New Methods for Indistinguishability Obfuscation: Bootstrapping and Instantiation

Shweta Agrawal[*]

Abstract

Constructing indistinguishability obfuscation (iO) [BGI$^+$01] is a central open question in cryptography. We provide new methods to make progress towards this goal. Our contributions may be summarized as follows:

1. **Bootstrapping.** In a recent work, Lin and Tessaro [LT17] (LT) show that iO may be constructed using i) Functional Encryption (FE) for polynomials of degree L, ii) Pseudorandom Generators (PRG) with *blockwise locality* L and polynomial expansion, and iii) Learning With Errors (LWE). Since there exist constructions of FE for quadratic polynomials from standard assumptions on bilinear maps [Lin17, BCFG17], the ideal scenario would be to set $L = 2$, yielding iO from widely believed assumptions.

   Unfortunately, it was shown soon after [LV17, BBKK17] that PRG with block locality 2 and the expansion factor required by the LT construction, concretely $\Omega(n \cdot 2^{b(3+\epsilon)})$, where n is the input length and b is the block length, do not exist. In the worst case, these lower bounds rule out 2-block local PRG with stretch $\Omega(n \cdot 2^{b(2+\epsilon)})$. While [LV17, BBKK17] provided strong negative evidence for constructing iO based on bilinear maps, they could not rule out the possibility completely; a tantalizing gap has remained. Given the current state of lower bounds, the existence of 2 block local PRG with expansion factor $\Omega(n \cdot 2^{b(1+\epsilon)})$ remains open, although this stretch does not suffice for the LT bootstrapping, and is hence unclear to be relevant for iO.

   In this work, we improve the state of affairs as follows.

   (a) *Weakening requirements on Boolean PRGs:* In this work, we show that the narrow window of expansion factors left open by lower bounds *do* suffice for iO. We show a new method to construct FE for NC_1 from i) FE for degree L polynomials, ii) PRGs of block locality L and expansion factor $\tilde{\Omega}(n \cdot 2^{b(1+\epsilon)})$, and iii) LWE (or RLWE). Our method of bootstrapping is completely different from all known methods and does not go via randomizing polynomials.

   This re-opens the possibility of realizing iO *from* 2 *block local* PRG*,* SXDH *on Bilinear maps and* LWE*.*

   (b) *Broadening class of sufficient randomness generators*: Our bootstrapping theorem may be instantiated with a broader class of pseudorandom generators than hitherto considered for iO, and may circumvent lower bounds known for the arithmetic degree

[*]IIT Madras, India. Email: `shweta.a@cse.iitm.ac.in`.

of iO-sufficient PRGs [LV17, BBKK17]; in particular, these may admit instantiations with arithmetic degree 2, yielding iO with the additional assumptions of SXDH on Bilinear maps and LWE. In more detail, we may use the following two classes of PRG:

i. *Non-Boolean PRGs*: We may use pseudorandom generators whose inputs and outputs need not be Boolean but may be integers restricted to a small (polynomial) range. Additionally, the outputs are not required to be pseudorandom but must only satisfy a milder indistinguishability property[1]. We tentatively propose initializing these PRGs using the multivariate quadratic assumption MQ which has been widely studied in the literature [MI88, Wol05, DY09] and against the general case of which, no efficient attacks are known.

ii. *Correlated Noise Generators*: We introduce an even weaker class of pseudorandom generators, which we call correlated noise generators (CNG) which may not only be non-Boolean but are required to satisfy an even milder (seeming) indistinguishability property.

(c) *Assumptions and Efficiency.* Our bootstrapping theorems can be based on the hardness of the Learning With Errors problem or its ring variant (LWE/RLWE) and can compile FE for degree L polynomials directly to FE for NC_1. Previous work compiles FE for degree L polynomials to FE for NC_0 to FE for NC_1 to iO [LV16, Lin17, AS17, GGH$^+$13c]. Our method for bootstrapping to NC_1 does not go via randomized encodings as in previous works, which makes it simpler and more efficient than in previous works.

2. **Instantiating Primitives.** In this work, we provide the first direct candidate of FE for constant degree polynomials from new assumptions on lattices. Our construction is new and does not go via multilinear maps or graded encoding schemes as all previous constructions. In more detail, let $\mathcal{F}$ be the class of circuits with depth d and output length ℓ. Then, for any $f \in \mathcal{F}$, our scheme achieves $\mathsf{Time}(\mathsf{KeyGen}) = O\big(\mathrm{poly}(\kappa, |f|)\big)$, and $\mathsf{Time}(\mathsf{Enc}) = O(|\mathbf{x}| \cdot 2^d \cdot \mathrm{poly}(\kappa))$ where κ is the security parameter. This suffices to instantiate the bootstrapping step above. Our construction is based on the ring learning with errors assumption (RLWE) as well as new untested assumptions on NTRU rings.

   We provide a detailed security analysis and discuss why previously known attacks in the context of multilinear maps, especially zeroizing attacks and annihilation attacks, do not appear to apply to our setting. We caution that the assumptions underlying our construction must be subject to rigorous cryptanalysis before any confidence can be gained in their security. However, their significant departure from known multilinear map based constructions make them, we feel, a potentially fruitful new direction to explore. Additionally, being based entirely on lattices, we believe that security against classical attacks will likely imply security against quantum attacks.

[1]For the knowledgeable reader, we do not require the polynomials computing our PRGs to be sparse and hence the general attack of [BBKK17] does not rule out existence of degree 2 instantiations to the best of our knowledge.

Contents

1 Introduction

Indistinguishability Obfuscation. Program obfuscation aims to make a program "unintelligible" while preserving its functionality. Indistinguishability obfuscation [BGI$^+$01] (iO) is a flavour of obfuscation, which converts a circuit C to an obfuscated circuit $\mathcal{O}(C)$ such that any two circuits that have the same size and compute the same function are indistinguishable to a computationally bounded adversary.

While it is non-obvious at first glance what this notion is useful for, recent work has demonstrated the tremendous power of iO. iO can be used to construct almost any cryptographic object that one may desire – ranging (non-exhaustively) from classical primitives such as one way functions [KMN$^+$14], trapdoor permutations [BPW16], public key encryption [SW14] to deinable encryption [SW14], fully homomorphic encryption [CLTV15], functional encryption [GGH$^+$13c], succinct garbling schemes [CHJV15, BGL$^+$15, KLW15, LPST16] and many more.

The breakthrough work of Garg et al. [GGH$^+$13c] presented the first candidate construction of iO from the beautiful machinery of graded encoding schemes [GGH13a]. This work heralded substantial research effort towards understanding iO: from cryptanalysis to new constructions to understanding and weakening underlying assumptions to applications. On the cryptanalysis front, unfortunately, all known candidate graded encoding schemes [GGH13a, CLT13, GGH15] as well as several candidates of iO have been broken [CHL$^+$15, CGH$^+$15, HJ15, CJL, CFL$^+$, MSZ16, CLLT16, ADGM16]. Given the power of iO, a central question in cryptography is to construct iO from better understood hardness assumptions.

Functional Encryption. Functional encryption (FE) [SW05, SW] is a generalization of public key encryption in which secret keys correspond to programs rather than users. In more detail, a secret key embeds inside it a circuit, say f, so that given a secret key SK_f and ciphertext $\mathsf{CT}_{\mathbf{x}}$ encrypting a message $\mathbf{x}$, the user may run the decryption procedure to learn the value $f(\mathbf{x})$. Security of the system guarantees that nothing beyond $f(\mathbf{x})$ can be learned from $\mathsf{CT}_{\mathbf{x}}$ and SK_f. Recent years have witnessed significant progress towards constructing functional encryption for advanced functionalities, even from standard assumptions [BF01, Coc01, BW06, BW07, GPV08, CHKP10, ABB10, GPSW06, BSW07, KSW08, LOS$^+$10, AFV11, Wat12, GVW13, GGH$^+$13d, GGH$^+$13c, GVW15]. However, most constructions supporting general functionalities severely restrict the attacker in the security game: she must request only a bounded number of keys [GVW12, AR17, GKP$^+$13], or may request unbounded number of keys but from a restricted space[2] [GVW15, Agr17]. Schemes that may be proven secure against a general adversary are restricted to compute linear or quadratic functions [ABCP15, ALS16, Lin17, BCFG17].

Constructing iO from FE. Recent work [AJ15, BV15, AJS15] provided an approach for constructing iO via FE. While we do not have any candidate constructions for FE that satisfy the security and efficiency requirements for constructing iO (except constructions that themselves rely on graded encoding schemes or iO [GGH$^+$13c, GGHZ14]), FE is a primitive that is closer to what cryptographers know to construct and brings iO nearer the realm of reachable cryptography.

An elegant sequence of works [Lin16, LV16, Lin17, AS17, LT17] has attempted to shrink the functionality of FE that suffices for iO, and construct FE for this minimal functionality from graded

[2]Referred to in the literature as "predicate encryption"

encoding schemes or multilinear maps. Concretely, the question is: what is the smallest L such that FE supporting polynomials of degree L suffices for constructing iO? At a high level, these works follow a two step approach described below:

1. **Bootstrapping FE to iO.** The so called "bootstrapping" theorems have shown that general purpose iO can be built from one of the following: i) sub-exponentially secure FE for NC_1 [AJ15, BV15, AJS15, BNPW16], or ii) sub-exponentially secure FE for NC_0 and PRG in NC_0 [LV16] iii) PRGs with locality L and FE for computing degree L polynomials [Lin17] or iv) PRGs with *blockwise* locality L and FE for computing degree L polynomials [LT17].

   At a high level, all bootstrapping theorems make use of *randomized encodings* [IK00, AIK06] to reduce computation of a polynomial sized circuit to computation of low degree polynomials.

2. **Instantiating Primitives.** Construct FE supporting degree L polynomials based on graded encodings or multilinear maps [LV16, Lin17].

1.1 Bootstrapping, the Ideal.

Since we have candidates of FE for quadratic polynomials from standard assumptions on bilinear maps [Lin17, BCFG17], a dream along this line of work would be to reduce the degree required to be supported by FE all the way down to 2, yielding iO, from bilinear maps and other widely believed assumptions (like LWE and PRG with constant locality). The recent work of Lin and Tessaro [LT17] (LT) came closest to achieving this, by leveraging a new notion of PRG they termed *blockwise local* PRG. A PRG has blockwise locality L and block-size b, if when viewing the input seed as a matrix of b rows and n columns, every output bit depends on input bits in at most L columns. As mentioned above, they showed that PRGs with blockwise locality L and certain polynomial stretch, along with FE for computing degree L polynomials and LWE suffice for iO.

Unfortunately, it was shown soon after [LV17, BBKK17] that PRG with block locality 2 and the stretch required by the LT construction, concretely $\Omega(n \cdot 2^{b(3+\epsilon)})$, do not exist. In the worst case, these lower bounds rule out 2-block local PRG with stretch $\Omega(n \cdot 2^{b(2+\epsilon)})$. On the other hand, these works suggest that 3 block local PRG are likely to exist, thus shrinking the iO-sufficient degree requirement on FE to 3.

While [LV17, BBKK17] provided strong negative evidence for constructing iO based on 2 block local PRG and hence bilinear maps, they could not rule out the possibility completely; a tantalizing gap has remained. Roughly speaking, the construction of candidate PRG (first suggested by Goldreich [Gol00]) must choose a hyper-graph with variables on vertices, then choose predicates that are placed on each hyper-edge of the graph and output the values of the edge-predicates on the vertex-variables. The lower bounds provided by [LV17, BBKK17] vary depending on the choices made in the above construction. The following table by [LV17] summarises our current understanding on the existence of 2 block local PRG:

Stretch	Worst case versus Random Predicate	Worst case versus Random Graph	Different versus Same Predicate per output bit	Reference
$\tilde{\Omega}(n \cdot 2^{b(1+\epsilon)})$	Random	Random	Different	[BBKK17]
$\tilde{\Omega}(n \cdot 2^{b(2+\epsilon)})$	Worst Case	Worst Case	Different	[BBKK17]
$\tilde{\Omega}(n \cdot 2^{b(1+\epsilon)})$	Worst Case	Worst Case	Same	[LV17]
$\tilde{\Omega}(n \cdot 2^{b(1+\epsilon)})$	Worst Case	Worst Case	Different	Open

As we see in the above table, the existence of 2 block local PRG with carefully chosen graph and predicates with stretch $\tilde{\Omega}(n \cdot 2^{b(1+\epsilon)})$ is open. However, even in the case that these exist, it is not clear whether its even useful, since the Lin-Tessaro compiler requires larger stretch $\Omega(n \cdot 2^{b(3+\epsilon)})$, which is ruled out by row 2 above. In the current version of their paper, [LT17] remark that "Strictly speaking, our results leave a narrow window of expansion factors open where block-wise PRGs could exist, but we are not aware whether our approach could be modified to use such low-stretch PRGs."

Bootstrapping: Our Results. In this work, we show that the narrow window of expansion factors left open by lower bounds *do* suffice for iO. Moreover, we define a larger class of pseudorandomness generators than those considered so far, which may admit lower degree instantiations. We then show that these generators with the same expansion suffice for iO. We discuss each of these contributions below.

Weakening requirements on PRGs: We show a new method to construct FE for NC_1 from FE for degree L polynomials, sub-exponentially secure PRGs of block locality L and LWE (or RLWE). Since FE for NC_1 implies iO for P/Poly [AJ15, BV15, BNPW16], this suffices for bootstrapping all the way to iO for P/Poly. Our transformation requires the PRG to only have expansion $n \cdot 2^{b(1+\epsilon)}$ which is not ruled out as discussed above. This re-opens the possibility of realizing 2 block local PRG with our desired expansion factor (see below for a detailed discussion), which would imply iO from 2 block local PRG, SXDH on Bilinear maps and LWE. A summary of the state of art in PRG based bootstrapping is provided in Figure 1.1.

Broadening class of sufficient PRGs: Our bootstrapping theorem may be instantiated with a broader class of pseudorandom generators than hitherto considered for iO, and may circumvent lower bounds known for the arithmetic degree of iO-sufficient PRGs [LV17, BBKK17]; in particular, these may admit instantiations with arithmetic degree 2, yielding iO along with the additional assumptions of SXDH on Bilinear maps and LWE. In more detail, we may use the following two classes of PRG:

1. *Non-Boolean PRGs*: We may use pseudorandom generators whose inputs and outputs need not be Boolean but may be integers restricted to a small (polynomial) range. Additionally, the outputs are not required to be pseudorandom but must only satisfy a milder indistinguishability property[3]. We tentatively propose initializing these PRGs using the multivariate quadratic assumption MQ which has been widely studied in the literature [MI88, Wol05, DY09] and against the general case of which, no efficient attacks are known.

[3]For the knowledgeable reader, we do not require the polynomials computing our PRGs to be sparse and hence the general attack of [BBKK17] does not rule out existence of degree 2 instantiations to the best of our knowledge.

2. *Correlated Noise Generators*: We introduce an even weaker class of pseudorandom generators, which we call correlated noise generators (CNG) which may not only be non-Boolean but are required to satisfy an even milder (seeming) indistinguishability property.

Assumptions and Efficiency. Our bootstrapping theorems can be based on the hardness of LWE or its ring variant RLWE and compiles FE for degree L polynomials directly to FE for NC_1. Our method for bootstrapping to NC_1 does not go via randomized encodings as in previous works. Saving the transformation to randomized encodings makes bootstrapping to NC_1 more efficient than in previous works. For instance, [LT17] require Q PRG seeds, where Q is the (polynomial) length of random tapes needed by the randomized encodings. On the other hand we only need 2 PRG seeds, since we avoid using randomized encodings, yielding a ciphertext that is shorter by a factor of Q, as well as (significantly) simpler pre-processing.

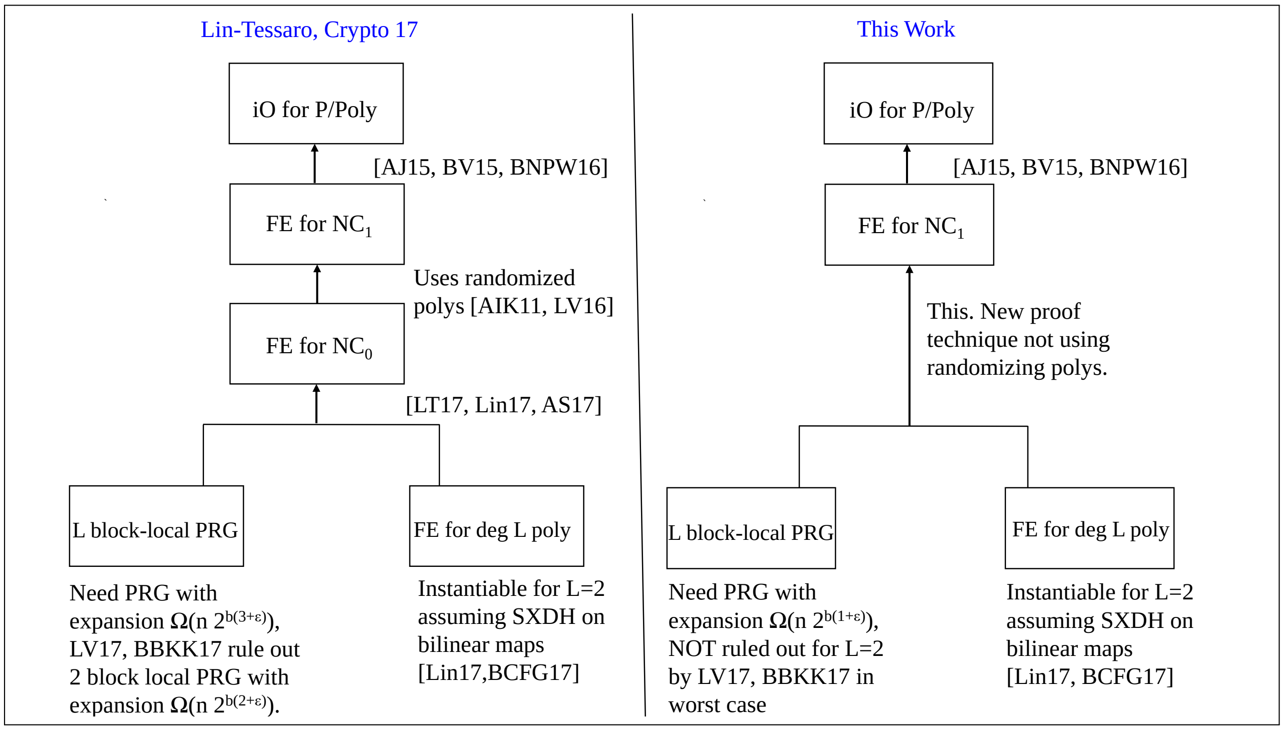
Lin-Tessaro, Crypto 17
iO for P/Poly
[AJ15, BV15, BNPW16]
FE for NC1
Uses randomized polys [AIK11, LV16]
FE for NC0
[LT17, Lin17, AS17]
L block-local PRG
FE for deg L poly
Need PRG with expansion Ω(n 2^b(3+ε)), LV17, BBKK17 rule out 2 block local PRG with expansion Ω(n 2^b(2+ε)).
Instantiable for L=2 assuming SXDH on bilinear maps [Lin17,BCFG17]
This Work
iO for P/Poly
[AJ15, BV15, BNPW16]
FE for NC1
This. New proof technique not using randomizing polys.
L block-local PRG
FE for deg L poly
Need PRG with expansion Ω(n 2^b(1+ε)), NOT ruled out for L=2 by LV17, BBKK17 in worst case
Instantiable for L=2 assuming SXDH on bilinear maps [Lin17, BCFG17]

Figure 1.1: State of the Art in Bootstrapping FE to iO. In the present work, we may bootstrap directly to FE for NC_1 without going through NC_0.

