}}$. Compute $\Pi_{i,j,k} \mathsf{ans}_{i,j,k}^{c_{i,j,k}} = \mathsf{ans}^*$. Note that $\mathsf{ans}^* = [f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) - r^2 f(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})]_T$
- Compute $e(\mathbf{k}_f, \mathsf{CT}_0,)$ to get Θ^*. Note that $\Theta^* = [r^2 f(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})]$
- Compute $out = \mathsf{ans}^* \cdot \Theta^*$. Check if $out = [g]_T$ for some $g \in [-B, B]$. If so, output the value g, otherwise output $\perp$.

We omit the correctness argument since it follows from the description of the scheme.

Efficiency: We now bound the size of the circuit computing the ciphertext. Each cipher-text consists of $3n + 1$ slotted encodings and a vector $\boldsymbol{x}$. Each encoding is computable by a circuit of size polynomial in $\log_2 \mathbf{p} \cdot poly(\lambda)$. This proves the result.

11.1 Security

Theorem 10. *Assuming the existence of a degree two slotted encoding scheme with five slots in the bilinear generic group model, the construction* $\mathsf{3FE}$ *is a secure three-restricted functional encryption scheme in the generic bilinear map model.*

We first describe the semi-functional algorithms.

$\underline{\mathsf{sfKG}(\mathsf{MSK}, f, \theta)}$: On input master secret key MSK, function f and a value θ ,

- Compute $\mathbf{k}_f = [0, 0, 0, V, \theta]_2$. Here V is computed as before. That is, $V = f(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$.

Output the resulting semi-functional key $sk[f, \theta] = (\{\mathbf{k}_{j,k}\}_{j\in[n],k\in[n]}, \mathbf{k}_f)$.

$\underline{\mathsf{sfEnc}(\mathsf{MSK}, \boldsymbol{x}, 1^{|\boldsymbol{y}|}, 1^{|\boldsymbol{z}|})}$: On input $\boldsymbol{x} \in \mathbb{Z}_{\mathbf{p}}^n$ and length $1^{|\boldsymbol{y}|}$, $1^{|\boldsymbol{z}|}$ (which are equal to n), where $\boldsymbol{x}$ is the public attribute and $\boldsymbol{y}, \boldsymbol{z}$ is the private message. Denote by x_i to be the i^{th} component in the vector of $\boldsymbol{x}$. Perform the following operations:

- Sample $r \in \mathbb{Z}_{\mathbf{p}}$.
- Compute $\mathsf{CT}_{2,j} = [0, r \cdot \beta_j, 0, 0, 0]_1$, for every $j \in [n]$.
- Compute $\mathsf{CT}_{3,k} = [0, -r \cdot \gamma_k, 0, 0, 0]_2$, for every $k \in [n]$.
- Compute $\mathsf{CT}_{1,i} = [0, 0, (x_i - \alpha_i) \cdot r^2, 0, 0]_1$, for every $i \in [n]$.
- Compute $\mathsf{CT}_0 = [0, 0, 0, r^2, 1]_1$.

Output the semi-functional ciphertext $\mathsf{ct}_{\mathsf{sf}} = (\boldsymbol{x}, \{\mathsf{CT}_{2,i}\}_{i\in[n]}, \{\mathsf{CT}_{3,i}\}_{i\in[n]}, \{\mathsf{CT}_i^x\}_{i\in[n]}, \mathsf{CT}_0)$.

First, we recall the generic (slotted) bilinear group model below. We use this model to argue security.

Generic Bilinear Group Model We describe the generic bilinear group model [BBG05] tailored to the slotted asymmetric setting. This model is parameterized by slotted encodings SE, which encodes five dimensional vectors over a prime field $\mathbb{Z}_{\mathbf{p}}$ at level 1 and 2, and it encodes element from $\mathbb{Z}_{\mathbf{p}}$ at the target level T. The encodings are done over level 1, 2 and the target T. The multiplication operation computes encoding at level T. The adversary in this model has access to an oracle $\mathcal{O}$. Initially, the adversary is handed out *handles* (sampled uniformly at random) instead of being handed out actual encodings. A handle is an element in a ring $\mathbb{Z}$ of order $\mathbf{p}$. The oracle $\mathcal{O}$ maintains a list L consisting of tuples $(e, \mathbf{Y}[e], u)$, where e is the handle issued, $\mathbf{Y}[e]$ is the formal expression associated with e and e is associated with encoding at level $u \in \{1, 2, T\}$.

The adversary is allowed to submit the following types of queries to the oracle:

The adversary is allowed to submit the following types of queries to the oracle:

- *Addition/ Subtraction*: The adversary submits (e_1, u_1) and (e_2, u_2) along with the operation '+'(or '-') to the oracle where $u_1, u_2 \in \{1, 2, T\}$. If $u_1 \neq u_2$ or If there is no tuple associated with either e_1 or e_2, the oracle sends $\perp$ back to the adversary. Otherwise, it replies according to the following cases:

– $u_1 \in \{1,2\}$: In this case it locates $(e_1, p_{1,e_1}, p_{2,e_1}, p_{3,e_1}, p_{4,e_1}, p_{5,e_1}, u_1)$ and $(e_2, p_{1,e_2}, p_{2,e_2}, p_{3,e_2}, p_{4,e_2}, p_{5,e_2}, u_2)$. It creates a new handle e' (sampled uniformly at random from $\mathcal{R}$) and appends $(e', p_{1,e_1} + p_{1,e_2}, p_{2,e_1} + p_{2,e_2}, p_{3,e_1} + p_{3,e_2}, p_{4,e_1} + p_{4,e_2}, p_{5,e_1} + p_{5,e_2}, u_1)$ to the list (in case of subtractions the polynomials are subtracted). It outputs e' to the adversary.

– $u_1 = u_2 = T$: In this case the adversary locates the tuples (e_1, p_{e_1}, u_1) and (e_2, p_{e_2}, u_2). It creates a new handle e' (sampled uniformly at random from $\mathcal{R}$) and appends $(e', p_{e_1} + p_{e_2}, u_1)$ (or $(e', p_{e_1} - p_{e_2}, u_1)$) to the list. The oracle sends e' to the adversary.

- *Multiplication*: The adversary submits (e_1, u_1) and (e_2, u_2) to the oracle. If there is no tuple associated with either e_1 or e_2, the oracle sends $\perp$ back to the adversary. If $u_1 = u_2$, $u_1 = T$ or $u_2 = T$, the oracle outputs $\perp$. Otherwise, it locates the tuples $(e_1, p_{1,e_1}, p_{2,e_1}, p_{3,e_1}, p_{4,e_1}, p_{5,e_1}, u_1)$ and $(e_2, p_{1,e_2}, p_{2,e_2}, p_{3,e_2}, p_{4,e_2}, p_{5,e_2}, u_2)$. It creates a new handle e' (sampled uniformly at random from $\mathcal{R}$) and appends $(e', \Sigma_{j\in[5]} p_{j,e_1} * p_{j,e_2}, T)$ to the list.

- *Zero Test*: The adversary submits element (e_1, u_1) to the oracle. If there is no tuple associated to e_1 it outputs $\perp$. Otherwise, if $u_1 = 1$ or $u_1 = 2$, it locates the tuples $(e_1, p_{1,e_1}, p_{2,e_1}, p_{3,e_1}, p_{4,e_1}, p_{5,e_1}, u_1)$. It outputs 1 if $p_{j,e_1} = 0$ for all $j \in [5]$ otherwise it outputs 0. If $u_1 = T$, it locates the tuples (e_1, p_{1,e_1}, u_1). It outputs 1 if $p_{1,e_1} = 0$, otherwise it outputs 0.

Now we describe a lemma that will be crucial for the rest of the proof.

Lemma 2 (Schwartz-Zippel-DeMillo-Lipton)**.** *Consider a polynomial $h \in \mathbb{Z}_{\mathbf{p}}[y_1, \ldots, y_n]$ for a prime $\mathbf{p}$. Suppose the degree of h is at most deg then,*

$$\Pr_{y_1,\ldots,y_n \xleftarrow{\$} \mathbb{Z}_{\mathbf{p}}} [h(y_1, \ldots, y_n) = 0] \leq \frac{\mathsf{deg}}{\mathbf{p}}$$

Now we consider three scenarios.

- **Case 1:** The adversary is given normal function keys and normal ciphertexts.
- **Case 2:** The adversary is given semi functional keys and normal ciphertexts.
- **Case 3:** The adversary is given semi-functional keys and one semi-functional ciphertext along with remaining normal ciphertexts.

To argue indistinguishability of semi-functional keys property we need to argue that **Case 1** is indistinguishable to **Case 2**. To argue indistinguishability of semi-functional ciphertext property, we need to argue **Case 2** is indistinguishable to **Case 3**. We will argue this in the following manner:

- We assume that the adversary is given some set of encodings (depending on which case he is in). Then the adversary submits a polynomial P for zero-test.
- Adversary wins if P evaluates to 0 in one case and non-zero in another.
- By a case analysis on P, we will show the if the adversary wins with non-negligible probability, then P must contradict the shwartz-zippel lemma.

In another words, we will show that if P evaluates to 0 with non-negligible probability in one case, then it should also evaluate to 0 with almost the same probability in other cases. Let us analyse these cases separately.

<u>Case 1:</u> In this case the adversary is given ciphertexts and keys which contain encodings at level 1 and 2. The adversary can query for any function key for functions f_l for $l \in [\eta]$. He also gets challenge ciphertext CT_1 along with other ciphertexts for CT_q for $q \in [2, \Gamma]$. Each key for f_l consists of the following encodings (variables denoted by α, β, γ, r are chosen at random from $\mathbb{Z}_\mathbf{p}$):

- $\mathbf{k}_{j,k} = [0, 0, \beta_j \cdot \gamma_k, 0, 0]_2$, for every $j, k \in [n]$
- $\mathbf{k}_f = [0, 0, 0, V_l, 0]_2$ for $l \in [\eta]$.

Each ciphertext CT_q consists of the following encodings:

- $\mathsf{CT}^q_{2,j} = [y_{q,j}, r_q\beta_j, 0, 0, 0]_1$, for every $j \in [n], q \in [\Gamma]$.
- $\mathsf{CT}^q_{3,k} = [z_{q,k}, -r_q \cdot \gamma_k, 0, 0, 0]_2$, for every $k \in [n], q \in [\Gamma]$.
- $\mathsf{CT}^q_{1,i} = [0, 0, (x_{q,i} - \alpha_i) \cdot r_q^2, 0, 0]_1$, for every $i \in [n], q \in [\Gamma]$.
- $\mathsf{CT}^q_0 = [0, 0, 0, r_q^2, 0]_1$ for every $q \in [\Gamma]$.

We now analyze the polynomials queried by the adversary to the oracle $\mathcal{O}$. There are two types of polynomials the adversary can submit: linear and quadratic.

If the adversary submits a linear polynomial, the oracle will output 0 only with probability at most $2/\mathbf{p}$. This follows from lemma 2 and analyzing the structure of the encodings.

We now turn to the case when the adversary submits quadratic polynomials to the oracle. To analyze this case, we look at all possible monomials/terms formed by evaluating multiplication of encodings of level 1 and level 2:

1. $r_q^2 f_l(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ for $q \in [\Gamma]$, $l \in [\eta]$.
2. $y_{q_1,j} z_{q_2,k} - r_{q_1} r_{q_2} \beta_j \gamma_k$. for $j, k \in [n]$ and $q_1, q_2 \in [\Gamma]$
3. $(x_{q,i} - \alpha_i) r_q^2 \beta_j \gamma_k$ for $i, j, k \in [n]$ and $q \in [\Gamma]$
4. Constant term 1. This is generated from the encoding of 1 at the level T.

Consider a zero test polynomial query P of this kind to the oracle $\mathcal{O}$.

Structure of P: Let us now consider a polynomial P which is a linear combination of monomials with coefficients in $\mathbb{Z}_\mathbf{p}$. Any monomial of type $i \in [4]$ can have a coefficient of the form $c_{i,\cdots}$ where the $(i, \cdots)$ is replaced with quantifiers of the variables in the monomials. For example, the coefficient of first monomial is represented as $c_{1,l,q}$ for $q \in [\Gamma], l \in [\eta]$. This polynomial P can be represented as: $k_0 + \Sigma_q k_{1,q} r_q^2 + \Sigma_{q_1,q_2} k_{2,q_1,q_2} r_{q_1} r_{q_2}$ where each term $k's$ are a function of variables independent of r_q. Now by Schwartz-Zippel lemma with probability at least $1 - 2/\mathbf{p}$ the coefficients $k_{2,q_1,q_2} = 0$ for $q_1 \neq q_2$. Then, we write $k_{2,q_1,q_2} = \Sigma_{j,k} c_{2,q_1,q_2,j,k} \beta_j \gamma_k$. With probability at least $1 - 2/\mathbf{p}$ each coefficient $c_{2,q_1,q_2,j,k} = 0$.

Now consider coefficients of r_q^2. By Schwartz-Zippel lemma, with probability at least $1 - 2/\mathbf{p}$, the coefficients should be 0.

Coefficient of r_q^2: The coefficient of r_q^2 for $q \in [\Gamma]$ is $\Sigma_{l,i,j,k} c_{1,q,l} f_l(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) - c_{2,q,q,j,k}\beta_j\gamma_k + c_{3,q,i,j,k}(x_{q,i} - \alpha_i)\beta_j\gamma_k$. Now applying Schwartz-Zippel lemma again and setting coefficients of $\alpha_i\beta_j\gamma_k$ and $\beta_j\gamma_k$ to be 0, we observe the following conditions. With probability at least $1 - 3/\mathbf{p}$, we have:

- $-c_{2,q,q,j,k} + \Sigma_i c_{3,q,i,j,k} x_{q,i} = 0$
- $-c_{3,q,i,j,k} + \Sigma_l c_{1,q,l} f_{l,i,j,k} = 0$

Coefficient of 1: The coefficient of 1 is, $c = c_4 + \Sigma_{q,j,k} c_{2,q,q,j,k} y_{q,j} z_{q,k}$. Note from the previous claim, we observe that this coefficient is equal to:

$$c = c_4 + \Sigma_{q,j,k}\Sigma_i c_{3,q,i,j,k} x_{q,i} y_{q,j} z_{q,k}$$

From the second sub claim in the previous claim we observe:

$$c = c_4 + \Sigma_{q,j,k}\Sigma_i\Sigma_l c_{1,q,l} f_{1,i,j,k} x_{q,i} y_{q,j} z_{q,k}$$

Thus

$$c = c_4 + \Sigma_{q,l} c_{1,q,l} f_l(\boldsymbol{x}_q, \boldsymbol{y}_q, \boldsymbol{z}_k)$$

By Shwartz-Zippel lemma, $c = 0$. From this, we can conclude that the oracle returns 0 to the polynomials submitted by the adversary only in the case when these polynomials are linear functions in the decryptions $f_l(M_q)$ for $l \in [\eta], q \in [\Gamma]$. Here M_q denotes the q^{th} message encrypted.

<u>Case 2:</u> In this case the keys are semi-functional while ciphertexts are honestly computed. The encodings given to the adversary in this case are:

- $\mathbf{k}_{j,k} = [0, 0, \beta_j \cdot \gamma_k, 0, 0]_2$, for every $j, k \in [n]$
- $\mathbf{k}_f = [0, 0, 0, V_l, \theta_l]_2$ for $l \in [\eta]$. Here θ_l for $l \in [\eta]$ are hardwirings.

Each ciphertext CT_q consists of the following encodings:

- $\mathsf{CT}^q_{2,j} = [y_{q,j}, r_q \cdot \beta_j, 0, 0, 0]_1$, for every $j \in [n], q \in [\Gamma]$.
- $\mathsf{CT}^q_{3,k} = [z_{q,k}, -r_q \cdot \gamma_k, 0, 0, 0]_2$, for every $k \in [n], q \in [\Gamma]$.
- $\mathsf{CT}^q_{1,i} = [0, 0, (x_{q,i} - \alpha_i) \cdot r_q^2, 0, 0]_1$, for every $i \in [n], q \in [\Gamma]$.
- $\mathsf{CT}^q_0 = [0, 0, 0, r_q^2, 0]_1$ for every $q \in [\Gamma]$.

The terms computed by pairing these encodings are as follows.

1. $r_q^2 f_l(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ for $q \in [\Gamma]$, $l \in [\eta]$.
2. $y_{q_1,j} z_{q_2,k} - r_{q_1} r_{q_2} \beta_j \gamma_k$. for $j, k \in [n]$ and $q_1, q_2 \in [\Gamma]$
3. $(x_{q,i} - \alpha_i) r_q^2 \beta_j \gamma_k$ for $i, j, k \in [n]$ and $q \in [\Gamma]$
4. Constant term 1. This is generated from the encoding of 1 at the level T.

The terms generated in this case are identical to terms in **Case 1**. Thus, these cases are indistinguishable.

Case 3: In this case, the challenge ciphertext as well as the functional keys are semi-functionally generated. Let $M = (\boldsymbol{x}_1, \boldsymbol{y}_1, \boldsymbol{z}_1)$ be the challenge. Set $\theta_i = f_i(M)$ for $i \in [\eta]$. Here are the encodings:

- $\mathbf{k}_{j,k} = [0, 0, \beta_j \cdot \gamma_k, 0, 0]_2$, for every $j, k \in [n]$
- $\mathbf{k}_f = [0, 0, 0, V_l, \theta_l]_2$ for $l \in [\eta]$. Here θ_l for $l \in [\eta]$ are hardwirings.

Each ciphertext CT_q consists of the following encodings for $q \in [2, \Gamma]$:

- $\mathsf{CT}^q_{2,j} = [y_{q,j}, r_q\beta_j, 0, 0, 0]_1$, for every $j \in [n], q \in [\Gamma]$.
- $\mathsf{CT}^q_{3,k} = [z_{q,k}, -r_q \cdot \gamma_k, 0, 0, 0]_2$, for every $k \in [n], q \in [\Gamma]$.
- $\mathsf{CT}^q_{1,i} = [0, 0, (x_{q,i} - \alpha_i) \cdot r_q^2, 0, 0]_1$, for every $i \in [n], q \in [\Gamma]$.
- $\mathsf{CT}^q_0 = [0, 0, 0, r_q^2, 0]_1$ for every $q \in [\Gamma]$.

Ciphertext CT_1 consists of the following encodings:

- $\mathsf{CT}^1_{2,j} = [0, r_1\beta_j, 0, 0, 0]_1$, for every $j \in [n]$.
- $\mathsf{CT}^1_{3,k} = [0, -r_1 \cdot \gamma_k, 0, 0, 0]_2$, for every $k \in [n]$.
- $\mathsf{CT}^1_{1,i} = [0, 0, (x_{1,i} - \alpha_i) \cdot r_1^2, 0, 0]_1$, for every $i \in [n]$.
- $\mathsf{CT}^1_0 = [0, 0, 0, r_1^2, 0]_1$.

The terms computed by pairing these encodings are:

1. $r_q^2 f_l(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ for $q \in [2, \Gamma]$, $l \in [\eta]$.
2. $r_1^2 f_l(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) + \theta_l$ for $l \in [\eta]$.
3. $y_{q_1,j} z_{q_2,k} - r_{q_1} r_{q_2} \beta_j \gamma_k$. for $j, k \in [n]$ and $q_1, q_2 \in [2, \Gamma]$
4. $0 - r_1 r_{q_2} \beta_j \gamma_k$. for $j, k \in [n]$ and $q_2 \in [1, \Gamma]$
5. $(x_{q,i} - \alpha_i) r_q^2 \beta_j \gamma_k$ for $i, j, k \in [n]$ and $q \in [\Gamma]$
6. Constant term 1. This is generated from the encoding of 1 at the level T.

Structure of P: The analysis of this case is similar to **Case 1**. The only difference are terms where $q = 1$. Since $\theta_l = f_l(\boldsymbol{x}_1, \boldsymbol{y}_1, \boldsymbol{z}_1)$ for $l \in [\eta]$, the conditions on the polynomials turn out to be the same. We elaborate now.

Let us now consider a polynomial P over $\mathbb{Z}_\mathbf{p}$. Any monomial of type $i \in [4]$ can have a coefficient of the form $c_{i,\cdots}$ where the $(i, \cdots)$ is replaced with quantifiers of the variables in the monomials. For example, the coefficient of first monomial is represented as $c_{1,q,l}$ for $q \in [\Gamma], l \in [\Gamma]$. This polynomial P can be represented as: $k_0 + \Sigma_q k_{1,q} r_q^2 + \Sigma_{q_1,q_2} k_{2,q_1,q_2} r_{q_1} r_{q_2}$ where each term $k's$ are a function of variables independent of r_q. Now by Schwartz-Zippel lemma with probability at least $1 - 2/\mathbf{p}$ the coefficients $k_{2,q_1,q_2} = 0$ for $q_1 \neq q_2$ and $q_1, q_2 \neq 1$. Then, we write $k_{2,q_1,q_2} = \Sigma_{j,k} c_{3,q_1,q_2,j,k} \beta_j \gamma_k$. With probability at least $1 - 2/\mathbf{p}$ each coefficient $c_{2,q_1,q_2,j,k} = 0$. Similarly, we observe that $c_{4,q_2,j,k} = 0$ for $q_2 \neq 1$. Now consider coefficients of r_q^2. By Schwartz-Zippel lemma, with probability at least $1 - 2/\mathbf{p}$, the coefficients should be 0.

Coefficient of r_q^2 for $q \neq 1$: The coefficient of r_q^2 for $q \in [2, \Gamma]$ is $\Sigma_l c_{1,q,l} f_l(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) - \Sigma_{l,j,k} c_{3,q,q,j,k} \beta_j \gamma_k + \Sigma_{l,i,j,k} c_{5,q,i,j,k}(x_{q,i} - \alpha_i)\beta_j\gamma_k$. Now applying Schwartz-Zippel lemma again and setting coefficients of $\alpha_i\beta_j\gamma_k$ and $\beta_j\gamma_k$ to be 0, we observe the following conditions. With probability at least $1 - 3/\mathbf{p}$, we have:

- $-c_{3,q,q,j,k} + \Sigma_i c_{5,q,i,j,k} x_{q,i} = 0$
- $-c_{5,q,i,j,k} + \Sigma_l c_{1,q,l} f_{l,i,j,k} = 0$

Coefficient of r_q^2 for $q = 1$: The coefficient of r_1^2 is $\Sigma_l c_{2,l} f_l(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) - \Sigma_{j,k} c_{4,j,k}\beta_j\gamma_k + \Sigma_{l,i,j,k} c_{5,1,i,j,k}(x_{1,i} - \alpha_i)\beta_j\gamma_k$. Now applying Schwartz-Zippel again and setting coefficients of $\alpha_i\beta_j\gamma_k$ and $\beta_j\gamma_k$ to be 0, we observe the following conditions. With probability at least $1 - 3/\mathbf{p}$, we have:

- $-c_{4,j,k} + \Sigma_i c_{5,1,i,j,k} x_{1,i} = 0$
- $-c_{5,1,i,j,k} + \Sigma_l c_{2,l} f_{l,i,j,k} = 0$

Coefficient of 1: The coefficient of 1 is, $c = c_6 + \Sigma_{q \neq 1,j,k} c_{3,q,q,j,k} y_{q,j} z_{q,k} + \Sigma_l c_{2,l}\theta_l$. Note from the previous claim, we observe that this coefficient is equal to:

$$c = c_6 + \Sigma_{q \neq 1,j,k} \Sigma_i c_{5,q,i,j,k} x_{q,i} y_{q,j} z_{q,k} + \Sigma_l c_{2,l}\theta_l$$

From the second claim in the previous case we observe:

$$c = c_6 + \Sigma_{q \neq 1,j,k} \Sigma_i \Sigma_l c_{1,q,l} f_{1,i,j,k} x_{q,i} y_{q,j} z_{q,k} + \Sigma_l c_{2,1}\theta_1$$

Thus

$$c = c_6 + \Sigma_{q \neq 1,l} c_{1,q,l} f_l(\boldsymbol{x}_q, \boldsymbol{y}_q, \boldsymbol{z}_k) + \Sigma_l c_{2,l} f(\boldsymbol{x}_1, \boldsymbol{y}_1, \boldsymbol{z}_1)$$

By Schwartz-Zippel lemma $c = 0$.

Thus, this shows that both **Case 1** and **Case 3** are indistinuishable as they lead to the same condition. The oracle returns 0 to the polynomials submitted by the adversary only in the case when these polynomials are linear functions in the decryptions $f_l(M_q)$ for $l \in [\eta], q \in [\Gamma]$. Here M_q denotes the q^{th} message encrypted.

12 Step 3: Construction of Semi-Functional FE for Cubic Polynomials

In this section we construct and then prove correctness, efficiency and security for semi-functional functional encryption for cubic functions (referred as FE_3). For this construction we assume the existence of a three-restricted 3FE for a specific function class (defined below) and a tempered cubic encoding scheme, TCE.