1.2 Instantiation: the Ideal.

To instantiate iO via FE for constant degree polynomials, [LT17] rely on the FE for degree L polynomials constructed by Lin [Lin17], which relies on SXDH on noiseless algebraic multilinear maps of degree L, for which no candidates of degree greater than 2 are known to exist. As discussed by [Lin17], instantiating her construction with noisy multilinear maps causes the proof to fail, in addition to the SXDH assumption itself being false on existing noisy multilinear map candidates. We refer the reader to [LT17, Lin17] for a detailed discussion.

Evidently, one ideal instantiation for iO would be to construct noiseless multilinear maps of degree at least 3[4], on which the SXDH assumption is believed to hold. At the moment, we have no evidence that such objects exist. Another ideal instantiation would be to provide a direct construction of FE for constant degree polynomials from well understood hardness assumptions, satisfying the requisite compactness properties for implication to iO. Constructing FE from well-understood hardness assumptions has received significant attention in recent years, and for the moment we do not have any constructions that suffice for iO excepting those that themselves rely on multilinear maps or iO.

Thus, at present, all concrete instantiations of the FE to iO compiler must go via noisy multilinear maps on which SXDH fails.

Instantiation: Our Results. In our work, we take a different approach to the question of instantiation. We propose to construct FE *directly*, without going through multilinear maps or graded encoding schemes, and use this FE to instantiate the transformation to iO. We believe this new approach has the following advantages:

1. **May be Simpler**: Construction of iO-sufficient FE might not need the full power of asymmetric multilinear maps, since FE is not known to imply asymmetric multilinear maps equipped with SXDH to the best of our knowledge [5]. Hence, constructing FE directly may be simpler.

2. **Yield new and possibly more robust assumptions:** Attempts to construct FE directly for low degree polynomials yield new hardness assumptions which are likely different from current assumptions on noisy multilinear maps. This direction may yield more resilient candidates than those that go via multilinear maps.

In this work, we provide the first direct candidate of symmetric key FE for constant degree polynomials from new assumptions on lattices. Let $\mathcal{F}$ be the class of circuits with depth d and output length ℓ. Then, for any $f \in \mathcal{F}$, our scheme achieves $\mathsf{Time}(\mathsf{KeyGen}) = O\big(\operatorname{poly}(\kappa, |f|)\big)$, and $\mathsf{Time}(\mathsf{Enc}) = O(|\mathbf{x}| \cdot 2^d \cdot \operatorname{poly}(\kappa))$ where κ is the security parameter. This suffices to instantiate the bootstrapping step above. Our construction is based on the ring learning with errors assumption (RLWE) as well as new untested assumptions on NTRU rings. We provide a detailed security analysis with necessary and sufficient conditions for security. In particular, we provide a proof based on a new assumption in a simplified variant of our scheme; we view this as a first step to provable security. Please see Section 8 for details. The assumptions underlying our construction must be subject to rigorous cryptanalysis before any confidence can be gained in their security. However, even if our particular attempt proves insecure, we believe there is significant value in taking this route and hope it inspires other candidates.

1.3 Our Techniques: Bootstrapping

In this section, we assume familiarity of the reader with RLWE and Regev's public key encryption scheme [Reg09, GPV08]. Please see Section 2 for a refresher. Although our bootstrapping can also be based on standard LWE, we describe it using RLWE here since it is simpler.

[4]Ideally degree 5, so as to remove the reliance on even blockwise local PRG and rely directly on 5 local PRG which are better understood.

[5]A line of work can traverse the route of FE to iO to PiO (probabilistic iO) to *symmetric* multilinear maps (see [FHHL18] and references therein) using multiple complex subexponential reductions, still not yielding *asymmetric* multilinear maps with SXDH.

Ciphertext and Public Key Evaluation by [AR17]. The starting point of our work is the bounded collusion FE constructed recently by Agrawal and Rosen [AR17]. In this work, the authors design a new encryption algorithm and a new ciphertext evaluation algorithm $\mathsf{Eval}_{\mathsf{CT}}$ so that the decryptor, given the encoding of some input $\mathbf{x}$ and some (arithmetic) circuit $f \in \mathsf{NC}_1$ can execute $\mathsf{Eval}_{\mathsf{CT}}$ to compute a functional ciphertext that encodes $f(\mathbf{x})$, *obliviously* of $\mathbf{x}$.

In more detail, if $\mathsf{CT}_{(f(\mathbf{x}))} = \mathsf{Eval}_{\mathsf{CT}}(\underset{i\in[d]}{\cup} \mathcal{C}^i, f)$ where d is the depth of the circuit and $\underset{i\in[d]}{\cup} \mathcal{C}^i$ is a set of encodings provided by the encryptor, then the functional ciphertext has the following structure:

$$\mathsf{CT}_{(f(\mathbf{x}))} = \langle \mathsf{Lin}_f,\ \mathcal{C}^d \rangle + \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1})$$

for some f-dependent linear function Lin_f and polynomial Poly_f. Here, Lin_f and Poly_f may be computed by a corresponding public key evaluation algorithm, denoted by $\mathsf{Eval}_{\mathsf{PK}}$, given only the public key and function f. Moreover, upon decrypting $\mathsf{CT}_{(f(\mathbf{x}))}$, we get

$$f(\mathbf{x}) + \mathsf{noise}_{f(\mathbf{x})} = \langle \mathsf{Lin}_f,\ \mathcal{M}^d \rangle + \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) \tag{1.1}$$

where $\mathcal{M}^d$ is the message vector encoded in level d encodings $\mathcal{C}^d$. Here, $f(\mathbf{x}) \in R_{p_0}$ for some ring R_{p_0} and $\mathsf{noise}_{f(\mathbf{x})}$ is a noise term which may be modded out using standard techniques to recover $f(\mathbf{x})$ as desired.

Given the above algorithms, an approach to compute $f(\mathbf{x})$ is to leverage functional encryption for linear functions [ABCP15, ALS16], denoted by LinFE. Recall the functionality of LinFE: the encryptor provides a ciphertext $\mathsf{CT}_{\mathbf{z}}$ for some vector $\mathbf{z} \in R^n$, the key generator provides a key $\mathsf{SK}_{\mathbf{v}}$ for some vector $\mathbf{v} \in R^n$ and the decryptor learns $\langle \mathbf{z}, \mathbf{v} \rangle \in R$. Thus, we may use LinFE to enable the decryptor to compute $\langle \mathsf{Lin}_f,\ \mathcal{M}^d \rangle$, let the decryptor compute $\mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1})$ herself to recover $f(\mathbf{x}) + \mathsf{noise}_{f(\mathbf{x})}$.

Unfortunately, this approach is insecure as is, as discussed in [AR17]. For bounded collusion FE, the authors achieve security by having the encryptor encode a fresh, large noise term $\mathsf{noise}_{\mathsf{fld}}$ for each requested key f which "floods" $\mathsf{noise}_{f(\mathbf{x})}$. This noise is forcibly added to the decryption equation so that the decryptor recovers $f(\mathbf{x}) + \mathsf{noise}_{f(\mathbf{x})} + \mathsf{noise}_{\mathsf{fld}}$, which by design is statistically indistinguishable from $f(\mathbf{x}) + \mathsf{noise}_{\mathsf{fld}}$. [AR17] show that with this modification the scheme can be shown to achieve strong simulation style security, by relying just on security of LinFE. However, encoding a fresh noise term per key causes the ciphertext size to grow at least linearly with the number of function keys requested, or in the case of single key FE, with the output length of the function. Note that to suffice for iO, we are required to construct this scheme with ciphertext size sublinear in the output length of the function, say ℓ [AJ15, BV15]. Thus, the ciphertext size of [AR17] violates the efficiency required from an FE scheme to be sufficient for iO.

Noisy Linear Functional Encryption. In this work, we show that the approach of [AR17] can be extended to construct a single key FE for NC_1 with ciphertext size sublinear in the output length, by replacing linear functional encryption LinFE with *noisy linear functional encryption*, denoted by NLinFE. Noisy linear functional encryption is like like regular linear functional encryption [ABCP15, ALS16], except that the function value is recovered only up to some bounded additive error/noise, and indistinguishability holds even if the challenge messages evaluated on all the function keys are only "approximately" and not exactly equal. The functionality of NLinFE is as follows: given a ciphertext $\mathsf{CT}_{\mathbf{z}}$ which encodes vector $\mathbf{z} \in R^n$ and a secret key $\mathsf{SK}_{\mathbf{v}}$ which encodes vector $\mathbf{v} \in R^n$, the decryptor recovers $\langle \mathbf{z},\ \mathbf{v} \rangle + \mathsf{noise}_{\mathbf{z},\mathbf{v}}$ where $\mathsf{noise}_{\mathbf{z},\mathbf{v}}$ is specific to the message and function being evaluated.

Let $f \in \mathsf{NC}_1$ and let the output of f be of size ℓ. Let $f_1, \ldots, f_\ell$ be the functions that output the i^{th} bit of f for $i \in [\ell]$. At a high level, our FE for NC_1 will enable the decryptor to compute $\langle \mathsf{Lin}_{f_i},\ \mathcal{M}^d \rangle + \mathsf{noise}_{\mathsf{fld}\,i}$ as in [AR17] but instead of having the encryptor encode ℓ noise terms during encryption, we use NLinFE to add noise terms $\mathsf{noise}_{\mathsf{fld}\,i}$ into the decryption equation. In more detail, the encryptor must provide encodings $\underset{i \in [d-1]}{\cup} \mathcal{C}^i$ as well as an NLinFE encryption of the level d message encodings $\mathcal{M}^d$ (please see Equation 1.1). The key generator provides an NLinFE key for Lin_f so that the decryptor may compute $\mathsf{Poly}_{f_i}(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) + \langle \mathsf{Lin}_{f_i},\ \mathcal{M}^d \rangle + \mathsf{noise}_{\mathsf{fld}\,i}$ as desired.

Noisy linear functional encryption provides the right abstraction (in our opinion) for the smallest functionality that may be bootstrapped to FE for NC_1 using our methods. NLinFE captures the precise requirements on noise that is required for the security of the construction and integrates seamlessly with our new proof technique. Moreover, NLinFE may be constructed in different ways from different assumptions – we provide three constructions from various assumptions (more on this below), and properties such as ciphertext size and collusion resistance achieved by NLinFE are inherited by FE for NC_1. Please see Sections 4 and Section 6 for details.

We remark that our construction crucially uses a powerful property of the ciphertext evaluation algorithm of [AR17], namely that computing a *linear* function on secret values plus a noise term suffices, along with additional public computation, to compute a function in NC_1. This is because the deep computation is performed on the public encodings as $\mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1})$ and constrained decryption is only required for a much shallower function. In this work we show that by assuming *indistinguishability* based security NLinFE, we may prove *indistinguishability* based security of FE for NC_1.

Comparison with [AR17]. Even though the present work uses the ciphertext and public key evaluation algorithms developed by Agrawal and Rosen [AR17], our construction of FE for NC_1 and particularly our proof technique are quite different. Firstly, [AR17] is in the bounded collusion setting with non-compact ciphertext, and achieves a simulation based security which is known to be impossible in our setting where compact ciphertext is crucial. Hence, we must give an indistinguishability style proof, which requires using a new proof technique developed in this work. Moreover, [AR17] adds statistical large flooding noise which is oblivious of the distribution of noise it needs to drown, whereas we will analyze and leverage the distribution carefully. Most importantly, [AR17] can make do with linear FE whereas we crucially need noisy linear FE[6].

Comparison with predicate encryption [GVW15]. An alternate successful approach to constructing functional encryption schemes with *one sided security* is the predicate encryption scheme by Gorbunov et al. [GVW15] and its extension by Agrawal [Agr17]. Roughly speaking, these schemes make use of an attribute based encryption scheme in conjunction with a fully homomorphic encryption scheme to achieve a system where the input $\mathbf{x}$ is hidden only as long as the adversary's key requests obey a certain restriction w.r.t the challenge messages. In more detail, security holds as long as the adversary does not obtain keys SK_f for any circuit f such that $f(\mathbf{x}) = 0$. Given

[6]We remark that a weak version of NLinFE in the bounded collusion setting was developed in an earlier version of [AR17] (see [AR16]) but was found to be redundant and was subsequently removed. The current, published version of [AR17] relies on LinFE alone. Our definition of NLinFE is significantly more general.

an adversary who obeys this one sided restriction, functionality is general, i.e. the adversary may request for a key corresponding to any polynomial sized circuit.

Our approach is orthogonal to [GVW15, Agr17] from the very first step, and insists on a two sided adversary at any cost to functionality. We maintain the two sidedness of our adversary as an invariant and start with the modest functionality of linear functional encryption [ABCP15, ALS16] and look at the mildest possible strengthening of this functionality, namely one that supports computation of linear functions plus noise that satisfies a relatively mild statistical property. We then show that this notion of FE may be bootstrapped all the way to iO.

Constructing NLinFE. Next, we discuss three methods to construct NLinFE which imply FE for NC_1 from various assumptions. Together with the bootstrapping of NLinFE to FE for NC_1 described above, this suffices for applying the FE to iO compiler of [AJ15, BV15].

Our first method makes use of a compact FE scheme which is powerful enough to compute PRG/blockwise local PRG [LT17]. Let PrgFE be a functional encryption scheme that supports evaluation of a PRG with polynomial stretch. Then, we may construct NLinFE and hence FE for NC_1 by leveraging PrgFE to compute the the noise to be added by NLinFE as the output of a PRG.

In more detail, by the discussion above, we would like the decryptor to compute:

$$f(\mathbf{x}) + \mathsf{noise}_{f(\mathbf{x})} + \mathsf{noise}_{\mathsf{fld}} = \langle \mathsf{Lin}_f,\ \mathcal{M}^d \rangle + \mathsf{noise}_{\mathsf{fld}} + \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1})$$

where $\mathsf{noise}_{f(\mathbf{x})} + \mathsf{noise}_{\mathsf{fld}}$ is statistically indistinguishable from $\mathsf{noise}_{\mathsf{fld}}$. Say that the norm of $\mathsf{noise}_{f(\mathbf{x})}$ may be bounded above by value $\mathsf{B}_{\mathsf{nse}}$. Then, it suffices to sample a uniformly distributed noise term $\mathsf{noise}_{\mathsf{fld}}$ of norm bounded by $\mathsf{B}_{\mathsf{fld}}$, where $\mathsf{B}_{\mathsf{fld}}$ is superpolynomially larger than $\mathsf{B}_{\mathsf{nse}}$ for the above indistinguishability to hold. We will use PrgFE to compute $\mathsf{noise}_{\mathsf{fld}}$.

In more detail, let G be a PRG with polynomial stretch which outputs ℓ uniform ring elements of norm bounded by $\mathsf{B}_{\mathsf{fld}}$, and let G_i be the function that selects the i^{th} output symbol of G, namely $G_i(\mathsf{seed}) = G(\mathsf{seed})[i]$ where seed is the seed of the PRG. Then,

1. The encryptor may provide PrgFE encryptions of $(\mathcal{M}^d, \mathsf{seed})$ along with encodings $\underset{i\in[d-1]}{\cup} \mathcal{C}^i$,
2. The key generator may provide a PrgFE secret key for polynomial $P_i(\mathbf{z}_1, \mathbf{z}_2) = \langle \mathsf{Lin}_{f_i},\ \mathbf{z}_1 \rangle + G_i(\mathbf{z}_2)$
3. The decryptor may compute $\langle \mathsf{Lin}_{f_i},\ \mathcal{M}^d \rangle + G_i(\mathsf{seed})$ as well as $\mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1})$, to recover $f(\mathbf{x}) + \mathsf{noise}_{f(\mathbf{x})} + G_i(\mathsf{seed})$ by Equation 1.1 as desired.

It is crucial to note that the degree of the polynomial P_i is equal to the degree required to compute G_i, because Lin_{f_i} is a linear function. Moreover, the degree is unchanged even if we make use of a standard PRG with binary range. To see this, take a binary PRG that requires degree L to compute, and apply the standard (linear) powers of two transformation to convert binary output to larger alphabet. For a more in-depth discussion, please see Section 7.

Thus, an FE scheme that supports polynomials of degree L, where L is the degree required to compute a PRG, suffices to construct NLinFE and hence FE for NC_1. Moreover, we may pre-process the seed of the PRG as in [LT17] to leverage "blockwise locality"; our construction allows the PRG to have smaller expansion factor than that required by the Lin-Tessaro construction, as discussed next.

Analyzing the Expansion Factor of the PRG. Above, the polynomial P_i we are required to construct must compute a function that is degree 1 in the PRG output plus an additional linear function of the encoded messages. By comparison, the underlying degree L FE in [LV16, Lin17, LT17] must natively compute a polynomial which has degree 3 in output of a PRG. In more detail, the FE of [Lin17, LT17] must compute a randomizing polynomial [AIK11], which contains terms of the form $r_i r_j r_k$ where r_i, r_j, r_k are random elements, each computed using a PRG. Thus, if L is the locality of the PRG, the total degree of the polynomial is $3L$, which is reduced to L using a clever precomputing trick developed by Lin [Lin17]. In contrast, our FE must compute a polynomial of the form $\langle \mathsf{Lin}_f,\ \mathcal{M}^d \rangle + \mathsf{PRG}(\mathsf{seed})$ as discussed above, thus natively yielding a polynomial of degree L.

Further, Lin and Tessaro [LT17] construct a method to leverage the blockwise locality L of the PRG, which is believed to be smaller than locality as discussed above. Recall that the PRG seed is now a matrix of b rows and n columns, and each output bit depends on input bits in at most L columns. Then, to begin, [LT17] suggest computing all possible monomials in any block, for all blocks, resulting in $O(n \cdot 2^b)$ total monomials. At this point, the output of a PRG can be computed using a degree L polynomial. However, since the final polynomial has degree 3 in the PRG outputs, LT further suggest precomputing all degree 3 products of the monomials constructed so far, leading to a total of $O(n \cdot 2^{3b})$ terms. To maintain ciphertext compactness then, the expansion of the PRG must be at least $\Omega(n \cdot 2^{b(3+\epsilon)})$. We refer the reader to [BBKK17, Appendix B] for an excellent overview of the LT construction, and to [LT17] for complete details.

Since the polynomial in our FE must compute has degree only 1 rather than 3 in PRG output, we need not compute degree 3 products in $O(n \cdot 2^b)$ monomials as required by [LT17], thus requiring to encode a total number of $O(n \cdot 2^b)$ monomials. Hence, to maintain ciphertext compactness (which is necessary for implication to iO [AJ15, BV15]), the expansion of the PRG in our case must be $\Omega(n \cdot 2^{b(1+\epsilon)})$. Thus, our transformation requires a lower expansion factor than that of [LT17]. Moreover, by sidestepping the need to compute randomized encodings, our bootstrapping becomes simpler and more efficient than that of [LV16, Lin17, LT17]. Please see Section 7.2.1 for more details.

Correlated Noise Generators. As discussed above, FE for implementing PRG suffices to construct NLinFE and hence FE for NC_1. However, examining carefully the requirements on the noise that must be added to decryption by NLinFE, reveals that using a PRG to compute the noise is wasteful; a weaker object appears to suffice. Specifically, we observe that the noise terms $\mathsf{noise}_{f_i(\mathbf{x})}$ for $i \in [\ell]$, which must be flooded are low entropy, correlated random variables, constructed as polynomials in $O(L)$ noise terms where $L = |\bigcup_{k \in [d]} \mathcal{C}_k|$. A PRG mimics ℓ i.i.d noise terms where $\ell > L$, i.e. $O(\ell)$ bits of entropy, whereas the random variables that must be flooded have only $O(L)$ bits of entropy.

Indeed, even to flood ℓ functions on L noise terms *statistically*, only L fresh noise terms are needed. For instance, let us say that we are required to flood $(f_i(\boldsymbol{\mu}))_{i\in[\ell]}$ for $\boldsymbol{\mu} \in R^L$. Then, it suffices to choose $\boldsymbol{\beta} \in R^L$ such that $\boldsymbol{\beta}$ is superpolynomially larger than $\boldsymbol{\mu}$ to conclude that

$$\mathsf{SD}\Big(\boldsymbol{\beta},\ \boldsymbol{\beta} + \boldsymbol{\mu}\Big) = \mathrm{negl}(\kappa)$$

This implies that

$$\mathsf{SD}\Big(\big(f_1(\boldsymbol{\beta}), \ldots, f_\ell(\boldsymbol{\beta})\big),\ \big(f_1(\boldsymbol{\beta}+\boldsymbol{\mu}), \ldots, f_\ell(\boldsymbol{\beta}+\boldsymbol{\mu})\big)\ \Big) = \mathrm{negl}(\kappa)$$

Considering that we must only generate $O(L)$ bits of pseudoentropy, can we make do with something weaker than a PRG?

To make this question precise, we define the notion of a correlated noise generator, denoted by CNG. A CNG captures computational flooding of correlated noise, to mimic statistical flooding described above. In more detail, we will use a CNG to generate flooding terms $g_1(\boldsymbol{\beta}), \ldots, g_\ell(\boldsymbol{\beta})$ such that to any computationally bounded adversary Adv, it holds that

$$\big(g_1(\boldsymbol{\beta}), \ldots, g_\ell(\boldsymbol{\beta})\big) \overset{c}{\approx} \big((g_1(\boldsymbol{\beta}) + f_1(\boldsymbol{\mu}), \ldots g_\ell(\boldsymbol{\beta}) + f_\ell(\boldsymbol{\mu})\big)$$

Note that if we denote by g_i the function for computing the i^{th} element of the PRG output, then by choosing the range of PRG superpolynomially larger than $|f_i(\boldsymbol{\mu})|$, the above condition is satisfied. Thus, a PRG implies a CNG. However, implication in the other direction does not hold, since CNG only generates strictly fewer bits of pseudoentropy than a PRG.

Moreover, matters are even nicer in the case of CNG, because g_i can be chosen after seeing f_i, and the distribution of $\boldsymbol{\mu}$ is known at the time of choosing g_i. So each g_i can be different depending on what it needs to "swallow". Additionally, we may leverage the fact that a CNG posits that a distribution must be indistinguishable from *itself* plus a fixed function, not indistinguishable from uniform.

Our hope is that since a CNG appears weaker than a PRG, it may sidestep the lower bounds known for the blockwise-locality of polynomial stretch PRGs, thereby providing a new route to iO from bilinear maps. Suggesting candidates for CNG that have lower degree than PRG is outside the scope of this work but we believe it is useful to identify the weakest object that suffices for bootstrapping to iO. For the precise definition of CNG, please see Section 5.

Non Boolean PRG. A notion of randomness generators that interpolates CNG and Boolean PRG is that of non-Boolean PRG, which allows the inputs and outputs to lie in a bounded (polynomial) sized interval over the integers and must only satisfy the computational flooding property described above. Taking a step back, we note that in prior work [Lin17, LT17, AS17], Boolean PRGs were required in order to compute the binary randomness needed for constructing randomizing polynomials. In our case, the PRG output need not be binary since we do not require these as input to randomized encodings. Additionally, they must satisfy a much weaker property than indistinguishability to uniform i.i.d random variables as discussed above. In more detail, say we can bound $|f_i(\boldsymbol{\mu})| \leq \epsilon$ for $i \in [\ell]$. Then we require the PRG output $G_i(\boldsymbol{\beta})$ to computationally flood $f_i(\boldsymbol{\mu})$ for $i \in [\ell]$, i.e. $G_i(\boldsymbol{\beta}) + f_i(\boldsymbol{\mu})$ must be computationally indistinguishable from $G_i(\boldsymbol{\beta})$. We conjecture that degree 2 polynomials over $\mathbb{Z}$ may be used to compute $G_i(\boldsymbol{\beta})$ via the multivariate quadratic MQ assumption.

At a high level, the MQ assumption states that given ℓ multivariate quadratic polynomials $p_i(\mathbf{x})$ in n variables $\mathbf{x} = (x_1, \ldots, x_n)$, where $\ell > n$, over a finite field $\mathbb{F}$, it is infeasible to recover $\mathbf{x}$ for appropriate values of ℓ, n and choice of polynomials p_i. The MQ assumption has been studied extensively [MI88, BFP09, BFSS13, BFS03, Wol05, DY09, YDH+15, WP05, TW10, AHKI+17] and has formed the basis of several cryptosystems, please see [DY09] for a survey. MQ is known to be NP complete in the worst case, and the general method to solve MQ involves computing Gröbner bases, which are notoriously hard to compute [Wol05]. We remark that a significant effort in design of MQ based cryptosystems stems from embedding hidden trapdoors in systems of equations, something that is unnecessary in our setting. Hence, using MQ to design PRG with non-Boolean output appears significantly simpler than using it to design public key encryption, which has been the focus of much past research. Please see Section 5 for more discussion. We conjecture that randomly chosen polynomials with coefficients super-polynomially larger than ϵ suffice for the aforementioned flooding. We leave further analysis of such randomness generators to future work.

1.4 Our Techniques: Direct Construction of NLinFE

Next, we provide a direct construction of NLinFE based on new (untested) assumptions that strengthen ring learning with errors (RLWE) and NTRU. Our construction is quite different from known constructions and does not rely on multilinear maps or graded encoding schemes.

As discussed above, flooding correlated noise terms appears qualitatively easier than generating uniform pseudorandom variables. Recall that ℓ is the output length of the function and L is sublinear in ℓ. In this section, we discuss a method to provide L encodings of a seed vector $\boldsymbol{\beta}$ in a way that the decryptor can compute ℓ encodings of $\{g_i(\boldsymbol{\beta})\}_{i\in[\ell]}$ on the fly. The careful reader may suspect that we are going in circles: if we could compute encodings of $g_i(\boldsymbol{\beta})$ on the fly, could we not just compute encodings of $f_i(\mathbf{x})$ on the fly?

We resolve this circularity by arguing that the demands placed on noise in lattice based constructions are significantly weaker than the demands placed on messages. In particular, while computation on messages must maintain integrity, noise need only create some perturbation, the exact value of this perturbation is not important. Therefore if, in our attempt to compute an encoding $g_i(\boldsymbol{\beta})$, we instead compute an encoding of $g'_i(\boldsymbol{\beta}')$, this still suffices for functionality. Intuitively $g'_i(\boldsymbol{\beta}')$ will be a polynomial equation of $\boldsymbol{\beta}$ designed to flood $f_i(\boldsymbol{\mu})$.

In order to construct FE that supports the computation of noisy linear equations, we begin with an FE that supports computation of linear equations, denoted by LinFE, provided by [ABCP15, ALS16]. All our constructions use the blueprint provided in [ALS16] to support linear equations, and develop new techniques to add noise. In order to interface with the LinFE construction of [ALS16], we are required to provide encodings of noise terms $\boldsymbol{\beta}$ such that:

1. Given encodings of $\boldsymbol{\beta}$ and g_i the decryptor may herself compute on these to construct an encoding of $g_i(\boldsymbol{\beta})$.

2. The functional encoding of $g_i(\boldsymbol{\beta})$ must have the form $h_{g,i} \cdot s + \mathsf{noise}_{g,i} + g_i(\boldsymbol{\beta})$ where $h_{g,i}$ *is computable by the key generator given only the public/secret key and the function value. In particular,* $h_{g,i}$ *should not depend on the ciphertext.*

In order to compute efficiently on encodings of noise, we introduce a strengthening of the RLWE and NTRU assumptions. Let $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$ and $R_{p_1} = R/(p_1 \cdot R)$, $R_{p_2} = R/(p_2 \cdot R)$ for some primes $p_1 < p_2$. Then, the following assumptions are necessary (but not sufficient) for security of our scheme:

1. *We assume that the* NTRU *assumption holds even if multiple samples have the same denominator.* This assumption has been discussed by Peikert [Pei16, 4.4.4], denoted as the *NTRU learning problem* and is considered a reasonable assumption. In more detail, for $i \in \{1, \dots, w\}$, sample f_{1i}, f_{2i} and g_1, g_2 from a discrete Gaussian over ring R. If g_1, g_2 are not invertible over R_{p_2}, resample. Set

$$h_{1i} = \frac{f_{1i}}{g_1}, \quad h_{2i} = \frac{f_{2i}}{g_2} \in R_{p_2}$$

We assume that the samples $\{h_{1i}, h_{2j}\}$ for $i, j \in [w]$ are indistinguishable from random. Note that NTRU requires the denominator to be chosen afresh for each sample, i.e. h_{1i} (resp. h_{2i}) should be constructed using denominator g_{1i} (resp. g_{2i}), for $i \in [w]$.

2. *We assume that* RLWE *with small secrets remains secure if the noise terms in* RLWE *samples live in some secret ideal.* In more detail, for $i \in [w]$, let $\widehat{\mathcal{D}}(\Lambda_2), \widehat{\mathcal{D}}(\Lambda_1)$ be discrete Gaussian distributions over lattices Λ_2 and Λ_1 respectively. Then, sample

$$e_{1i} \leftarrow \widehat{\mathcal{D}}(\Lambda_2), \quad \text{where } \Lambda_2 \triangleq g_2 \cdot R. \text{ Let } \quad e_{1i} = g_2 \cdot \xi_{1i} \in \mathsf{small},$$
$$e_{2i} \leftarrow \widehat{\mathcal{D}}(\Lambda_1), \quad \text{where } \Lambda_1 \triangleq g_1 \cdot R. \text{ Let } \quad e_{2i} = g_1 \cdot \xi_{2i} \in \mathsf{small},$$

Above, small is a place-holder term that implies the norm of the relevant element can be bounded well below the modulus size, $p_2/5$, say. We use it for intuition when the precise bound on the norm is not important. Hence, for $i, j \in [w]$, it holds that:

$$h_{1i} \cdot e_{2j} = f_{1i} \cdot \xi_{2j}, \quad h_{2j} \cdot e_{1i} = f_{2j} \cdot \xi_{1i} \quad \in \; \mathsf{small}$$

Now, sample small secrets t_1, t_2 and for $i \in [w]$, compute

$$d_{1i} = h_{1i} \cdot t_1 + p_1 \cdot e_{1i} \in R_{p_2}$$
$$d_{2i} = h_{2i} \cdot t_2 + p_1 \cdot e_{2i} \in R_{p_2}$$

We assume that the elements d_{1i}, d_{2j} for $i, j \in [w]$ are pseudorandom. The powerful property that this assumption provides is that the product of the samples $d_{1i} \cdot d_{2j}$ do not suffer from large cross terms for any $i, j \in [w]$ – since the error of one sample is chosen to annihilate with the large element of the other sample, the product yields a well behaved RLWE sample whose label is a product of the original labels. In more detail,

$$d_{1i} \cdot d_{2j} = \big(h_{1i} \cdot h_{2j}\big) \cdot (t_2 \; t_2) + p_1 \cdot \mathsf{noise}$$
$$\text{where } \mathsf{noise} = p_1 \cdot \big(f_{1i} \cdot \xi_{2j} \cdot t_1 + f_{2j} \cdot \xi_{1i} \cdot t_2 + p_1 \cdot g_1 \cdot g_2 \cdot \xi_{1i} \cdot \xi_{2j}\big) \in \mathsf{small}$$

If we treat each d_{1i}, d_{2j} as an RLWE sample, then we may use these samples to encode noise terms so that direct multiplication of samples is well behaved. Note that the noise terms we wish to compute on, are the messages encoded by the "RLWE sample" d_{1i}, hence d_{1i} must contain two kinds of noise: the noise required for RLWE security and the noise that behaves as the encoded message. This requires some care, but can be achieved by nesting these noise terms in different ideals as:

$$d_{1i} = h_{1i} \cdot t_1 + p_1 \cdot \tilde{e}_{1i} + p_0 \cdot e_{1i} \in R_{p_2}$$
$$d_{2i} = h_{2i} \cdot t_2 + p_1 \cdot \tilde{e}_{2i} + p_0 \cdot e_{2i} \in R_{p_2}$$

Here, $(p_1 \cdot \tilde{e}_{1i}, \; p_1 \cdot \tilde{e}_{2i})$ behave as RLWE noise and $(p_0 \cdot e_{1i}, \; p_0 \cdot e_{2i})$ behave as the encoded messages. Both $\tilde{e}_{1i}, e_{1i}$ as well as $\tilde{e}_{2i}, e_{2i}$ are chosen from special ideals as before. Now, we may compute quadratic polynomials on the encodings "on-the-fly" as $\sum_{i,j} d_{1i} d_{2j}$ to obtain a structured-noise RLWE sample whose label is computable by the key generator. If we treat this dynamically generated encoding as an RLWE encoding of correlated noise, then we can use this to add noise to the NLinFE decryption equation by generalizing techniques from [ALS16]. The decryptor can, using all the machinery developed so far, recover $f_i(\mathbf{x}) + \mathsf{noise}_{f(\mathbf{x})} + \mathsf{noise}_{\mathsf{fld}\,i}$ where $\mathsf{noise}_{\mathsf{fld}\,i}$ is constructed as a quadratic polynomial of noise terms that live in special ideals. Indeed, there is no need to stop at quadratic polynomials – we can compute any constant degree polynomial, see Section 8 for details. Again, we remind the reader that setting $\mathsf{noise}_{\mathsf{fld}\,i} = \mathsf{PRG}_i(\mathsf{seed})$ would make the scheme provably secure, so an approach is to choose the bounded degree polynomial to be computed as PRG_i.

Mixing Ideals. While it suffices for functionality to choose the correlated noise term as a polynomial evaluated on noise living in special ideals, the question of security is more worrisome. By using the new "on-the-fly" encodings of noise, the decryptor recovers noise which lives in special, secret ideals, and learning these ideals would compromise security. In more detail, the noise term we constructed above is a random linear combination of terms $(g_1 \cdot g_2)$, $\{f_{1i}\}_i$, $\{f_{2j}\}_j$, which must be kept secret for semantic security of d_{1i}, d_{2j} to hold. Indeed, if we over-simplify and assume that the attacker can recover noise terms that live in the ideal generated by $g_1 \cdot g_2$, then recovering $g_1 \cdot g_2$ from these terms becomes an instance of the *principal ideal problem* [Gen09, CDPR16].

While the principal ideal problem has itself resisted efficient classical algorithms so far, things in our setting can be made significantly better by breaking the ideal structure using additional tricks. We describe these next.

1. *Mixing ideals.* Instead of computing a single set of pairs $\Big((h_{1i}, d_{1i}), (h_{2i}, d_{2i}) \Big)$, we now compute k of them, for some polynomial k fixed in advance. Thus, we sample f_{1i}^j, f_{2i}^j and g_1^j, g_2^j for $i \in \{1, \dots, w\}$, $j \in \{1, \dots, k\}$, where w, k are fixed polynomials independent of function output length ℓ, and set
$$h_{1i}^j = \frac{f_{1i}^j}{g_1^j}, \quad h_{2i}^j = \frac{f_{2i}^j}{g_2^j} \in R_{p_2}$$
The encoding of a noise term constructed corresponding to the $(i,j)^{th}$ monomial is $d_{ii'} = \sum\limits_{j \in [k]} d_{1i} d_{2i'}$. Thus, the resultant noise term that gets added to the decryption equation looks like:
$$p_0 \cdot \left[\sum_{j \in [k]} \left(g_2^j \cdot g_1^j \cdot \Big(p_0 \cdot (\xi_{1i}^j \cdot \xi_{2i'}^j) \Big) + \Big(f_{1i}^j \cdot \xi_{2i'}^j \cdot t_1 + f_{2i'}^j \cdot \xi_{1i}^\ell \cdot t_2 \Big) \right) \right] \tag{1.2}$$
Thus, by adding together noise terms from multiple ideals, we "spread" it out over the entire ring rather than restricting it to a single secret ideal. Also, we note that it is only the higher degree noise terms that must live in special ideals; if the polynomial contains linear terms, these may be chosen from the whole ring without any restrictions. In more detail, above, we computed a noise term corresponding to a quadratic monomial which required multiplying and summing encodings. If we modify the above quadratic polynomial to include a linear term, we will need to add a degree 1 encoding into the above equation. The degree 1 encoding which does not participate in products, may encode noise that is chosen without any restrictions, further randomizing the resultant noise.

2. *Adding noise generated collectively by ciphertext and key.* Aside from computing polynomials over structured noise terms encoded in the ciphertext, we suggest an additional trick which forces noise terms into the decryption equation. These noise terms are quadratic polynomials

where each monomial is constructed jointly by the encryptor and the key generator. This trick relies on the structure of the key and ciphertext in our construction. We describe the relevant aspects of the key and ciphertext here, for details please see Section 8. The key for function f_i is a short vector $\mathbf{k}$ such that:

$$\langle \mathbf{w},\ \mathbf{k} \rangle = u_{f_i}$$

where $\mathbf{w}$ is part of the master secret, and u_{f_i} is computed by the key generator using the $\mathsf{Eval}_{\mathsf{PK}}$ function. The encryptor provides an encoding

$$\mathbf{c} = \mathbf{w} \cdot s + p_1 \cdot \mathsf{noise}_0$$

As part of decryption, the decryptor computes $\langle \mathbf{k}, \mathbf{c} \rangle$ to obtain $u_{f_i} \cdot s + p_1 \cdot \langle \mathbf{k},\ \mathsf{noise}_0 \rangle$. Moreover by running $\mathsf{Eval}_{\mathsf{CT}}(\mathcal{C}^1, \ldots, \mathcal{C}^d)$, she also obtains $u_{f_i} \cdot s + f(\mathbf{x}) + p_0 \cdot \mathsf{noise} + p_1 \cdot \mathsf{noise}'$ (please see Appendix E for details). Subtracting these and reducing modulo p_1 and then modulo p_0 yields $f(\mathbf{x})$ as desired.

Intuitively, the structured noise computed above is part of the noise in the sample computed by $\mathsf{Eval}_{\mathsf{CT}}$, i.e. part of $\big(\ p_0 \cdot \mathsf{noise} + p_1 \cdot \mathsf{noise}'\ \big)$ in the notation above. Our next trick shows how to add noise to $\langle \mathbf{k}, \mathbf{c} \rangle$.