Function class of interest for FE_3: We construct a semi-functional functional encryption scheme for cubic homogenous polynomials over variables over integers $\mathbb{Z}$. Formally, consider a set of functions $\mathcal{F}_{\mathsf{FE}_3,\lambda,n} = \{f : [-\rho, \rho]^n \rightarrow \mathbb{Z}\}$. Here n is seen as a function of λ and ρ is a constant. Each $f \in \mathcal{F}_{\mathsf{FE}_3,\lambda,n}$ takes as input $\boldsymbol{x} = (x_1, .., x_n) \in [-\rho, \rho]^n$ and computes a polynomial of the form $\Sigma c_{i,j,k} x_i x_j x_k$ over $\mathbb{Z}$ (where some variables can repeat) and each coefficient $c_{i,j,k} \in [-\rho, \rho]$ and $\Sigma_{j,k}|c_{i,j,k}| < w(\lambda)$ for some fixed polynomial $w(\lambda)$ independent of n. In order to implement semi-functional functional encryption for this class of functions we use a $\mathsf{3FE}$ scheme over some large prime $\mathbf{p}$ and a TCE scheme with a plain-text space is $\mathbb{Z} \cap [-\Delta, \Delta]$ for some large enough Δ. Note that if $\mathbf{p}$ is large the result of computation is the same as the computation done over $\mathbb{Z}$.

Setting parameters of TCE: We require the following notational properties of TCE which can be instanitated as in Section 10.1.

1. We require the plain-text space $\mathbb{Z}$ to be $\mathbb{Z} \cap [-\Delta, \Delta]$ for some polynomial Δ. Δ should be larger than $w(\lambda)\rho^3$. This is so as to allow the computations of $\mathcal{F}_{\mathsf{FE}_3}$ to be done over $\mathbb{Z}$ (instead of $\mathbb{Z}$) as $\mathcal{F}_{\mathsf{FE}_3}$ contain polynomials that act on inputs in $[-\rho, \rho]$. This idea will be more clear when we describe the construction.

2. (Representation) The encoding of any element $a \in R$ at any level $l \in \{1, 2, 3\}$ should consist of three parts as described now: $[\mathbf{a}]_l = ([\mathbf{a}]_l.\mathsf{pub}, [\mathbf{a}]_l.\mathsf{priv}(1), [\mathbf{a}]_l.\mathsf{priv}(2))$. Each part is thought of as a vector of dimension $\mathsf{d} = \mathsf{d}(\lambda)$ over $\mathbb{Z}_\mathbf{p}$ for some prime $\mathbf{p} = \mathbf{p}(\lambda)$.

3. **Security:** We require that TCE scheme satisfy $(\mathcal{S}_\eta)-$Tempered Security. We will prove that if TCE satisfies $\mathcal{S}_\eta-$tempered security, the semi-functional FE scheme for cubic polynomials will satisfy $\mathcal{S}_\eta-$Bounded semi-functional security. Thus, to construct a semi-functional FE scheme for cubic polynomials for class of functions $\mathcal{S}_\eta$, we need TCE to satisfy $\mathcal{S}_\eta-$tempered security. Denote by $\mathcal{S}_\eta = (\mathcal{F}_{\mathsf{FE}_3,\lambda,n})^\eta$. Here, η is the maximum number of key queries handled by the scheme.

4. **Cubic Evaluation:** We require that $\mathsf{TCE.Decode}(q, g, \cdot)$ for any cubic homogeneous polynomial amounts to evaluating another cubic homogeneous polynomial $\phi_{q,g}$ on $\mathbb{Z}_\mathbf{p}$ over encodings (with partial degree 1 in public as well as private components). This follows from the cubic evaluation property of Tempered Cubic Encoding.

Function class for $\mathsf{3FE}$: To allow compatability with TCE we will use $\mathsf{3FE}$ for the following class of functions. $\mathcal{F}_{\mathsf{3FE},\lambda,3n\mathsf{d},\mathbf{p}} = \{f : \{\mathbb{Z}_\mathbf{p}^{3n\mathsf{d}}\}^3 \rightarrow \mathbb{Z}_\mathbf{p}\}$ where $\mathbb{Z}_\mathbf{p}$ is a finite field of order $\mathbf{p}(\lambda)$ takes as input $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ where each vector over $\mathbb{Z}_\mathbf{p}$ is of length $3n\mathsf{d}$ and computes a polynomial of the form $\Sigma c_{i,j,k} x_i y_j z_k$ over $\mathbb{Z}_\mathbf{p}$.

12.1 Construction

Now we formally present our construction.

<u>$\mathsf{FE}_3.\mathsf{Setup}(1^\lambda, 1^n)$:</u> On input the security parameter 1^λ the setup algorithm does the following:

1. Compute $\mathsf{TCE.Setup}(1^\lambda, 1^n) \rightarrow \mathsf{params}$.

2. Sample $(q_1, ..., q_\eta) \leftarrow \mathsf{TCE.SetupDec}(\mathsf{params})$.

3. Let $\mathbb{Z}_{\mathbf{p}}$ denote the prime field associated with TCE encodings. Let $[-\rho, \rho]^n$ for some $n = n(\lambda)$ denote the plaintext space of which the scheme needs to be constructed. Let $\mathsf{d} = \mathsf{d}(\lambda)$ be the dimension of each part of encoding of TCE.

4. Now let $\mathsf{3FE}$ denote the scheme for the function class $\mathcal{F}_{\mathsf{3FE}} = \mathcal{F}_{\mathsf{3FE},\lambda,3n\mathsf{d},\mathbf{p}}$. Run $\mathsf{3FE.Setup}(1^\lambda) \to sk$.

5. Then sample $\mathsf{sp} \leftarrow \mathsf{TCE.SetupEnc}(\mathsf{params})$. Encode vector $\boldsymbol{z}$ with $z_i = 0$ for $i \in [n]$ at all the three levels. That is, compute $[\mathbf{z_i}]_{i,j} \leftarrow \mathsf{Encode}(\mathsf{sp}, z_i, i, j)$ for every $i \in [n]$ and $j \in [3]$. Denote $[\mathbf{z_i}]_{i,j} = ([\mathbf{z_i}]_{i,j}.\mathsf{pub}, [\mathbf{z_i}]_{i,j}.\mathsf{priv}(1), [\mathbf{z_i}]_{i,j}.\mathsf{priv}(2))$. Here, both public and private components belong to $\mathbb{Z}_{\mathbf{p}}^{\mathsf{d}}$. These encodings are used only in the semi-functional algorithms.

6. Output $\mathsf{MSK} = (\mathsf{params}, sk, \mathsf{sp}, \{[\mathbf{z_i}]_{i,j}\}_{i\in[n],j\in[3]}, \{q_j\}_{j\in[\eta]})$.

<u>$\mathsf{FE}_3.\mathsf{Enc}(\mathsf{MSK}, \mathbf{x} = (x_1, .., x_n) \in [-\rho, \rho]^n)$</u> : On input the encryption key and the plaintext message in $[-\rho, \rho]^n$ the encryption algorithm does the following:

1. Run $\mathsf{TCE.SetupEnc}(\mathsf{params}) \to \mathsf{sp}_1$.

2. Encode each x_i for $i \in [n]$ at all the three levels. That is compute $[\mathbf{x_i}]_{i,j} \leftarrow \mathsf{Encode}(\mathsf{sp}_1, x_i, i, j)$ for every $i \in [n]$ and $j \in [3]$. Denote $[\mathbf{x_i}]_{i,j} = ([\mathbf{x_i}]_{i,j}.\mathsf{pub}, [\mathbf{x_i}]_{i,j}.\mathsf{priv}(1), [\mathbf{x_i}]_{i,j}.\mathsf{priv}(2))$. Here $[\mathbf{x_i}]_{i,j}.\mathsf{pub}$, $[\mathbf{x_i}]_{i,j}.\mathsf{priv}(1)$ and $[\mathbf{x_i}]_{i,j}.\mathsf{priv}(2)$ belong to $\mathbb{Z}_{\mathbf{p}}^{\mathsf{d}}$

3. Construct three vectors $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ in $\mathbb{Z}_{\mathbf{p}}^{3n\mathsf{d}}$ as follows.

   - Set $\mathbf{A}$ as the vector of level pub parts of encodings. That is, $\mathbf{A} = \left(\{[\mathbf{x_i}]_{i,j}.\mathsf{pub}\}_{i\in[n],j\in[3]}\right)$.
   - Set $\mathbf{B}$ as the vector of level $\mathsf{priv}(1)$ part of encodings. That is, $\mathbf{B} = \left(\{[\mathbf{x_i}]_{i,j}.\mathsf{priv}(1), \}_{i\in[n],j\in[3]}\right)$.
   - Set $\mathbf{C}$ as the vector of level $\mathsf{priv}(3)$ part of encodings. That is, $\mathbf{C} = \left(\{[\mathbf{x_i}]_{i,j}.\mathsf{priv}(2), \}_{i\in[n],j\in[3]}\right)$.

4. Encrypt these encodings using $\mathsf{3FE}$ scheme and output the resulting ciphertext. Formally, output $\mathsf{CT} \leftarrow \mathsf{3FE.Enc}(sk, \mathbf{A}, \mathbf{B}, \mathbf{C})$

<u>$\mathsf{FE}_3.\mathsf{KeyGen}(\mathsf{MSK}, i, f \in \mathcal{F}_{\mathsf{FE}_3,\lambda,n})$</u> : The key generation on input the master secret key MSK, an index i and a cubic integer polynomial f with coefficients over n variables from $[-\rho, \rho]$ does the following:

1. Parse $\mathsf{MSK} = (\mathsf{params}, sk, \mathsf{sp}, \{[\mathbf{z_i}]_{i,j}\}_{i\in[n],j\in[3]}, \{q_j\}_{j\in[\eta]})$.

2. See f as a polynomial with short coefficients over $\mathbb{Z}$. Let $\phi_{q_i,f}$ denote the resulting polynomial in $\mathcal{F}_{\mathsf{3FE}}$ that computes $\mathsf{TCE.Decode}(q_i, f, \cdot) \in \mathbb{Z}_{\mathbf{p}}$,

3. Compute a key for the function $sk_f \leftarrow \mathsf{3FE.KeyGen}(sk, \phi_{q_i,f})$. Output (q_i, sk_f).

<u>$\mathsf{FE}_3.\mathsf{Dec}\,((q, sk_f), \mathsf{CT})$</u> : The decryption algorithm on input a $\mathsf{3FE}$ functional key sk_f and TCE decoding parameter q and a ciphertext CT does the following.

1. Compute $\mathsf{temp} \leftarrow \mathsf{3FE.Dec}(sk_f, 1^{B_{\mathsf{FE}_3}}, \mathsf{CT})$ for some large enough polynomial B_{FE_3} to ensure correctness (described shortly).

2. temp is either a value in $[-B_{\mathsf{FE}_3}, B_{\mathsf{FE}_3}]$ or $\perp$. If it is $\perp$, output 1 otherwise output 0.

Now we argue the properties associated with the scheme.

Correctness: We argue correctness now. Consider the following:

- Ciphertext, $\mathsf{CT} \leftarrow \mathsf{3FE.Enc}(sk, \{[\mathbf{x_i}]_{i,j}.\mathsf{pub}\}_{i\in[n],j\in[3]}, \{[\mathbf{x_i}]_{i,j}.\mathsf{priv}(1)\}_{i\in[n],j\in[3]}, \{[\mathbf{x_i}]_{i,j}.\mathsf{priv}(2)\}_{i\in[n],j\in[3]})$.

- Function key for f, $sk_f \leftarrow \mathsf{3FE.KeyGen}(sk, \phi_{q,f})$. Here q is the decoding parameter.

Due to the correctness of the scheme $\mathsf{3FE}$ and cubic evaluation property of TCE, the decryption function , $\mathsf{3FE.Dec}(sk_f, 1^{B_{\mathsf{FE}_3}}, \mathsf{CT})$ does the following:
It checks $|\mathsf{Decode}(q, f, \{[\mathbf{x_i}]_{i,j}\}_{i\in[n],j\in[3]})| < B_{\mathsf{FE}_3}$. If this is the case it outputs $\mathsf{Decode}(q, f, \{[\mathbf{x_i}]_{i,j}\}_{i\in[n],j\in[3]})$, otherwise it outputs $\perp$. Now there are two cases:

- If $f(x_1, .., x_n) = 0$ then $|\mathsf{Decode}(q, f, \{[\mathbf{x_i}]_{i,j}\}_{i\in[n],j\in[3]})| < B_{\mathsf{FE}_3}$ due to correctness of TCE. In this case we always output 0. Thus $B_{\mathsf{FE}_3} = \mathsf{TCEbound}(\lambda, n)$

- If $f(x_1, .., x_n) \neq 0$ then $|\mathsf{Decode}(q, f, \{[\mathbf{x_i}]_{i,j}\}_{i\in[n],j\in[3]})| > \mathsf{TCEbound}(\lambda, n)$ with overwhelming probability due to correctness of TCE. In this case, we output 1 with overwhelming probability as $\mathsf{3FE}$ decryption outputs a $\perp$ with overwhelming probability.

Efficiency: We now bound the size of the circuit computing ciphertext to encrypt $\mathbf{x} = (x_1, .., x_n) \in [-\rho, \rho]^n$. Encryption of $\mathbf{x}$ consists of $\mathsf{3FE}$ encryption of three encoding parts of x_i for $i \in [n]$. The size of circuit computing each encoding $[\mathbf{x_i}]_{i,j}$ is polynomial in $3\log_2 p \cdot \mathsf{d} < poly(\lambda, \log n)$ for some polynomial $poly$, due to the efficiency of the TCE scheme. Due to the linear efficiency of $\mathsf{3FE}$ the size of circuit computing CT is less than $n \cdot poly'(\lambda, \log n)$ for some polynomial $poly'$. Note that $n < 2^\lambda$, hence, the claim follows.

12.2 Security Proof

Now we argue security.

Theorem 11. *If* $\mathsf{3FE}$ *is a secure three-restricted functional encryption scheme and* TCE *satisfies* $\mathcal{S}_\eta-$*tempered security, then the scheme described in Section 12.1 is a* $\mathcal{S}_\eta-$*bounded secure semi-functional functional encryption scheme for homogenous degree three polynomials according to definition 18.*

Proof. First we present the semi-functional algorithms and then prove $\mathcal{S}_\eta-$bounded indistinguishability of semi-functional ciphertexts and $\mathcal{S}_\eta-$bounded indistinguishability of semi-functional keys separately.

$\underline{\mathsf{FE}_3.\mathsf{sfKG}(\mathsf{MSK}, k, f \in \mathcal{F}_{\mathsf{FE}_3,\lambda,n}, \theta):}$ The key generation on input the master secret key $\mathsf{MSK} = (\mathsf{params}, sk, \mathsf{sp}, \{[\mathbf{z_i}]_{i,j}\}_{i\in[n],j\in[3]}\ \{q_j\}_{j\in[\eta]})$, a cubic integer polynomial f over n variables from $[-\rho, \rho]$, an index $k \in [\eta]$ along with a value $\theta \in \mathbb{Z}_\mathbf{p}$ does the following:

1. Compute $\mathsf{leak}_{sim} \leftarrow \mathsf{TCE.Sim}\left(q_k, f, \{[\mathbf{z_i}]_{i,1}\}_{i\in[n]}, \{[\mathbf{z_i}]_{i,2}\}_{i\in[n]}, \{[\mathbf{z_i}]_{i,3}\}_{i\in[n]}, \theta\right)$.
2. Compute a 3FE semi-functional key for the function $\phi_{q_k,f}$, $sk_{f,\theta} \leftarrow \mathsf{3FE.sfKG}(sk, \phi_{q_k,f}, \mathsf{leak}_{sim})$. Output $sk_{f,\theta}$.

We now describe the semi-functional encryption algorithm:

$\underline{\mathsf{FE}_3.\mathsf{sfEnc}(\mathsf{MSK}, 1^n)}$: On input the encryption key $\mathsf{MSK} = (\mathsf{params}, sk, \mathsf{sp}, \left\{[\mathbf{z_i}]_{\{j\}}\right\}_{i\in[n],j\in[3]}, \{q_j\}_{j\in[\eta]})$ and the length of the plaintext message n, the encryption algorithm does the following:

1. Parse $\mathbf{A} = \left(\{[\mathbf{z_i}]_{i,j}.\mathsf{pub}\}_{i\in[n],j\in[3]}\right)$ in $\mathbb{Z}_{\mathbf{p}}^{3n\mathsf{d}}$.
2. Encrypt $\mathbf{A}$ using the semi-functional encryption algorithm of 3FE scheme and output the resulting ciphertext. Formally, output $\mathsf{ct_{sf}} \leftarrow \mathsf{3FE.sfEnc}(sk, \mathbf{A}, 1^{3nd}, 1^{3n\mathsf{d}})$.

We now prove the indistinguishability of semi-functional key property.

$\mathcal{S}_\eta-$bounded Indistinguishability of semi-functional key property: We do this by presenting two hybrids, where the first hybrid correspond to the security where when the function keys are honestly generated whereas the last hybrid corresponds to the security game when the functional keys are semi-functional.

$\mathbf{Hybrid}_0$: This corresponds to the security game with challenge bit $\mathbf{b} = 0$:

1. Adversary outputs message queries $\mathbf{X}^k = (x_1^k, .., x_n^k)$ for $k \in [q]$.
2. Challenger runs Setup to get 3FE encryption key sk, the encodings $\{[\mathbf{z_i}]_{i,j}\}_{i\in[n],j\in[3]}$ and the TCE public parameters params and encoding parameter sp.It also samples decoding parameters $\{q_i\}_{i\in[\eta]}$.
3. The challenger computes $\mathsf{CT}_k \leftarrow \mathsf{Enc}(\mathsf{MSK}, \mathbf{X}^k)$ for $k \in [q]$.
4. Now the adversary requests functions $(f_1, .., f_\eta) \in \mathcal{S}_\eta$. It specifies values θ_i for $i \in [\eta]$.

   The polynomials (q_i, f_i) uniquely defines a cubic polynomial ϕ_{q_i,f_i} for each $i \in [\eta]$.
5. The challenger computes $\mathsf{leak}_i \leftarrow \mathsf{TCE.Sim}(q_i, f_i, \{[\mathbf{z_i}]_{i,j}\}_{i\in[n],j\in[3]}, f_i, \theta_i)$
6. Challenger outputs $(q_i, sk_{f_i} \leftarrow \mathsf{3FE.KeyGen}(sk, \phi_{q_i,f_i}))$ as the function key for $i \in [\eta]$.
7. Adversary outputs b'.

$\mathbf{Hybrid}_1$: This corresponds to the real security game with challenge bit $\mathbf{b} = 1$. The change is marked with the boldfaced word [**Change**]:

1. Adversary outputs message queries $\mathbf{X}^k = (x_1^k, .., x_n^k)$ for $k \in [q]$.
2. Challenger runs Setup to get 3FE encryption key sk, the encodings $\{[\mathbf{z_i}]_{i,j}\}_{i\in[n],j\in[3]}$ and the TCE public parameters params and encoding parameter sp. It also samples decoding parameters $\{q_i\}_{i\in[\eta]}$.

3. The challenger computes $\mathsf{CT}_k \leftarrow \mathsf{Enc}(\mathsf{MSK}, \mathbf{X}^k)$ for $k \in [q]$.

4. Now the adversary requests functions $(f_1, .., f_\eta) \in \mathcal{S}_\eta$. It specifies values θ_i for $i \in [\eta]$. The polynomials (q_i, f_i) uniquely defines a cubic polynomial ϕ_{q_i, f_i} for each $i \in [\eta]$.

5. The challenger computes $\mathsf{leak}_i \leftarrow \mathsf{TCE.Sim}(q_i, f_i, \{[\mathbf{z_i}]_{i,j}\}_{i \in [n], j \in [3]}, f_i, \theta_i)$

6. [**Change**] Challenger outputs $(q_i, sk_{f_i, \theta_i} \leftarrow \mathsf{3FE.sfKG}(sk, \phi_{f_i}, \mathsf{leak}_i))$ as the function key for $i \in [\eta]$.

7. Adversary outputs b'.

Lemma 3. *If* 3FE *scheme satisfies indistinguishability of semi-functional key property then there exists some constant $c > 0$ such that for any adversary of size 2^{λ^c}, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_0) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_1) = 1]| < 2^{-\lambda^c}$.*

Proof. (Sketch) The only way in which the above two hybrids differ is the way the keys for functions f_i for each $i \in [\eta]$ are generated. In $\mathbf{Hybrid}_0$ they are generated using 3FE.KeyGen, while in $\mathbf{Hybrid}_1$ they are generated using 3FE.sfKG algorithm. Note that in both the hybrids 3FE.sfEnc is not used. The reduction can be sketched as follows. The reduction generates the TCE parameters and encodings itself. Then, it gets from the adversary the messages $\mathbf{X}^k$, functions $(f_1, .., f_\eta)$ and values $(\theta_1, .., \theta_\eta)$. It generates values leak_i (using TCE parameters) and passes it along with messages and functions to the challenger of 3FE scheme. Challenger can then encrypt the cipher-text honestly. It flips a coin and either sends functional keys or the semi-functional keys. These are then used to simulate rest of the game. In the end the adversary outputs bit b' which the reduction outputs as it is. The indistinguishability now follows from the indistinguishability of semi-functional keys. □

$\mathcal{S}_\eta-$bounded Indistinguishability of semi-functional ciphertext property Fix messages $M_i = \{(\boldsymbol{x}_i)\}_{i \in \Gamma}$ for some polynomial Γ and a challenge $M^* = (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. Also fix $f_1, ..., f_\eta \in \mathcal{S}_\eta$. This defines $\mathsf{aux} = (1^\lambda, 1^n, \Gamma, M_i = \{(\boldsymbol{x}_i)\}_{i \in \Gamma}, M^* = (\boldsymbol{x^*}), f_1, .., f_\eta)$. Set $\theta_i = f_i(M^*)$ for all $i \in [\eta]$.

We now prove security by describing hybrids and arguing indistinguishability between them.
$\mathbf{Hybrid}_0$: This corresponds to the security game with challenge bit $\mathbf{b} = 0$:

1. Challenger runs Setup to get 3FE encryption key sk, the encodings $\{[\mathbf{z_i}]_{i,j}\}_{i \in [n], j \in [3]}$ and the TCE public parameters params and encoding parameter sp and decoding parameters $\{q_i\}_{i \in [\eta]}$.

2. The challenger computes $\mathsf{CT}_k \leftarrow \mathsf{Enc}(\mathsf{MSK}, M_i)$ for $i \in [\Gamma]$.

3. The challenger also encrypts M^*.

   (a) Run $\mathsf{TCE.SetupEnc}(\mathsf{params}) \rightarrow \mathsf{sp}_1$.

   (b) Encode each x_i^* for $i \in [n]$ at all the three levels. That is compute $[\mathbf{x_i^*}]_{i,j} \leftarrow \mathsf{Encode}(\mathsf{sp}_1, x_i^*, i, j)$ for every $i \in [n]$ and $j \in [3]$. Denote $[\mathbf{x_i^*}]_{i,j} = ([\mathbf{x_i^*}]_{i,j}.\mathsf{pub}, [\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(1), [\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(2))$. Here $[\mathbf{x_i^*}]_{i,j}.\mathsf{pub}$, $[\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(1)$ and $[\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(2)$ belong to $\mathbb{Z}_\mathbf{p}^\mathsf{d}$

   (c) Construct three vectors $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ in $\mathbb{Z}_\mathbf{p}^{3n\mathsf{d}}$ as follows.

   - Set $\mathbf{A}$ as the vector of level pub parts of encodings. That is, $\mathbf{A} = \left(\{[\mathbf{x_i^*}]_{i,j}.\mathsf{pub}\}_{i \in [n], j \in [3]}\right)$.

- Set $\mathbf{B}$ as the vector of level $\mathsf{priv}(1)$ part of encodings. That is, $\mathbf{B} = \left(\{[\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(1),\}_{i\in[n],j\in[3]}\right)$.
- Set $\mathbf{C}$ as the vector of level $\mathsf{priv}(3)$ part of encodings. That is, $\mathbf{C} = \left(\{[\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(2),\}_{i\in[n],j\in[3]}\right)$.

(d) Encrypt these encodings using 3FE scheme and output the resulting ciphertext. Formally, output $\mathsf{CT} \leftarrow \mathsf{3FE.Enc}(sk, \mathbf{A}, \mathbf{B}, \mathbf{C})$

4. The polynomials (q_i, f_i) uniquely defines a cubic polynomial ϕ_{q_i,f_i} for each $i \in [\eta]$.

5. The challenger computes $\mathsf{leak}_i \leftarrow \mathsf{TCE.Sim}(q_i, f_i, \{[\mathbf{z_i}]_{i,j}\}_{i\in[n],j\in[3]}, f_i, \theta_i)$

6. Challenger outputs $(q_i, sk_{f_i} \leftarrow \mathsf{3FE.sfKG}(sk, \phi_{q_i,f_i}, \mathsf{leak}_i))$ as the function key for $i \in [\eta]$.