We modify KeyGen so that instead of choosing a single $\mathbf{k}$, it now chooses a pair $(\mathbf{k}^1, \mathbf{k}^2)$ such that:

$$\langle \mathbf{w},\ \mathbf{k}^1 \rangle = u_{f_i} + p_0 \cdot \Delta^1 + p_1 \cdot \tilde{\Delta}^1$$
$$\langle \mathbf{w},\ \mathbf{k}^2 \rangle = u_{f_i} + p_0 \cdot \Delta^2 + p_1 \cdot \tilde{\Delta}^2$$

Here, Δ^1, Δ^2, $\tilde{\Delta}^1$, $\tilde{\Delta}^2$ are discrete Gaussians sampled by the key generator *unique to the key for* f_i. Additionally, the encryptor splits $\mathbf{c}$ as:

$$\mathbf{c}_{01} = \mathbf{w} \cdot s_1 + p_1 \cdot \boldsymbol{\nu}_1$$
$$\mathbf{c}_{02} = \mathbf{w} \cdot s_2 + p_1 \cdot \boldsymbol{\nu}_2$$

where $s_1 + s_2 = s$ and s_1, s_2 are small, then,

$$\langle \mathbf{k}^1,\ \mathbf{c}_{01} \rangle + \langle \mathbf{k}^2,\ \mathbf{c}_{02} \rangle = u_{f_i} \cdot s + p_0 \cdot \big(\Delta^1 \cdot s_1 + \Delta^2 \cdot s_2\big) + p_1 \cdot \big(\tilde{\Delta}^1 \cdot s_1 + \tilde{\Delta}^2 \cdot s_2\big) + p_1 \cdot \mathsf{noise}$$

Thus, we forced the quadratic polynomial $p_0 \cdot \big(\Delta^1 \cdot s_1 + \Delta^2 \cdot s_2\big) + p_1 \cdot \big(\tilde{\Delta}^1 \cdot s_1 + \tilde{\Delta}^2 \cdot s_2\big)$ into the noise, where Δ^1, Δ^2 and $\tilde{\Delta}^1, \tilde{\Delta}^2$ are chosen by the key generator for the particular key request and the terms s_1 and s_2 are chosen by the encryptor unique to that ciphertext. Note that $\mathbf{w}$ can be hidden from the view of the adversary since it is not required for decryption, hence the adversary may not compute $\langle \mathbf{w},\ \mathbf{k}^1 \rangle$, $\langle \mathbf{w},\ \mathbf{k}^2 \rangle$ in the clear. For more details, please see Section 8.

Putting it together. Put together, an overview of our transformation is provided in Figure 1.2.

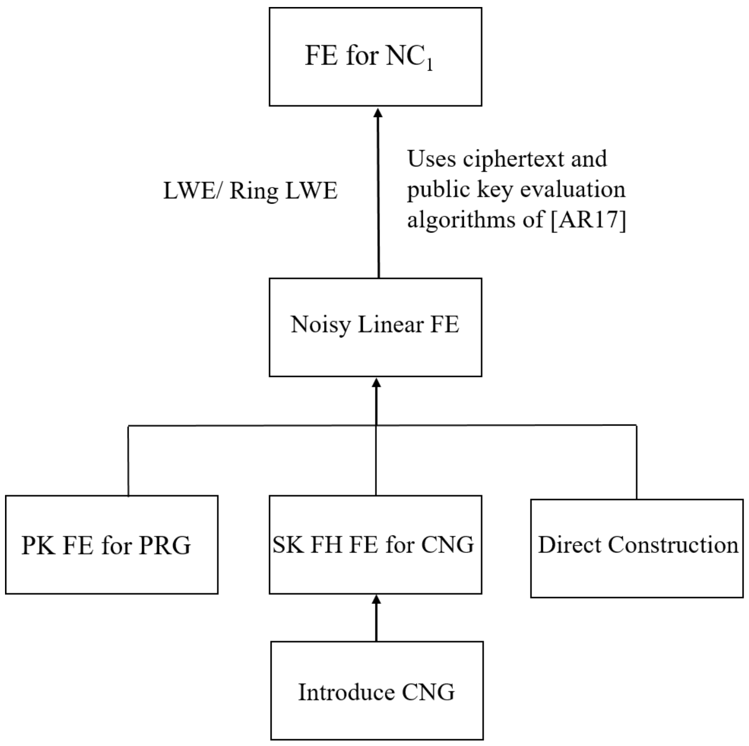

Figure 1.2: Overview of our transformation. Above, FH refers to function hiding.

1.5 Organization of the paper

In Section 2 we provide the definitions and preliminaries we require. In Section 3 we define the notion of noisy linear functional encryption NLinFE. In Section 4 we provide our warm-up construction of Quadratic FE from Noisy Linear FE and PRG. In Section 5 we define the notion of correlated noise generators CNG and non-Boolean PRG that we require. In Section 6, we provide our full construction of FE for NC_1 using NLinFE, RLWE and CNG. In Section 7 we provide our bootstrapping theorems, namely, construction of NLinFE from FE for polynomials of degree L, where L is the blockwise locality of PRG in the public setting or CNG in the symmetric setting. In Section 8 we provide our new construction of succinct, symmetric key noisy linear FE based on new assumptions, and we reason about its security in Section 8.1. We discuss parameters in Section 9 and conclude in Section 10.

2 Preliminaries

In this section, we define the preliminaries we require for our constructions. We begin with defining the notation that we will use throughout the paper.

Notation. We say a function $f(n)$ is *negligible* if it is $O(n^{-c})$ for all $c > 0$, and we use negl(n) to denote a negligible function of n. We say $f(n)$ is *polynomial* if it is $O(n^c)$ for some $c > 0$, and we use poly(n) to denote a polynomial function of n. We say an event occurs with *overwhelming* *probability* if its probability is $1 - \text{negl}(n)$. The function $\log x$ is the base 2 logarithm of x. The notation $\lfloor x \rceil$ denotes the nearest integer to x, rounding towards 0 for half-integers.

Throughout the writeup, we use the term noise as a place-holder for RLWE noise when its precise value is not important. Similarly we use the term small as a place holder that implies the norm of the relevant element (β (say) when we have $\beta \in \mathsf{small}$), can be bounded well below the modulus size, $\mathsf{modulus}/5$, say. We use it for intuition when the precise bound on the norm is not important.

2.1 Pseudorandom Generators

Definition 2.1 (Pseudorandom Generator (PRG))**.** A function $G : \{0,1\}^n \to \{0,1\}^m$ is a pseudorandom generator (PRG) if it has the following properties:

1. G is computable in (uniform) time $\text{poly}(n)$
2. For any probabilistic polynomial time adversary Adv, it holds that

$$\left| \Pr_{\boldsymbol{\beta} \leftarrow \{0,1\}^n} \big(1 \leftarrow \mathsf{Adv}(G(\boldsymbol{\beta})\big) - \Pr_{\mathbf{r} \leftarrow \{0,1\}^m} \big(1 \leftarrow \mathsf{Adv}(\mathbf{r})\big) \right| = \text{negl}(\kappa)$$

Here, $\boldsymbol{\beta}$ is called the *seed* of the PRG G, and $m - n$ is called the *stretch* of the PRG. In this paper, we focus on the polynomial stretch regime, namely where $m = O(n^c)$ for some constant $c > 1$. If G is computable in NC_0, then we define the *locality* of G to be the maximum number of input elements on which any output element of G depends.

Lin and Tessaro [LT17] defined the notion of *blockwise locality* as follows.

Definition 2.2 (Blockwise Locality[LT17])**.** Let n, m, L and b be polynomials. We say an (n, b, m) PRG has blockwise locality L if the input of the PRG may be viewed as a matrix with b rows and n columns, and each output bit of the PRG depends on input bits that are contained in at most L of b columns.

2.2 Indistinguishability Obfuscation

A uniform P.P.T machine iO is an indistinguishability obfuscator for a class of circuits $\{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$, if the following conditions are satisfied:

1. **Correctness.** For all security parameters $\kappa \in \mathbb{N}$, for any $C \in \mathcal{C}_\kappa$ and every input $\mathbf{x}$ from the domain of C, we have that:

$$\Pr\left[C' \leftarrow \mathsf{iO}(1^\kappa, C) : C'(\mathbf{x}) = C(\mathbf{x})\right] = 1$$

where the probability is taken over the coin-tosses of the obfuscator iO.

2. **Indistinguishability of Equivalent Circuits.** For every ensemble of pairs of circuits $\{C_{0,\kappa}, C_{1,\kappa}\}_{\kappa \in \mathbb{N}}$, and $C_{0,\kappa}(\mathbf{x}) = C_{1,\kappa}(\mathbf{x})$ for every $\mathbf{x}$, we have that the following ensembles of pairs of distributions are indistinguishable to any P.P.T Adv:

$$\left\{C_{0,\kappa}, C_{1,\kappa}, \mathsf{iO}(1^\kappa, C_{0,\kappa})\right\} \overset{c}{\approx} \left\{C_{0,\kappa}, C_{1,\kappa}, \mathsf{iO}(1^\kappa, C_{1,\kappa})\right\}$$

2.3 Functional Encryption

Let $\mathcal{X} = \{\mathcal{X}_\kappa\}_{\kappa\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\kappa\}_{\kappa\in\mathbb{N}}$ denote ensembles where each $\mathcal{X}_\kappa$ and $\mathcal{Y}_\kappa$ is a finite set. Let $\mathcal{C} = \left\{\mathcal{C}_\kappa\right\}_{\kappa\in\mathbb{N}}$ denote an ensemble where each $\mathcal{C}_\kappa$ is a finite collection of circuits, and each circuit $g \in \mathcal{C}_\kappa$ takes as input a string $x \in \mathcal{X}_\kappa$ and outputs $g(x) \in \mathcal{Y}_\kappa$.

A functional encryption scheme FE for $\mathcal{C}$ consists of four algorithms $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ defined as follows.

- $\mathsf{FE.Setup}(1^\kappa)$ is a p.p.t. algorithm takes as input the unary representation of the security parameter and outputs the master public and secret keys $(\mathsf{PK}, \mathsf{MSK})$.
- $\mathsf{FE.Keygen}(\mathsf{MSK}, g)$ is a p.p.t. algorithm that takes as input the master secret key MSK and a circuit $g \in \mathcal{C}_\kappa$ and outputs a corresponding secret key SK_g.
- $\mathsf{FE.Enc}(\mathsf{PK}, x)$ is a p.p.t. algorithm that takes as input the master public key PK and an input message $x \in \mathcal{X}_\kappa$ and outputs a ciphertext CT.
- $\mathsf{FE.Dec}(\mathsf{SK}_g, \mathsf{CT}_x)$ is a deterministic algorithm that takes as input the secret key SK_g and a ciphertext CT_x and outputs $g(x)$.

Definition 2.3 (Correctness)**.** A functional encryption scheme FE is correct if for all $g \in \mathcal{C}_\kappa$ and all $x \in \mathcal{X}_\kappa$,

$$\Pr\left[\begin{array}{l} (\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa); \\ \mathsf{FE.Dec}\Big(\mathsf{FE.Keygen}(\mathsf{MSK}, g), \mathsf{FE.Enc}(\mathsf{PK}, x)\Big) \neq g(x) \end{array}\right] = \mathrm{negl}(\kappa)$$

where the probability is taken over the coins of $\mathsf{FE.Setup}$, $\mathsf{FE.Keygen}$, and $\mathsf{FE.Enc}$.

Security. In this paper we will consider the standard indistinguishability based definition.

Definition 2.4. A functional encryption scheme FE for a function family $\mathcal{C}$ is secure in the adaptive indistinguishability game, denoted as IND secure, if for all probabilistic polynomial-time adversaries Adv, the advantage of Adv in the following experiment is negligible in the security parameter κ:

1. **Public Key.** Challenger Ch returns PK to Adv.
2. **Pre-Challenge Key Queries.** Adv may adaptively request keys for any functions $g_1, \ldots, g_{\ell'} \in \mathcal{C}$. In response, Adv is given the corresponding keys SK_{g_i}.
3. **Challenge.** Adv outputs the challenges $(\mathbf{x}_0, \mathbf{x}_1) \in \mathcal{X}$ to the challenger, subject to the restriction that $g_i(\mathbf{x}_0) = g_i(\mathbf{x}_1)$ for all $i \in [\ell']$. The challenger chooses a random bit b, and returns the ciphertext $\mathsf{CT}_{\mathbf{x}_b}$.
4. **Post-Challenge Key Queries.** The adversary may continue to request keys for additional functions g_i, subject to the restriction that $g_i(\mathbf{x}_0) = g_i(\mathbf{x}_1)$ for all $i \in \{\ell' + 1, \ldots, \ell\}$. In response, Adv is given the corresponding keys SK_{g_i}.
5. **Guess.** Adv outputs a bit b', and succeeds if $b' = b$.

The *advantage* of Adv is the absolute value of the difference between its success probability and $1/2$. In the *selective* game, the adversary must announce the challenge in the first step, before receiving the public key. Note that without loss of generality, in the selective game, the challenge ciphertext can be returned along with the public key. In the *semi-adaptive* game, the adversary must announce the challenge after seeing the public key but before making any key requests.

We also define a very restricted notion of security, called $\mathsf{Full\text{-}Sel}$ security, in which the adversary must announce both the challenge messages and key requests in the very beginning of the game, before seeing even the public key. This notion of security is sufficient for building iO from FE (see for instance [LT17, Sec 2.5.1]).

Compactness. Intuitively, an FE scheme is compact if the length of its ciphertext does not depend on the size of the circuit it supports [AJ15, BV15]. Formally, we say an FE scheme is compact if the encryption algorithm runs in time $\mathrm{poly}(\kappa, |\mathbf{x}|, \log S)$ where $\mathbf{x}$ is the input and S is the size of the circuit. The notion of weak compactness asks that the encryption algorithm run in time $\mathrm{poly}(\kappa, |\mathbf{x}|, S)^{1-\epsilon}$ for any constant $\epsilon \in (0, 1)$.

2.4 Function Hiding Symmetric Key Encryption

The symmetric key version of functional encryption is defined analogously to the public key version defined above except:

1. The encryptor also makes use of the master secret key to perform the encrypt operation.
2. In the security game, the adversary can make ciphertext queries in addition to key queries, since he may no longer compute ciphertexts on his own.

In the symmetric key setting, we may additionally define the notion of function hiding as follows.

Definition 2.5 (Function hiding)**.** A symmetric key FE scheme FE is function-hiding, if every admissible PPT adversary Adv has negligible advantage in the following game:

1. Key Generation. The challenger Ch samples $\mathsf{MSK} \leftarrow \mathsf{FE.Setup}(1^\kappa)$.
2. The challenger Ch chooses a random bit b and repeats the following with Adv for an arbitrary number of times determined by Adv:
   - Function Queries. Upon Adv choosing a pair of functions (f_0, f_1), Ch sends Adv a function key $\mathsf{SK} \leftarrow \mathsf{FE.Keygen}(\mathsf{MSK}, f_b)$.
   - Message Queries. Upon Adv choosing a pair of messages $(\mathbf{x}_0, \mathbf{x}_1)$, Ch sends Adv a ciphertext $\mathsf{CT} \leftarrow \mathsf{FE.Enc}(\mathsf{PK}, \mathbf{x}_b)$.
3. The adversary outputs a guess b' for the bit b and wins if $b = b'$.

We say an adversary is admissible if for all function and message queries, it holds that $f_0(\mathbf{x}_0) = f_1(\mathbf{x}_1)$.

2.5 Lattice Preliminaries

An *m-dimensional lattice* Λ is a full-rank discrete subgroup of $\mathbb{R}^m$. A *basis* of Λ is a linearly independent set of vectors whose span is Λ.

Gaussian distributions. For any vector $\mathbf{c} \in \mathbb{R}^n$ and any positive parameter $\sigma \in \mathbb{R}_{>0}$, let $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) := \mathsf{Exp}\left(-\pi\|\mathbf{x}-\mathbf{c}\|^2/\sigma^2\right)$ be the Gaussian function on $\mathbb{R}^n$ with center $\mathbf{c}$ and parameter σ. Let $\rho_{\sigma,\mathbf{c}}(\Lambda) := \sum_{\mathbf{x}\in\Lambda} \rho_{\sigma,\mathbf{c}}(\mathbf{x})$ be the discrete integral of $\rho_{\sigma,\mathbf{c}}$ over L, and let $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}$ be the discrete Gaussian distribution over Λ with center $\mathbf{c}$ and parameter σ. Specifically, for all $\mathbf{y} \in \Lambda$, we have $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}(\mathbf{y}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{y})}{\rho_{\sigma,\mathbf{c}}(\Lambda)}$. For notational convenience, $\rho_{\sigma,\mathbf{0}}$ and $\mathcal{D}_{\Lambda,\sigma,\mathbf{0}}$ are abbreviated as ρ_σ and $\mathcal{D}_{\Lambda,\sigma}$, respectively.

We will also need the definition of a B-bounded distribution.

Definition 2.6 (B-bounded distribution)**.** A distribution ensemble $(\chi_\kappa)_{\kappa\in\mathbb{N}}$ is called B-bounded if

$$\Pr_{e\leftarrow\chi_\kappa}(\|e\| > B) = \text{negl}(\kappa)$$

The following lemma gives a bound on the length of vectors sampled from a discrete Gaussian.

Lemma 2.7 ([MR07, Lemma 4.4])**.** *Let Λ be an n-dimensional lattice, let $\mathbf{T}$ be a basis for Λ, and let $\|\mathbf{T}\|_{\mathsf{GS}}$ denote the Gram Schmidt norm of $\mathbf{T}$. Suppose $\sigma \geq \|\mathbf{T}\|_{\mathsf{GS}} \cdot \omega(\sqrt{\log n})$. Then for any $\mathbf{c} \in \mathbb{R}^n$ we have*

$$\Pr\left[\|\mathbf{x}-\mathbf{c}\| > \sigma\sqrt{n} : \mathbf{x} \overset{\mathrm{R}}{\leftarrow} \mathcal{D}_{\Lambda,\sigma,\mathbf{c}}\right] \;\leq\; \text{negl}(n)$$

We will also need the "noise smudging" or "noise flooding lemma" as follows.

Lemma 2.8. *[GKPV10] Let $n \in \mathbb{N}$. For any real $\sigma = \omega(\sqrt{\log n})$, and any $\mathbf{c} \in \mathbb{Z}^n$,*

$$\mathsf{SD}(\mathcal{D}_{\mathbb{Z}^n,\sigma},\ \mathcal{D}_{\mathbb{Z}^n,\sigma,\mathbf{c}}) \leq \|\mathbf{c}\|/\sigma$$

2.6 Hardness Assumptions.

Our constructions are based on the hardness of the learning with errors problem LWE [Reg09] or its ring variant RLWE problem [LPR10]. We define these next.

Definition 2.9 (LWE)**.** For security parameter κ, let $n = n(\kappa)$ be an integer dimension, let $q = q(\kappa) \geq 2$ be an integer and let $\chi = \chi(\lambda)$ be a distribution over $\mathbb{Z}$. The $\mathsf{LWE}_{\mathsf{n,q},\chi}$ problem is to distinguish the following two distributions: in the first distribution, sample $(\mathbf{a}_i, b_i)$ uniformly from $\mathbb{Z}_q^{n+1}$. In the second distribution, one first draws $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ uniformly and then samples $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^{n+1}$ by sampling $\mathbf{a}_i \leftarrow \mathbb{Z}_q^n$ uniformly, $e_i \leftarrow \chi$ and setting $b_i = \langle \mathbf{a}_i, \mathbf{s}\rangle + e_i$. The $\mathsf{LWE}_{\mathsf{n,q},\chi}$ assumption is that the $\mathsf{LWE}_{\mathsf{n,q},\chi}$ problem is infeasible.

Regev [Reg09] proved that for certain moduli q and Gaussian error distributions χ, the $\mathsf{LWE}_{\mathsf{n,q},\chi}$ assumption is true as long as certain worst-case lattice problems are hard to solve using a quantum algorithm. This result was de-quantized by Peikert for exponential modulus [Pei09] and by Brakerski, Langlois, Peikert, Regev, Oded and Stehlé for polynomial modulus [BLP+13].

Next, we define the ring variant of the LWE problem.

Definition 2.10 (RLWE)**.** For security parameter κ, let $f(x) = x^n + 1$ where n is a power of 2. Let $q = q(\kappa)$ be an integer. Let $R = \mathbb{Z}[x]/f(x)$ and let $R_q = R/qR$. Let χ be a probability distribution on R. For $s \in R_q$, let $A_{s,\chi}$ be the probability distribution on $R_q \times R_q$ obtained by choosing an element $a \in R_q$ uniformly at random, choosing $e \leftarrow \chi$ and outputting $(a, a \cdot s + e)$. The decision $\mathsf{RLWE}_{n,q,\chi}$ problem is to distinguish between samples that are either (all) from $A_{s,\chi}$ or (all) uniformly random in $R_q \times R_q$. The $\mathsf{RLWE}_{\mathsf{n,q},\chi}$ assumption is that the $\mathsf{RLWE}_{\mathsf{n,q},\chi}$ problem is infeasible.

Theorem 2.11 ([LPR10]). *Let $r \geq \omega(\sqrt{\log n})$ be a real number and let R, q be as above. Then, there is a randomized reduction from $2^{\omega(\log n)} \cdot (q/r)$ approximate RSVP to $\mathsf{RLWE}_{n,q,\chi}$ where χ is the discrete Gaussian distribution with parameter r. The reduction runs in time* $\text{poly}(n, q)$.

NTRU. In Section 8, we will also need the **NTRU** assumption defined by [HPS98] which roughly states that it is hard to distinguish a fraction of small elements over R_q from random.

Definition 2.12 ($\mathsf{NTRU}_{q,\chi}$). The NTRU problem $\mathsf{NTRU}_{q,\chi}$ is to distinguish between the following two distributions: in the first distribution sample a polynomial $h = g/f$ where $f, g \leftarrow \chi$, conditioned on f being invertible in R_q and in the second distribution sample a polynomial h uniformly over R_q.

Stehlé and Steinfeld [SS11] showed that the $\mathsf{NTRU}_{q,\chi}$ problem is hard even for unbounded adversaries for χ chosen as the discrete Gaussian distribution with parameter $r > \sqrt{q} \cdot \text{poly}(n)$. However, we will need to make the assumption for much smaller $r = \text{poly}(n)$ as in many prior works (eg. [LATV12]).

NTRU Learning Problem. Peikert [Pei16] defines the NTRU learning problem as follows.

Definition 2.13 (NTRU Learning Problem). For an invertible $g \in R_q^*$ and a distribution χ on R, define $N_{g,\chi}$ to be the distribution that outputs f/g where $f \leftarrow \chi$. The *NTRU learning problem* is: given independent samples $h_i \in R_q$ where every sample is distributed according to either $N_{g,\chi}$ for some randomly chosen $g \in R_q^*$ (fixed for all samples), or the uniform distribution, distinguish which is the case with non-negligible advantage.

Peikert observes that just as with the normal form of ring-LWE, we can assume without loss of generality that the secret denominator g is chosen from the distribution χ, restricted to units in R_q. Note that in the NTRU cryptosystem, the public key is one sample f/g for a short g, and the ciphertexts essentially correspond to additional samples with the same denominator, and less short numerators. In our construction provided in Section 8, we will make use of a public key which contains multiple elements of the form f_i/g for varying f_i and fixed g. We refer the reader to [Pei16, 4.4.4] for a discussion on the hardness of this problem.

The Multivariate Quadratic Polynomial Problem (MQ)**:** [MI88, BFSS13, Wol05, DY09] The multivariate quadratic polynomial problem, denoted by MQ, is: given m quadratic polynomials $f_1, \ldots, f_m$ in n variables $x_1, \ldots, x_n$, with coefficients chosen from a field $\mathbb{F}$, find a solution $\mathbf{z} \in \mathbb{F}^n$ such that $f_i(\mathbf{z}) = 0$ for $i \in [m]$.

2.7 Sampling and Trapdoors

Ajtai [Ajt99] showed how to sample a random lattice along with a trapdoor that permits sampling short vectors from that lattice. Recent years have seen significant progress in refining and extending this result [SSTX09, AP09, MP12]. Our construction in Section 8 requires trapdoor generation in polynomial lattices.

Let $R = \mathbb{Z}[x]/(\phi)$ where $\phi = x^n + 1$ and n is a power of 2. Let $R_q \triangleq R/qR$ where q is a large prime satisfying $q = 1 \mod 2n$. For $r \in R$, we use $\|r\|$ to refer to the Euclidean norm of r's coefficient vector.

We will make use of the following algorithms from [MP12]:

1. $\mathsf{TrapGen}(n, m, q)$: The $\mathsf{TrapGen}$ algorithm takes as input the dimension of the ring n, a sufficiently large integer $m = O(n \log q)$ and the modulus size q and outputs a vector $\mathbf{w} \in R_q^m$ such that the distribution of $\mathbf{w}$ is negligibly far from uniform, along with a "trapdoor" $\mathbf{T_w} \in R^{m \times m}$ for the lattice $\Lambda_q^{\perp}(\mathbf{w}) = \{\mathbf{x} : \langle \mathbf{w}, \mathbf{x} \rangle = 0 \mod q\}$.

2. $\mathsf{SamplePre}(\mathbf{w}, \mathbf{T_w}, a, \sigma)$: The $\mathsf{SamplePre}$ algorithm takes as input a vector $\mathbf{w} \in R_q^m$ along with a trapdoor $\mathbf{T_w}$ and a syndrome $a \in R_q$ and a sufficiently large $\sigma = O(\sqrt{n \log q})$ and outputs a vector $\mathbf{e}$ from a distribution within negligible distance to $\mathcal{D}_{\Lambda_q^a(\mathbf{w}), \sigma \cdot \omega(\sqrt{\log n})}$ where $\Lambda_q^a(\mathbf{w}) = \{\mathbf{x} : \langle \mathbf{w}, \mathbf{x} \rangle = a \mod q\}$.

3 Noisy Linear Functional Encryption

In this section, we define the notion of noisy linear functional encryption. At a high level, noisy linear functional encryption is like regular linear functional encryption [ABCP15, ALS16], except that the function value is recovered only up to some bounded additive error (which we informally call noise), and indistinguishability holds even if the challenge messages evaluated on all the function keys are only "approximately" equal, i.e. they differ by an additive term of low norm.

In our constructions, R is a ring, instantiated either as the ring of integers $\mathbb{Z}$ or the ring of polynomials $\mathbb{Z}[x]/f(x)$ where $f(x) = x^n + 1$ for n a power of 2. We let $R_q = R/qR$ for some prime q. Let $\mathcal{D}$ be a distribution over R, $\mathcal{F}$ be a class of functions $\mathcal{F} : R^\ell \to R$ and $B \in \mathbb{R}^+$ a bounding value on the norm of the decryption error. In general, we require $B << q$. Then,

Definition 3.1. A $(\mathcal{D}, \mathcal{F}, B)$-noisy linear functional encryption scheme FE is a tuple $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ of four probabilistic polynomial-time algorithms with the following specifications:

- $\mathsf{FE.Setup}(1^\kappa, R_q^\ell)$ takes as input the security parameter κ and the space of message and function vectors R_q^ℓ and outputs the public key and the master secret key pair $(\mathsf{PK}, \mathsf{MSK})$.
- $\mathsf{FE.Keygen}(\mathsf{MSK}, \mathbf{v})$ takes as input the master secret key MSK and a vector $\mathbf{v} \in R_q^\ell$ and outputs the secret key $\mathsf{SK}_\mathbf{v}$.
- $\mathsf{FE.Enc}(\mathsf{PK}, \mathbf{z})$ takes as input the public key PK and a message $\mathbf{z} \in R_q^\ell$ and outputs the ciphertext $\mathsf{CT}_\mathbf{z}$.
- $\mathsf{FE.Dec}(\mathsf{SK}_\mathbf{v}, \mathsf{CT}_\mathbf{z})$ takes as input the secret key of a user $\mathsf{SK}_\mathbf{v}$ and the ciphertext $\mathsf{CT}_\mathbf{z}$, and outputs $y \in R_q \cup \{\perp\}$.

Definition 3.2 (Approximate Correctness)**.** A noisy linear functional encryption scheme FE is correct if for all $\mathbf{v}, \mathbf{z} \in R_q^\ell$,

$$\Pr\left[\begin{array}{l} (\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa); \\ \mathsf{FE.Dec}\Big(\mathsf{FE.Keygen}(\mathsf{MSK}, \mathbf{v}), \mathsf{FE.Enc}(\mathsf{PK}, \mathbf{z})\Big) = \langle \mathbf{v}, \mathbf{z} \rangle + e \end{array}\right] = 1 - \mathrm{negl}(\kappa)$$

where $e \in R$ with $\|e\| \le B$ and the probability is taken over the coins of $\mathsf{FE.Setup}$, $\mathsf{FE.Keygen}$, and $\mathsf{FE.Enc}$.

Security. Security is defined in the standard indistinguishability setting, which requires that no admissible adversary should be able to distinguish $\mathsf{CT}(\mathbf{z}_0)$ from $\mathsf{CT}(\mathbf{z}_1)$ with non negligible advantage. However, we place severe restrictions on our adversary as defined next.

Definition 3.3 (Admissible Adversary)**.** We say an adversary is *admissible* if for any pair of challenge messages $\mathbf{z}_0, \mathbf{z}_1 \in R_q^\ell$ and any queried key $\mathbf{v}_i \in R_q^\ell$, it holds that $\langle \mathbf{v}_i, \mathbf{z}_0 - \mathbf{z}_1 \rangle = f_i(\boldsymbol{\mu})$ where $\boldsymbol{\mu} \leftarrow \mathcal{D}^\ell$ and $f_i \in \mathcal{F}$.

While it may appear strange to restrict the adversary to choosing messages and functions that satisfy a strong constraint such as the above, such a restricted adversary suffices for our main application of constructing quadratic functional encryption QuadFE in Section 4 and its generalisation to NC_1 in Appendix 6. At a high level, a QuadFE adversary makes queries for quadratic functions, which translate to queries against NLinFE which obey the restrictions above. Constructing an NLinFE scheme secure against such a restricted adversary will prove significantly simpler as we shall see.

Formally, we define the notion of $\mathsf{Noisy\text{-}IND}$ security as follows.

Definition 3.4 ($\mathsf{Noisy\text{-}IND}$ security)**.** A $(\mathcal{D}, \mathcal{F}, B)$ noisy linear FE scheme NLinFE is $\mathsf{Noisy\text{-}IND}$ secure if for all admissible probabilistic polynomial-time adversaries Adv, the advantage of Adv in the following experiment is negligible in the security parameter κ.

1. **Public Key:** Challenger returns PK to the adversary.

2. **Pre-Challenge Queries:** Adv may adaptively request keys for any functions $\mathbf{v}_i \in R_q^\ell$ for $i \in [k]$ for some polynomial k. In response, Adv is given the corresponding keys $\mathsf{SK}(\mathbf{v}_i)$.

3. **Challenge Ciphertexts:** Adv outputs the challenge message pairs $(\mathbf{z}_0^i, \mathbf{z}_1^i) \in R_q^\ell \times R_q^\ell$, for $i \in [Q]$, where Q is some polynomial, to the challenger. The challenger chooses a random bit b, and returns the ciphertexts $\{\mathsf{CT}(\mathbf{z}_b^i)\}_{i \in [Q]}$.

4. **Post-Challenge Queries:** Adv may request additional keys for functions of its choice and is given the corresponding keys. Adv may also output additional challenge message pairs which are handled as above.

5. **Guess.** Adv outputs a bit b', and succeeds if $b' = b$.

The *advantage* of Adv is the absolute value of the difference between the adversary's success probability and 1/2.

In the *selective* game, the adversary must announce the challenge in the first step, before receiving the public key. In the *semi-adaptive* game, the adversary must announce the challenge after seeing the public key but before making any key requests.

The symmetric key version of noisy functional encryption is defined analogously to functional encryption, please see Section 2.

4 Warm-up: Quadratic Functional Encryption

In this section, we provide our warm-up construction of functional encryption for quadratic polynomials. For ease of exposition, we present first our simplest construction, which relies on the RLWE assumption, a noisy linear functional encryption scheme $(\mathcal{D}, \mathcal{F}, B)$-NLinFE and a PRG with polynomial expansion. We may improve assumptions to use LWE in place of RLWE, and CNG (please see Section 1) in place of PRG – this is discussed in Appendix A. The parameters of NLinFE are instantiated below in Section 4.3.1.

The construction relies on the ciphertext and public key evaluation procedures developed recently by Agrawal and Rosen [AR17]. Let $R = \mathbb{Z}[x]/(x^d + 1)$ and $R_p = R/pR$. Let $u_i, s \leftarrow R_{p_1}$ and $x_i \in R_{p_0}$ for $i \in [w]$. [AR17] observe that given "Regev encodings" of x_i as:

$$c_i \approx u_i \cdot s + x_i, \qquad c_j \approx u_j \cdot s + x_j$$

one may compute a Regev encoding of the product $x_i x_j$ by re-purposing a trick used by Brakerski and Vaikuntanathan [BV11b]. In more detail, we write

$$x_i x_j \approx c_i c_j + u_i u_j (s^2) - u_j (c_i s) - u_i (c_j s)$$

Here, since c_i, c_j are known to the decryptor she may compute the cross term $c_i c_j$ herself. The remaining linear term may be written as an inner product

$$\big\langle (u_i u_j, -0-, u_i, -0-, u_j, -0-),\ (s^2, c_1 s, \ldots, c_w s) \big\rangle$$

of which the first vector is known to the key generator and the second to the encryptor. If the encryptor provides Regev encodings c_i as well as a LinFE encryption of vector $(s^2, c_1 s, \ldots, c_w s)$, and the key generator provides a LinFE secret key for the linear function $(u_i u_j, -0-, u_i, -0-, u_j, -0-)$, then the decryptor may recover the inner product using a linear FE scheme and compute (an approximation of) $x_i x_j$ as above.

The above intuitive description is insecure and [AR17] suggest a way to secure it but their approach results in non-compact ciphertext. In this work, we show that by replacing the usage of LinFE by NLinFE, we can make the ciphertext compact.

Formally, quadratic polynomials will be represented by a circuit with a multiplication layer, followed by an addition layer. For the construction below, we will require two prime moduli $p_0 < p_1$ where p_0 serves as the message space for the quadratic scheme, and p_1 serves as the message space for the NLinFE scheme. Below, the distribution $\mathcal{D}_0$ is a discrete Gaussian with width σ_0 and $\mathcal{D}_1$ is a discrete Gaussian with width σ_1. Parameters are instantiated in Appendix 9.

Choosing the PRG. Below, we will modify slightly (albeit superficially) the standard definition of PRG so that it has non-Boolean output. In more detail, we choose $G : \{-1, 1\}^n \rightarrow \mathsf{R_{fld}}^L$ where $L = |\{\ (i,j) : 1 \leq j \leq i \leq w\}|$, $\mathsf{R_{fld}} \subset R_{p_1}$ is the set of elements in R_{p_1} with norm $\leq \mathsf{B_{fld}}$, and $\mathsf{B_{fld}}$ is a parameter which is set in Section 4.3.1. We denote the seed of the PRG as $\boldsymbol{\beta}$ and denote by $G_{ij}(\boldsymbol{\beta})$ the output symbol corresponding to the pair (i,j) in the L length output of G. Note that this formulation of PRG is implied trivially by the standard binary PRG by generating each coefficient of the polynomial $G_{ij}(\boldsymbol{\beta})$ in its binary representation and using the standard powers of two transformation to map it to an element in $\mathbb{Z}_{p_1}^d$.

4.1 Construction

We proceed to describe our construction, which we denote by QuadFE.

$\mathsf{QuadFE.Setup}(1^\kappa, R_{p_0}^w)$: On input a security parameter κ and the space of message vectors $R_{p_0}^w$, do:

1. Sample $\mathbf{u} \leftarrow R_{p_1}^w$.
2. Invoke $\mathsf{NLinFE.Setup}(1^\kappa, R_{p_1}^{w+2})$ to obtain $\mathsf{NLinFE.PK}$ and $\mathsf{NLinFE.MSK}$.
3. Sample a PRG $G : \{-1,1\}^n \to {\mathsf{R_{fld}}}^L$ where $L = |\{\ (i,j) : 1 \le j \le i \le w\}|$ as discussed above.
4. Sample $t_0, \ldots, t_w \leftarrow R_{p_1}$ and let $\mathbf{t} = (t_0, \ldots, t_w)$.
5. Output $\mathsf{PK} = (\mathsf{NLinFE.PK}, \mathbf{u})$, $\mathsf{MSK} = (\mathsf{NLinFE.MSK}, \boldsymbol{\beta}, \mathbf{t})$.

$\mathsf{QuadFE.Enc}(\mathsf{PK}, \mathbf{x})$: On input public parameters PK, and message vector $\mathbf{x} \in R_{p_0}^w$ do:

1. Sample $s_1 \leftarrow R_{p_1}$ and $\boldsymbol{\mu} \leftarrow \mathcal{D}_0^w$, and compute the encoding of the message
$$\mathbf{c} = \mathbf{u} \cdot s_1 + p_0 \cdot \boldsymbol{\mu} + \mathbf{x} \in R_{p_1}^w.$$
2. Let $\mathbf{b} = \mathsf{NLinFE.Enc}\ (s_1^2, c_1 s_1, \ldots, c_w s_1, 0)$.
3. Output $\mathsf{CT} = (\mathbf{c}, \mathbf{b})$

$\mathsf{QuadFE.KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathbf{g})$: On input the public parameters $\mathsf{PK} = (\mathsf{NLinFE.PK}, \mathbf{u})$, the master secret key $\mathsf{MSK} = (\mathsf{NLinFE.MSK}, \boldsymbol{\beta}, \mathbf{t})$, and a function $\mathbf{g}(\mathbf{x}) = \sum_{1 \le j \le i \le w} g_{ij} x_i x_j$, represented as a coefficient vector $(g_{ij}) \in \mathbb{Z}_{p_0}^L$ do:

1. Compute
$$\mathbf{u_g} = \sum_{1 \le j \le i \le w} g_{ij}\,(u_i u_j, 0....0, -u_i, 0...0, -u_j, 0...0) \in R_{p_1}^{w+1}.$$
2. Define
$$\mathsf{key}_{ij} = u_i u_j t_0 - u_j t_i - u_i t_j - p_0 \cdot G_{ij}(\boldsymbol{\beta}), \quad \forall 1 \le j \le i \le w,$$
$$\mathsf{key}_{\mathbf{g}} = \sum_{1 \le j \le i \le w} g_{ij} \mathsf{key}_{ij}, \qquad G_{\mathbf{g}}(\boldsymbol{\beta}) = \sum_{1 \le j \le i \le w} g_{ij} \cdot G_{ij}(\boldsymbol{\beta})$$
Note that, $\mathsf{key}_{\mathbf{g}} = \langle \mathbf{u_g},\ \mathbf{t} \rangle - p_0 \cdot G_{\mathbf{g}}(\boldsymbol{\beta}) \ \in R_{p_1}$
3. Compute $\mathsf{SK}_{\mathbf{g}} = \mathsf{NLinFE.KeyGen}\big(\mathsf{MSK}, (\mathbf{u_g} | \mathsf{key}_{\mathbf{g}})\big)$ and output it.

$\mathsf{QuadFE.Dec}(\mathsf{PK}, \mathsf{SK}_{\mathbf{g}}, \mathsf{CT}_{\mathbf{x}})$: On input the public parameters PK, a secret key $\mathsf{SK}_{\mathbf{g}}$ for polynomial $\sum_{1 \le j \le i \le w} g_{ij} x_i x_j$, and a ciphertext $\mathsf{CT}_{\mathbf{x}} = (\mathbf{c}, \mathbf{b})$, compute
$$\sum_{1 \le j \le i \le w} g_{ij} c_i c_j + \mathsf{NLinFE.Dec}(\mathbf{b}, \mathsf{SK}_{\mathbf{g}}) \bmod p_0$$
and output it.