7. Adversary outputs b'.

Hybrid$_1$: This corresponds is the same as the previous hybrid, except that function keys are generated using a semi-functional key with different hardwired values. The change is marked with boldfaced word [**Change**].

1. Challenger runs Setup to get 3FE encryption key sk, the encodings $\{[\mathbf{z_i}]_{i,j}\}_{i\in[n],j\in[3]}$ and the TCE public parameters params and encoding parameter sp and decoding parameters $\{q_i\}_{i\in[\eta]}$.

2. The challenger computes $\mathsf{CT}_k \leftarrow \mathsf{Enc}(\mathsf{MSK}, M_i)$ for $i \in [\Gamma]$.

3. The challenger also encrypts M^*.

   (a) Run $\mathsf{TCE.SetupEnc}(\mathsf{params}) \rightarrow \mathsf{sp}_1$.

   (b) Encode each x_i^* for $i \in [n]$ at all the three levels. That is compute $[\mathbf{x_i^*}]_{i,j} \leftarrow \mathsf{Encode}(\mathsf{sp}_1, x_i^*, i, j)$ for every $i \in [n]$ and $j \in [3]$. Denote $[\mathbf{x_i^*}]_{i,j} = ([\mathbf{x_i^*}]_{i,j}.\mathsf{pub}, [\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(1), [\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(2))$. Here $[\mathbf{x_i^*}]_{i,j}.\mathsf{pub}$, $[\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(1)$ and $[\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(2)$ belong to $\mathbb{Z}_\mathbf{p}^\mathsf{d}$

   (c) Construct three vectors $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ in $\mathbb{Z}_\mathbf{p}^{3n\mathsf{d}}$ as follows.

   - Set $\mathbf{A}$ as the vector of level pub parts of encodings. That is, $\mathbf{A} = \left(\{[\mathbf{x_i^*}]_{i,j}.\mathsf{pub}\}_{i\in[n],j\in[3]}\right)$.
   - Set $\mathbf{B}$ as the vector of level $\mathsf{priv}(1)$ part of encodings. That is, $\mathbf{B} = \left(\{[\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(1),\}_{i\in[n],j\in[3]}\right)$.
   - Set $\mathbf{C}$ as the vector of level $\mathsf{priv}(3)$ part of encodings. That is, $\mathbf{C} = \left(\{[\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(2),\}_{i\in[n],j\in[3]}\right)$.

   (d) Encrypt these encodings using 3FE scheme and output the resulting ciphertext. Formally, output $\mathsf{CT} \leftarrow \mathsf{3FE.Enc}(sk, \mathbf{A}, \mathbf{B}, \mathbf{C})$

4. The polynomials (q_i, f_i) uniquely defines a cubic polynomial ϕ_{q_i,f_i} for each $i \in [\eta]$.

5. [**Change**] The challenger computes $\mathsf{leak}_i \leftarrow \mathsf{TCE.Dec}(q_i, f_i, \{[\mathbf{x_i^*}]_{i,j}\}_{i\in[n],j\in[3]})$

6. Challenger outputs $(q_i, sk_{f_i} \leftarrow \mathsf{3FE.sfKG}(sk, \phi_{q_i,f_i}, \mathsf{leak}_i))$ as the function key for $i \in [\eta]$.

7. Adversary outputs b'.

Lemma 4. *If 3FE satisfies semi-functional security, then there exists a constant $c > 0$ such that for any adversary $\mathcal{A}$ of size 2^{λ^c}, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_0) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_1) = 1| < 2^{-\lambda^c}$.*

Proof. (Sketch) The only difference between the two hybrids is the hardwirings while computing semi-functional functional keys. In both hybrids, ciphertexts are generated by using $\mathsf{3FE.Enc}$ algorithm. The indistinguishability follows from the indistinguishability of semi-functional keys property of the $\mathsf{3FE}$ scheme. □

Hybrid$_2$: This corresponds is the same as the previous hybrid, except that challenge ciphertext is generated using semi-functional encryption algorithm of $\mathsf{3FE}$ scheme.

1. Challenger runs Setup to get $\mathsf{3FE}$ encryption key sk, the encodings $\{[\mathbf{z_i}]_{i,j}\}_{i\in[n],j\in[3]}$ and the TCE public parameters params and encoding parameter sp and decoding parameters $\{q_i\}_{i\in[\eta]}$.

2. The challenger computes $\mathsf{CT}_k \leftarrow \mathsf{Enc}(\mathsf{MSK}, M_i)$ for $i \in [\Gamma]$.

3. The challenger also encrypts M^*.

   (a) Run $\mathsf{TCE.SetupEnc}(\mathsf{params}) \rightarrow \mathsf{sp}_1$.

   (b) Encode each x_i^* for $i \in [n]$ at all the three levels. That is compute $[\mathbf{x_i^*}]_{i,j} \leftarrow \mathsf{Encode}(\mathsf{sp}_1, x_i^*, i, j)$ for every $i \in [n]$ and $j \in [3]$. Denote $[\mathbf{x_i^*}]_{i,j} = ([\mathbf{x_i^*}]_{i,j}.\mathsf{pub}, [\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(1), [\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(2))$. Here $[\mathbf{x_i^*}]_{i,j}.\mathsf{pub}$, $[\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(1)$ and $[\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(2)$ belong to $\mathbb{Z}_\mathbf{p}^\mathsf{d}$

   (c) Construct three vectors $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ in $\mathbb{Z}_\mathbf{p}^{3n\mathsf{d}}$ as follows.

   - Set $\mathbf{A}$ as the vector of level pub parts of encodings. That is, $\mathbf{A} = \left(\{[\mathbf{x_i^*}]_{i,j}.\mathsf{pub}\}_{i\in[n],j\in[3]}\right)$.
   - Set $\mathbf{B}$ as the vector of level $\mathsf{priv}(1)$ part of encodings. That is, $\mathbf{B} = \left(\{[\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(1), \}_{i\in[n],j\in[3]}\right)$.
   - Set $\mathbf{C}$ as the vector of level $\mathsf{priv}(3)$ part of encodings. That is, $\mathbf{C} = \left(\{[\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(2), \}_{i\in[n],j\in[3]}\right)$.

   (d) [**Change**] Encrypt these encodings using $\mathsf{3FE}$ scheme and output the resulting ciphertext. Formally, output $\mathsf{CT} \leftarrow \mathsf{3FE.sfEnc}(sk, \mathbf{A}, 1^{3nd}, 1^{3n\mathsf{d}})$.

4. The polynomials (q_i, f_i) uniquely defines a cubic polynomial ϕ_{q_i,f_i} for each $i \in [\eta]$.

5. The challenger computes $\mathsf{leak}_i \leftarrow \phi_{q_i,f_i}(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \mathsf{TCE.Dec}(q_i, f_i, \{[\mathbf{x_i^*}]_{i,j}\}_{i\in[n],j\in[3]})$.

6. Challenger outputs $(q_i, sk_{f_i} \leftarrow \mathsf{3FE.sfKG}(sk, \phi_{q_i,f_i}, \mathsf{leak}_i))$ as the function key for $i \in [\eta]$.

7. Adversary outputs b'.

Lemma 5. *If* $\mathsf{3FE}$ *is a secure three restricted FE scheme, then there exists a constant* $c > 0$ *such that for any adversary* $\mathcal{A}$ *of size* 2^{λ^c}, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_1) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_2) = 1| < 2^{-\lambda^c}$.

Proof. (Sketch.) The only difference between the two hybrids is the way CT^* is generated. In $\mathbf{Hybrid}_1$ it is generated using $\mathsf{3FE.Enc}$ algorithm, encrypting $(\mathbf{A}, \mathbf{B}, \mathbf{C})$. In $\mathbf{Hybrid}_2$, it is generated as $\mathsf{3FE.sfEnc}(sk, \mathbf{A}, 1^{3nd}, 1^{3n\mathsf{d}})$. Note that function keys are generated according to the requirement of the indistinguishability of semi-functional ciphertexts security game. The indistinguishability follows from the indistinguishability of semi-functional ciphertext property of the $\mathsf{3FE}$ scheme. □

Hybrid$_3$: This corresponds to challenge bit $\mathbf{b} = 1$. Namely, this hybrid is the same as the previous one except that both $\mathbf{A}$ and leak_i are generated as in TCE security game with challenge bit 1.

1. Challenger runs Setup to get 3FE encryption key sk, the encodings $\{[\mathbf{z_i}]_{i,j}\}_{i\in[n],j\in[3]}$ and the TCE public parameters params and encoding parameter sp and decoding parameters $\{q_i\}_{i\in[\eta]}$.
2. The challenger computes $\mathsf{CT}_k \leftarrow \mathsf{Enc}(\mathsf{MSK}, M_i)$ for $i \in [\Gamma]$.
3. The challenger also encrypts M^*.
   (a) Run $\mathsf{TCE.SetupEnc}(\mathsf{params}) \rightarrow \mathsf{sp}_1$.
   (b) Encode each x_i^* for $i \in [n]$ at all the three levels. That is compute $[\mathbf{x_i^*}]_{i,j} \leftarrow \mathsf{Encode}(\mathsf{sp}_1, x_i^*, i, j)$ for every $i \in [n]$ and $j \in [3]$. Denote $[\mathbf{x_i^*}]_{i,j} = ([\mathbf{x_i^*}]_{i,j}.\mathsf{pub}, [\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(1), [\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(2))$. Here $[\mathbf{x_i^*}]_{i,j}.\mathsf{pub}$, $[\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(1)$ and $[\mathbf{x_i^*}]_{i,j}.\mathsf{priv}(2)$ belong to $\mathbb{Z}_\mathbf{p}^\mathsf{d}$
   (c) Construct three vectors $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ in $\mathbb{Z}_\mathbf{p}^{3n\mathsf{d}}$ as follows.
      - [**Change**] Set $\mathbf{A}$ as the vector of level pub parts of encodings. That is, $\mathbf{A} = \left(\{[\mathbf{z_i}]_{i,j}.\mathsf{pub}\}_{i\in[n],j\in[3]}\right)$.
   (d) Encrypt these encodings using 3FE scheme and output the resulting ciphertext. Formally, output $\mathsf{CT} \leftarrow \mathsf{3FE.sfEnc}(sk, \mathbf{A}, 1^{3n\mathsf{d}}, 1^{3n\mathsf{d}})$.
4. The polynomials (q_i, f_i) uniquely defines a cubic polynomial ϕ_{q_i,f_i} for each $i \in [\eta]$.
5. [**Change**] The challenger computes $\mathsf{leak}_i \leftarrow \mathsf{TCE.Sim}(q_i, f_i, \{[\mathbf{z_i}]_{i,j}\}_{i\in[n],j\in[3]}, f_i, \theta_i)$.
6. Challenger outputs $(q_i, sk_{f_i} \leftarrow \mathsf{3FE.sfKG}(sk, \phi_{q_i,f_i}, \mathsf{leak}_i))$ as the function key for $i \in [\eta]$.
7. Adversary outputs b'.

Lemma 6. *If* TCE *satisfies* $\mathcal{S}_\eta-$*tempered security, then there exists a constant* $c > 0$ *such that for any adversary* $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_2) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_3) = 1| < 1 - 1/\lambda + \mathsf{negl}(\lambda)$.

Proof. (Sketch.) Note that both the hybrids depend only on public part of encoding $\mathbf{A}$ and leakages θ_i. In $\mathbf{Hybrid}_2$, they correspond to actual encoding of M^*, in $\mathbf{Hybrid}_3$, they are simulated. The indistinguishability follows from $\mathcal{S}_\eta$ tempered security of TCE scheme. □

Thus we get the result from the above three lemmas.

□

13 Step 4: (Sublinear) Semi-Functional Secret Key FE from Semi-Functional FE for Cubic Polynomials

13.1 Randomizing Polynomials

A randomizing polynomials scheme defined over a field $\mathbb{Z}_\mathbf{p}$ consists of probabilistic polynomial time algorithms ($\mathsf{CktEncd}, \mathsf{InpEncd}, \mathsf{Decd}$) and is associated with a class of circuits,

$$\mathcal{F}_{n,s} = \{C : \{0,1\}^n \rightarrow \{0,1\}^m \ : \ C \text{ is of size } s\}$$

- $\mathsf{CktEncd}(1^\lambda, C)$: On input security parameter λ, a circuit C, it outputs polynomials $(p_1, \ldots, p_N)$ over $\mathbb{Z}_\mathbf{p}$. This is a deterministic algorithm.

- $\mathsf{InpEncd}(x; R)$: On input x, randomness R, it outputs the input encoding $\mathbf{x}$.
- $\mathsf{Decd}(p_1(\mathbf{x}), \ldots, p_N(\mathbf{x}))$: On input $p_1(\mathbf{x}), \ldots, p_N(\mathbf{x})$, it outputs the decoded value y.

Definition 19. *A tuple of algorithms* $\mathsf{RP} = (\mathsf{CktEncd}, \mathsf{InpEncd}, \mathsf{Decd})$ *is a randomizing polynomials scheme with* ε*-sublinear randomness complexity for a class of circuits* $\mathcal{F}_{n,s}$ *over* $\mathbb{Z}_\mathbf{p}$ *if the following properties are satisfied:*

- **Correctness***: For every* $C \in \mathcal{F}_{n,s}$*, input* $x \in \{0,1\}^n$*, for sufficiently large* $\lambda \in \mathbb{N}$*, we have* $\Pr[\mathsf{Decd}(p_1(\mathbf{x}), \ldots, p_N(\mathbf{x})) = C(x)] \geq 1 - \mathsf{negl}(\lambda)$*, for some negligible function* negl*, where:*

  - $(p_1, \ldots, p_N) \leftarrow \mathsf{CktEncd}(1^\lambda, C)$
  - $\mathbf{x} \leftarrow \mathsf{InpEncd}(x; R)$*, where* R *is sampled from uniform distribution.*

- adv-**Security***: There exists a simulator* Sim *such that the following holds: for every* $C \in \mathcal{F}_{n,s}, x \in \{0,1\}^n$*, sufficiently large* $\lambda \in \mathbb{N}$*, consider* $\mathsf{CktEncd}(1^\lambda, C) \rightarrow (p_1, \ldots, p_N)$ *and* $\mathsf{InpEncd}(x) \rightarrow \mathbf{x}$*. Then, for all adversaries* $\mathcal{A}$ *of size at most* $O(2^\lambda)$*,*

$$\Pr[\mathcal{A}(p_1(\mathbf{x}), ..., p_N(\mathbf{x})) = 1] - \Pr[\mathcal{A}(\mathsf{Sim}(1^\lambda, C, C(x))) = 1] < \mathsf{adv}(\lambda) + \mathsf{negl}(\lambda)$$

- ε**-Sublinear Input Encoding***: We require that the size of the circuit computing* $\mathsf{InpEncd}(x; R)$ *is* $(n + s^{\frac{1}{1+\varepsilon}}) \cdot poly(\lambda)$*.*

Moreover, we say that RP *is a degree-*d *randomizing polynomials scheme if every polynomial* p_i *is homogenous and has degree exactly* d*.*

Definition 20. *Let* λ *be the security parameter. By* $\mathcal{C}_{n,s}$ *we denote the set of circuits* $C : \{0,1\}^n \rightarrow \{0,1\}^*$ *with size bounded by some polynomial* $s(n, \lambda)$ *and depth* λ*. In particular, this class contains* NC^1 *circuits of size* $s(n, \lambda)$*.*

We construct a sublinear semi-functional secret key FE for $\mathcal{C}_{n,s}$ for $s = n^{1+\epsilon}$ for some $\epsilon > 0$ starting from semi-functional FE for $\mathcal{F}_{\lambda,3}$, where $\mathcal{F}_\lambda$ consists of all polynomial-sized (in λ) circuits and $\mathcal{F}_{\lambda,3}$ consists of all cubic polynomials over $\mathbb{Z}$. As an intermediate tool, we consider the notion degree three randomizing polynomials with ε-sublinear randomness complexity $\mathsf{RP} = (\mathsf{CktEncd}, \mathsf{InpEncd}, \mathsf{Decd})$ for some $\varepsilon > 0$. Let $\mathsf{Sim}_{\mathsf{RP}}$ be the simulator associated with the randomizing polynomials scheme. Such RP was constructed in [LT17]:

Theorem 12 (Imported Theorem [LT17])**.** *Assuming there exists pseudorandom generators with*

- *block locality three and stretch* $n^{1+\epsilon'}$ *for some* $\epsilon' > 0$*.*
- *distinguishing gap bounded by* adv *for adversaries of size* 2^λ

there exist a adv*-secure degree three randomizing polynomials scheme with* $\frac{1}{1+\epsilon'}$*−sublinear efficiency.*

We now describe the ingredients of the construction:

Ingredients.

- A degree 3 randomizing polynomials scheme RP for $\mathcal{C}_{n,s}$ with $\epsilon = \frac{1}{1+\epsilon'}$ sublinear complexity. Here $\epsilon' > 0$ is some constant. For any circuit $C \in \mathcal{C}_{n,s}$, let N denote the number such that $\mathsf{CktEncd}(1^\lambda, C) = (p_1, ..., p_N)$. N is upper bounded by $s poly(\lambda)$ for some polynomial $poly$. Also, each polynomial p_i is such that the sum of the absolute values of the coefficients are bounded by a fixed polynomial $w(\lambda)$. Let RP satisfy $\mathsf{adv}_{\mathsf{RP}}-$security.

- A Semi-functional FE scheme for cubic polynomials to be $\mathsf{sFE}_3 = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ associated with semi-functional algorithms ($\mathsf{sFE}_3.\mathsf{sfEnc}, \mathsf{sFE}_3.\mathsf{sfKG}$). We require sFE_3 to satisfy $\mathcal{S}_\eta-$Bounded semi-functional security. $\mathcal{S}_\eta$ is the set $\mathcal{F}^N$, where the set $\mathcal{F}$ denotes the set of all homogeneous cubic polynomials with sum of absolute values of coefficients weight bounded by $w(\lambda)$. Note that the polynomials generated by RP are in class $\mathcal{F}$. In Section 12, we construct such a notion with distinguishing gap bounded by $1 - 1/\lambda$, but for a general exposition we assume that it is bounded by some advantage $\mathsf{adv}_{\mathsf{sFE}_3}$.

We denote the scheme we construct to be sFE.

<u>$\mathsf{Setup}(1^\lambda, 1^n)$</u>: On input security parameter λ, input length n, it executes the setup of the underlying semi-functional FE scheme to obtain $\mathsf{sFE}_3.\mathsf{MSK} \leftarrow \mathsf{sFE}.\mathsf{Setup}(1^\lambda, 1^{n'})$. It outputs secret key $\mathsf{MSK} = \mathsf{sFE}_3.\mathsf{MSK}$. Here n' is the output length of $|\mathsf{InpEncd}(\cdot, \cdot)|$. Note that $n' = npoly(\lambda)$ for some fixed polynomial $poly$.

<u>$\mathsf{KeyGen}(\mathsf{MSK}, C)$</u>: It takes as input master secret key MSK and circuit C.

- Compute the polynomials associated with the randomizing polynomials scheme; $(p_1, \ldots, p_N) \leftarrow \mathsf{CktEncd}(1^\lambda, C)$.

- Compute the sFE_3 keys associated with the polynomials $(p_1, \ldots, p_N)$; for every $i \in [N]$, compute $\mathsf{sFE}_3.\mathsf{KeyGen}(\mathsf{sFE}_3.\mathsf{MSK}; p_i)$ to obtain $\mathsf{sFE}_3.sk_{p_i}$.

Output the functional key $sk_C = (\mathsf{sFE}_3.sk_{p_1}, \ldots, \mathsf{sFE}_3.sk_{p_N})$. Note that $N = |C|poly(\lambda)$ for some polynomial $poly$.

<u>$\mathsf{Enc}(\mathsf{MSK}, x)$</u>: It takes as input the master secret key MSK and input x, of length n. It samples a binary string R uniformly at random of length ℓ_R. Here, ℓ_R is the length of randomness used in algorithm $\mathsf{InpEncd}$ to encode a circuit of size $|C|$ and input length n. It then computes $\mathbf{x} \leftarrow \mathsf{RP}.\mathsf{InpEncd}(x, R)$. It then computes $\mathsf{CT} \leftarrow \mathsf{sFE}_3.\mathsf{Enc}(\mathsf{MSK}, \mathbf{x})$. It outputs the ciphertext $\mathsf{sFE}.\mathsf{CT} = \mathsf{sFE}_3.\mathsf{CT}$.

<u>$\mathsf{Dec}(sk_C, \mathsf{CT})$</u>: It takes as input functional key sk_C and ciphertext CT. It executes the following steps:

- Parse sk_C as $(sk_{p_1}, \ldots, sk_{p_N})$. Compute $\mathsf{sFE}_3.\mathsf{Dec}(sk_{p_i}, \mathsf{CT})$ to obtain $\widetilde{y_i}$, for ever $i \in [N]$.

- Compute $\mathsf{RP}.\mathsf{Decode}(\widetilde{y_1}, \ldots, \widetilde{y_N})$ to obtain the output z.

Output z.

Correctness. Consider a circuit $C \in \mathcal{F}_\lambda$ and input x. Let the functional key, $sk_C = (sk_{p_1}, \ldots, sk_{p_N})$ and ciphertext, $\mathsf{CT} = \mathsf{sFE}_3.\mathsf{Enc}(\mathsf{MSK}, \mathbf{x})$, where $\mathbf{x} \leftarrow \mathsf{RP.InpEncd}(x, R)$, be as generated in the above scheme. From the correctness of sFE_3, we have that $\mathsf{sFE}_3.\mathsf{Dec}(sk_{p_i}, \mathsf{CT})$ yields $p_i(\mathbf{x})$ for every $i \in [N]$. Moreover, from the correctness of randomizing polynomials, we have that the output of $\mathsf{RP.Decode}$ on input $(p_1(\mathbf{x}), \ldots, p_N(\mathbf{x}))$ is $C(x)$.

Encryption Complexity. From the multiplicative overhead property in encryption complexity of sFE_3, we have:

$$\begin{aligned} |\mathsf{Enc}(\mathsf{MSK}, x)| &= |\mathbf{x}| \cdot poly_1(\lambda) \\ &= n' \cdot poly_1(\lambda) \\ &= npoly_2(\lambda) \cdot poly(\lambda) \\ &\leq |C|^{\varepsilon} \cdot poly(|x|, \lambda) \ (\because \varepsilon\text{-sublinear randomness complexity of } \mathsf{RP}) \end{aligned}$$

Thus, the encryption complexity is ε-sublinear in $|C|$, as intended.

13.2 Security

We show that sFE satisfies semi-functional security. Before we show that, we need to demonstrate the semi-functional algorithms.

$\underline{\mathsf{sFE.sfEnc}(\mathsf{MSK}, 1^{|x|})}$: On input master secret key MSK, length of input $1^{|x|}$, we compute $\mathsf{sFE}_3.\mathsf{FkCT} \leftarrow \mathsf{sFE}_3.\mathsf{sfEnc}(\mathsf{MSK}, 1^{|x|})$. Output the semi-functional ciphertext, $\mathsf{FkCT} = \mathsf{sFE}_3.\mathsf{FkCT}$.

$\underline{\mathsf{sFE.sfKG}(\mathsf{MSK}, C, \Theta)}$: On input master secret key MSK, circuit C, value θ, compute $\mathsf{Sim}(1^\lambda, C, \Theta)$ to obtain $(\theta_1, \ldots, \theta_N)$. It then computes $\mathsf{sFE}_3.\mathsf{fk.sk}_i \leftarrow \mathsf{sFE}_3.\mathsf{sfKG}(\mathsf{MSK}, p_i, \theta_i)$ for every $i \in [N]$, where $(p_1, \ldots, p_N) \leftarrow \mathsf{CktEncd}(1^\lambda, C)$. Output $sk_C = (\mathsf{sFE}_3.sk_1, \ldots, \mathsf{sFE}_3.sk_N)$.

Now we describe our theorem. Below we prove $(\mathsf{size}_\mathcal{A}, \mathsf{adv})-$semi-functional security for the scheme where $\mathsf{adv} = 1 - 1/\lambda$ and $\mathsf{size}_\mathcal{A} = 2^{\lambda^c}$ for some constant $c > 0$. We omit referring them for notational simplicity.