Ciphertext Size. The ciphertext in the above scheme is comprised of:

$$\mathbf{c} = \mathbf{u} \cdot s_1 + p_0 \cdot \boldsymbol{\mu} + \mathbf{x} \in R_{p_1}^w, \quad \mathbf{b} = \mathsf{NLinFE.Enc}\left(s_1^2, c_1 s_1, \ldots, c_w s_1, 0\right)$$

Clearly, $\mathbf{c}$ enjoys linear efficiency. Additionally, note that the message size in $\mathbf{b}$ is linear, hence efficiency of QuadFE ciphertext is inherited from that of NLinFE ciphertext.

4.2 Correctness.

We show correctness of the quadratic FE scheme.

Theorem 4.1. *If* NLinFE *is correct, then the* QuadFE *is correct.*

Proof. Let $1 \leq j \leq i \leq w$. By definition

$$x_i + p_0 \cdot \mu_i = c_i - u_i s_1, \quad x_j + p_0 \cdot \mu_j = c_j - u_j s_1$$

Let

$$\mu_{ij} = x_i \mu_j + x_j \mu_i + p_0 \mu_i \mu_j \tag{4.1}$$

$$x_i x_j + p_0 \cdot \mu_{ij} = c_i c_j - c_i u_j s_1 - c_j u_i s_1 + u_i u_j s_1^2 \tag{4.2}$$

We denote the noise corresponding to monomial $x_i x_j$, added by the scheme NLinFE by $p_0 \cdot \rho_{ij}$. We note that the noise added by NLinFE can easily be chosen to be a multiple of p_0, since this is chosen by the encryptor of the NLinFE scheme, please see Section 8 for details.

By correctness of the linear scheme NLinFE, we have that

$$\mathsf{NLinFE.Dec}(\mathbf{b}, \mathsf{SK}_{\mathbf{g}}) = \sum_{1 \leq j \leq i \leq w} g_{ij}\left(u_i u_j s_1^2 - u_j c_i s_1 - u_i c_j s_1 + p_0 \cdot \rho_{ij}\right)$$

$$\begin{aligned}\sum_{1 \leq j \leq i \leq w} g_{ij} c_i c_j + \mathsf{NLinFE.Dec}(\mathbf{b}, \mathsf{SK}_{\mathbf{g}}) &= \sum_{1 \leq j \leq i \leq w} g_{ij}\Big(c_i c_j - c_i u_j s_1 - c_j u_i s_1 + u_i u_j s_1^2 + p_0 \cdot \rho_{ij}\Big) \\ &= \sum_{1 \leq j \leq i \leq w} g_{ij}\left(x_i x_j + p_0 \cdot \mu_{ij} + p_0 \cdot \rho_{ij}\right) \\ &= \sum_{1 \leq j \leq i \leq w} g_{ij}\ x_i x_j \bmod p_0 \quad \text{as long as } p_0 \cdot (\mu_{ij} + \rho_{ij}) \leq \frac{p_1}{5}.\end{aligned}$$

□

Note that the error recovered in decryption is $p_0 \cdot \sum_{1 \leq j \leq i \leq w} g_{ij}\ (\mu_{ij} + \rho_{ij})$.

4.3 Indistinguishability Based Security

Theorem 4.2. *Assume that the noisy linear FE scheme* NLinFE *satisfies semi-adaptive indistinguishability based security as in Definition 3.4, that the* RLWE *assumption holds and that G is a secure* PRG *for the parameters discussed in Section 4.3.1. Then, the construction* QuadFE *in Section 4 achieves semi-adaptive indistinguishability based security provided in Definition 2.4.*

Proof. We will prove the theorem via a sequence of hybrids, where the first hybrid is the real world with challenge $\mathbf{x}_0$ and the last hybrid is the real world with challenge $\mathbf{x}_1$.

The Hybrids. Our Hybrids are described below.

Hybrid 0. This is the real world with message $\mathbf{x}_0$. In hybrid 0, $\mathsf{key}_{\mathbf{g}}$ is picked as follows:

1. Sample $t_0, \ldots, t_w \leftarrow R_{p_1}$ and let $\boldsymbol{\beta} \leftarrow \{-1,1\}^n$ be the seed of the PRG.
2. For $1 \le j \le i \le w$, set $\mathsf{key}_{ij} = \big(u_i u_j t_0 - u_j t_i - u_i t_j - p_0 \cdot G_{ij}(\boldsymbol{\beta})\big) \mod p_1$
3. Let $\mathsf{key}_{\mathbf{g}} = \sum_{1 \le j \le i \le w} g_{ij} \mathsf{key}_{ij} \mod p_1$.

Hybrid 1. In this hybrid, the only thing that is different is that the challenger picks $\mathsf{key}_{\mathbf{g}}$ to depend on the challenge ciphertext $(c_1, \ldots, c_w)$ and the function value $\mathbf{g}(\mathbf{x})$. Specifically,

1. Sample $t_0, \ldots, t_w \leftarrow R_{p_1}$ and let $\boldsymbol{\beta} \leftarrow \{-1,1\}^n$ be the seed of the PRG.
2. Set

$$\begin{aligned}\mathsf{key}_{ij} &= \big(x_i x_j - c_i c_j\big) - \big(u_i u_j t_0 - u_j t_i - u_i t_j\big) - p_0 \cdot G_{ij}(\boldsymbol{\beta}) \quad \forall\, 1 \le j \le i \le w \\ \mathsf{key}_{\mathbf{g}} &= \sum_{1 \le j \le i \le w} g_{ij} \mathsf{key}_{ij} \mod p_1\end{aligned}$$

Hybrid 2. In this hybrid, we change the input for $\mathsf{NLinFE.Enc}$ to $(t_0, t_1, \ldots, t_w, 1)$ where t_i are chosen as in Hybrid 1.

Hybrid 3. In this hybrid, we change the message vector in $\mathbf{c}$ to $\mathbf{x}_1$.

Hybrids 4 and 5. In Hybrid 4 we change the input to $\mathsf{NLinFE.Enc}$ to $(s_1^2, c_1 s_1, \ldots, c_w s_1, 0)$ as in Hybrid 1. In Hybrid 5, we change $\mathsf{key}_{\mathbf{g}}$ to be chosen independent of the ciphertext as in Hybrid 0. This is the real world with message $\mathbf{x}_1$.

Indistinguishability of Hybrids. Below we establish that consecutive hybrids are indistinguishable.

Claim 4.3. *Hybrid 0 and Hybrid 1 are indistinguishable assuming the security of the* PRG G *for parameters as discussed in Section 4.3.1.*

Proof. The only difference between Hybrid 0 and Hybrid 1 is in the way $\mathsf{key}_{\mathbf{g}}$ is sampled. In Hybrid 1, $\mathsf{key}_{\mathbf{g}}$ is picked as follows:

1. Sample $t_0, \ldots, t_w \leftarrow R_{p_1}$ and $\boldsymbol{\beta} \leftarrow \{-1,1\}^n$ be the seed of the PRG.
2. Let $\mathsf{key}_{ij} = \big(x_i x_j - c_i c_j\big) - \big(u_i u_j t_0 - u_j t_i - u_i t_j\big) - p_0 \cdot G_{ij}(\boldsymbol{\beta})$ for $1 \le j \le i \le w$.

To argue that key_{ij} (and hence $\mathsf{key}_{\mathbf{g}}$) are indistinguishable, we rely on the security of PRG. Note that by Equation 4.2,

$$x_i x_j - c_i c_j = u_i u_j s_1^2 - c_i u_j s_1 - c_j u_i s_1 - p_0 \cdot \Big(p_0 \cdot \mu_i \mu_j + x_i \mu_j + x_j \mu_i\Big)$$

Hence,

$$\begin{aligned}\big(x_ix_j - c_ic_j\big) - \big(u_iu_jt_0 - u_jt_i - u_it_j\big) &= \big(u_iu_js_1^2 - u_ic_js_1 - u_jc_is_1 - p_0\cdot(p_0\cdot\mu_i\mu_j + x_i\mu_j + x_j\mu_i)\big)\\ &\quad - \big(u_iu_jt_0 - u_jt_i - u_it_j\big)\\ &= u_iu_j(s_1^2 - t_0) - u_i(c_js_1 - t_j) - u_j(c_is_1 - t_i)\\ &\quad - p_0\cdot(p_0\cdot\mu_i\mu_j + x_i\mu_j + x_j\mu_i)\end{aligned}$$

Recall $\mu_{ij} = p_0\cdot\mu_i\mu_j + x_i\mu_j + x_j\mu_i$, and let $t'_0 = s_1^2 - t_0,\;\; t'_i = c_is_1 - t_i,\;\; \forall\, i\in[w]$

Then, $\big(x_ix_j - c_ic_j\big) - \big(u_iu_jt_0 - u_jt_i - u_it_j\big) = u_iu_jt'_0 - u_it'_j - u_jt'_i - p_0\cdot\mu_{ij}$

Thus, in Hybrid 1, we have

$$\mathsf{key}_{ij} = u_iu_jt'_0 - u_it'_j - u_jt'_i - p_0\cdot\big(\mu_{ij} + G_{ij}(\boldsymbol{\beta})\big) \tag{4.3}$$

where $t'_0,\ldots,t'_w$ are uniform in R_{p_1}. Next, $\mathsf{key}_{\mathbf{g}} = \sum_{1\le j\le i\le w} g_{ij}\mathsf{key}_{ij}$.

In Hybrid 0, we have:

1. Sample $t_0,\ldots,t_w \leftarrow R_{p_1}$ and $\boldsymbol{\beta}\leftarrow\{-1,1\}^n$ be the seed of the PRG.
2. Set $\mathsf{key}_{ij} = u_iu_jt_0 - u_jt_i - u_it_j - p_0\cdot G_{ij}(\boldsymbol{\beta})$
3. Set $\mathsf{key}_{\mathbf{g}} = \sum_{1\le j\le i\le w} g_{ij}\mathsf{key}_{ij}$.

Thus, it suffices to argue that $\big(\mu_{ij} + G_{ij}(\boldsymbol{\beta})\big)$ is indistinguishable from $G_{ij}(\boldsymbol{\beta})$. We have by security of the PRG G that $\big(\mu_{ij} + G_{ij}(\boldsymbol{\beta})\big)$ is indistinguishable from $\big(\mu_{ij} + \mathsf{Unif}_{\mathsf{R_{fld}}}\big)$, where $\mathsf{R_{fld}}$ is the range of G_{ij}. Next, we have by our choice of parameters (please see Section 4.3.1) that $\big(\mu_{ij} + \mathsf{Unif}_{\mathsf{R_{fld}}}\big)$ is statistically indistinguishable from $\mathsf{Unif}_{\mathsf{R_{fld}}}$, and again by security of PRG that $\mathsf{Unif}_{\mathsf{R_{fld}}}$ is indistinguishable from $G_{ij}(\boldsymbol{\beta})$ as desired. ☐

Claim 4.4. *Assuming semi-adaptive IND security of the noisy linear FE scheme* NLinFE*, Hybrid 1 and Hybrid 2 are indistinguishable.*

Proof. Given an adversary $\mathcal{B}$ who distinguishes between Hybrid 1 and 2, we construct an adversary $\mathcal{A}$ who breaks the semi-adaptive IND security of the noisy linear FE scheme NLinFE. $\mathcal{A}$ runs as follows:

1. The challenger of the NLinFE scheme provides NLinFE.PK to $\mathcal{A}$. $\mathcal{A}$ picks $\mathbf{u}\in R_{p_1}^w$, and sends $\mathsf{PK} = (\mathsf{NLinFE.PK}, \mathbf{u})$ to $\mathcal{B}$.
2. $\mathcal{B}$ outputs challenges $\mathbf{x}_0,\mathbf{x}_1\in R_{p_0}^w$. $\mathcal{A}$ picks $s_1\leftarrow R_{p_1}$, and $\mathbf{t}\leftarrow R_{p_1}^{w+1}$ and computes

$$c_i = u_is_1 + p_0\cdot\mu_i + \mathbf{x}_0[i] \quad \forall i\in[w].$$

$\mathcal{A}$ chooses challenge messages $\mathbf{z}_0,\mathbf{z}_1$ as:

$$\begin{aligned}\mathbf{z}_0 &= \big(s_1^2, c_1s_1,\ldots,c_ws_1,\ 0\big) &\in R_{p_1}^{w+2}\\ \mathbf{z}_1 &= (t_0,\ t_1,\quad\ldots\quad,t_w,\ 1) &\in R_{p_1}^{w+2}\end{aligned}$$

$\mathcal{A}$ returns $(\mathbf{z}_0,\mathbf{z}_1)$ to the NLinFE challenger.

3. When the NLinFE challenger sends the challenge $\mathsf{NLinFE.CT}(\mathbf{z}_b)$ to $\mathcal{A}$, it sends $\mathsf{CT} = \big(\mathsf{NLinFE.CT}(\mathbf{z}_b), \mathbf{c}\big)$ to $\mathcal{B}$.
4. When $\mathcal{B}$ requests a key for quadratic polynomial $\sum\limits_{1 \le j \le i \le w} g_{ij} x_i x_j$, $\mathcal{A}$ computes $\mathsf{key}_{\mathbf{g}} \in R_{p_1}$ as in Hybrid 1, namely, for $1 \le j \le i \le w$, let

$$\mathsf{key}_{ij} = \big(x_i x_j - c_i c_j\big) - \big(u_i u_j t_0 - u_j t_i - u_i t_j\big) - p_0 \cdot G_{ij}(\boldsymbol{\beta}), \quad \mathsf{key}_{\mathbf{g}} = \sum_{1 \le j \le i \le w} g_{ij} \mathsf{key}_{ij}$$

   It forwards $(\mathbf{u}_{\mathbf{g}} \| \mathsf{key}_{\mathbf{g}}) \in R_{p_1}^{w+2}$ to the NLinFE challenger. It's response is relayed to $\mathcal{B}$.
5. When $\mathcal{B}$ outputs a guess bit b, $\mathcal{A}$ forwards this guess to the NLinFE challenger.

Now, we argue that for all the requested keys, the difference between the decryption values on the challenge messages $\mathbf{z}_0$ and $\mathbf{z}_1$ has the required distribution.

Consider a key request for quadratic function $\mathbf{g}(\mathbf{x}) = \sum\limits_{1 \le j \le i \le w} g_{ij} x_i x_j$, i.e. a NLinFE key request for $(\mathbf{u}_{\mathbf{g}} \| \mathsf{key}_{\mathbf{g}}) \in R_{p_1}^{w+2}$.

In Hybrid 1, the function $(\mathbf{u}_{\mathbf{g}} \| \mathsf{key}_{\mathbf{g}})$ evaluated on $\mathbf{z}_0$ yields:

$$\sum_{1 \le j \le i \le w} g_{ij} \big(u_i u_j s_1^2 - u_i c_j s_1 - u_j c_i s_1\big)$$

In Hybrid 2 the function $(\mathbf{u}_{\mathbf{g}} \| \mathsf{key}_{\mathbf{g}})$ evaluated on $\mathbf{z}_1$ yields:

$$\begin{aligned}
&\sum_{1 \le j \le i \le w} g_{ij} \Big(u_i u_j t_0 - u_i t_j - u_j t_i + \mathsf{key}_{ij}\Big) \\
&= \sum_{1 \le j \le i \le w} g_{ij} \Big(u_i u_j t_0 - u_i t_j - u_j t_i + \big(u_i u_j (s_1^2 - t_0) - u_j (c_i s_1 - t_i) - u_i (c_j s_1 - t_j) \\
&\qquad - p_0 \cdot \mu_{ij} - p_0 \cdot G_{ij}(\boldsymbol{\beta})\big)\Big) \\
&= \sum_{1 \le j \le i \le w} g_{ij} \Big(u_i u_j s_1^2 - u_i c_j s_1 - u_j c_i s_1 - p_0 \cdot (\mu_{ij} + G_{ij}(\boldsymbol{\beta}))\Big)
\end{aligned}$$

Thus, the decryption values in both worlds are the same upto an additive error of

$$p_0 \cdot \sum_{1 \le j \le i \le w} g_{ij} (\mu_{ij} + G_{ij}(\boldsymbol{\beta}))$$

Thus, the difference between the 2 hybrids is distributed as $\sum\limits_{1 \le j \le i \le w} g_{ij} \big(f_{ij}(\mathbf{x}, \boldsymbol{\mu}) + G_{ij}(\boldsymbol{\beta})\big)$ We will instantiate NLinFE so that $\sum\limits_{1 \le j \le i \le w} g_{ij} \big(f_{ij}(\mathbf{x}, \boldsymbol{\mu}) + G_{ij}(\boldsymbol{\beta})\big) \in \mathcal{F}_{\mathsf{NFE}}$. Then, by the guarantee of Noisy Linear FE we obtain that Hybrids 1 and 2 are indistinguishable. $\square$

Claim 4.5. *Assume Regev public key encryption is semantically secure. Then, Hybrid 2 is indistinguishable from Hybrid 3.*

Proof. Recall that by semantic security of Regev's (dual) public key encryption, we have that the ciphertext $\mathbf{c} = \mathbf{u} \cdot s_1 + p_0 \cdot \boldsymbol{\mu} + \mathbf{x}_0$ is indistinguishable from $\mathbf{c} = \mathbf{u} \cdot s_1 + p_0 \cdot \boldsymbol{\mu} + \mathbf{x}_1$, where $\mathbf{u}$ is part of the public key and $\boldsymbol{\mu} \leftarrow \mathcal{D}_0$ is suitably chosen noise. We refer the reader to [GPV08] for more details.

Given an adversary $\mathcal{B}$ who distinguishes between Hybrid 2 and Hybrid 3, we build an adversary $\mathcal{A}$ who breaks the semantic security of Regev public key encryption. The adversary $\mathcal{A}$ receives $\mathsf{PK} = \mathbf{u}$ upon which, it simulates the view of $\mathcal{B}$ as follows:

- Run NLinFE.Setup to obtain NLinFE.PK and NLinFE.MSK. Return $\mathsf{PK} = (\mathsf{NLinFE.PK}, \mathbf{u})$ to $\mathcal{B}$.
- When $\mathcal{B}$ outputs challenges $\mathbf{x}_0, \mathbf{x}_1$, $\mathcal{A}$ forwards these to the PKE challenger.
- $\mathcal{A}$ receives $\mathbf{c}$ where $\mathbf{c} = \mathbf{u} \cdot s_1 + p_0 \cdot \boldsymbol{\mu} + \mathbf{x}_b$ for a random bit b.
- $\mathcal{A}$ computes $\mathbf{b} = \mathsf{NLinFE.CT}(t_0, \ldots, t_w, 1)$ and returns $(\mathbf{c}, \mathbf{b})$ to $\mathcal{B}$.
- When $\mathcal{B}$ requests a key for quadratic polynomial $\mathbf{g}$, construct it using NLinFE.MSK and with $\mathsf{key}_{\mathbf{g}}$ as in the previous hybrid.
- Finally, when $\mathcal{B}$ outputs a guess bit b, $\mathcal{A}$ outputs the same.

Clearly, if $b = 0$, then $\mathcal{B}$ sees the distribution of Hybrid 2, whereas if $b = 1$, it sees the distribution of Hybrid 3. Also, the advantage of the attacker $\mathcal{B}$ in distinguishing Hybrids 2 and 3 translates directly to the advantage of the attacker $\mathcal{A}$ in breaking the semantic security of Regev public key encryption. Hence the claim follows. □

Hybrid 4 is analogous to Hybrid 1 and indistinguishability can be argued along the same lines as that between Hybrids 1 and 2, while Hybrid 5 is analogous to Hybrid 0 but with message $\mathbf{x}_1$. This is the real world with message $\mathbf{x}_1$. Indistinguishability between Hybrids 4 and 5 can be argued along the same lines as that between Hybrids 0 and 1. □

4.3.1 Instantiating The Ingredients.

Below, we discuss how to instantiate the PRG and the NLinFE scheme.

1. **Pseudorandom Generator** PRG. Intuitively, we need the PRG to make the distribution of key_{ij} match in the real world and the simulation. As discussed in the proof of Lemma 4.3, we need the PRG output $G_{ij}(\boldsymbol{\beta})$ to flood the noise term μ_{ij}, that is,

$$\mathsf{SD}\Big(G_{ij}(\boldsymbol{\beta}), G_{ij}(\boldsymbol{\beta}) + \mu_{ij}\Big) = \text{negl}(\kappa) \tag{4.4}$$

Recall that $G_{ij}(\boldsymbol{\beta})$ is distributed uniformly in $\mathsf{R}_{\mathsf{fld}}$ which is the set of polynomials with norm at most $\mathsf{B}_{\mathsf{fld}}$. Moreover, we have by equation 4.1 that

$$\mu_{ij} = x_i \mu_j + x_j \mu_i + p_0 \mu_i \mu_j$$

where $x_i \in R_{p_0}$ and $\mu_i \leftarrow \mathcal{D}_0$ are B_{init} bounded distributions (please see Definition 2.6), so we may bound $\|\mu_{ij}\|$ by $p_0 \cdot B_{\mathsf{init}}^2$. Now, by choosing $\mathsf{B}_{\mathsf{fld}}$ to satisfy

$$\frac{|\mu_{ij}|}{\mathsf{B}_{\mathsf{fld}}} = \text{negl}(\kappa) \tag{4.5}$$

we claim equation 4.4 by a standard probabilistic argument[7].

2. **Noisy linear functional encryption NLinFE.** In Claim 4.4, we saw that decryption in Hybrid 1 and 2 is the same up to an additive error of

$$p_0 \cdot \sum_{1 \le j \le i \le w} g_{ij}\big(\mu_{ij} + G_{ij}(\boldsymbol{\beta})\big)$$

where μ_{ij} is defined in equation 4.1 and G_{ij} is an output symbol of the PRG G as discussed above.

We require a $(\mathcal{D}_{\mathsf{NFE}}, \mathcal{F}_{\mathsf{NFE}}, B_{\mathsf{NFE}})$-$\mathsf{NLinFE}$ scheme where (i). $(\mathbf{x}_0, \boldsymbol{\mu}, \boldsymbol{\beta}) \leftarrow \mathcal{D}_{\mathsf{NFE}}$ when $\mathbf{x} \in R_{p_0}^w$ is chosen arbitrarily, $\boldsymbol{\mu} \leftarrow \mathcal{D}_0^w$ and $\boldsymbol{\beta} \leftarrow \{-1,1\}^n$ is the seed of the PRG (ii). $\mathcal{F}_{\mathsf{NFE}} = \big\{\sum_{1 \le j \le i \le w} g_{ij}\; h_{ij}(\cdot,\cdot,\cdot)\big\}_{1 \le j \le i \le w}$ with

$$h_{ij}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\beta}) = p_0 \cdot \big(\mu_{ij} + G_{ij}(\boldsymbol{\beta})\big)$$

and (iii) B_{NFE} is chosen to satisfy $B_{\mathsf{NFE}} < p_1/5$ to maintain decryption correctness as discussed in Section 4.2. Our constructions of NLinFE will impose an additional constraint on B_{NFE}, please see Section 7 for details.

For parameters, please see Section 9.

5 Broader Classes of Randomness Generators

In this section we define broader classes of randomness generators that suffice for our bootstrapping.

5.1 Correlated Noise Generators

In this section we define the notion of a correlated noise generator, which we denote by CNG. Since the intuition was discussed in Section 1, we proceed directly with the formalism.

We denote by R the ring of integers $\mathbb{Z}$ or the ring of polynomials $\mathbb{Z}[x]/f(x)$ where $f(x) = x^d + 1$ as discussed in Section 2. Let $\mathcal{D}_1$ be a distribution over R and $\mathcal{F} : R^w \to R$ be a set of deterministic functions. Let $\mathsf{Dom}_{\mathsf{Cng}}, \mathsf{Rg}_{\mathsf{Cng}}$ be finite subsets of R, let $\mathcal{G} : \mathsf{Dom}_{\mathsf{Cng}}^n \to \mathsf{Rg}_{\mathsf{Cng}}^m$ be a family of deterministic functions and $\mathcal{D}_2$ be a distribution over $\mathsf{Dom}_{\mathsf{Cng}}$. We require that n be linear in w, i.e. $n = O(w, \text{poly}(\kappa))$.

Definition 5.1 ($(\mathcal{D}_1, \mathcal{F})$- Correlated Noise Generator)**.** We say that $(\mathcal{D}_2, \mathcal{G})$ is a $(\mathcal{D}_1, \mathcal{F})$- *Correlated Noise Generator* (CNG) if the advantage of any P.P.T adversary $\mathcal{A}$ is negligible in the following game:

1. Challenger chooses n i.i.d samples $\boldsymbol{\beta} \leftarrow \mathcal{D}_2^n$.
2. The adversary $\mathcal{A}$ does the following:
   (a) It chooses m functions $f_1, \ldots, f_m \in \mathcal{F}$.

[7]We note that this is the usual “flooding” argument, but here we flood using a uniform random variable rather than the more standard Gaussian. This makes the argument even simpler.

   (b) It samples $\boldsymbol{\mu} \leftarrow \mathcal{D}_1^w$.

   (c) It returns $\{f_i, f_i(\boldsymbol{\mu})\}_{i\in[m]}$ to the challenger.

3. The challenger chooses m functions $G_1, \ldots, G_m \in \mathcal{G}$. It tosses a coin b. If $b = 0$, it returns $\{f_i(\boldsymbol{\mu}) + G_i(\boldsymbol{\beta})\}_{i\in[m]}$, else it returns $\{G_i(\boldsymbol{\beta})\}_{i\in[m]}$.

4. The adversary outputs a guess for the bit b and wins if correct.

We will refer to $\boldsymbol{\beta}$ as the *seed* of the CNG. We say that an CNG has polynomial stretch if $m = n^{1+c}$ for some constant $c > 0$.

Blockwise local CNG is defined analogously to blockwise local PRG, please see Section 2.1 for details.

Discussion. As discussed in Section 1, a correlated noise generator is weaker than a pseudorandom generator, in that, a PRG implies an CNG but not the other way around. As discussed in Section 4.3.1, by choosing the range of a PRG G to have norm bounded by a value which is superpolynomially larger than $\|f_i(\boldsymbol{\mu})\|$ for all $i \in [m]$, we have that $G_i(\boldsymbol{\beta}) + f_i(\boldsymbol{\mu})$ is indistinguishable from $G_i(\boldsymbol{\beta})$, by a standard "flooding" argument.

Our construction will use CNG for flooding noise terms as was done by PRG in Section 4, claim 4.3, but now leveraging the fact that the noise terms that must be flooded are correlated in a special way, not i.i.d. as to require PRG. Intuitively, flooding is performed to wipe out leakage otherwise provided by noise terms, but the flooded noise term must still be small enough to maintain correctness of the scheme. Hence the norm of the CNG output will be chosen so as to be smaller than $p_1/5$; we discuss this in detail in Section 7.

5.2 Non Boolean Pseudorandom Generators

In this section, we describe the properties we require from non-Boolean PRGs and discuss methods to instantiate these. As discussed in Section 1, in prior work [Lin17, LT17], Boolean PRGs were required in order to compute the binary randomness needed for constructing randomizing polynomials. Note that when instantiated with low degree candidates such as Goldreich's PRG [Gol00], the PRG output can be expressed as a degree 2 polynomial over $\mathbb{Z}_2$, however to compute this polynomial within a QuadFE scheme which only supports computations over $\mathbb{Z}$, the polynomial must be arithmetized, and this blows up the degree to the locality of the PRG, which is 5 for the case of [Gol00].

In our case, the PRG output must satisfy a much weaker property than indistinguishability to uniform as discussed above. Say we can bound $\|f_i(\boldsymbol{\mu})\| \leq \epsilon$ for $i \in [m]$. Then we require the PRG output $G_i(\boldsymbol{\beta})$ to computationally flood $f_i(\boldsymbol{\mu})$ for $i \in [m]$, i.e. $G_i(\boldsymbol{\beta}) + f_i(\boldsymbol{\mu})$ must be computationally indistinguishable from $G_i(\boldsymbol{\beta})$. We conjecture that degree 2 polynomials over $\mathbb{Z}$ may be used to compute $G_i(\boldsymbol{\beta})$ via the multivariate quadratic MQ assumption.

At a high level, the MQ assumption states that given m multivariate quadratic polynomials $p_i(\mathbf{x})$ in n variables $\mathbf{x} = (x_1, \ldots, x_n)$, where $m > n$, over a finite field $\mathbb{F}$, it is infeasible to recover $\mathbf{x}$ for appropriate values of m, n and choice of polynomials p_i. The MQ assumption has been studied extensively [MI88, BFP09, BFSS13, BFS03, Wol05, DY09, YDH+15, WP05, TW10, AHKI+17] and has formed the basis of several cryptosystems, please see [DY09] for a survey. MQ is known to be NP complete in the worst case, and the general method to solve MQ involves computing Gröbner bases, which are notoriously hard to compute [Wol05]. Please see Section 2 for a precise statement of the assumption.

We note that our setting is different from the standard setting of MQ in that in our case, the quadratic polynomials are computed over (a bounded range of) $\mathbb{Z}$ rather than over a finite field. Even for this setting however, the problem is no easier than the standard case of finite fields to the best of our knowledge, indeed, computing Gröbner bases in this setting typically involves reducing the equations modulo some prime and then applying the tools of the standard finite field setting [Wol05].

We conjecture that MQ can be used to instantiate the non-Boolean PRG we require by choosing parameters so that the output of the polynomials is super-polynomially larger than ϵ, the size of the term it seeks to flood. We do not insist on a concrete candidate here, but note that randomly chosen polynomials are often considered good candidates (please see [Wol05] for a discussion on how to construct hard MQ problems) for m, n as in our case, please see Section 7. We leave further studies on such PRG to future work, but note that we do not impose any sparsity requirement on the non-Boolean PRG (please see Section 7), and hence the general attack of [BBKK17] does not seem to apply.

6 Functional Encryption for NC_1

In this section, we construct a functional encryption scheme for NC_1, denoted by FeNC_1, from a correlated noise generator CNG, the RLWE assumption and a noisy linear functional encryption scheme NLinFE. Our construction is a generalisation of the quadratic FE scheme provided in Section 4 and makes use of the public key and ciphertext evaluation algorithms developed by Agrawal and Rosen [AR17].

Background. Let $R = \mathbb{Z}[x]/(\phi)$ where $\phi = x^d + 1$ and d is a power of 2. Let $R_p \triangleq R/pR$ for any large prime p satisfying $p = 1 \mod 2n$.

We consider arithmetic circuits $\mathcal{F} : R_{p_0}^w \to R_{p_0}$ of depth d, consisting of alternate addition and multiplication layers. For circuits with long output, say ℓ, we consider ℓ functions, one computing each output bit. For $k \in [d]$, layer k of the circuit is associated with a modulus p_k. For an addition layer at level k, the modulus p_k will be the same as the previous modulus p_{k-1}; for a multiplication layer at level k, we require $p_k > p_{k-1}$. Thus, we get a tower of moduli $p_1 < p_2 = p_3 < p_4 = \ldots < p_d$. We define encoding functions $\mathcal{E}^k$ for $k \in [d]$ such that $\mathcal{E}^k : R_{p_{k-1}} \to R_{p_k}$. The message space of the scheme FeNC_1 is R_{p_0}.

At level k, the encryptor will provide L^k encodings, denoted by $\mathcal{C}^k$, for some $L^k = O(2^k)$. For $i \in [L^k]$ we define

$$\mathcal{E}^k(y_i) = u_i^k \cdot s + p_{k-1} \cdot \eta_i^k + y_i.$$

Here $u_i^k \in R_{p_k}$ is called the "label" or "public key" of the encoding, η_i^k is noise chosen from some distribution χ_k, $s \leftarrow R_{p_1}$ is the RLWE secret, and $y_i \in R_{p_{k-1}}$ is the message being encoded. We will refer to $\mathcal{E}^k(y_i)$ as the *Regev encoding* of y_i. We denote:

$$\mathsf{PK}\big(\mathcal{E}^k(y_i)\big) \triangleq u_i^k, \quad \mathsf{Nse}(\mathcal{C}^k) \triangleq p_{k-1} \cdot \eta_i^k$$

The messages encoded in level k encodings $\mathcal{C}^k$ are denoted by $\mathcal{M}^k$.

Agrawal and Rosen [AR17] show that at level k, the decryptor is able to compute a Regev encoding of functional message $f^k(\mathbf{x})$ where f^k is the circuit f restricted to level k. Formally:

Theorem 6.1. *[AR17] There exists a set of encodings $\mathcal{C}^i$ for $i \in [d]$, such that:*

1. **Encodings have size sublinear in circuit.** $\forall i \in [d]\ |\mathcal{C}^i| = O(2^i)$.

2. **Efficient public key and ciphertext evaluation algorithms.** *There exist efficient algorithms $\mathsf{Eval}_{\mathsf{PK}}$ and $\mathsf{Eval}_{\mathsf{CT}}$ so that for any circuit f of depth d, if $\mathsf{PK}_f \leftarrow \mathsf{Eval}_{\mathsf{PK}}(\mathsf{PK}, f)$ and $\mathsf{CT}_{(f(\mathbf{x}))} \leftarrow \mathsf{Eval}_{\mathsf{CT}}(\underset{i\in[d]}{\cup} \mathcal{C}^i, f)$, then $\mathsf{CT}_{(f(\mathbf{x}))}$ is a "Regev encoding" of $f(\mathbf{x})$ under public key PK_f. Specifically, for some LWE secret s, we have:*
$$\mathsf{CT}_{(f(\mathbf{x}))} = \mathsf{PK}_f \cdot s + p_{d-1} \cdot \eta_f^{d-1} + \mu_{f(\mathbf{x})} + f(\mathbf{x}) \tag{6.1}$$
where $p_{d-1} \cdot \eta_f^{d-1}$ is RLWE noise and $\mu_{f(\mathbf{x})} + f(\mathbf{x})$ is the desired message $f(\mathbf{x})$ plus some noise $\mu_{f(\mathbf{x})}$[8].

3. **Ciphertext and public key structure.** *The structure of the functional ciphertext is as:*
$$\mathsf{CT}_{f(\mathbf{x})} = \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) + \langle \mathsf{Lin}_f, \mathcal{C}^d \rangle \tag{6.2}$$
where $\mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) \in R_{p_{d-1}}$ is a high degree polynomial value obtained by computing a public f-dependent function on level $k \le d-1$ encodings $\{\mathcal{C}^k\}_{k\in[d-1]}$ and $\mathsf{Lin}_f \in R_{p_d}^{L_d}$ computed by $\mathsf{Eval}_{\mathsf{PK}}(\mathsf{PK}, f)$ is an f-dependent linear function. We also have
$$f(\mathbf{x}) + \mu_{f(\mathbf{x})} = \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) + \langle \mathsf{Lin}_f, \mathcal{M}^d \rangle \tag{6.3}$$
where $\mathcal{M}^d$ are the messages encoded in $\mathcal{C}^d$ and $\mu_{f(\mathbf{x})}$ is functional noise. The public key for the functional ciphertext is structured as:
$$\mathsf{PK}\big(\mathsf{CT}_{f(\mathbf{x})}\big) = \Big\langle \mathsf{Lin}_f,\ \big(\mathsf{PK}(\mathcal{C}_1^d), \ldots, \mathsf{PK}(\mathcal{C}_{L_d}^d)\big) \Big\rangle \tag{6.4}$$

The Encodings. The encodings $\mathcal{C}^k$ for $k \in [d]$ are defined recursively as:

1. $\mathcal{C}^1 \triangleq \{\mathcal{E}^1(x_i), \mathcal{E}^1(s)\}$
2. If k is a multiplication layer, $\mathcal{C}^k = \{\mathcal{E}^k(\mathcal{C}^{k-1}), \mathcal{E}^k(\mathcal{C}^{k-1} \cdot s), \mathcal{E}^k(s^2)\}$[9]. If k is an addition layer, let $\mathcal{C}^k = \mathcal{C}^{k-1}$.

As in the case of QuadFE, we will use NLinFE to enable the decryptor to compute $\langle \mathsf{Lin}_f, \mathcal{M}^d \rangle + G_f(\boldsymbol{\beta})$ where $G_f(\boldsymbol{\beta})$ is a large noise term that is meant to "flood" functional noise $\mu_{f(\mathbf{x})}$. She may then compute $\mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1})$ herself and by Equation 6.3 recover $f(\mathbf{x}) + \mu_{f(\mathbf{x})} + G_f(\boldsymbol{\beta})$.

In Section 4, G_f was chosen to be a PRG; here, we will use the weaker primitive of correlated noise generator CNG to instantiate G_f. As discussed in Section 1 and Section 5, CNG leverages the fact that the noise it must flood has special structure and in particular lower entropy than i.i.d random variables.

[8]Here $\mu_{f(\mathbf{x})}$ is clubbed with the message $f(\mathbf{x})$ rather than the RLWE noise $p_{d-1} \cdot \eta_f^{d-1}$ since $\mu_{f(\mathbf{x})} + f(\mathbf{x})$ is what will be recovered after decryption of $\mathsf{CT}_{f(\mathbf{x})}$.

[9]Here, we use the same secret s for all RLWE samples, but this is for ease of exposition – it is possible to have a different secret at each level so that circular security need not be assumed. We do not describe this extension here.

6.1 Construction

We will instantiate the parameters of NLinFE and CNG in Section 6.4. Next, we proceed to describe the construction. The construction below supports a single function of output length ℓ or equivalently ℓ functions with constant size output (however, in this case ℓ must be fixed in advance and input to all algorithms). The ciphertext size is sublinear in the size of the circuit, as discussed in Section 6.2.

$\mathsf{FeNC_1.Setup}(1^\kappa, 1^w, 1^d)$: Upon input the security parameter κ, the message dimension w, and the circuit depth d, do:

1. For $k \in [d]$, let $L_k = |\mathcal{C}^k|$ where $\mathcal{C}^k$ is as defined in Theorem 6.1. For $k \in [d-1]$, $i \in [L_k]$, choose uniformly random $u_i^k \in R_{p_k}$. Denote $\mathbf{u}^k = (u_i^k) \in R_{p_k}^{L_k}$.
2. Invoke $\mathsf{NLinFE.Setup}(1^\kappa, 1^{L_d}, p_d)$ to obtain $\mathsf{PK} = \mathsf{NLinFE.PK}$ and $\mathsf{MSK} = \mathsf{NLinFE.MSK}$.
3. Sample a CNG seed $\boldsymbol{\beta} \leftarrow \mathcal{D}^n_{\mathsf{seed}}$. Sample $t_0, \ldots, t_{L_d} \leftarrow R_{p_{d-1}}$ and let $\mathbf{t} = (t_0, \ldots, t_{L_d})$.
4. Output $\mathsf{PK} = (\mathbf{u}^1, \ldots, \mathbf{u}^{d-1}, \mathsf{NLinFE.PK})$ and $\mathsf{MSK} = (\mathsf{NLinFE.MSK}, \boldsymbol{\beta}, \mathbf{t})$.