Theorem 13. *Assuming the $\mathcal{S}_\eta$bounded indistinguishability of semi-functional keys property of* sFE_3*, the scheme* sFE *satisfies indistinguishability of semi-functional keys property.*

Proof. (Sketch.) There are just two hybrids in the proof. First hybrid corresponds to the case when key and the ciphertexts are functionally generated. Second hybrid corresponds to the case when ciphertexts are functionally encrypted while the key is semi-functionally generated. Since sFE_3 scheme satisfies indistinguishability of semi-functional keys property, the claim follows. □

Theorem 14. *Assuming* $\mathsf{adv}_\mathsf{RP} + \mathsf{adv}_{\mathsf{sFE}_3} < 1 - 1/\lambda$*, the $\mathcal{S}_\eta$bounded semi-functional security of* sFE_3 *and* adv_RP*−security of* RP*, the scheme* sFE *satisfies indistinguishability of semi-functional ciphertext property.*

Proof. (Sketch.) We now list hybrids. First hybrid corresponds to the case when the ciphertext is functionally encrypted and the keys are semi-functional, whereas, in the last hybrid the ciphertext is semi-functionally encrypted and the keys are semi-functionally encrypted.

Hybrid$_0$:

1. Adversary $\mathcal{A}$ on input 1^λ, outputs challenge message x^*, message queries $\{x_i\}_{i\in[\Gamma]}$ and circuit C.

2. The challenger samples $\mathsf{MSK} \leftarrow \mathsf{sFE.Setup}(1^\lambda)$.

3. Encrypt message queries honestly $\mathsf{CT} \leftarrow \mathsf{sFE.Enc}(\mathsf{MSK}, x_i)$ for $i \in [\Gamma]$.

4. To encrypt the challenge message do the following:

   - Sample a binary string R uniformly at random of length ℓ_R. Here, ℓ_R is the length of randomness used in algorithm $\mathsf{InpEncd}$ to encode a circuit of size $|C|$ and input length n.
   - Compute $\mathbf{x} \leftarrow \mathsf{RP.InpEncd}(x^*, R)$.
   - Compute $\mathsf{CT}^* \leftarrow \mathsf{sFE}_3.\mathsf{Enc}(\mathsf{MSK}, \mathbf{x})$
   - Set $\theta = C(x^*)$

5. To generate the function key, do the following.

   - On input master secret key MSK, circuit C, value θ, compute $\mathsf{Sim}(1^\lambda, C, \Theta)$ to obtain $(\theta_1, \ldots, \theta_N)$. It then computes $\mathsf{sFE}_3.\mathsf{fk.sk}_i \leftarrow \mathsf{sFE}_3.\mathsf{sfKG}(\mathsf{MSK}, p_i, \theta_i)$ for every $i \in [N]$, where $(p_1, \ldots, p_N) \leftarrow \mathsf{CktEncd}(1^\lambda, C)$. Output $sk_C = (\mathsf{sFE}_3.\mathsf{fk.sk}_1, \ldots, \mathsf{sFE}_3.\mathsf{fk.sk}_N)$.

6. . Give the following to the adversary:

   - Challenge ciphertext CT^*.
   - Ciphertext queries $\{\mathsf{CT}_i\}_{i\in\Gamma}$
   - Function key sk_C.

7. $\mathcal{A}$ guesses bit b'.

Hybrid$_1$: This hybrid is the same as the previous one except that hardwirings in the semi-functional keys are done differently. We describe the hybrid now. The change is described with boldfaced word [**Change**].

1. Adversary $\mathcal{A}$ on input 1^λ, outputs challenge message x^*, message queries $\{x_i\}_{i\in[\Gamma]}$ and circuit C.

2. The challenger samples $\mathsf{MSK} \leftarrow \mathsf{sFE.Setup}(1^\lambda)$.

3. Encrypt message queries honestly $\mathsf{CT} \leftarrow \mathsf{sFE.Enc}(\mathsf{MSK}, x_i)$ for $i \in [\Gamma]$.

4. To encrypt the challenge message do the following:

   - Sample a binary string R uniformly at random of length ℓ_R. Here, ℓ_R is the length of randomness used in algorithm $\mathsf{InpEncd}$ to encode a circuit of size $|C|$ and input length n.
   - Compute $\mathbf{x} \leftarrow \mathsf{RP.InpEncd}(x^*, R)$.
   - Compute $\mathsf{CT}^* \leftarrow \mathsf{sFE}_3.\mathsf{Enc}(\mathsf{MSK}, \mathbf{x})$

- Set $\theta = C(x^*)$

5. [**Change**] To generate the function key, do the following.

    - On input master secret key MSK, circuit C, do the following.
    - Let $(p_1, \dots, p_N) \leftarrow \mathsf{CktEncd}(1^\lambda, C)$.
    - Set $(\theta_1, \dots, \theta_N) = (p_1(\mathbf{x}), \dots, p_N(\mathbf{x}))$
    - compute $\mathsf{sFE}_3.\mathsf{fk}.\mathsf{sk}_i \leftarrow \mathsf{sFE}_3.\mathsf{sfKG}(\mathsf{MSK}, p_i, \theta_i)$ for every $i \in [N]$. Set $sk_C = (\mathsf{sFE}_3.\mathsf{fk}.\mathsf{sk}_1, \dots, \mathsf{sFE}_3.\mathsf{fk}.\mathsf{sk}_N)$.

6. . Give the following to the adversary:

    - Challenge ciphertext CT^*.
    - Ciphertext queries $\{\mathsf{CT}_i\}_{i \in \Gamma}$
    - Function key sk_C.

7. $\mathcal{A}$ guesses bit b'.

Lemma 7. *If sFE_3 satisfies $\mathcal{S}_\eta$ indistinguishability of semi-functional key property then there exits a constant c_0 for any adversary D of size $2^{\lambda^{c_0}}$,* $|\Pr[D(\mathbf{Hybrid}_0) = 1] - \Pr[D(\mathbf{Hybrid}_1) = 1]| < 2^{-\lambda^{c_0}}$.

Proof. (Sketch). The only difference between the two hybrids is the way hardwirings θ_i for the keys $\mathsf{fk}.\mathsf{sk}_i$ for $i \in [N]$ are generated. In $\mathbf{Hybrid}_0$, they are generated using $\mathsf{RP}.\mathsf{Sim}$, while in $\mathbf{Hybrid}_1$, they are generated as $p_i(\mathbf{x})$ Note that in both hybrids CT^* is functionally generated. The claim then follows from the security of sFE_3 scheme.

□

Hybrid$_2$: This hybrid is the same as the previous one except that ciphertext is generated using $\mathsf{sFE}_3.\mathsf{sfEnc}$ algorithm. We describe the hybrid now.

1. Adversary $\mathcal{A}$ on input 1^λ, outputs challenge message x^*, message queries $\{x_i\}_{i \in [\Gamma]}$ and circuit C.

2. The challenger samples $\mathsf{MSK} \leftarrow \mathsf{sFE}.\mathsf{Setup}(1^\lambda)$.

3. Encrypt message queries honestly $\mathsf{CT} \leftarrow \mathsf{sFE}.\mathsf{Enc}(\mathsf{MSK}, x_i)$ for $i \in [\Gamma]$.

4. To encrypt the challenge message do the following:

    - Sample a binary string R uniformly at random of length ℓ_R. Here, ℓ_R is the length of randomness used in algorithm $\mathsf{InpEncd}$ to encode a circuit of size $|C|$ and input length n.
    - Compute $\mathbf{x} \leftarrow \mathsf{RP}.\mathsf{InpEncd}(x^*, R)$.
    - [**Change**] Compute $\mathsf{CT}^* \leftarrow \mathsf{sFE}_3.\mathsf{Enc}(\mathsf{MSK}, 1^\lambda)$
    - Set $\theta = C(x^*)$

5. To generate the function key, do the following.

- On input master secret key MSK, circuit C, do the following.
- Let $(p_1, \ldots, p_N) \leftarrow \mathsf{CktEncd}(1^\lambda, C)$.
- Set $(\theta_1, \ldots, \theta_N) = (p_1(\mathbf{x}), \ldots, p_N(\mathbf{x}))$
- Compute $\mathsf{sFE}_3.\mathsf{fk.sk}_i \leftarrow \mathsf{sFE}_3.\mathsf{sfKG}(\mathsf{MSK}, p_i, \theta_i)$ for every $i \in [N]$. Set $sk_C = (\mathsf{sFE}_3.\mathsf{fk.sk}_1, \ldots, \mathsf{sFE}_3.\mathsf{fk.sk}_N)$.

6. Give the following to the adversary:
   - Challenge ciphertext CT^*.
   - Ciphertext queries $\{\mathsf{CT}_i\}_{i \in \Gamma}$
   - Function key sk_C.
7. $\mathcal{A}$ guesses bit b'.

Lemma 8. *If* sFE_3 *satisfies* $\mathcal{S}_\eta-$*bounded indistinguishability of semi-functional ciphertexts then there exists a constant* c_1 *for any adversary* D *of size* $2^{\lambda^{c_1}}$, $|\Pr[D(\mathbf{Hybrid}_1) = 1] - \Pr[D(\mathbf{Hybrid}_2) = 1]| < \mathsf{adv}_{\mathsf{sFE}_3} + \mathsf{negl}(\lambda)$.

Proof. (Sketch). The only difference between the two hybrids is the way CT^* is generated. In $\mathbf{Hybrid}_1$, they are generated using $\mathsf{sFE}_3.\mathsf{Enc}$ algorithm, while in $\mathbf{Hybrid}_2$, they are generated using $\mathsf{sFE}_3.\mathsf{sfEnc}$ algorithm. Note that the keys are semi-functionally generated with $\theta_i = p_i(\mathbf{x})$, as required by indistinguishability of semi-functional ciphertexts property game of sFE_3. The claim then follows from the security of sFE_3 scheme.

□

$\mathbf{Hybrid}_3$: This hybrid is the same as the previous one except that the function key is generated using $\mathsf{sFE}_3.\mathsf{sfKG}$ algorithm. We describe the hybrid now.

1. Adversary $\mathcal{A}$ on input 1^λ, outputs challenge message x^*, message queries $\{x_i\}_{i \in [\Gamma]}$ and circuit C.
2. The challenger samples $\mathsf{MSK} \leftarrow \mathsf{sFE}.\mathsf{Setup}(1^\lambda)$.
3. Encrypt message queries honestly $\mathsf{CT} \leftarrow \mathsf{sFE}.\mathsf{Enc}(\mathsf{MSK}, x_i)$ for $i \in [\Gamma]$.
4. To encrypt the challenge message do the following:
   - Sample a binary string R uniformly at random of length ℓ_R. Here, ℓ_R is the length of randomness used in algorithm $\mathsf{InpEncd}$ to encode a circuit of size $|C|$ and input length n.
   - Compute $\mathbf{x} \leftarrow \mathsf{RP}.\mathsf{InpEncd}(x^*, R)$.
   - Compute $\mathsf{CT}^* \leftarrow \mathsf{sFE}_3.\mathsf{Enc}(\mathsf{MSK}, 1^\lambda)$
   - Set $\theta = C(x^*)$
5. [**Change**] To generate the function key, do the following.

- On input master secret key MSK, circuit C, value θ, compute $\mathsf{Sim}(1^\lambda, C, \Theta)$ to obtain $(\theta_1, \ldots, \theta_N)$. It then computes $\mathsf{sFE}_3.\mathsf{fk.sk}_i \leftarrow \mathsf{sFE}_3.\mathsf{sfKG}(\mathsf{MSK}, p_i, \theta_i)$ for every $i \in [N]$, where $(p_1, \ldots, p_N) \leftarrow \mathsf{CktEncd}(1^\lambda, C)$. Output $sk_C = (\mathsf{sFE}_3.\mathsf{fk.sk}_1, \ldots, \mathsf{sFE}_3.\mathsf{fk.sk}_N)$.

6. . Give the following to the adversary:
   - Challenge ciphertext CT^*.
   - Ciphertext queries $\{\mathsf{CT}_i\}_{i \in \Gamma}$
   - Function key sk_C.
7. $\mathcal{A}$ guesses bit b'.

Lemma 9. *If* RP *is* $\mathsf{adv}_{\mathsf{RP}}-$*secure then for any adversary* D *of size* 2^λ, $|\Pr[D(\mathbf{Hybrid}_2) = 1] - \Pr[D(\mathbf{Hybrid}_3) = 1]| < \mathsf{adv}_{\mathsf{RP}}$.

Proof. (Sketch). The only difference between the two hybrids is the way hardwirings θ_i are generated. In $\mathbf{Hybrid}_2$, they are generated as $p_i(\mathbf{x})$ where $\mathbf{x} \leftarrow \mathsf{RP.InpEncd}(x^*, R)$. In $\mathbf{Hybrid}_3$ they are simulated using simulator of the RP scheme. Note that in both the hybrids CT^* is semi-functionally encrypted and x^* is absent. The claim then follows from the security of RP scheme. □

From the lemmas above, as long as the sum of advantages $\mathsf{adv}_{\mathsf{RP}} + \mathsf{adv}_{\mathsf{sFE}_3} + \mathsf{negl} < 1 - 1/\lambda + \mathsf{negl}(\lambda)$, the claim goes through.

□

Remark 8. *From the above proof, we observe that as long as* $\mathsf{adv}_{\mathsf{RP}} + \mathsf{adv}_{\mathsf{sFE}_3} < 1 - 1/\lambda$, *the proof goes through. Thus we can allow a trade off in the required level of security between a three block local PRGs and* $\Delta\mathsf{RG}$. *This is because* $\mathsf{adv}_{\mathsf{RP}} = \mathsf{adv}_{PRG}$ *and* $\mathsf{adv}_{\mathsf{sFE}_3} = \mathsf{adv}_{\Delta\mathsf{RG}}$ *upto negligible factors. Here* adv_{PRG} *and* $\mathsf{adv}_{\Delta\mathsf{RG}}$ *is the allowed distinguishing gap for a three block local PRG and* $\Delta\mathsf{RG}$ *respectively.*

14 Step 5: Amplification

In this section, we construct $(\mathsf{size}_{\mathsf{FE}}, \mathsf{adv}_{\mathsf{FE}})-$secure ρ-sublinear secret key FE from the following ingredients:

Ingredients: Assume there exist:

- PRF in NC^1 with distinguishing gap $O(\mathsf{adv}_{\mathsf{PRF}})$ against adversaries of size $O(\mathsf{size}_{\mathsf{PRF}})$.
- A compact $(\mathsf{size}_{\mathsf{TFHE}}, \mathsf{adv}_{\mathsf{TFHE}})-$secure threshold fully homomorphic encryption scheme TFHE for $\mathcal{C}_{n,s}$, according to Definition 5. We note that there exist fixed polynomials p_1, p_2 such that for any circuit $C \in \mathcal{C}_{n,s}$ according to Definition 19 and any PRF in NC^1, the circuit $\mathsf{PartDec}(\cdot, \mathsf{Eval}(C, \cdot); \mathsf{PRF}(\cdot, \cdot))$ is in $\mathcal{C}_{n',s'}$ for $n' = n \cdot p_1(\lambda)$ and $s' = s \cdot p_2(\lambda)$.
- Statistically binding commitments Com in the CRS model, with $(\mathsf{size}_{\mathsf{Com}}, \mathsf{adv}_{\mathsf{Com}})$-security. These can be instantiated based on $(\mathsf{size}_{\mathsf{Com}}, \mathsf{adv}_{\mathsf{Com}})$-secure one-way functions.
- Sublinear Semi-Functional FE scheme sFE for circuit class $\mathcal{C}_{n',s'}$ with $(\mathsf{size}, \mathsf{adv})$ semi-functional security according to Section 9, Definition 17, where $\mathsf{adv} = 1 - 1/p(\lambda)$ for some polynomial p.

We describe the the construction below.

- $\underline{\mathsf{Setup}(1^\lambda):}$
  1. Set $t = \lambda \cdot p(\lambda)$.
  2. For $i \in [t]$, compute $sk_i \leftarrow \mathsf{sFE.Setup}(1^\lambda)$.
  3. Output $\mathsf{MSK} = (sk_1, ..., sk_t)$

- $\underline{\mathsf{Enc}(\mathsf{MSK}, \mathsf{m}):}$
  1. Parse $\mathsf{MSK} = (sk_1, ..., sk_t)$.
  2. Compute $(\mathsf{fpk}, \mathsf{fsk}_1, ..., \mathsf{fsk}_t) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^t)$.
  3. Compute $\mathsf{fct} \leftarrow \mathsf{TFHE.Enc}(\mathsf{fpk}, m)$.
  4. For $i \in [t]$, sample $K_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$.
  5. For $i \in [t]$, compute $\mathsf{CT}_i \leftarrow \mathsf{sFE.Enc}(sk_i, (\mathsf{fct}, \mathsf{fsk}_i, K_i))$.
  6. Compute $Z = \mathsf{Com}(K_1, ..., K_t, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$.
  7. Output $(Z, \mathsf{CT}_1, \ldots, \mathsf{CT}_t)$.

- $\underline{\mathsf{KeyGen}(\mathsf{MSK}, \mathsf{C}):}$
  1. Parse $\mathsf{MSK} = (sk_1, ..., sk_t)$.
  2. Let F be the circuit described in Figure 3. For $i \in [t]$, compute $sk_{C,i} \leftarrow \mathsf{sFE.KeyGen}(sk_i, F)$.
  3. Output $sk_C = (sk_{C,1}, ..., sk_{C,t})$.

- $\underline{\mathsf{Dec}(\mathsf{sk_C}, \mathsf{CT}):}$
  1. Parse $sk_C = (sk_{C,1}, ..., sk_{C,t})$ and $\mathsf{CT} = (Z, \mathsf{CT}_1, ..., \mathsf{CT}_t)$.
  2. For $i \in [t]$, compute $p_i \leftarrow \mathsf{sFE.Dec}(sk_{C,i}, \mathsf{CT}_i)$.
  3. Output $\mathsf{TFHE.FinDec}(p_1, ..., p_t)$.

F

Input: TFHE ciphertext fct, Partial Decryption Key fsk_i and a PRF key K_i

- Compute $\mathsf{fct}_C \leftarrow \mathsf{TFHE.Eval}(C, \mathsf{fct})$.
- Compute $r \leftarrow \mathsf{PRF}(K, \mathsf{fct}_C)$.
- Output $\mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{fct}_C; r)$

Figure 3: Description of the Circuit F.

Remark 9. *We note that the commitment in the scheme described above is only useful in the security proof.*

We prove the correctness, efficiency and security properties of the above described sub-linear secret key FE scheme. Once we prove that, we will have the following theorem.

Theorem 15. *Let* $(\mathsf{size}_{\mathsf{FE}}, \mathsf{adv}_{\mathsf{FE}})$ *be some size and advantage parameters. Let* $\mathsf{size}_{\mathsf{FE}} > poly(\lambda)$ *for some fixed polynomial. Then, assuming:*

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

There exists a $(\mathsf{size}_{\mathsf{FE}}, 2^{-\lambda^{c_1}} + \lambda \cdot p(\lambda) \cdot \mathsf{adv}_{\mathsf{FE}})-$*secure* $p(\lambda)^{c_2} \cdot \lambda^{c_2} s^{1-\epsilon} poly(\lambda, n)-$*efficient secret key FE scheme for* $\mathcal{C}_{n,s}$. *Here* $c_1, c_2 > 0$ *and* $1 > \epsilon > 0$ *are constants.*

Remark 10. *Although the theorem above is a compiler from a semi-functional FE scheme to a sublinear FE scheme (refer Section A for the definition), the theorem can be seen as security amplification for any sublinear FE scheme. This is because, assuming one-way functions, one can construct a sublinear semi-functional FE from a sublinear FE scheme (as shown in Section A) and then use the theorem above.*

The following corollary is an immediate consequence of the theorem above:

Corollary 1. *Assuming there exists a constant* $c > 0$ *and there exists:*

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

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

The following corollary is an immediate consequence of the theorem above:

Corollary 2. *Assuming there exists a constant* $c > 0$ *and there exists:*

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

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

We now prove the correctness, efficiency and security properties of the scheme.

Correctness: Correctness of this scheme follows from the correctness of underlying sFE scheme and TFHE scheme.

$t^c \cdot s^{1-\epsilon} poly(\lambda, n)$**-Efficiency.** We now prove that the scheme above satisfies $t^c \cdot s^{1-\epsilon} poly(\lambda, n)$-efficiency if underlying scheme is sublinear for the class $\mathcal{C}_{n,s}$. Here $c > 0$ is some constant. We now bound the size of the FE encryption circuit for class $\mathcal{C}_{n,s}$. Note that for the encryption algorithm,

1. Step 1 takes time $O(t \cdot poly(\lambda))$.
2. Step 2 takes time $O(poly(t, \lambda))$.
3. Step 3 takes time $poly(\lambda, n, t)$ due to compactness of TFHE. Here n is the length of the message.
4. Step 4 takes time $O(poly(t, \lambda))$.
5. Step 5 takes time $O(t \cdot T')$, where T' is the time taken by sFE.*enc*. Note that T' is sublinear in $s' = s \cdot poly(\lambda, t)$, due to sublinear efficiency of sFE. Hence it is also sublinear in s.
6. Steps 6 and 7 take time $O(poly(\lambda, t))$.

Therefore, the total time required by Enc is $O(poly(\lambda, t, n) \cdot T')$.

Thus if t is independent of the circuit size s, then the resulting scheme is also sublinear.

Security. We first present an overview of the proof of security.

Proof Overview. Before we go into the details of our proof, we will begin a quick overview. Recall that Yao's XOR lemma states that the process of XOR-ing of the outputs of many instantiations of a function f amplifies the hardness of f. The proof of this lemma can be based on Impagliazzo's hard core lemma [Imp95] which identifies a hard core measure such that it suffices to sample the inputs of f according to this hard core measure to ensure the hardness of f. Among the many instantiations of f (in the XOR lemma), there must exist at least one instantiation where the inputs to this instantiation are sampled from the hard core set and this instantiation renders the hardness of the resulting function.

A naive (and unsuccessful) attempt to prove our result would be to mimic the proof of Yao's XOR lemma. In the first hybrid, the challenge message m_b^*, for a random bit b, is encrypted, where (m_0, m_1) is the challenge message submitted by the adversary. First, we invoke the security of indistinguishability of semi-functional keys property to switch the functional keys to semi-functional keys; the hardwired values correspond to the outputs of the functions on m_b^*. We can't yet invoke the indistinguishability of semi-functional ciphertexts since the advantage is only inverse polynomial. However, similar to the argument in Yao's XOR lemma, with overwhelming probability, there must exist some instantiation of semi-functional FE such that the randomness for this instantiation is sampled according to the hardcore measure. The hope would be to remove the threshold FHE key associated with this instantiation and replace the ciphertext to be an encryption of 0 – however, this argument is flawed; we have no bound on the computational complexity of sampling from the hardcore measure. Sampling from the measure could potentially involve breaking the security of the FHE ciphertext and then sampling randomness based on the information contained inside the FHE ciphertext.

How can we fix this issue? Suppose, instead, that this measure could be sampled by a computationally bounded sampler, with runtime[4] at most 2^λ. Then by suitably increasing the security of the threshold FHE scheme, we can ensure that the computationally bounded sampler cannot break the threshold FHE scheme which will allow us to complete the security proof of the threshold FHE.

All that is left is to show how to replace the hard-core measure with a distribution that can be efficiently sampled. The first step is to emulate sampling from a hard-core measure by instead uniformly sampling from a hard-core set Set, using a theorem using Holenstein [Hol06]. Then, we can invoke a result from [CCL18] to argue that we can simulate uniformly sampling from a set using a computationally bounded sampler. However, the issue is that the running time of the sampler guaranteed by [CCL18] is exponential in $poly(\lambda)$, for some polynomial $poly$, where $poly(\lambda)$ can be much larger than the size of the FHE ciphertexts and thus, this sampler can indeed break the security of threshold FHE. Indeed, looking closely into [CCL18], we realize that the length of the strings in Set depend on the total randomness used to create all the semi-functional ciphertexts and semi-functional keys.

To get around this issue, we crucially use the fact that the *density* of the hardcore measure must be relatively high. More specifically, instead of sampling from the set Set, we first efficiently sample a random set SetR and inefficiently sample (in particular, this sampling algorithm breaks the underlying commitment scheme) an index from the set SetR. We note that the size of the index can be upper bounded by λ and thus, using [CCL18] the sampling algorithm sampling this index can be simulated using a circuit of size exponential in λ. Putting together, we have accomplished the task of simulating the hard-core measure using a circuit with runtime at most 2^λ.

We now present the hybrids and argue indistinguishability between them. While the first hybrid encrypts a challenge message m_b^*, the last one is independent of b. We will analyze the distinguishing advantage of the adversary between these hybrids.

[4]This is not accurate; we will only be able to bound the sampler by $2^{\lambda p(\lambda)}$, where p is as stated in the theorem. This is still sufficient for us since we can set the TFHE parameters in such a way that the security of TFHE is guaranteed even against adversaries running in time $2^{\lambda p(\lambda)}$

Hybrid$_0$: In this hybrid, we have the following:

1. Adversary gets as input the security parameter 1^λ and outputs a circuit $C \in \mathcal{C}_{n,s}$. He also gives out some messages $m_0, ..., m_\Gamma, m_0^*, m_1^* \in \{0,1\}^n$ such that $C(m_0^*) = C(m_1^*)$.

2. Sample a bit $b \in \{0, 1\}$.

3. To encrypt challenge ciphertext, compute an intermediate message $x^* = (x_1^*, ..., x_t^*)$ as follows.

   - Run $\mathsf{TFHE.Setup}(1^\lambda, 1^t) \rightarrow (\mathsf{fpk}, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$.
   - Compute $\mathsf{fct} \leftarrow \mathsf{TFHE.Enc}(\mathsf{fpk}, m_b^*)$.
   - Sample t PRF keys $K_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $i \in [t]$.
   - Set $x_i^* = (\mathsf{fct}, \mathsf{fsk}_i, K_i)$.

4. Similarly, compute x^j as intermediate messages corresponding to m_j for $j \in \Gamma$, as described above. Let x^j be denoted as $(x_1^j, \ldots x_t^j)$.

5. Compute $Z^* = \mathsf{Com}(K_1, ..., K_t, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$. For other ciphertext queries $j \in [\Gamma]$, compute Z^j similarly (using respective PRF and partial decryption keys).

6. Then run the setup of the FE as follows: compute $\mathsf{sFE.Setup}(1^\lambda) \rightarrow sk_i$ for $i \in [t]$ and set $\mathsf{MSK} = (sk_1, ..., sk_t)$.

7. Generate $sk_C = (sk_{C,1}, .., sk_{C,t})$ where $sk_{C,i} \leftarrow \mathsf{sFE.KeyGen}(sk_i, F)$ for the circuit F described in the key generation algorithm.

8. For $j \in [\Gamma]$, compute $\mathsf{CT}^j = (Z^j, \mathsf{CT}_1^j, .., \mathsf{CT}_t^j)$. Here, $\mathsf{CT}_i^j \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^j)$ for $i \in [t]$.

9. Compute $\mathsf{CT}^* = (Z^*, \mathsf{CT}_1^*, .., \mathsf{CT}_t^*)$. Here, $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^*)$ for $i \in [t]$.

10. Give the following to adversary $(sk_C, \{\mathsf{CT}_i^j\}_{j \in [\Gamma], i \in [t]}, \mathsf{CT}^*)$

11. Adversary guesses $b' \in \{0, 1\}$

Hybrid$_1$: This hybrid is the same as the previous one except that the function keys are generated using semi-functional key generation algorithm. We describe the changes from the previous hybrid using bold faced word **change**.

1. Adversary gets as input the security parameter 1^λ and outputs a circuit $C \in \mathcal{C}_{n,s}$. He also gives out some messages $m_0, ..., m_\Gamma, m_0^*, m_1^* \in \{0,1\}^n$ such that $C(m_0^*) = C(m_1^*)$.

2. Sample a bit $b \in \{0,1\}$.

3. To encrypt challenge ciphertext, compute an intermediate message $x^* = (x_1^*, ..., x_t^*)$ as follows.

   - Run $\mathsf{TFHE.Setup}(1^\lambda, 1^t) \rightarrow (\mathsf{fpk}, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$.
   - Compute $\mathsf{fct} \leftarrow \mathsf{TFHE.Enc}(\mathsf{fpk}, m_b^*)$.
   - Sample t PRF keys $K_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $i \in [t]$.
   - Set $x_i^* = (\mathsf{fct}, \mathsf{fsk}_i, K_i)$.

4. [**Change**] Let F be the circuit described in the key generation algorithm. Set $\theta_i = F(x_i^*) = \mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{Eval}(C, \mathsf{fct}); \mathsf{PRF}(K_i, \mathsf{Eval}(C, \mathsf{fct})))$ for $i \in [t]$.

5. Similarly, compute x^j as intermediate messages corresponding to m_j for $j \in \Gamma$, as described above. Let x^j be denoted as $(x_1^j, \ldots x_t^j)$.

6. Compute $Z^* = \mathsf{Com}(K_1, ..., K_t, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$. For other ciphertext queries $j \in [\Gamma]$, compute Z^j similarly (using respective PRF and partial decryption keys).

7. Then run the setup of the FE as follows: compute $\mathsf{sFE.Setup}(1^\lambda) \rightarrow sk_i$ for $i \in [t]$ and set $\mathsf{MSK} = (sk_1, ..., sk_t)$.

8. Generate $sk_C = (sk_{C,1}, .., sk_{C,t})$ where $sk_{C,i} \leftarrow \mathsf{sFE.sfKG}(sk_i, F, \theta_i)$ for the circuit F described in the key generation algorithm.

9. For $j \in [\Gamma]$, compute $\mathsf{CT}^j = (Z^j, \mathsf{CT}_1^j, .., \mathsf{CT}_t^j)$. Here, $\mathsf{CT}_i^j \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^j)$ for $i \in [t]$.

10. Compute $\mathsf{CT}^* = (Z^*, \mathsf{CT}_1^*, .., \mathsf{CT}_t^*)$. Here, $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^*)$ for $i \in [t]$.

11. Give the following to adversary $(sk_C, \{\mathsf{CT}_i^j\}_{j \in [\Gamma], i \in [t]}, \mathsf{CT}^*)$

12. Adversary guesses $b' \in \{0,1\}$

Lemma 10. *If* sFE *scheme satisfies indistinguishability of semi-functional key property then for any adversary* $\mathcal{A}$ *of size* $O(\mathsf{size})$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_0) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_1) = 1]| < 2^{-\lambda^{c_0}}$ *for some constant* $c_0 > 0$ [5].

Proof. (Sketch) In both the above hybrids, the ciphertexts are generated using honest encryption algorithm. The only way the hybrids differ is the way functional keys are generated. In **Hybrid**$_0$ they are functional while in **Hybrid**$_1$ they are semi-functional. We can invoke a series of t hybrids, where one by one in each system the key is generated semi-functionally instead of functionally. The claim thus follows from indistinguishability of the semi-functional keys property of sFE scheme. □

[5]Note that the definition in Section 9 requires the adversary to have advantage $\mathsf{negl}(\lambda)$ in the indistinguishability of semi-function key security game. However, for our application to iO, it suffices to replace negl with a subexponentially small function. For our amplification theorem, this just means that the additive negligible security loss is subexponentially small as opposed to some negligible function.

Before, we describe the next hybrid, we recall the following theorem about the scheme sFE proved in Section 9:

Theorem 16. *Fix $1^\lambda, 1^n, \Gamma, \{M_i\}, M^*, C$ as above. Define two functions E_b for $b \in \{0,1\}$, that takes as input $\{0,1\}^{\ell_b}$. Here ℓ_b is the length of randomness required to compute the following. The functions do the following.*

Consider the following process:

1. *Compute* $\mathsf{MSK} \leftarrow \mathsf{sFE.Setup}(1^\lambda)$.
2. *Compute* $\mathsf{CT}_i \leftarrow \mathsf{sFE.Enc}(\mathsf{MSK}, M_i)$ *for* $i \in [\Gamma]$.
3. *Set* $\theta = C(M^*)$. *Compute* $sk_C \leftarrow \mathsf{sFE.sfKG}(\mathsf{MSK}, C, \theta)$.
4. *If* $b = 0$, *compute* $\mathsf{CT}^* = \mathsf{sFE.Enc}(\mathsf{MSK}, M^*)$ *and if* $b = 1$, *compute* $\mathsf{CT}^* = \mathsf{sFE.sfEnc}(\mathsf{MSK}, 1^\lambda)$.
5. *For* $b \in \{0,1\}$, E_b *on input* $r \in \{0,1\}^{\ell_b}$ *outputs* $\{\mathsf{CT}_i\}_{i\in\Gamma}, sk_C, \mathsf{CT}^*$.

If sFE *satisfies* $(\mathsf{size_{sFE}}, \mathsf{adv_{sFE}})-$*indistinguishability of semi-functional ciphertexts property, then, there exists two computable (not necessarily efficient) measures $\mathcal{M}_0$ and $\mathcal{M}_1$ ($\mathcal{M}_b$ defined over $\{0,1\}^{\ell_b}$ for $b \in \{0,1\}$) of density exactly $1 - \mathsf{adv_{sFE}}/2$ such that, for all circuits $\mathcal{A}$ of size* $\mathsf{size}'_{\mathsf{sFE}} > \mathsf{size_{sFE}} \cdot \mathsf{adv}'^2_{\mathsf{sFE}}/128(\ell_0 + \ell_1 + 1)$,

$$|\Pr_{u\leftarrow\mathcal{D}_{\mathcal{M}_0}}[\mathcal{A}(E_0(u)) = 1] - \Pr_{v\leftarrow\mathcal{D}_{\mathcal{M}_1}}[\mathcal{A}(E(v)) = 1]| < \mathsf{adv}'_{\mathsf{sFE}}$$

Here both measures may depend on $(\{M_i\}_{i\in\Gamma}, C, M^*)$

Hybrid$_2$: Let adv' be a parameter denoting a distinguishing advantage and size' be a size parameter. This hybrid is inefficient. Set $\mathsf{size}' = \mathsf{size} \cdot \mathsf{adv}'^2/(128(\ell_0 + \ell_1 + 1))$, where ℓ_0, ℓ_1 be the lengths described in the theorem above. Let $\mathcal{M}_{0,i}$ denote the (scaled) measure of density $1/2p(\lambda)$ corresponding to encryption algorithm as described by the theorem above and let $\mathcal{M}_{1,i}$ denote corresponding measure for semi-functional encryption algorithm. For any measure $\mathcal{M}$, let $\overline{\mathcal{M}}$ denote the measure $1 - \mathcal{M}$. Now we describe the hybrid in more detail.

1. Adversary gets as input the security parameter 1^λ and outputs a circuit $C \in \mathcal{C}_{n,s}$. He also gives out some messages $m_0, ..., m_\Gamma, m_0^*, m_1^* \in \{0,1\}^n$ such that $C(m_0^*) = C(m_1^*)$.
2. Sample a bit $b \in \{0,1\}$.
3. [**Change**] Sample a string $y \in \{0,1\}^t$ such that for every $i \in [t]$, set $y_i = 1$ with probability $1/2p(\lambda)$ and $y_i = 0$ with probability $(1 - 1/2p(\lambda))$. Here, each bit y_i is chosen independently.
4. To encrypt challenge ciphertext, compute an intermediate message $x^* = (x_1^*, ..., x_t^*)$ as follows.
   - Run $\mathsf{TFHE.Setup}(1^\lambda, 1^t) \to (\mathsf{fpk}, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$.
   - Compute $\mathsf{fct} \leftarrow \mathsf{TFHE.Enc}(\mathsf{fpk}, m_b^*)$.
   - Sample t PRF keys $K_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $i \in [t]$.
   - Set $x_i^* = (\mathsf{fct}, \mathsf{fsk}_i, K_i)$.

5. Let F be the circuit described in the key generation algorithm. Set $\theta_i = F(x_i^*) = \mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{Eval}(C, \mathsf{fct}); \mathsf{PRF}(K_i, \mathsf{Eval}(C, \mathsf{fct})))$ for $i \in [t]$.

6. Similarly, compute x^j as intermediate messages corresponding to m_j for $j \in \Gamma$, as described above. Let x^j be denoted as $(x_1^j, \dots x_t^j)$.

7. Compute $Z^* = \mathsf{Com}(K_1, ..., K_t, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$. For other ciphertext queries $j \in [\Gamma]$, compute Z^j similarly (using respective PRF and partial decryption keys).

8. [**Change**] For every $i \in [t]$, to compute the following steps, we generate randomness $R_i = (r_{1,i}, r_{2,i}, \{r_{3,j,i}\}_{j \in [\Gamma]}, r_{4,i})$ as follows. If $y_i = 1$ sample $R_i \leftarrow \mathcal{D}_{\mathcal{M}_{0,i}}$, otherwise, sample $R_i \leftarrow \mathcal{D}_{\overline{\mathcal{M}_{0,i}}}$. We note here $\mathcal{M}_{0,i}$ and $\mathcal{M}_{1,i}$ may depend on $(C, x_i^*, \{x_i^j\}_{j \in [\Gamma]})$

9. [**Change**] Then run the setup of the FE as follows: compute $\mathsf{sFE.Setup}(1^\lambda; r_{1,i}) \rightarrow sk_i$ for $i \in [t]$ and set $\mathsf{MSK} = (sk_1, ..., sk_t)$.

10. Generate $sk_C = (sk_{C,1}, .., sk_{C,t})$ where $sk_{C,i} \leftarrow \mathsf{sFE.sfKG}(sk_i, F, \theta_i; r_{2,i})$ for the circuit F described in the key generation algorithm.

11. For $j \in [\Gamma]$, compute $\mathsf{CT}^j = (Z^j, \mathsf{CT}_1^j, .., \mathsf{CT}_t^j)$. Here, $\mathsf{CT}_i^j \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^j; r_{3,j,i})$ for $i \in [t]$.

12. Compute $\mathsf{CT}^* = (Z^*, \mathsf{CT}_1^*, .., \mathsf{CT}_t^*)$. Here, $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^*; r_{4,i})$ for $i \in [t]$.

13. Give the following to adversary $(sk_C, \{\mathsf{CT}_i^j\}_{j \in [\Gamma], i \in [t]}, \mathsf{CT}^*)$

14. Adversary guesses $b' \in \{0, 1\}$

Lemma 11. *For any adversary $\mathcal{A}$,* $|\Pr[\mathcal{A}(\mathbf{Hybrid}_1) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_2) = 1]| = 0$.

Proof. (Sketch) These hybrids are identical. Note that measure generated by $\mathcal{M}_{0,i}$ for every $i \in [t]$, have density exactly $1/2p(\lambda)$. With probability $1/2p(\lambda)$, uniform randomness to generate encryption can be thought of as if it was sampled from $\mathcal{M}_{0,i}$ and with $1 - 1/2p(\lambda)$ from its complement. □

Hybrid$_3$: This hybrid is the same as the previous the hybrid except that challenger aborts if $y = 0^t$. The exact change is marked in red color.

1. Adversary gets as input the security parameter 1^λ and outputs a circuit $C \in \mathcal{C}_{n,s}$. He also gives out some messages $m_0, ..., m_\Gamma, m_0^*, m_1^* \in \{0,1\}^n$ such that $C(m_0^*) = C(m_1^*)$.

2. Sample a bit $b \in \{0, 1\}$.

3. [**Change**] Sample a string $y \in \{0,1\}^t$ such that for every $i \in [t]$, set $y_i = 1$ with probability $1/2p(\lambda)$ and $y_i = 0$ with probability $(1 - 1/2p(\lambda))$. Here, each bit y_i is chosen independently. If $y = 0^t$, abort.

4. To encrypt challenge ciphertext, compute an intermediate message $x^* = (x_1^*, ..., x_t^*)$ as follows.
   - Run $\mathsf{TFHE.Setup}(1^\lambda, 1^t) \to (\mathsf{fpk}, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$.
   - Compute $\mathsf{fct} \leftarrow \mathsf{TFHE.Enc}(\mathsf{fpk}, m_b^*)$.
   - Sample t PRF keys $K_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $i \in [t]$.
   - Set $x_i^* = (\mathsf{fct}, \mathsf{fsk}_i, K_i)$.

5. Let F be the circuit described in the key generation algorithm. Set $\theta_i = F(x_i^*) = \mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{Eval}(C, \mathsf{fct}); \mathsf{PRF}(K_i, \mathsf{Eval}(C, \mathsf{fct})))$ for $i \in [t]$.

6. Similarly, compute x^j as intermediate messages corresponding to m_j for $j \in \Gamma$, as described above. Let x^j be denoted as $(x_1^j, \ldots x_t^j)$.

7. Compute $Z^* = \mathsf{Com}(K_1, ..., K_t, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$. For other ciphertext queries $j \in [\Gamma]$, compute Z^j similarly (using respective PRF and partial decryption keys).

8. For every $i \in [t]$, to compute the following steps, we generate randomness $R_i = (r_{1,i}, r_{2,i}, \{r_{3,j,i}\}_{j \in [\Gamma]}, r_{4,i})$ as follows. If $y_i = 1$ sample $R_i \leftarrow \mathcal{D}_{\mathcal{M}_{0,i}}$, otherwise, sample $R_i \leftarrow \mathcal{D}_{\overline{\mathcal{M}_{0,i}}}$. We note here $\mathcal{M}_{0,i}$ and $\mathcal{M}_{1,i}$ may depend on $(C, x_i^*, \{x_i^j\}_{j \in [\Gamma]})$

9. Then run the setup of the FE as follows: compute $\mathsf{sFE.Setup}(1^\lambda; r_{1,i}) \to sk_i$ for $i \in [t]$ and set $\mathsf{MSK} = (sk_1, ..., sk_t)$.

10. Generate $sk_C = (sk_{C,1}, .., sk_{C,t})$ where $sk_{C,i} \leftarrow \mathsf{sFE.sfKG}(sk_i, F, \theta_i; r_{2,i})$ for the circuit F described in the key generation algorithm.

11. For $j \in [\Gamma]$, compute $\mathsf{CT}^j = (Z^j, \mathsf{CT}_1^j, .., \mathsf{CT}_t^j)$. Here, $\mathsf{CT}_i^j \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^j; r_{3,j,i})$ for $i \in [t]$.

12. Compute $\mathsf{CT}^* = (Z^*, \mathsf{CT}_1^*, .., \mathsf{CT}_t^*)$. Here, $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^*; r_{4,i})$ for $i \in [t]$.

13. Give the following to adversary $(sk_C, \{\mathsf{CT}_i^j\}_{j \in [\Gamma], i \in [t]}, \mathsf{CT}^*)$

14. Adversary guesses $b' \in \{0, 1\}$

Lemma 12. *For any adversary $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_2) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_3) = 1]| < 2^{-c_2 \cdot \lambda}$ for some constant c_2.*

Proof. (Sketch) These hybrids are statistically close. The probability that the string $y = 0^t$ is exactly $(1 - 1/2p(\lambda))^t$. Substituting $t = p(\lambda)\lambda$, the claim follows. ◻

Hybrid$_4$: This hybrid is inefficient. In this hybrid we use the security of sFE and switch to encrypting challenge ciphertexts semi-functionally whenever $y_i = 1$. The exact change is marked in red. This hybrid is now described as follows:

1. Adversary gets as input the security parameter 1^λ and outputs a circuit $C \in \mathcal{C}_{n,s}$. He also gives out some messages $m_0, ..., m_\Gamma, m_0^*, m_1^* \in \{0,1\}^n$ such that $C(m_0^*) = C(m_1^*)$.
2. Sample a bit $b \in \{0,1\}$.
3. Sample a string $y \in \{0,1\}^t$ such that for every $i \in [t]$, set $y_i = 1$ with probability $1/2p(\lambda)$ and $y_i = 0$ with probability $(1 - 1/2p(\lambda))$. Here, each bit y_i is chosen independently. If $y = 0^t$, abort.
4. To encrypt challenge ciphertext, compute an intermediate message $x^* = (x_1^*, ..., x_t^*)$ as follows.
   - Run $\mathsf{TFHE.Setup}(1^\lambda, 1^t) \to (\mathsf{fpk}, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$.
   - Compute $\mathsf{fct} \leftarrow \mathsf{TFHE.Enc}(\mathsf{fpk}, m_b^*)$.
   - Sample t PRF keys $K_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $i \in [t]$.
   - Set $x_i^* = (\mathsf{fct}, \mathsf{fsk}_i, K_i)$.
5. Let F be the circuit described in the key generation algorithm. Set $\theta_i = F(x_i^*) = \mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{Eval}(C, \mathsf{fct}); \mathsf{PRF}(K_i, \mathsf{Eval}(C, \mathsf{fct})))$ for $i \in [t]$.
6. Similarly, compute x^j as intermediate messages corresponding to m_j for $j \in \Gamma$, as described above. Let x^j be denoted as $(x_1^j, \ldots x_t^j)$.
7. Compute $Z^* = \mathsf{Com}(K_1, ..., K_t, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$. For other ciphertext queries $j \in [\Gamma]$, compute Z^j similarly (using respective PRF and partial decryption keys).
8. [**Change**] For every $i \in [t]$, to compute the following steps, we generate randomness $R_i = (r_{1,i}, r_{2,i}, \{r_{3,j,i}\}_{j\in[\Gamma]}, r_{4,i})$ as follows. If $y_i = 1$ sample $R_i \leftarrow \mathcal{D}_{\mathcal{M}_{1,i}}$, otherwise, sample $R_i \leftarrow \mathcal{D}_{\overline{\mathcal{M}_{0,i}}}$. We note here $\mathcal{M}_{0,i}$ and $\mathcal{M}_{1,i}$ may depend on $(C, x_i^*, \{x_i^j\}_{j\in[\Gamma]})$
9. Then run the setup of the FE as follows: compute $\mathsf{sFE.Setup}(1^\lambda; r_{1,i}) \to sk_i$ for $i \in [t]$ and set $\mathsf{MSK} = (sk_1, ..., sk_t)$.
10. Generate $sk_C = (sk_{C,1}, .., sk_{C,t})$ where $sk_{C,i} \leftarrow \mathsf{sFE.sfKG}(sk_i, F, \theta_i; r_{2,i})$ for the circuit F described in the key generation algorithm.
11. For $j \in [\Gamma]$, compute $\mathsf{CT}^j = (Z^j, \mathsf{CT}_1^j, .., \mathsf{CT}_t^j)$. Here, $\mathsf{CT}_i^j \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^j; r_{3,j,i})$ for $i \in [t]$.
12. [**Change**] Compute $\mathsf{CT}^* = (Z^*, \mathsf{CT}_1^*, .., \mathsf{CT}_t^*)$. Here, for every $i \in [t]$, if $y_i = 0$, $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^*; r_{4,i})$ otherwise $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.sfEnc}(sk_i, 1^\lambda, 1^\lambda; r_{4,i})$.
13. Give the following to adversary $(sk_C, \{\mathsf{CT}_i^j\}_{j\in[\Gamma], i\in[t]}, \mathsf{CT}^*)$
14. Adversary guesses $b' \in \{0,1\}$

Lemma 13. *If* sFE *satisfies indistinguishability of semi-functional ciphertexts property, then for any adversary* $\mathcal{A}$ *of size* size′, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_3) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_4) = 1]| < t \cdot \mathsf{adv}'$.

Proof. (Sketch) This is a direct application of Theorem 16. Because these hybrids are inefficient, we need to perform non-uniform fixing. This proof goes by fixing the "best possible" string $y \in \{0,1\}^t$ which is sampled according to the distribution specified in the hybrids. The claim is that if $|\Pr[\mathcal{A}(\mathbf{Hybrid}_3) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_4) = 1]| > \epsilon$, then there must exists (a non zero) y such that $|\Pr[\mathcal{A}(\mathbf{Hybrid}_{3,y}) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_{4,y}) = 1]| > \epsilon$. Where $\mathbf{Hybrid}_{3,y}$ or $\mathbf{Hybrid}_{4,y}$ represents the corresponding hybrid where string y is fixed.

This is because $\Sigma_{y\in\{0,1\}^t} \Pr[y]|\Pr[\mathcal{A}(\mathbf{Hybrid}_{3,y}) = 1]-\Pr[\mathcal{A}(\mathbf{Hybrid}_{4,y}) = 1]| > |\Pr[\mathcal{A}(\mathbf{Hybrid}_3) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_4) = 1]| > \epsilon$. Since for every string y, $0 < \Pr[y] < 1$ (refer previous hybrid for the calculation), $\Sigma_y \Pr[y] = 1$, and $|\Pr[\mathcal{A}(\mathbf{Hybrid}_{3,0^t}) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_{4,0^t}) = 1]| = 0$ (as the experiment is aborted) the claim follows by pigeon hole principle.

Fix any such y. We can construct w indistinguishable hybrids, where w is the weight of the string y. For each such index i, with $y_i = 1$, we define an intermediate $\mathbf{Hybrid}_{3,y,i}$, where the encryptions for index $j \neq i$ are generated as in the previous hybrid but encryption of $j = i$ is generated differently as follows. Instead of being computed using $\mathsf{sFE.Enc}$ algorithm using randomness from $\mathcal{M}_{0,i}$, it is encrypted using $\mathsf{sFE.sfEnc}$ algorithm using the randomness generated from $\mathcal{M}_{1,i}$. Note that for last such index i, such that $y_i = 1$, $\mathbf{Hybrid}_{3,y,i}$ is the same as $\mathbf{Hybrid}_{4,y}$.

Once this fixing is done, each intermediate hybrid is indistinguishable due to the security of sFE. Note that to reduce to the security of sFE, reduction has to non-uniformly fix the randomness generated for other indices $j \neq i$. Informally, we use this to advice to generate encryptions for indices $j \neq i$ in $[t]$. For index i, we get ciphertexts and the keys from the challenger. They are either functionally encrypted using the randomness sampled from $\mathcal{M}_{0,i}$ or they are semi-functionally encrypted using the randomness sampled from $\mathcal{M}_{1,i}$. Since the encryptions for indices $j \neq i$ are generated using non-uniformly fixed randomness and encryption for index i comes from the challenger, the rest of the hybrid can be generated in polynomial time. Now the reduction can use the adversary's response to break the security of sFE.

□

We now restate theorem 17. We will use this theorem in this hybrid.

Theorem 17 (Imported Theorem [Hol06].)**.** *Let $\mathcal{M}$ be any measure on $\{0,1\}^n$ of density $\mu(\mathcal{M}) \geq 1-\rho(n)$. Let $\gamma(n) \in (0, 1/2)$ be any function. Then, for a random set* Set *chosen according to the measure $\mathcal{M}$ the following two holds with probability at least $1 - 2(2^{-2^n \gamma^2 (1-\rho)^4/64})$:*

- $(1 - \frac{\gamma(1-\rho)}{4})(1-\rho)2^n \leq |\mathsf{Set}| \leq (1 + \frac{\gamma(1-\rho)}{4})(1-\rho)2^n$
- *For such a random set* Set*, for any distinguisher $\mathcal{A}$ with size $|\mathcal{A}| \leq 2^n(\frac{\gamma^2(1-\rho)^4}{64n})$ satisfying*

$$|\Pr_{x \leftarrow \mathsf{Set}}[\mathcal{A}(x) = 1] - \Pr_{x \leftarrow \mathcal{D}_{\mathcal{M}}}[\mathcal{A}(x) = 1]| \leq \gamma$$

Hybrid$_5$: This hybrid is the same as the previous hybrid except that for every $i \in [t]$, instead of sampling from a measure $\mathcal{M}_i$ (either $\mathcal{M}_{1,i}$ or $\overline{\mathcal{M}}_{0,i}$), we sample a set Set_i from the corresponding measure, and then sample uniformly from Set_i. These sets are constructed according to theorem 17. Lets analyse it case by case. For the analysis, set the bound on distinguishing advantage γ to be $2^{-\lambda}$.

- If $y_i = 0$, measure $\mathcal{M}_i = \overline{\mathcal{M}}_{0,i}$ has density exactly $1 - 1/2p(\lambda)$. From Theorem 17 with probability at least $1 - 2(2^{-2^{\ell_0 - 2\lambda}/4p(\lambda^4)})$, density of Set_i is atleast $1/3p(\lambda)$ and the distinguishing advantage is bounded by $2^{-\lambda}$ for adversaries of size $2^{\ell_0 - 2\lambda}/(\ell_0 poly(\lambda))$.
- If $y_i = 1$, measure $\mathcal{M}_i = \mathcal{M}_{1,i}$ has density exactly $1/\lambda$. From Theorem 17 we observe following. With probability at least, $1 - 2(2^{-2^{\ell_1 - 2\lambda}(1-1/2p(\lambda))^4/64})$, density of Set_i is atleast $1/3p(\lambda)$ and the distinguishing advantage is bounded by $2^{-\lambda}$ for adversaries of size $2^{\ell_1 - 2\lambda}/(\ell_1 poly(\lambda))$.

Now we describe the hybrid in detail.

1. Adversary gets as input the security parameter 1^λ and outputs a circuit $C \in \mathcal{C}_{n,s}$. He also gives out some messages $m_0, ..., m_\Gamma, m_0^*, m_1^* \in \{0,1\}^n$ such that $C(m_0^*) = C(m_1^*)$.
2. Sample a bit $b \in \{0,1\}$.
3. Sample a string $y \in \{0,1\}^t$ such that for every $i \in [t]$, set $y_i = 1$ with probability $1/2p(\lambda)$ and $y_i = 0$ with probability $(1 - 1/2p(\lambda))$. Here, each bit y_i is chosen independently. If $y = 0^t$, abort.
4. To encrypt challenge ciphertext, compute an intermediate message $x^* = (x_1^*, ..., x_t^*)$ as follows.
   - Run $\mathsf{TFHE.Setup}(1^\lambda, 1^t) \rightarrow (\mathsf{fpk}, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$.
   - Compute $\mathsf{fct} \leftarrow \mathsf{TFHE.Enc}(\mathsf{fpk}, m_b^*)$.
   - Sample t PRF keys $K_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $i \in [t]$.
   - Set $x_i^* = (\mathsf{fct}, \mathsf{fsk}_i, K_i)$.
5. Let F be the circuit described in the key generation algorithm. Set $\theta_i = F(x_i^*) =$ $\mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{Eval}(C, \mathsf{fct}); \mathsf{PRF}(K_i, \mathsf{Eval}(C, \mathsf{fct})))$ for $i \in [t]$.
6. Similarly, compute x^j as intermediate messages corresponding to m_j for $j \in \Gamma$, as described above. Let x^j be denoted as $(x_1^j, \ldots x_t^j)$.

7. Compute $Z^* = \mathsf{Com}(K_1, ..., K_t, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$. For other ciphertext queries $j \in [\Gamma]$, compute Z^j similarly (using respective PRF and partial decryption keys).

8. [**Change**] For every $i \in [t]$, to compute the following steps, we generate randomness $R_i = (r_{1,i}, r_{2,i}, \{r_{3,j,i}\}_{j\in[\Gamma]}, r_{4,i}) \leftarrow \mathsf{Set}_i$ as follows. If $y_i = 1$ set Set_i is constructed using theorem 17 from measure $\mathcal{M}_{1,i}$. Otherwise, set Set_i is constructed using theorem 17 from measure $\overline{\mathcal{M}}_{0,i}$. We note here Set_i may depend on $(C, x_i^*, \{x_i^j\}_{j\in[\Gamma]})$.

9. Then run the setup of the FE as follows: compute $\mathsf{sFE.Setup}(1^\lambda; r_{1,i}) \to sk_i$ for $i \in [t]$ and set $\mathsf{MSK} = (sk_1, ..., sk_t)$.

10. Generate $sk_C = (sk_{C,1}, .., sk_{C,t})$ where $sk_{C,i} \leftarrow \mathsf{sFE.sfKG}(sk_i, F, \theta_i; r_{2,i})$ for the circuit F described in the key generation algorithm.

11. For $j \in [\Gamma]$, compute $\mathsf{CT}^j = (Z^j, \mathsf{CT}_1^j, .., \mathsf{CT}_t^j)$. Here, $\mathsf{CT}_i^j \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^j; r_{3,j,i})$ for $i \in [t]$.

12. Compute $\mathsf{CT}^* = (Z^*, \mathsf{CT}_1^*, .., \mathsf{CT}_t^*)$. Here, for every $i \in [t]$, if $y_i = 0$, $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^*; r_{4,i})$ otherwise $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.sfEnc}(sk_i, 1^\lambda, 1^\lambda; r_{4,i})$.

13. Give the following to adversary $(sk_C, \{\mathsf{CT}_i^j\}_{j\in[\Gamma], i\in[t]}, \mathsf{CT}^*)$

14. Adversary guesses $b' \in \{0, 1\}$

Lemma 14. *Due to theorem 17, with probability at least* $1 - 2^{-\lambda}$ *(over construction of* Set_i*), for any adversary* $\mathcal{A}$ *of size* $O(2^{2^\ell})$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_4) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_5) = 1]| < 2^{-\lambda}$. *Here* $\ell = min\{\ell_0, \ell_1\}$ *is the minimum length of the randomness used in the hybrid to generate encryptions for any index.*

Proof. (Sketch) This is a direct application of Theorem 17. This proof goes by fixing the "best possible" string $y \in \{0, 1\}^t$ which is sampled according to the distribution specified in the hybrids. This can be proven by a series of t intermediate hybrids. We can define t intermediate hybrids, $\mathbf{Hybrid}_{4,i}$ for $i \in [t]$. Here $\mathbf{Hybrid}_{4,i}$ is similar to its previous hybrid except that for system i, randomness is sampled from Set_i instead of $\mathcal{M}_i$. Note that $\mathbf{Hybrid}_{4,t}$ is the same as $\mathbf{Hybrid}_5$. Note that if there exists an adversary $\mathcal{A}$ (of size described in the theorem) that distinguish $\mathbf{Hybrid}_{4,i}$ from $\mathbf{Hybrid}_{4,i+1}$ with advantage $2^{-\lambda}$, we can build a reduction that refutes theorem 17. In doing so, reduction fixes non-uniformly the randomness for other systems $j \in [t]$ with $j \neq i$. In particular, reduction generates keys and ciphertext for all indices $j \neq i$, as in the previous hybrid using the non-uniformly fixed randomness. For index i, the keys and ciphertext are generated using the randomness given by the challenger. It is either generated using measure $\mathcal{M}_i$ or from the set Set_i. Due to theorem 17, the security holds. □

Hybrid$_6$: This hybrid is the same as the previous hybrid except that for every $i \in [t]$, the following happens. For every $i \in [t]$, construct a new set SetR_i as a set of $t = p(\lambda)\lambda$ random samples from $\{0,1\}^{\ell_i}$ (here, let $\{0,1\}^{\ell}_i$ denote the domain of measure $\mathcal{M}_i$). For every $i \in [t]$, instead of computing the challenge encryption using randomness sampled from Set_i, compute it from randomness sampled uniformly from $\mathsf{Set}_i \cap \mathsf{SetR}_i$. Abort if the intersection is empty. In this hybrid, let Mach_i denote the (unbounded probabilistic) machine that takes as input SetR_i along with $(C, x_i^*, \{x_i^j\}_{j\in[\Gamma]})$ to compute an index $j_i \in [t]$ of the randomness sampled from SetR_i.

We describe now the randomized algorithm Mach_i.

1. On input $L_i = (\mathsf{SetR}_i, C, x_i^*, \{x_i^j\}_{j\in[\Gamma]})$, sample the set Set_i as in the previous hybrid. If $y_i = 0$, it is sampled from measure $\mathcal{M}_i = \overline{M}_{0,i}$, otherwise from $\mathcal{M}_i = \mathcal{M}_{1,i}$.

2. Randomly sample from $\mathsf{Set}_i \cap \mathsf{SetR}_i$ and output the index of the element in j_i. Output $\perp$ if the intersection is empty.

Here is the hybrid description:

1. Adversary gets as input the security parameter 1^λ and outputs a circuit $C \in \mathcal{C}_{n,s}$. He also gives out some messages $m_0, ..., m_\Gamma, m_0^*, m_1^* \in \{0,1\}^n$ such that $C(m_0^*) = C(m_1^*)$.

2. Sample a bit $b \in \{0,1\}$.

3. Sample a string $y \in \{0,1\}^t$ such that for every $i \in [t]$, set $y_i = 1$ with probability $1/2p(\lambda)$ and $y_i = 0$ with probability $(1 - 1/2p(\lambda))$. Here, each bit y_i is chosen independently. Abort if $y = 0^t$.

4. To encrypt challenge ciphertext, compute $x^* = (x_1^*, ..., x_t^*)$ as follows.

   - Run $\mathsf{TFHE.Setup}(1^\lambda, 1^t) \to (\mathsf{fpk}, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$.
   - Compute $\mathsf{fct} \leftarrow \mathsf{TFHE.Enc}(\mathsf{fpk}, m_b^*)$.
   - Sample t PRF keys $K_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $i \in [t]$.
   - Set $x_i^* = (\mathsf{fct}, \mathsf{fsk}_i, K_i)$.

5. Let F be the circuit described in the key generation algorithm. Set $\theta_i = F(x_i^*) =$ $\mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{Eval}(C, \mathsf{fct}); \mathsf{PRF}(K_i, \mathsf{Eval}(C, \mathsf{fct})))$ for $i \in [t]$.

6. Similarly, compute x^j as intermediate messages corresponding to m_j for $j \in \Gamma$, as described above. Let x^j be denoted as $(x_1^j, \ldots x_t^j)$.

7. Compute $Z^* = \mathsf{Com}(K_1, ..., K_t, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$. For other ciphertext queries $j \in [\Gamma]$, compute Z^j similarly (using respective PRF and partial decryption keys).

8. [**Change**] For $i \in [t]$, sample SetR_i as a set of $t = p(\lambda)\lambda$ uniformly chosen inputs from domain of $\mathcal{M}_i$. Note that this measure is equal to $\mathcal{M}_{1,i}$ if $y_i = 1$ and $\overline{\mathcal{M}}_{0,i}$ otherwise.

9. [**Change**] For every $i \in [t]$, to compute the following steps, we generate randomness $R_i = (r_{1,i}, r_{2,i}, \{r_{3,j,i}\}_{j\in[\Gamma]}, r_{4,i})$ as follows. Run $\mathsf{Mach}_i(\mathsf{SetR}_i, C, x_i^*, \{x_i^j\}_{j\in[\Gamma]}) \to j_i$. Set R_i as the randomness with index j_i in the set SetR_i.

10. Then run the setup of the FE as follows: compute $\mathsf{sFE.Setup}(1^\lambda; r_{1,i}) \rightarrow sk_i$ for $i \in [t]$ and set $\mathsf{MSK} = (sk_1, ..., sk_t)$.

11. Generate $sk_C = (sk_{C,1}, .., sk_{C,t})$ where $sk_{C,i} \leftarrow \mathsf{sFE.sfKG}(sk_i, F, \theta_i; r_{2,i})$ for the circuit F described in the key generation algorithm.

12. For $j \in [\Gamma]$, compute $\mathsf{CT}^j = (Z^j, \mathsf{CT}^j_1, .., \mathsf{CT}^j_t)$. Here, $\mathsf{CT}^j_i \leftarrow \mathsf{sFE.Enc}(sk_i, x^j_i; r_{3,j,i})$ for $i \in [t]$.

13. Compute $\mathsf{CT}^* = (Z^*, \mathsf{CT}^*_1, .., \mathsf{CT}^*_t)$. Here, for every $i \in [t]$, if $y_i = 0$, $\mathsf{CT}^*_i \leftarrow \mathsf{sFE.Enc}(sk_i, x^*_i; r_{4,i})$ otherwise $\mathsf{CT}^*_i \leftarrow \mathsf{sFE.sfEnc}(sk_i, 1^\lambda, 1^\lambda; r_{4,i})$.

14. Give the following to adversary $(sk_C, \{\mathsf{CT}^j_i\}_{j \in [\Gamma], i \in [t]}, \mathsf{CT}^*)$

15. Adversary guesses $b' \in \{0, 1\}$

Lemma 15. *For any adversary $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_5) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_6) = 1]| < 2^{-c_5\lambda}$ for some constant $c_5 > 0$. This indistinguishability is statistical.*

Proof. (Sketch) These two hybrids are statistically close via construction of Mach_i. This can be proven by a series of t intermediate statistically close hybrids. Define $\mathbf{Hybrid}_{5,i}$ for $i \in [t]$, where randomness to encrypt challenge ciphertext is sampled as in previous hybrid for all indices $j \neq i$. For index i, it is generated using intersection of $\mathsf{Set}_i \cap \mathsf{SetR}_i$. Note that $\mathbf{Hybrid}_{5,t}$ is the same as $\mathbf{Hybrid}_6$. Let us calculate the statistical distance between the two hybrids. The statistical distance is bounded by the sum of probability that Set_i has a density less than $1/\lambda$ and the probability that intersection of SetR_i and Set_i is empty. This is because once SetR_i is chosen and has a large enough size, sampling SetR_i randomly and sampling from the intersection ensures that the probability of choosing any element from Set_i is identical by symmetry.

The probability that Set_i has a density smaller than $1/3p(\lambda)$ is bounded by at least $2^{-c \cdot \lambda}$ for some constant c (due to theorem 17). Let us bound the probability that intersection of Set_i and SetR_i is empty. This probability is bounded by $(1 - |\mathsf{Set}_i| 2^{-\ell_i})^{p(\lambda)\lambda}$. This is less than, $(1 - 1/3p(\lambda))^{p(\lambda)\lambda} \leq e^{-\lambda/3}$ with probability at least $1 - 2^{-\lambda}$ over the construction of Set_i (described by $\mathbf{Hybrid}_5$ according to theorem 17). □

Hybrid$_7$: This hybrid is the same as the previous hybrid except that the representation changes. Let Y_β denote the set of indices i where $y_i = \beta$ for $\beta \in \{0,1\}$. Let Mach' be an (unbounded) machine that computes the result of $\mathsf{Mach}_1, ..., \mathsf{Mach}_t$. Precisely, Mach' takes as input y, SetR_i for $i \in [t]$, circuit C, $\{x_i^j\}_{j\in[\Gamma],i\in[t]}$, Z^*, $\{x_i^*\}_{i\in Y_0}$ and hardwired partial decryption values $\{\theta_i\}_{i\in Y_1}$. Note that Mach' does not take as input x_i^* for $i \in Y_1$ and in order to compute the result, it may have to break commitment Z^* to construct x_i^* for $i \in Y_1$.

Denote by X the distribution $(y, \{\mathsf{SetR}_i\}_{i\in[t]}, C, \{x_i^j\}_{j\in[\Gamma],i\in[t]}, \{x_i^*\}_{i\in Y_0}, Z^*, \{\theta_i\}_{i\in Y_1}, \mathsf{fct}, \{Z^j\}_{j\in\Gamma})$. Thus, $\mathsf{Mach}'(X) \to (j_1, ..., j_t)$ where j_i is an index in $[t]$. Here is the pseudocode of Mach'.

1. On input $X = (y, \{\mathsf{SetR}_i\}_{i\in[t]}, C, \{x_i^j\}_{j\in[\Gamma],i\in[t]}, \{x_i^*\}_{i\in Y_0}, Z^*, \{\theta_i\}_{i\in Y_1}, \mathsf{fct}, \{Z^j\}_{j\in\Gamma})$,take following steps.
2. Break Z^* to compute $(K_1, ..., K_t, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$.
3. For $i \in Y_1$, compute $x_i^* = (\mathsf{fct}, \mathsf{fsk}_i, K_i)$.
4. Output $\Big(\mathsf{Mach}_i(C, x^*, \{x_i^j\}_{j\in\Gamma})\Big)_{i\in[t]}$

We describe the hybrid in detail now:

1. Adversary gets as input the security parameter 1^λ and outputs a circuit $C \in \mathcal{C}_{n,s}$. He also gives out some messages $m_0, ..., m_\Gamma, m_0^*, m_1^* \in \{0,1\}^n$ such that $C(m_0^*) = C(m_1^*)$.
2. Sample a bit $b \in \{0,1\}$.
3. Sample a string $y \in \{0,1\}^t$ such that for every $i \in [t]$, set $y_i = 1$ with probability $1/2p(\lambda)$ and $y_i = 0$ with probability $(1 - 1/2p(\lambda))$. Here, each bit y_i is chosen independently. Abort if $y = 0^t$.
4. To encrypt challenge ciphertext, compute $x^* = (x_1^*, ..., x_t^*)$ as follows.
   - Run $\mathsf{TFHE.Setup}(1^\lambda, 1^t) \to (\mathsf{fpk}, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$.
   - Compute $\mathsf{fct} \leftarrow \mathsf{TFHE.Enc}(\mathsf{fpk}, m_b^*)$.
   - Sample t PRF keys $K_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $i \in [t]$.
   - Set $x_i^* = (\mathsf{fct}, \mathsf{fsk}_i, K_i)$.
5. Let F be the circuit described in the key generation algorithm. Set $\theta_i = F(x_i^*) = \mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{Eval}(C, \mathsf{fct}); \mathsf{PRF}(K_i, \mathsf{Eval}(C, \mathsf{fct})))$ for $i \in [t]$.
6. Similarly, compute x^j as intermediate messages corresponding to m_j for $j \in \Gamma$, as described above. Let x^j be denoted as $(x_1^j, \ldots x_t^j)$.
7. Compute $Z^* = \mathsf{Com}(K_1, ..., K_t, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$. For other ciphertext queries $j \in [\Gamma]$, compute Z^j similarly (using respective PRF and partial decryption keys).
8. For $i \in [t]$, sample SetR_i as a set of t uniformly chosen inputs from support of $\mathcal{M}_i$ (which is equal to $\mathcal{M}_{1,i}$ if $y_i = 1$ and $\overline{\mathcal{M}_{0,i}}$ otherwise).

9. [**Change**] Define $X = (y, \{\mathsf{SetR}_i\}_{i \in [t]}, C, \{x_i^j\}_{j \in [\Gamma], i \in [t]}, \{x_i^*\}_{i \in Y_0}, Z^*, \{\theta_i\}_{i \in Y_1}, \mathsf{fct}, \{Z^j\}_{j \in \Gamma})$

10. [**Change**] For every $i \in [t]$, to compute the following steps, we generate randomness $R_i = (r_{1,i}, r_{2,i}, \{r_{3,j,i}\}_{j \in [\Gamma]}, r_{4,i})$ as follows. Run $\mathsf{Mach}'(X) \to (j_1, ..., j_t)$. Set R_i as the randomness with index j_i in the set SetR_i.

11. Then run the setup of the FE as follows: compute $\mathsf{sFE.Setup}(1^\lambda; r_{1,i}) \to sk_i$ for $i \in [t]$ and set $\mathsf{MSK} = (sk_1, ..., sk_t)$.

12. Generate $sk_C = (sk_{C,1}, .., sk_{C,t})$ where $sk_{C,i} \leftarrow \mathsf{sFE.sfKG}(sk_i, F, \theta_i; r_{2,i})$ for the circuit F described in the key generation algorithm.

13. For $j \in [\Gamma]$, compute $\mathsf{CT}^j = (Z^j, \mathsf{CT}_1^j, .., \mathsf{CT}_t^j)$. Here, $\mathsf{CT}_i^j \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^j; r_{3,j,i})$ for $i \in [t]$.

14. Compute $\mathsf{CT}^* = (Z^*, \mathsf{CT}_1^*, .., \mathsf{CT}_t^*)$. Here, for every $i \in [t]$, if $y_i = 0$, $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^*; r_{4,i})$ otherwise $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.sfEnc}(sk_i, 1^\lambda, 1^\lambda; r_{4,i})$.

15. Give the following to adversary $(sk_C, \{\mathsf{CT}_i^j\}_{j \in [\Gamma], i \in [t]}, \mathsf{CT}^*)$

16. Adversary guesses $b' \in \{0, 1\}$

Lemma 16. *Statistical distance between* $\mathbf{Hybrid}_7$ *and* $\mathbf{Hybrid}_6$ *is bounded by* $2^{-\lambda^c}$ *for some constant* $c > 0$.

Proof. If Com satisfies statistical binding then with probability at least $1 - 2^{-\lambda^c}$ for some constant $c > 0$ over the coins of the setup algorithm of the commitment scheme, the hybrids are identical. $\square$

We first restate theorem 18. We will use the following theorem in this hybrid.

Theorem 18. *Let $n, \ell \in \mathbb{N}$, $\epsilon > 0$ and $\mathcal{C}_{leak}$ be a family of distinguisher circuits from $\{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}$ of size $s(n)$. Then, for every distribution (X, Z) over $\{0,1\}^n \times \{0,1\}^\ell$, there exists a simulator $h : \{0,1\}^n \to \{0,1\}^\ell$ such that:*

- *h has size bounded by $s' = O(s2^\ell \epsilon^{-2})$.*
- *(X, Z) and $(X, h(X))$ are indistinguishable by $\mathcal{C}_{leak}$. That is for every $C \in \mathcal{C}_{leak}$,*

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

Hybrid$_8$: This hybrid is the same as the previous one except that we now simulate Mach' using theorem 18 by a leakage simulator h. Note that output length of Mach' is $t \log t$. To construct h_i using theorem 18, we set the size of distinguisher to be size_8 and advantage bound to be adv_8.

Constructing simulators h: To construct h we define corresponding distributions inductively as follows. Denote by X the distribution $(y, \{\mathsf{SetR}_i\}_{i\in[t]}, C, \{x_i^j\}_{j\in[\Gamma],i\in[t]}, \{x_i^*\}_{i\in Y_0}, Z^*, \{\theta_i\}_{i\in Y_1}, \mathsf{fct}, \{Z^j\}_{j\in\Gamma})$. Let $Z = \mathsf{Mach}'(X)$. Thus theorem 18 gives us a simulator h of size $\mathsf{size}_h = O(\mathsf{size}_8 \cdot \mathsf{adv}_8^{-2} \cdot 2^{t \log t})$. Once we use h, the hybrid is implementable by circuit of size bounded by $\mathsf{size}_h + poly(\lambda)$ for some polynomial $poly$.

1. Adversary gets as input the security parameter 1^λ and outputs a circuit $C \in \mathcal{C}_{n,s}$. He also gives out some messages $m_0, ..., m_\Gamma, m_0^*, m_1^* \in \{0,1\}^n$ such that $C(m_0^*) = C(m_1^*)$.
2. Sample a bit $b \in \{0,1\}$.
3. Sample a string $y \in \{0,1\}^t$ such that for every $i \in [t]$, set $y_i = 1$ with probability $1/2p(\lambda)$ and $y_i = 0$ with probability $(1 - 1/2p(\lambda))$. Here, each bit y_i is chosen independently. Abort if $y = 0^t$.
4. To encrypt challenge ciphertext, compute $x^* = (x_1^*, ..., x_t^*)$ as follows.
   - Run $\mathsf{TFHE.Setup}(1^\lambda, 1^t) \to (\mathsf{fpk}, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$.
   - Compute $\mathsf{fct} \leftarrow \mathsf{TFHE.Enc}(\mathsf{fpk}, m_b^*)$.
   - Sample t PRF keys $K_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $i \in [t]$.
   - Set $x_i^* = (\mathsf{fct}, \mathsf{fsk}_i, K_i)$.
5. Let F be the circuit described in the key generation algorithm. Set $\theta_i = F(x_i^*) = \mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{Eval}(C, \mathsf{fct}); \mathsf{PRF}(K_i, \mathsf{Eval}(C, \mathsf{fct})))$ for $i \in [t]$.
6. Similarly, compute x^j as intermediate messages corresponding to m_j for $j \in \Gamma$, as described above. Let x^j be denoted as $(x_1^j, \ldots x_t^j)$.
7. Compute $Z^* = \mathsf{Com}(K_1, ..., K_t, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$. For other ciphertext queries $j \in [\Gamma]$, compute Z^j similarly (using respective PRF and partial decryption keys).
8. For $i \in [t]$, sample SetR_i as a set of t uniformly chosen inputs from support of $\mathcal{M}_i$ (which is equal to $\mathcal{M}_{1,i}$ if $y_i = 1$ and $\overline{\mathcal{M}}_{0,i}$ otherwise).

9. Define $X = (y, \{\mathsf{SetR}_i\}_{i\in[t]}, C, \{x_i^j\}_{j\in[\Gamma],i\in[t]}, \{x_i^*\}_{i\in Y_0}, Z^*, \{\theta_i\}_{i\in Y_1}, \mathsf{fct}, \{Z^j\}_{j\in\Gamma})$

10. [**Change**] For every $i \in [t]$, to compute the following steps, we generate randomness $R_i = (r_{1,i}, r_{2,i}, \{r_{3,j,i}\}_{j\in[\Gamma]}, r_{4,i})$ as follows. Run $h(X) \to (j_1, ..., j_t)$. Let R_i be the randomness with index j_i in the set SetR_i.

11. Then run the setup of the FE as follows: it compute $\mathsf{sFE.Setup}(1^\lambda; r_{1,i}) \to sk_i$ for $i \in [t]$ and set $\mathsf{MSK} = (sk_1, ..., sk_t)$.

12. Generate $sk_C = (sk_{C,1}, .., sk_{C,t})$ where $sk_{C,i} \leftarrow \mathsf{sFE.sfKG}(sk_i, F, \theta_i; r_{2,i})$ for the circuit F described in the key generation algorithm.

13. For $j \in [\Gamma]$, compute $\mathsf{CT}^j = (Z^j, \mathsf{CT}_1^j, .., \mathsf{CT}_t^j)$. Here, $\mathsf{CT}_i^j \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^j; r_{3,j,i})$ for $i \in [t]$.

14. Compute $\mathsf{CT}^* = (Z^*, \mathsf{CT}_1^*, .., \mathsf{CT}_t^*)$. Here, for every $i \in [t]$, if $y_i = 0$, $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^*; r_{4,i})$ otherwise $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.sfEnc}(sk_i, 1^\lambda, 1^\lambda; r_{4,i})$.

15. Give the following to adversary $(sk_C, \{\mathsf{CT}_i^j\}_{j\in[\Gamma],i\in[t]}, \mathsf{CT}^*)$

16. Adversary guesses $b' \in \{0, 1\}$

Lemma 17. *Due to theorem 18 the following holds, for any adversary $\mathcal{A}$ of size $O(\mathsf{size}_8)$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_7) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_8) = 1]| < \mathsf{adv}_8$. Here $\mathsf{size}_8 > \lambda^{c_8}$ for some constant $c_8 > 0$.*

Proof. This proof is a direct application of theorem 18. Here, is our reduction.

Let $X = (y, \{\mathsf{SetR}_i\}_{i\in[t]}, C, \{x_i^j\}_{j\in[\Gamma],i\in[t]}, \{x_i^*\}_{i\in Y_0}, Z^*, \{\theta_i\}_{i\in Y_1}, \mathsf{fct}, \{Z^j\}_{j\in\Gamma})$ as defined in stage $1-8$ of $\mathbf{Hybrid}_7$ and $\mathbf{Hybrid}_8$. Reduction recieves X and samples $(j_1, ..., j_t)$ which are either generated using Mach' or using simulator h. Assume that we have an adversary $\mathcal{A}$ (of size size_8) that distinguishes the hybrids with probability greater than adv_8. Then the reduction proceeds as follows:

1. Parse $X = (y, \{\mathsf{SetR}_i\}_{i\in[t]}, C, \{x_i^j\}_{j\in[\Gamma],i\in[t]}, \{x_i^*\}_{i\in Y_0}, Z^*, \{\theta_i\}_{i\in Y_1}, \{Z^j\}_{j\in\Gamma})$.

2. For $i \in Y_0$, set $\theta_i = F(x_i^*)$.

3. Parse $\mathsf{aux} = (j_1, ..., j_t)$. Set R_i as the randomness with index j_i in the set SetR_i for $i \in [t]$. Parse $R_i = (r_{1,i}, r_{2,i}, \{r_{3,j,i}\}_{j\in[\Gamma]}, r_{4,i})$.

4. Then run the setup of the FE as follows: compute $\mathsf{sFE.Setup}(1^\lambda; r_{1,i}) \to sk_i$ for $i \in [t]$ and set $\mathsf{MSK} = (sk_1, ..., sk_t)$.

5. Generate $sk_C = (sk_{C,1}, .., sk_{C,t})$ where $sk_{C,i} \leftarrow \mathsf{sFE.sfKG}(sk_i, F, \theta_i; r_{2,i})$ for the circuit F described in the key generation algorithm.

6. For $j \in [\Gamma]$, compute $\mathsf{CT}^j = (Z^j, \mathsf{CT}_1^j, .., \mathsf{CT}_t^j)$. Here, $\mathsf{CT}_i^j \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^j; r_{3,j,i})$ for $i \in [t]$.

7. Compute $\mathsf{CT}^* = (Z^*, \mathsf{CT}_1^*, .., \mathsf{CT}_t^*)$. Here, for every $i \in [t]$, if $y_i = 0$, $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^*; r_{4,i})$ otherwise $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.sfEnc}(sk_i, 1^\lambda, 1^\lambda; r_{4,i})$.

8. Give the following to adversary $(sk_C, \{\mathsf{CT}_i^j\}_{j\in[\Gamma],i\in[t]}, \mathsf{CT}^*)$

9. Adversary guesses $b' \in \{0, 1\}$

10. Output adversary's guess as its own output.

Note that the reduction emulates the either $\mathbf{Hybrid}_7$ (if aux is generated using Mach') or $\mathbf{Hybrid}_8$ (if aux is generated as h). Hence, the advantage of $\mathcal{A}$ is exactly the same as the advantage of reduction to win in the game of theorem 6. Note that if $\mathcal{A}$ has size size_8, the size of reduction is $\mathsf{size}_8 + poly(\lambda)$ for some fixed polynomial $poly$, which is also $O(\mathsf{size}_8)$ assuming size_8 is large enough. Claim now follows from the way the parameters are set in theorem 18.

□

Hybrid$_9$: This hybrid is the same as the previous one except that Z^* is now a commitment of 0.

1. Adversary gets as input the security parameter 1^λ and outputs a circuit $C \in \mathcal{C}_{n,s}$. He also gives out some messages $m_0, ..., m_\Gamma, m_0^*, m_1^* \in \{0,1\}^n$ such that $C(m_0^*) = C(m_1^*)$.
2. Sample a bit $b \in \{0,1\}$.
3. Sample a string $y \in \{0,1\}^t$ such that for every $i \in [t]$, set $y_i = 1$ with probability $1/2p(\lambda)$ and $y_i = 0$ with probability $(1 - 1/2p(\lambda))$. Here, each bit y_i is chosen independently. Abort if $y = 0^t$.
4. To encrypt challenge ciphertext, compute $x^* = (x_1^*, ..., x_t^*)$ as follows.
   - Run $\mathsf{TFHE.Setup}(1^\lambda, 1^t) \to (\mathsf{fpk}, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$.
   - Compute $\mathsf{fct} \leftarrow \mathsf{TFHE.Enc}(\mathsf{fpk}, m_b^*)$.
   - Sample t PRF keys $K_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $i \in [t]$.
   - Set $x_i^* = (\mathsf{fct}, \mathsf{fsk}_i, K_i)$.
5. Let F be the circuit described in the key generation algorithm. Set $\theta_i = F(x_i^*) = \mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{Eval}(C, \mathsf{fct}); \mathsf{PRF}(K_i, \mathsf{Eval}(C, \mathsf{fct})))$ for $i \in [t]$.
6. Similarly, compute x^j as intermediate messages corresponding to m_j for $j \in \Gamma$, as described above. Let x^j be denoted as $(x_1^j, \ldots x_t^j)$.
7. [**Change**] Compute $Z^* = \mathsf{Com}(0^\kappa)$ where κ is the length of $(K_1, .., k_t, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$. For other ciphertext queries $j \in [\Gamma]$, compute Z^j as in the previous hybrids (using respective PRF and partial decryption keys).
8. For $i \in [t]$, sample SetR_i as a set of t uniformly chosen inputs from support of $\mathcal{M}_i$ (which is equal to $\mathcal{M}_{1,i}$ if $y_i = 1$ and $\overline{\mathcal{M}}_{0,i}$ otherwise).
9. Define $X = (y, \{\mathsf{SetR}_i\}_{i \in [t]}, C, \{x_i^j\}_{j \in [\Gamma], i \in [t]}, \{x_i^*\}_{i \in Y_0}, Z^*, \{\theta_i\}_{i \in Y_1}, \mathsf{fct}, \{Z^j\}_{j \in \Gamma})$
10. For every $i \in [t]$, to compute the following steps, we generate randomness $R_i = (r_{1,i}, r_{2,i}, \{r_{3,j,i}\}_{j \in [\Gamma]}, r_{4,i})$ as follows. Run $h(X) \to (j_1, ..., j_t)$. Let R_i be the randomness with index j_i in the set SetR_i.
11. Then run the setup of the FE as follows: it compute $\mathsf{sFE.Setup}(1^\lambda; r_{1,i}) \to sk_i$ for $i \in [t]$ and set $\mathsf{MSK} = (sk_1, ..., sk_t)$.
12. Generate $sk_C = (sk_{C,1}, .., sk_{C,t})$ where $sk_{C,i} \leftarrow \mathsf{sFE.sfKG}(sk_i, F, \theta_i; r_{2,i})$ for the circuit F described in the key generation algorithm.
13. For $j \in [\Gamma]$, compute $\mathsf{CT}^j = (Z^j, \mathsf{CT}_1^j, .., \mathsf{CT}_t^j)$. Here, $\mathsf{CT}_i^j \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^j; r_{3,j,i})$ for $i \in [t]$.
14. Compute $\mathsf{CT}^* = (Z^*, \mathsf{CT}_1^*, .., \mathsf{CT}_t^*)$. Here, for every $i \in [t]$, if $y_i = 0$, $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^*; r_{4,i})$ otherwise $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.sfEnc}(sk_i, 1^\lambda, 1^\lambda; r_{4,i})$.
15. Give the following to adversary $(sk_C, \{\mathsf{CT}_i^j\}_{j \in [\Gamma], i \in [t]}, \mathsf{CT}^*)$

16. Adversary guesses $b' \in \{0,1\}$

Lemma 18. *If* Com *is* $(\mathsf{size}_{\mathsf{Com}}, \mathsf{adv}_{\mathsf{Com}})$ *secure and* $\mathsf{size}_{\mathsf{Com}} > \mathsf{size}_h + poly(\lambda)$ *for some fixed polynomial poly, then for any adversary* $\mathcal{A}$ *of size* $O(\mathsf{size}_h + poly(\lambda))$*,* $|\Pr[\mathcal{A}(\mathbf{Hybrid}_8) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_9) = 1]| < \mathsf{adv}_{\mathsf{Com}}$.

Proof. Note that both hybrids $\mathbf{Hybrid}_8$ and $\mathbf{Hybrid}_9$ can be computed in size the sum of size_h (used for simulator h and a fixed polynomial $poly(\lambda)$ to generate the hybrid distribution). The reduction to the commitment scheme works by using the challenge commitment Z^* (either a commitment of 0 or respective PRF keys and TFHE partial decryption keys) to generate a hybrid. In $\mathbf{Hybrid}_8$, Z^* is a commitment of PRF keys and partial decryption keys. In $\mathbf{Hybrid}_9$, it is a commitment of 0. If there is an adversary $\mathcal{A}$, that distinguishes the hybrids with probability greater than $\mathsf{adv}_{\mathsf{Com}}$, the reduction uses the challenge commitment to generate either $\mathbf{Hybrid}_8$ or $\mathbf{Hybrid}_9$ (depending on Z^*) and runs $\mathcal{A}$ on it. Then it just outputs response of $\mathcal{A}$ as its guess. The advantage of $\mathcal{A}$ then becomes the advantage of the reduction. Note that reduction runs in time bounded in running time of generating the hybrid and running time of $\mathcal{A}$. Since, Com is secure against circuits of this size, the claim follows. □

Hybrid$_{10}$: This hybrid is the same as the previous one except that for $i \in Y_1$, θ_i is computed honestly using the $\mathsf{PartDec}$ algorithm using true randomness, instead of using PRF key K_i.

1. Adversary gets as input the security parameter 1^λ and outputs a circuit $C \in \mathcal{C}_{n,s}$. He also gives out some messages $m_0, ..., m_\Gamma, m_0^*, m_1^* \in \{0,1\}^n$ such that $C(m_0^*) = C(m_1^*)$.
2. Sample a bit $b \in \{0,1\}$.
3. Sample a string $y \in \{0,1\}^t$ such that for every $i \in [t]$, set $y_i = 1$ with probability $1/2p(\lambda)$ and $y_i = 0$ with probability $(1 - 1/2p(\lambda))$. Here, each bit y_i is chosen independently. Abort if $y = 0^t$.
4. To encrypt challenge ciphertext, compute $x^* = (x_1^*, ..., x_t^*)$ as follows.
   - Run $\mathsf{TFHE.Setup}(1^\lambda, 1^t) \to (\mathsf{fpk}, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$.
   - Compute $\mathsf{fct} \leftarrow \mathsf{TFHE.Enc}(\mathsf{fpk}, m_b^*)$.
   - [**Change**] Sample PRF keys $K_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $i \in Y_0$.
   - [**Change**] Set $x_i^* = (\mathsf{fct}, \mathsf{fsk}_i, K_i)$ for $i \in Y_0$.
5. [**Change**] Let F be the circuit described in the key generation algorithm. Set $\theta_i = F(x_i^*) = \mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{Eval}(C, \mathsf{fct}); \mathsf{PRF}(K_i, \mathsf{Eval}(C, \mathsf{fct})))$ for $i \in Y_0$, otherwise set $\theta_i = \mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{Eval}(C, \mathsf{fct}))$ using fresh and independent randomness.
6. Similarly, compute x^j as intermediate messages corresponding to m_j for $j \in \Gamma$, as described above. Let x^j be denoted as $(x_1^j, \ldots x_t^j)$.
7. Compute $Z^* = \mathsf{Com}(0^\kappa)$ where κ is the length of $(K_1, .., k_t, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$. For other ciphertext queries $j \in [\Gamma]$, compute Z^j as in the previous hybrids (using respective PRF and partial decryption keys).
8. For $i \in [t]$, sample SetR_i as a set of t uniformly chosen inputs from support of $\mathcal{M}_i$ (which is equal to $\mathcal{M}_{1,i}$ if $y_i = 1$ and $\overline{\mathcal{M}}_{0,i}$ otherwise).
9. Define $X = (y, \{\mathsf{SetR}_i\}_{i \in [t]}, C, \{x_i^j\}_{j \in [\Gamma], i \in [t]}, \{x_i^*\}_{i \in Y_0}, Z^*, \{\theta_i\}_{i \in Y_1}, \mathsf{fct}, \{Z^j\}_{j \in \Gamma})$
10. For every $i \in [t]$, to compute the following steps, we generate randomness $R_i = (r_{1,i}, r_{2,i}, \{r_{3,j,i}\}_{j \in [\Gamma]}, r_{4,i})$ as follows. Run $h(X) \to (j_1, ..., j_t)$. Let R_i be the randomness with index j_i in the set SetR_i.
11. Then run the setup of the FE as follows: it compute $\mathsf{sFE.Setup}(1^\lambda; r_{1,i}) \to sk_i$ for $i \in [t]$ and set $\mathsf{MSK} = (sk_1, ..., sk_t)$.
12. Generate $sk_C = (sk_{C,1}, .., sk_{C,t})$ where $sk_{C,i} \leftarrow \mathsf{sFE.sfKG}(sk_i, F, \theta_i; r_{2,i})$ for the circuit F described in the key generation algorithm.
13. For $j \in [\Gamma]$, compute $\mathsf{CT}^j = (Z^j, \mathsf{CT}_1^j, .., \mathsf{CT}_t^j)$. Here, $\mathsf{CT}_i^j \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^j; r_{3,j,i})$ for $i \in [t]$.
14. Compute $\mathsf{CT}^* = (Z^*, \mathsf{CT}_1^*, .., \mathsf{CT}_t^*)$. Here, for every $i \in [t]$, if $y_i = 0$, $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^*; r_{4,i})$ otherwise $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.sfEnc}(sk_i, 1^\lambda, 1^\lambda; r_{4,i})$.

15. Give the following to adversary $(sk_C, \{\mathsf{CT}_i^j\}_{j\in[\Gamma],i\in[t]}, \mathsf{CT}^*)$

16. Adversary guesses $b' \in \{0,1\}$

Lemma 19. *If* PRF *is* $(\mathsf{size}_{\mathsf{PRF}}, \mathsf{adv}_{\mathsf{PRF}})-$*secure with* $\mathsf{size}_{\mathsf{PRF}} > \mathsf{size}_h + poly(\lambda)$ *for some fixed polynomial* $poly(\lambda)$*, then for any adversary* $\mathcal{A}$ *of size* $O(\mathsf{size}_{\mathsf{PRF}})$*,* $|\Pr[\mathcal{A}(\mathbf{Hybrid}_9) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_{10}) = 1]| < \mathsf{adv}_{\mathsf{PRF}}$.

Proof. Note that both hybrids $\mathbf{Hybrid}_9$ and $\mathbf{Hybrid}_{10}$ can be computed in time roughly the size of running h along with polynomial overheads, and can be bounded by $\mathsf{size}_h + poly(\lambda)$. The only difference between the hybrids is the way θ_i is generated for $i \in Y_1$. This is proven by fixing a best possible y. In $\mathbf{Hybrid}_9$ it is generated using randomness derived from PRF keys. In $\mathbf{Hybrid}_{10}$, they are generated using true randomness. Note that for $i \in Y_1$, the PRF keys are absent. The security then holds due to the security of PRF against adversaries of size $\mathsf{size}_{\mathsf{PRF}}$. □

Hybrid$_{11}$: This hybrid is the same as the previous one except that for first index $i_0 \in Y_1$, θ_{i_0} is simulated using the simulator of the TFHE partial decryption keys $\{\mathsf{fsk}_i\}_{i \neq i_0}$. That is, set $\theta_{i_0} = \mathsf{TFHE.Sim}(\{\mathsf{fsk}_i\}_{i \neq i_0}, \mathsf{fct}, C, C(m_0))$.

1. Adversary gets as input the security parameter 1^λ and outputs a circuit $C \in \mathcal{C}_{n,s}$. He also gives out some messages $m_0, ..., m_\Gamma, m_0^*, m_1^* \in \{0,1\}^n$ such that $C(m_0^*) = C(m_1^*)$.
2. Sample a bit $b \in \{0,1\}$.
3. Sample a string $y \in \{0,1\}^t$ such that for every $i \in [t]$, set $y_i = 1$ with probability $1/2p(\lambda)$ and $y_i = 0$ with probability $(1 - 1/2p(\lambda))$. Here, each bit y_i is chosen independently. Abort if $y = 0^t$.
4. To encrypt challenge ciphertext, compute $x^* = (x_1^*, ..., x_t^*)$ as follows.
   - Run $\mathsf{TFHE.Setup}(1^\lambda, 1^t) \rightarrow (\mathsf{fpk}, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$.
   - Compute $\mathsf{fct} \leftarrow \mathsf{TFHE.Enc}(\mathsf{fpk}, m_b^*)$.
   - [**Change**] Sample PRF keys $K_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $i \in Y_0$.
   - [**Change**] Set $x_i^* = (\mathsf{fct}, \mathsf{fsk}_i, K_i)$ for $i \in Y_0$.
5. [**Change**] Let F be the circuit described in the key generation algorithm. Let i_0 be the first index in Y_1. Set $\theta_i = F(x_i^*) =$ $\mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{Eval}(C, \mathsf{fct}); \mathsf{PRF}(K_i, \mathsf{Eval}(C, \mathsf{fct})))$ for $i \in Y_0$, otherwise set $\theta_i = \mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{Eval}(C, \mathsf{fct}))$ for $i \in Y_1 \setminus i_0$. Set $\theta_{i_0} = \mathsf{TFHE.Sim}(\{\mathsf{fsk}_i\}_{i \neq i_0}, \mathsf{fct}, C, C(m_0))$
6. Similarly, compute x^j as intermediate messages corresponding to m_j for $j \in \Gamma$, as described above. Let x^j be denoted as $(x_1^j, \ldots x_t^j)$.
7. Compute $Z^* = \mathsf{Com}(0^\kappa)$ where κ is the length of $(K_1, .., k_t, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$. For other ciphertext queries $j \in [\Gamma]$, compute Z^j as in the previous hybrids (using respective PRF and partial decryption keys).
8. For $i \in [t]$, sample SetR_i as a set of t uniformly chosen inputs from support of $\mathcal{M}_i$ (which is equal to $\mathcal{M}_{1,i}$ if $y_i = 1$ and $\overline{\mathcal{M}}_{0,i}$ otherwise).
9. Define $X = (y, \{\mathsf{SetR}_i\}_{i \in [t]}, C, \{x_i^j\}_{j \in [\Gamma], i \in [t]}, \{x_i^*\}_{i \in Y_0}, Z^*, \{\theta_i\}_{i \in Y_1}, \mathsf{fct}, \{Z^j\}_{j \in \Gamma})$
10. For every $i \in [t]$, to compute the following steps, we generate randomness $R_i = (r_{1,i}, r_{2,i}, \{r_{3,j,i}\}_{j \in [\Gamma]}, r_{4,i})$ as follows. Run $h \rightarrow (j_1, ..., j_t)$. Let R_i be the randomness with index j_i in the set SetR_i.
11. Then run the setup of the FE as follows: it compute $\mathsf{sFE.Setup}(1^\lambda; r_{1,i}) \rightarrow sk_i$ for $i \in [t]$ and set $\mathsf{MSK} = (sk_1, ..., sk_t)$.
12. Generate $sk_C = (sk_{C,1}, .., sk_{C,t})$ where $sk_{C,i} \leftarrow \mathsf{sFE.sfKG}(sk_i, F, \theta_i; r_{2,i})$ for the circuit F described in the key generation algorithm.
13. For $j \in [\Gamma]$, compute $\mathsf{CT}^j = (Z^j, \mathsf{CT}_1^j, .., \mathsf{CT}_t^j)$. Here, $\mathsf{CT}_i^j \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^j; r_{3,j,i})$ for $i \in [t]$.

14. Compute $\mathsf{CT}^* = (Z^*, \mathsf{CT}^*_1, .., \mathsf{CT}^*_t)$. Here, for every $i \in [t]$, if $y_i = 0$, $\mathsf{CT}^*_i \leftarrow \mathsf{sFE.Enc}(sk_i, x^*_i; r_{4,i})$ otherwise $\mathsf{CT}^*_i \leftarrow \mathsf{sFE.sfEnc}(sk_i, 1^\lambda, 1^\lambda; r_{4,i})$.

15. Give the following to adversary $(sk_C, \{\mathsf{CT}^j_i\}_{j\in[\Gamma],i\in[t]}, \mathsf{CT}^*)$

16. Adversary guesses $b' \in \{0, 1\}$

Lemma 20. *If* TFHE *is statistically simulation secure, then there exists constant* $c_{10} > 0$ *such that for any adversary* $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_{10}) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_{11}) = 1]| < 2^{-\lambda^{c_{10}}}$.

Proof. The only difference between the hybrids is the way commitment θ_{i_0} is generated for first $i_0 \in Y_1$. This is proven by fixing a best possible y. In $\mathbf{Hybrid}_{10}$ it is generated using TFHE.PartDec. In $\mathbf{Hybrid}_{11}$, it is generated using TFHE.Sim. Note that these two distributions are statistically close. The security then holds due to the security of TFHE. □

Hybrid$_{12}$: This hybrid is the same as the previous one except that now we generate fct as an encryption of 0

1. Adversary gets as input the security parameter 1^λ and outputs a circuit $C \in \mathcal{C}_{n,s}$. He also gives out some messages $m_0, ..., m_\Gamma, m_0^*, m_1^* \in \{0,1\}^n$ such that $C(m_0^*) = C(m_1^*)$.
2. Sample a bit $b \in \{0,1\}$.
3. Sample a string $y \in \{0,1\}^t$ such that for every $i \in [t]$, set $y_i = 1$ with probability $1/2p(\lambda)$ and $y_i = 0$ with probability $(1 - 1/2p(\lambda))$. Here, each bit y_i is chosen independently. Abort if $y = 0^t$.
4. To encrypt challenge ciphertext, compute $x^* = (x_1^*, ..., x_t^*)$ as follows.
   - Run $\mathsf{TFHE.Setup}(1^\lambda, 1^t) \to (\mathsf{fpk}, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$.
   - [**Change**] Compute $\mathsf{fct} \leftarrow \mathsf{TFHE.Enc}(\mathsf{fpk}, 0^{|m_0|})$.
   - Sample PRF keys $K_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $i \in Y_0$.
   - Set $x_i^* = (\mathsf{fct}, \mathsf{fsk}_i, K_i)$ for $i \in Y_0$.
5. Let F be the circuit described in the key generation algorithm. Let i_0 be the first index in Y_1. Set $\theta_i = F(x_i^*) = \mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{Eval}(C, \mathsf{fct}); \mathsf{PRF}(K_i, \mathsf{Eval}(C, \mathsf{fct})))$ for $i \in Y_0$, otherwise set $\theta_i = \mathsf{PartDec}(\mathsf{fsk}_i, \mathsf{Eval}(C, \mathsf{fct}))$ for $i \in Y_1 \setminus i_0$. Set $\theta_{i_0} = \mathsf{TFHE.Sim}(\{\mathsf{fsk}_i\}_{i \neq i_0}, \mathsf{fct}, C, C(m_0))$
6. Similarly, compute x^j as intermediate messages corresponding to m_j for $j \in \Gamma$, as described above. Let x^j be denoted as $(x_1^j, \ldots x_t^j)$.
7. Compute $Z^* = \mathsf{Com}(0^\kappa)$ where κ is the length of $(K_1, .., k_t, \mathsf{fsk}_1, ..., \mathsf{fsk}_t)$. For other ciphertext queries $j \in [\Gamma]$, compute Z^j as in the previous hybrids (using respective PRF and partial decryption keys).
8. For $i \in [t]$, sample SetR_i as a set of t uniformly chosen inputs from support of $\mathcal{M}_i$ (which is equal to $\mathcal{M}_{1,i}$ if $y_i = 1$ and $\overline{\mathcal{M}}_{0,i}$ otherwise).
9. Define $X = (y, \{\mathsf{SetR}_i\}_{i \in [t]}, C, \{x_i^j\}_{j \in [\Gamma], i \in [t]}, \{x_i^*\}_{i \in Y_0}, Z^*, \{\theta_i\}_{i \in Y_1}, \mathsf{fct}, \{Z^j\}_{j \in \Gamma})$
10. For every $i \in [t]$, to compute the following steps, we generate randomness $R_i = (r_{1,i}, r_{2,i}, \{r_{3,j,i}\}_{j \in [\Gamma]}, r_{4,i})$ as follows. Run $h \to (j_1, ..., j_t)$. Let R_i be the randomness with index j_i in the set SetR_i.
11. Then run the setup of the FE as follows: it compute $\mathsf{sFE.Setup}(1^\lambda; r_{1,i}) \to sk_i$ for $i \in [t]$ and set $\mathsf{MSK} = (sk_1, ..., sk_t)$.
12. Generate $sk_C = (sk_{C,1}, .., sk_{C,t})$ where $sk_{C,i} \leftarrow \mathsf{sFE.sfKG}(sk_i, F, \theta_i; r_{2,i})$ for the circuit F described in the key generation algorithm.
13. For $j \in [\Gamma]$, compute $\mathsf{CT}^j = (Z^j, \mathsf{CT}_1^j, .., \mathsf{CT}_t^j)$. Here, $\mathsf{CT}_i^j \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^j; r_{3,j,i})$ for $i \in [t]$.
14. Compute $\mathsf{CT}^* = (Z^*, \mathsf{CT}_1^*, .., \mathsf{CT}_t^*)$. Here, for every $i \in [t]$, if $y_i = 0$, $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.Enc}(sk_i, x_i^*; r_{4,i})$ otherwise $\mathsf{CT}_i^* \leftarrow \mathsf{sFE.sfEnc}(sk_i, 1^\lambda, 1^\lambda; r_{4,i})$.

15. Give the following to adversary $(sk_C, \{\mathsf{CT}_i^j\}_{j\in[\Gamma],i\in[t]}, \mathsf{CT}^*)$

16. Adversary guesses $b' \in \{0,1\}$

Lemma 21. *If* TFHE *is* $(\mathsf{size}_{\mathsf{TFHE}}, \mathsf{adv}_{\mathsf{TFHE}})-$*semantic secure and* $\mathsf{size}_{\mathsf{TFHE}} > \mathsf{size}_h + poly(\lambda)$ *for some fixed polynomial poly, then for any adversary* $\mathcal{A}$ *of size* $O(\mathsf{size}_{\mathsf{TFHE}})$, $|\Pr[\mathcal{A}(\mathbf{Hybrid}_{11}) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_{12}) = 1]| < \mathsf{adv}_{\mathsf{TFHE}}$.

Proof. To prove this, we non-uniformly fix y. Note that both hybrids $\mathbf{Hybrid}_{11,y}$ and $\mathbf{Hybrid}_{12,y}$ can be computed in time roughly the sum of time to implement sampler h and other polynomial overheads to generate hybrid distribution, and can be bounded by $\mathsf{size}_h + poly(\lambda)$. The only difference between the hybrids is the way encryption fct is generated. In $\mathbf{Hybrid}_{11}$ it is generated as an encryption of m_b^*, while in $\mathbf{Hybrid}_{12}$ it is generated as an encryption of 0. In both hybrids, for first $i_0 \in Y_1$, partial decryption key fsk_{i_0} is missing. The security then holds due to semantic security of TFHE. □

Lemma 22. $\mathbf{Hybrid}_{12}$ *is information theoretically independent of* b.

Proof. This claim follows by construction. □

From these lemmas we prove the theorem as follows, if we ensure that $\mathsf{size}_{\mathsf{FE}}$ as the minimum of the adversary size required for all the lemmas above, we can guarantee that the total advantage of such adversary is bounded twice the sum of advantages guaranteed by the lemmas above. Twice because in FE security game we need to argue indistinguishability between certain messages, thus the same sequence of hybrids appear twice. Hence, we prove the claim.
Now we ensure that for all hybrids, the size of adversary is at least $O(\mathsf{size}_{\mathsf{FE}})$. Let $\mathsf{adv}_{\mathsf{FE}}$ qualitatively denote the desired security level. All these conditions can be ensured if following relations are satisfied:

- $\mathsf{size}_{\mathsf{FE}} > poly(\lambda)$ for some fixed polynomial.
- Set $\mathsf{adv}' = \mathsf{adv}_{\mathsf{FE}}$ and $\mathsf{size}' = \mathsf{size}_{\mathsf{FE}}$.
- Set $\mathsf{size}_8 = \mathsf{size}_{\mathsf{FE}}$ and $\mathsf{adv}_8 = \mathsf{adv}_{\mathsf{FE}}$.
- $\mathsf{size}_h = O(\mathsf{size}_{\mathsf{FE}} \cdot \mathsf{adv}_{\mathsf{FE}}^{-2} \cdot 2^{t \log t})$.
- Thus $\mathsf{size} = O(\mathsf{size}_{\mathsf{FE}} \cdot \mathsf{adv}_{\mathsf{FE}}^{-2} \cdot poly(\lambda))$ for a fixed polynomial $poly$ in the length of the randomness in the hybrid. Note that this can be bounded by $\mathsf{size}_{\mathsf{FE}}$. Hence, $\mathsf{size} = O(\mathsf{size}_{\mathsf{FE}}^2 \cdot \mathsf{adv}_{\mathsf{FE}}^{-2})$.
- $\mathsf{size}_{\mathsf{Com}}, \mathsf{size}_{\mathsf{TFHE}}, \mathsf{size} > \mathsf{size}_h$.
- $\mathsf{adv}_{\mathsf{Com}}, \mathsf{adv}_{\mathsf{PRF}}, \mathsf{adv}_{\mathsf{TFHE}} = \mathsf{adv}_{\mathsf{FE}}$

Total advantage achieved by any adversary of size $O(\mathsf{size}_{\mathsf{FE}})$ is bounded by: $2^{-\lambda^c} + t\mathsf{adv}' + \mathsf{adv}_8 + \mathsf{adv}_{\mathsf{Com}} + \mathsf{adv}_{\mathsf{TFHE}} + \mathsf{adv}_{\mathsf{PRF}}$. This advantage is bounded by $O(2^{-\lambda^c} + t\mathsf{adv}_{\mathsf{FE}})$. Thus we prove the theorem.

15 Construction of iO

From Corollary 2, Section 14, we have the following result:

Theorem 19. *Assuming*

- *LWE secure against subexponential sized circuits.*
- *Secure Three restricted FE scheme* [6]*.*
- *PRGs with*
  - *Stretch of $k^{1+\epsilon}$ (length of input being k bits) for some constant $\epsilon > 0$.*
  - *Block locality three.*
  - *Security with* negl *distinguishing gap against adversaries of subexponential size*[7]*.*
- *Perturbation resilient generators implementable by three restricted FE scheme with*[8]*:*
  - *Stretch of $k^{1+\epsilon}$ for some $\epsilon > 0$.*
  - *Security with distinguishing gap $1 - 1/\lambda$ against adversaries of subexponential size.*

there exists subexponentially secure sublinear secret key FE for $\mathcal{C}_{n,s}$ for any polynomial $n(\lambda), s(\lambda)$ for $\lambda \in \mathbb{N}$.

In a follow-up to our work [JS18, LM18] showed a construction of a three-restricted FE scheme from SXDH over bilinear maps.

Theorem 20 ([JS18, LM18])**.** *Assuming SXDH over bilinear maps, there exists a construction of a three-restricted FE scheme.*

It was observed in [BNPW16] that any subexponentially secure secret key FE scheme for $\mathcal{C}_{n,s}$ can be used to build iO (further assuming LWE). Thus we have,

Theorem 21. *Assuming*

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

[6]See Section 11 for a construction of a three-restricted FE scheme from bilinear maps. The security of this construction is justified in the generic group model.

[7]As pointed before in Section 13, we allow a trade-off between the required level of security of ΔRG and a three-block local PRG.

[8]Refer Section 5.2 for instantiations.

– *Stretch of* $k^{1+\epsilon}$ *for some* $\epsilon > 0$.
– *Security with distinguishing gap* $1 - 1/\lambda$ *against adversaries of subexponential size.*

there exists an indistinguishability obfuscation scheme for P/poly.

Now we provide a more general theorem that allows a trade-off between the required level of security of $3\Delta\mathsf{RG}$ and a three-block local PRG. This follows from the results in Section 13 and Section 14.

Theorem 22. *Let* $\mathsf{adv}_1, \mathsf{adv}_2$ *be two distinghing gaps such that* $\mathsf{adv}_1 + \mathsf{adv}_2 \leq 1 - 1/p(\lambda)$ *for any fixed polynomial* $p(\lambda) > 1$*. Then assuming,*

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

there exists a secure iO scheme for P/poly.

Finally, if one was just interested in sublinear secret key FE scheme with polynomial security, then as observed in Section 14, we require the security of, $SXDH$. $3\Delta\mathsf{RG}$ and the block-local PRG to hold against polynomial sized circuits.

Theorem 23. *Let* $\mathsf{adv}_1, \mathsf{adv}_2$ *be two distinguishing gaps such that* $\mathsf{adv}_1 + \mathsf{adv}_2 \leq 1 - 1/p(\lambda)$ *for any fixed polynomial* $p(\lambda) > 1$*. Then assuming,*

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

there exists a secure sublinear secret key FE scheme for $\mathcal{C}_{n,s}$.

References

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

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

[Agr17] Shweta Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In *CRYPTO*, pages 3–35, 2017.

[Agr18] Shweta Agrawal. New methods for indistinguishability obfuscation: Bootstrapping and instantiation. *IACR Cryptology ePrint Archive*, 2018:633, 2018.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *CRYPTO*, pages 297–314, 2014.

[AS17] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. EUROCRYPT, 2017.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT*, pages 440–456, 2005.

[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.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, 2001.

[BFM14] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Indistinguishability obfuscation and uces: The case of computationally unpredictable sources. In *CRYPTO*, pages 188–205, 2014.

[BGG$^+$17] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2017, 2017.

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

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001.

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

[BHJ+18] Boaz Barak, Samuel Hopkins, Aayush Jain, Pravesh Kothari, and Amit Sahai. Sum-of-squares meets program obfuscation, revisited. *Unpublished Work*, 2018.

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

[BNPW16] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. Cryptology ePrint Archive, Report 2016/558, 2016. http://eprint.iacr.org/2016/558.

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

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

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

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

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

[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 *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 476–493, 2013.

[CLT15] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In *CRYPTO*, 2015.

[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.

[Fre10] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *EUROCRYPT*, pages 44–61, 2010.

[GGG+14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In *EUROCRYPT*, 2014.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013.

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

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *TCC*, pages 498–527, 2015.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 74–94, 2014.

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In *CRYPTO*, 2016.

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

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

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

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

[HJK+16] Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. How to generate and use universal samplers. In *ASIACRYPT*, pages 715–744, 2016.

[Hol06] Thomas Holenstein. *Strengthening key agreement using hard-core sets.* PhD thesis, ETH Zurich, 2006.

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

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

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

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

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

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

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 28–57. Springer, 2016.

[Lin17a] Huijia Lin. Indistinguishability obfuscation from sxdh on 5-linear maps and locality-5 prgs. In *CRYPTO*, pages 599–629. Springer, 2017.

[Lin17b] Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 prgs. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 599–629, 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 bilinear maps and block-wise local prgs. Cryptology ePrint Archive, Report 2017/250, 2017. http://eprint.iacr.org/2017/250.

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

[LV17] Alex Lombardi and Vinod Vaikuntanathan. On the non-existence of blockwise 2-local prgs with applications to indistinguishability obfuscation. *IACR Cryptology ePrint Archive*, 2017:301, 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 *EUROCRYPT*, pages 700–718, 2012.

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

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

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

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 500–517, 2014.

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

[RTTV08] Omer Reingold, Luca Trevisan, Madhur Tulsiani, and Salil P. Vadhan. Dense subsets of pseudorandom sets. In *FOCS*, pages 76–85, 2008.

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

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

A Sub-linear Functional Encryption for Circuits

In this section, we recall the notion of a sublinear secret key Functional Encryption which is known to imply indistinguishability obfuscation [BNPW16], and was considered in many works, most recently in [Lin17a, Lin17b, LT17, AS17].

Syntax. A sublinear secret key functional encryption FE for a message space $\chi = \{\chi_\lambda\}_{\lambda \in \mathbb{N}}$ and a function space $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ is a tuple of PPT algorithms with the following properties:

- **Setup,** $\mathsf{Setup}(1^\lambda)$: On input security parameter λ, it outputs the master secret key MSK.
- **Encryption,** $\mathsf{Enc}(\mathsf{MSK}, x)$: On input the encryption key MSK and a message $x \in \chi_\lambda$, it outputs ciphertext CT.

- **Key Generation,** $\mathsf{KeyGen}(\mathsf{MSK}, C)$: On input the master secret key MSK and a function $C \in \mathcal{C}_\lambda$, it outputs a functional key sk_C.
- **Decryption,** $\mathsf{Dec}(sk_C, \mathsf{CT})$: On input functional key sk_C and a ciphertext CT, it outputs the result out.

We define correctness property below.

Correctness. Consider any function $C \in \mathcal{C}_\lambda$ and any plaintext $x \in \chi_\lambda$. Consider the following process:

- $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$
- $sk_C \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, C)$.
- $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, x)$

The following should hold:

$$\mathsf{Pr}\left[\mathsf{Dec}(sk_C, \mathsf{CT}) = C(x)\right] \geq 1 - \mathsf{negl}(\lambda),$$

for some negligible function negl.

Sub-Linear Efficiency: We require that for any message $\boldsymbol{x} \in \chi_\lambda$ the following holds:

- Let $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$.
- Compute $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, x)$.

The size of circuit computing CT is less than $\ell^{1-\epsilon_C} \cdot poly(\lambda, |x|)$. $poly$ is some fixed polynomial, $\epsilon_C > 0$ is some constant, $|x|$ is the length of the message x and $\ell = max\{size(C)\}_{C \in \mathcal{C}_\lambda}$.

Now we define the notion of $(\mathsf{adv}, \mathsf{size}_\mathcal{A})-$ security. Here adv is a parameter denoting advantage of adversary and $\mathsf{size}_\mathcal{A}$ is the parameter denoting the size of the adversary.

Security Definition

Definition 21 (Indistinguishability of Ciphertexts)**.** *For a sublinear secret key FE scheme* FE *for a class of functions* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$*, the* $(\mathsf{adv}, \mathsf{size}_\mathcal{A})-$ **indistinguishability of semi-functional ciphertexts property** *is associated with two experiments. The experiments are parameterised with* $\mathsf{aux} = (1^\lambda, \Gamma, M_i = \{x_i\}_{i \in \Gamma}, M_0^*, M_1^*, C)$ *where* $C(M_0^*) = C(M_1^*)$.
<u>$\mathsf{Expt}_{\mathsf{aux}}(1^\lambda, \mathbf{b})$</u>*:*

1. *Compute* $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. *Compute* $sk_C \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, C)$.
3. $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{MSK}, M_i)$*, for every* $i \in [\Gamma]$.
4. *If* $\mathbf{b} = 0$*, compute* $\mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{MSK}, M_0^*)$.

5. *If* $\mathbf{b} = 1$ *compute* $\mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{MSK}, M_0^*)$.

6. *Output the following:*

   (a) CT_i *for* $i \in \Gamma$ *and* CT^*.

   (b) sk_C.

   (c) M_0^*, M_1^* *and* $\{M_i\}_{i\in\Gamma}$

   (d) C

A semi-functional FE scheme FE *associated with plaintext space* χ *is said to satisfy* ($\mathsf{size}_{\mathcal{A}}$, adv)**-indistinguishability of semi-functional ciphertexts property** *if the following happens:* $\forall\lambda > \lambda_0$, *any polynomial* Γ, *messages* $\{M_i\}_{i\in\Gamma} \in \chi_\lambda$, $M_0^*, M_1^* \in \chi_\lambda$, $C \in \mathcal{C}_\lambda$ *and any adversary* $\mathcal{A}$ *of size* $\mathsf{size}_{\mathcal{A}}$*:*

$$|\Pr[\mathcal{A}(\mathsf{Expt}_{\mathsf{aux}}(1^\lambda, 0) = 1] - \Pr[\mathcal{A}(\mathsf{Expt}_{\mathsf{aux}}(1^\lambda, 1)) = 1]| \leq \mathsf{adv}$$

where $\mathsf{aux} = (1^\lambda, \Gamma, M_i = \{x_i\}_{i\in\Gamma}, M_0^*, M_1^*, C)$

A.1 Equivalence of Semi-Functional FE and Sublinear FE

In this section, we show that any sublinear secret key FE with $(\mathsf{size}, \mathsf{adv})-$security for circuit class $\mathcal{C}_{n,s}$ imply a $(\mathsf{size}, \mathsf{adv})$ secure sublinear semi-functional secret key FE for $\mathcal{C}_{n',s'}$, assuming additionally a secret key encryption scheme with $(\mathsf{size}+poly(\lambda), \mathsf{negl}(\lambda))$ security for some negligible negl and polynomial $poly$. Here $n = n'poly(\lambda)$ and $s = s'poly(\lambda)$ for some polynomial $poly(\lambda)$.

Theorem 24. *Assuming:*

- *Sublinear secret key FE with* $(\mathsf{size}, \mathsf{adv})-$*security for circuit class* $\mathcal{C}_{n,s}$

- *Secret key encryption scheme with* $(\mathsf{size} + poly_1(\lambda), \mathsf{negl})$ *security*[9]*. Here* $poly_1(\lambda)$ *is some fixed polynomial in* λ

There exists $(\mathsf{size}, \mathsf{adv})$ *secure sublinear semi-functional secret key FE scheme for circuit class* $\mathcal{C}_{n',s'}$*. Here* $n = n'poly(\lambda)$ *and* $s = s'poly(\lambda)$ *for some fixed polynomial* $poly(\lambda)$ *independent of* n, s*.*

We now describe our construction of our semi-functional FE scheme sFE from sublinear FE FE. Here are our algorithms. Let E denote the secret key encryption scheme. We describe the the construction below.

- $\underline{\mathsf{Setup}(1^\lambda):}$

  1. Run $\mathsf{E.Setup}(1^\lambda) \rightarrow \mathsf{Esk}$
  2. Run $\mathsf{FE.Setup}(1^\lambda) \rightarrow sk$.
  3. Output $\mathsf{MSK} = (sk, \mathsf{Esk})$

- $\underline{\mathsf{Enc}(\mathsf{MSK}, \mathsf{m}):}$

  1. Parse $\mathsf{MSK} = (sk, \mathsf{Esk})$.

[9] For sub-exponential security we need negl to be sub-exponentially small

2. Compute $\mathsf{CT} = \mathsf{FE.Enc}(sk, (m, 0, 0^{|\mathsf{Esk}|}))$.
3. Output CT.

- $\underline{\mathsf{KeyGen}(\mathsf{MSK}, \mathsf{C}):}$

1. Parse $\mathsf{MSK} = (sk, \mathsf{Esk})$.
2. Compute $\mathsf{Ect} \leftarrow \mathsf{E.Enc}(\mathsf{Esk}, 0^{\ell_{out}})$. Here, ℓ_{out} is the output length of C.
3. Let G be the circuit described in Figure 4. Compute $sk_C \leftarrow \mathsf{FE.KeyGen}(sk, G)$.
4. Output sk_C.

- $\underline{\mathsf{Dec}(\mathsf{sk_C}, \mathsf{CT}):}$

1. Output $m^* = \mathsf{FE.Dec}(sk_C, \mathsf{CT})$.

G

Input: Message m, flag b and a string $x \in \{0,1\}^{|\mathsf{Esk}|}$
Hardwired: Secret key encryption ciphertext Ect.

- If $b = 1$, output $\mathsf{E.Dec}(x, \mathsf{Ect})$.
- Otherwise output $C(m)$

Figure 4: Description of the Circuit G.

Correctness: Correctness is immediate from the description of the scheme assuming the underlying schemes satisfy correctness.

Sub-linearity: If C takes inputs of size n and is of size s, the size of G is $s \cdot poly_1(\lambda)$ (as it just computes $\mathsf{E.Dec}$ over a hardwired output in addition to computing C). Also note that input length of G is $n' = |m| + |\mathsf{Esk}| + 1$. If $|m| = n$, then $n' = n + poly_2(\lambda)$. Here both polynomials are independent of n, s thus we prove the claim.

Indistinguishability of Semi-functional key We now describe the semi-functional key generation algorithm
$\underline{\mathsf{sfKG}(\mathsf{MSK}, \mathsf{C}, \theta):}$

1. Parse $\mathsf{MSK} = (sk, \mathsf{Esk})$.
2. [**Change**] Compute $\mathsf{Ect} \leftarrow \mathsf{E.Enc}(\mathsf{Esk}, \theta)$. This is the change from the honest encryption algorithm.
3. Let G be the circuit described in Figure 4. Compute $sk_C \leftarrow \mathsf{FE.KeyGen}(sk, G)$.
4. Output sk_C.

Note that in the game for indistinguishability of semi-functional keys, ciphertexts are honestly generated and hence Esk is never used in the clear. Thus, the security follows from the security of the encryption scheme.

Indistinguishability of Semi-functional Ciphertexts We now describe the semi-functional encryption algorithm. The change from the honest encryption algorithm is marked below.

$\underline{\mathsf{sfEnc}(\mathsf{MSK}, 1^\lambda):}$

1. Parse $\mathsf{MSK} = (sk, \mathsf{Esk})$.
2. [**Change**] Compute $\mathsf{CT}_{sf} = \mathsf{FE.Enc}(sk, (0, 1, \mathsf{Esk}))$.
3. Output CT_{sf}.

Note that in the game for indistinguishability of semi-functional ciphertexts the value θ hardwired in the functional key is set to be $C(m^*)$ where m^* is the challenge message. Thus, the security follows from the security of FE scheme.