$\mathsf{FeNC_1.KeyGen}(\mathsf{MSK}, f)$: Upon input the master secret key $\mathsf{NLinFE.MSK}$, CNG seed $\boldsymbol{\beta}$ and a circuit $f : R_{p_0}^w \to R_{p_0}$[10] of depth d, do:

1. Let $\mathsf{Lin}_f \leftarrow \mathsf{Eval_{PK}}(\mathsf{PK}, f) \in R_{p_d}^{L_d}$ as described in Equation 6.4.
2. Let G_f denote the CNG chosen corresponding to function f as described in Section 6.4.
3. Compute $\mathsf{key}_f = \langle \mathsf{Lin}_f, \mathbf{t} \rangle - G_f(\boldsymbol{\beta})$.
4. Let $\mathsf{SK}_f = \mathsf{NLinFE.KeyGen}(\mathsf{MSK}, \mathsf{Lin}_f \| \mathsf{key}_f)$.

$\mathsf{FeNC_1.Enc}(\mathbf{x}, \mathsf{PK})$: Upon input the public key and the input $\mathbf{x}$, do:

1. Compute the encodings $\mathcal{C}^k$ for $k \in [d-1]$ as defined in Theorem 6.1. Denote by s the RLWE secret used for these encodings.
2. Define $\mathcal{M}^d = \big(\mathcal{C}^{d-1},\ \mathcal{C}^{d-1} \cdot s,\ s^2 \big) \in R_{p_d}^{L_d}$. Compute $\mathcal{C}^d = \mathsf{NLinFE.Enc}(\mathsf{NLinFE.PK}, \mathcal{M}^d)$.
3. Output $\mathsf{CT}_{\mathbf{x}} = (\{\mathcal{C}^k\}_{k \in [d]})$.

$\mathsf{FeNC_1.Dec}(\mathsf{PK}, \mathsf{CT}_{\mathbf{x}}, \mathsf{SK}_f)$: Upon input a ciphertext $\mathsf{CT}_{\mathbf{x}}$ for vector $\mathbf{x}$, and a secret key SK_f for circuit f, do:

1. Compute $\mathsf{CT}_{f(\mathbf{x})} = \mathsf{Eval_{CT}}(\{\mathcal{C}^k\}_{k \in [d-1]}, f)$. Express $\mathsf{CT}_{f(\mathbf{x})} = \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) + \langle \mathsf{Lin}_f, \mathcal{C}^d \rangle$ as described in Equation 6.2.
2. Compute $\mathsf{NLinFE.Dec}(\mathsf{SK}_f, \mathcal{C}^d)$ to obtain $\langle \mathsf{Lin}_f, \mathcal{M}^d \rangle + \eta_f$ for some noise η_f added by NLinFE.
3. Compute $\mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) + \langle \mathsf{Lin}_f, \mathcal{M}^d \rangle + \eta_f \mod p_d \mod p_{d-1}, \ldots, \mod p_0$ and output it.

[10] We will let the adversary request ℓ functions

Correctness follows from correctness of $\mathsf{Eval}_{\mathsf{PK}}$, $\mathsf{Eval}_{\mathsf{CT}}$ and NLinFE. We have by correctness of $\mathsf{Eval}_{\mathsf{PK}}$, $\mathsf{Eval}_{\mathsf{CT}}$ that:

$$\begin{aligned} \mathsf{CT}_{f(\mathbf{x})} &= \langle \mathsf{Lin}_f, \mathcal{C}^d \rangle + \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) \\ \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) + \mathsf{NLinFE.Dec}(\mathsf{SK}_f, \mathcal{C}^d) &= \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) + \langle \mathsf{Lin}_f, \mathcal{M}^d \rangle + \eta_f \\ &= f(\mathbf{x}) + \mu_{f(\mathbf{x})} + \eta_f \text{ by theorem 6.1} \\ &= f(\mathbf{x}) \mod p_d \mod p_{d-1}, \ldots, \mod p_0 \end{aligned}$$

where the last step follows since $\mu_{f(\mathbf{x})}$ and η_f are linear combinations of noise terms, each noise term being a multiple of p_k for $k \in \{0, \ldots, d-1\}$. For details regarding the structure of the noise terms, please see Appendix E.

6.2 Ciphertext Size

The size of the ciphertext is $|\bigcup_{k\in[d-1]} \mathcal{C}^k| + |\mathsf{NLinFE.CT}(\mathcal{M}^d)|$. Note that $|\bigcup_{k\in[d-1]} \mathcal{C}^k| = O(2^d)$ and $|\mathcal{M}^d| = O(2^d)$ by Theorem 6.1. All our constructions of NLinFE will have compact ciphertext (please see Sections 7 and 8), hence the ciphertext of the above scheme is also sublinear in circuit size.

6.3 Proof of Security.

In this section, we provide the proof of security. The proof is nearly identical to that in Section 4, generalised with $\mathsf{Eval}_{\mathsf{PK}}$ and $\mathsf{Eval}_{\mathsf{CT}}$.

Theorem 6.2. *Let* NLinFE *and* CNG *be instantiated as described in Section 6.4. Assume the noisy linear FE scheme* NLinFE *satisfies semi-adaptive indistinguishability based security as in Definition 3.4 and that G is a secure* CNG *as defined in Definition 5.1. Then, the construction* FeNC_1 *achieves semi-adaptive indistinguishability based security in the single key game described in Definition 2.4.*

Proof. We will prove the theorem via a sequence of hybrids, where the first hybrid is the real world with challenge $\mathbf{x}_0$ and the last hybrid is the real world with challenge $\mathbf{x}_1$.

The Hybrids. Our Hybrids are described below.

Hybrid 0. This is the real world with message $\mathbf{x}_0$. In hybrid 0, the element key_f in the $\mathsf{FeNC}_1.\mathsf{KeyGen}$ procedure is picked as follows: sample $t_0, \ldots, t_{L_d} \leftarrow R_{p_1}$ and denote $\mathbf{t} = (t_0, \ldots, t_{L_d})$. Let

$$\mathsf{key}_f = \langle \mathsf{Lin}_f, \mathbf{t} \rangle - G_f(\boldsymbol{\beta})$$

Hybrid 1. In this hybrid, the only thing that is different is that the challenger picks key_f to depend on the challenge ciphertext. In more detail,

1. Sample $t_0, \ldots, t_{L_d} \leftarrow R_{p_{d-1}}$ and compute $G_f(\boldsymbol{\beta})$ as in Hybrid 0. Denote $\mathbf{t} = (t_0, \ldots, t_{L_d})$.
2. Set
$$\mathsf{key}_f = f(\mathbf{x}) - \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) - \langle \mathsf{Lin}_f, \mathbf{t} \rangle - G_f(\boldsymbol{\beta})$$

Hybrid 2. In this hybrid, we change the input for $\mathsf{NLinFE.Enc}$ to $(t_0, t_1, \ldots, t_{L_d}, 1)$ where t_i for $i \in [L_d]$, are chosen as in Hybrid 1.

Hybrid 3. In this hybrid, we change the message vector in $\underset{k\in[d-1]}{\cup} \mathcal{C}^k$ to $\mathbf{x}_1$.

Hybrids 4 and 5. In Hybrid 4 we change the input to $\mathsf{NLinFE.Enc}$ to $(\mathcal{M}^d)$ as in Hybrid 1. In Hybrid 5, we change key_f to be chosen independent of the ciphertext as in Hybrid 0. This is the real world with message $\mathbf{x}_1$.

Indistinguishability of Hybrids.

Lemma 6.3. *Hybrid 0 and Hybrid 1 are indistinguishable by the security of* CNG*.*

Proof. In Hybrid 0, we set

$$\mathsf{key}_{f_i} = \langle \mathsf{Lin}_{f_i}, \mathbf{t} \rangle - G_{f_i}(\boldsymbol{\beta}), \quad \forall i \in [\ell]$$

In Hybrid 1, we set

$$\mathsf{key}_{f_i} = f_i(\mathbf{x}) - \mathsf{Poly}_{f_i}(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) - \langle \mathsf{Lin}_{f_i}, \mathbf{t} \rangle - G_{f_i}(\boldsymbol{\beta})$$

We have by Theorem 6.1 that

$$\begin{aligned} \langle \mathsf{Lin}_{f_i}, \mathcal{M}^d \rangle + \mathsf{Poly}_{f_i}(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) &= f_i(\mathbf{x}) + \mu_{f_i(\mathbf{x})} \\ \text{Hence, } f_i(\mathbf{x}) - \mathsf{Poly}_{f_i}(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) &= \langle \mathsf{Lin}_{f_i}, \mathcal{M}^d \rangle - \mu_{f_i(\mathbf{x})} \end{aligned}$$

Hence, we have in Hybrid 1,

$$\begin{aligned} \mathsf{key}_{f_i} &= \langle \mathsf{Lin}_{f_i}, \mathcal{M}^d \rangle - \mu_{f_i(\mathbf{x})} - \langle \mathsf{Lin}_{f_i}, \mathbf{t} \rangle - G_{f_i}(\boldsymbol{\beta}) \\ &= \langle \mathsf{Lin}_{f_i}, \mathcal{M}^d - \mathbf{t} \rangle - (\mu_{f_i(\mathbf{x})} + G_{f_i}(\boldsymbol{\beta})) \\ &= \langle \mathsf{Lin}_{f_i}, \mathbf{t}' \rangle - (\mu_{f_i(\mathbf{x})} + G_{f_i}(\boldsymbol{\beta})) \end{aligned}$$

Above we set $\mathbf{t}' = \mathcal{M}^d - \mathbf{t}$. Since $\mathbf{t}$ is chosen randomly, we have that $\mathbf{t}'$ is distributed uniformly over $R_{p_{d-1}}$.

Next, we claim for CNG instantiated as in Section 6.4, we have $\mu_{f_i(\mathbf{x})} + G_{f_i}(\boldsymbol{\beta}) \overset{c}{\approx} G_{f_i}(\boldsymbol{\beta})$ by the security of CNG. Formally, given an adversary Adv who distinguishes between Hybrid 0 and 1, we construct an adversary $\mathsf{Adv}_{\mathsf{CNG}}$ against CNG 5 as follows:

1. $\mathsf{Adv}_{\mathsf{CNG}}$ expresses $\mu_{f_i(\mathbf{x})} = \hat{f}_i\big(\mathcal{M}^1, \mathsf{Nse}(\mathcal{C}^1), \ldots, \mathcal{M}^d, \mathsf{Nse}(\mathcal{C}^d)\big)$ as described in Section 6.4 for all $i \in [\ell]$ (corresponding to ℓ output bits) and sends $(\hat{f}_i, \mu_{f_i(\mathbf{x})}$ to the CNG challenger.

2. The CNG challenger chooses CNG G_{f_i} for $i \in [\ell]$ and seed $\boldsymbol{\beta} \leftarrow \mathcal{D}^n_{\mathsf{seed}}$, a random bit b and returns $z_i = G_{f_i}(\boldsymbol{\beta})$ if $b = 0$ and $z_i = G_{f_i}(\boldsymbol{\beta}) + \mu_{f_i(\mathbf{x})}$ if $b = 1$.

3. $\mathsf{Adv}_{\mathsf{CNG}}$ computes $\mathsf{key}_{f_i} = \langle \mathsf{Lin}_{f_i}, \mathbf{t}' \rangle - z_i$. It computes all other elements as in Hybrid 0 and returns this to Adv.

4. It outputs whatever Adv outputs.

Note that the reduction $\mathsf{Adv}_{\mathsf{CNG}}$ is a valid adversary against the CNG. Additionally, if $b = 0$, we are in Hybrid 0, else in Hybrid 1, hence the advantage of $\mathsf{Adv}_{\mathsf{CNG}}$ translates to an advantage of Adv. $\square$

Indistinguishability of remaining Hybrids is exactly as in Section 4.3. In more detail, Hybrid 1 and Hybrid 2 are indistinguishable by the security of NLinFE because the challenge decryption in both Hybrids is equal up to an additive error with the appropriate distribution. To see this, note that in Hybrid 1, NLinFE decryption gives:

$$\langle \mathsf{Lin}_f, \mathcal{M}^d \rangle$$

and in Hybrid 2, NLinFE decryption gives:

$$\begin{aligned}
\langle \mathsf{Lin}_f, \mathbf{t} \rangle + \mathsf{key}_f &= \langle \mathsf{Lin}_f, \mathbf{t} \rangle + \big(f(\mathbf{x}) - \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) - \langle \mathsf{Lin}_f, \mathbf{t} \rangle - G_f(\boldsymbol{\beta}) \big) \\
&= f(\mathbf{x}) - \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) - G_f(\boldsymbol{\beta}) \\
&= \langle \mathsf{Lin}_f, \mathcal{M}^d \rangle - \mu_{f(\mathbf{x})} - G_f(\boldsymbol{\beta}) \text{ by Theorem 6.1} \\
&= \langle \mathsf{Lin}_f, \mathcal{M}^d \rangle - \big(\mu_{f(\mathbf{x})} + G_f(\boldsymbol{\beta}) \big)
\end{aligned}$$

Thus, the challenge message evaluation on the requested key differs by an additive term of

$$\big(\mu_{f(\mathbf{x})} + G_f(\boldsymbol{\beta}) \big) \tag{6.5}$$

which, for our choice of parameters (see Section 6.4) and by guarantee of NLinFE implies indistinguishability of NLinFE ciphertexts in Hybrids 1 and 2. The formal reduction is exactly as in Claim 4.4.

Indistinguishability of Hybrids 2 and 3 follows exactly as in Claim 4.5. Intuitively, now that the NLinFE message is independent of the encodings $\underset{i \in [d-1]}{\cup} \mathcal{C}^k$, we may switch the message in the encodings to $\mathbf{x}_1$ by the semantic security of Regev encodings. Note that decryption still works correctly because the term key_{f_i} for $i \in [\ell]$ in the functional key compensates for the NLinFE message, exactly as in Section 4.3. $\square$

6.4 Instantiating CNG and NLinFE

Instantiating the CNG. To instantiate the CNG, we must analyse the distribution of the noise term $\mu_{f(\mathbf{x})}$ that occurs in Equation 6.3. The final noise term $\mu_{f(\mathbf{x})}$ is a high degree polynomial computed on $L = \big| \underset{i \in [d]}{\cup} \mathcal{C}^i \big|$ noise terms and message terms that are used in the encodings $\mathcal{C}^i$ for $i \in [d]$. A detailed analysis of this distribution is provided in Appendix E.2, here we give an intuitive overview. The disinterested reader may safely skip this section – it is not very important what is the exact distribution of $\mu_{f(\mathbf{x})}$ as long as it can be understood as a circuit computed on $O(L)$ noise terms where L is sublinear in $|f|$.

The distribution of $\mu_{f(\mathbf{x})}$ may be analysed by proceeding from bottom to top of the circuit as follows. At level 1, for computing a multiplication layer, we require $(\mathcal{D}^1_{\mathsf{CNG}}, \mathcal{F}^1_{\mathsf{CNG}})$-$\mathsf{CNG}$ where to sample from $\mathcal{D}^1_{\mathsf{CNG}}$, we choose $\mathbf{x} \in R^w_{p_0}$ arbitrarily, and $\boldsymbol{\mu} \leftarrow \mathcal{D}^w_0$, where $\mathcal{D}_0$ is a discrete Gaussian. Since $\mathbf{x}$ is the message encoded at level 1, it is denoted by $\mathcal{M}^1$ using the notation above and since $\boldsymbol{\mu}$

is the noise used in level 1 encodings, this is denoted by $\mathsf{Nse}(\mathcal{C}^1)$. Thus, $\mathcal{D}^1_{\mathsf{CNG}}$ is defined by sampling as $(\mathcal{M}^1, \mathsf{Nse}(\mathcal{C}^1))$ as specified by the encrypt algorithm. The function family $\mathcal{F}^1_{\mathsf{CNG}}$ is described as follows. Let k be a multiplication gate taking as input wires i and j and let $\ell^1 = |1 \leq j \leq i \leq w|$. Then, $\mathcal{F}^1_{\mathsf{CNG}} = \{\ \hat{f}^1_k\ \}_{k \in [\ell^1]}$ where,

$$\mathsf{Mult:}\ \ \hat{f}^1_k(\mathbf{x}, \boldsymbol{\mu}) = p_0 \cdot \big(p_0 \cdot \mu_i\mu_j + x_i\mu_j + x_j\mu_i\big) \tag{6.6}$$

At the next (addition) layer, $\mathcal{D}^2_{\mathsf{CNG}} = \mathcal{D}^1_{\mathsf{CNG}}$ and the function family $\mathcal{F}^2_{\mathsf{CNG}}$ is described as follows. Let g be an addition gate at level 2 taking input wires k and k' and let ℓ^2 denote the number of addition gates at level 2. Then, $\mathcal{F}^2_{\mathsf{CNG}} = \{\ \hat{f}^2_g\ \}_{g \in [\ell^2]}$

$$\mathsf{Add:}\ \ \hat{f}^2_g(\mathbf{x}, \boldsymbol{\mu}) = \hat{f}^1_k(\mathbf{x}, \boldsymbol{\mu}) + \hat{f}^1_{k'}(\mathbf{x}, \boldsymbol{\mu}) \tag{6.7}$$

Note that $\hat{f}^1_k(\mathbf{x}, \boldsymbol{\mu})$ and $\hat{f}^1_{k'}(\mathbf{x}, \boldsymbol{\mu})$ are computed as described in Equation 6.6. Generalizing, the distribution $\mathcal{D}^d_{\mathsf{CNG}}$ is defined by sampling from the vector $(\mathcal{M}^1, \mathsf{Nse}(\mathcal{C}^1), \ldots, \mathcal{M}^d, \mathsf{Nse}(\mathcal{C}^d))$ as specified by the encrypt algorithm. The function $\hat{f}^d$ corresponding to circuit f in the function key is computed bottom up using the analogue of Equation 6.6 for multiplication and the analogue of Equation 6.7 for addition.

We note that the final noise is a high degree polynomial in the message and noise terms that were used to compute the encodings $\{\mathcal{C}^k\}_{k \in [d]}$, i.e. a function of $\sum\limits_{k \in [d]} |\mathcal{C}_k| = O(2^d)$ terms.

Thus, we may instantiate the correlated noise generator $(\mathcal{D}^d_{\mathsf{CNG}}, \mathcal{F}^d_{\mathsf{CNG}})$-$\mathsf{CNG}$, with seed $\boldsymbol{\beta}$ so that if $\hat{f}$ is the circuit that computes the noise $\mu_{f(\mathbf{x})}$ corresponding to ciphertext evaluation on f, and G_f is the CNG chosen to flood $\mu_{f(\mathbf{x})}$, we have:

$$\hat{f}\big(\mathcal{M}^1, \mathsf{Nse}(\mathcal{C}^1), \ldots, \mathcal{M}^d, \mathsf{Nse}(\mathcal{C}^d)\big) + G_f(\boldsymbol{\beta}) \overset{c}{\approx} G_f(\boldsymbol{\beta})$$

Instantiating the Noisy Linear FE. We will also require a noisy linear functional encryption scheme as in Section 4, $(\mathcal{D}_{\mathsf{NFE}}, \mathcal{F}_{\mathsf{NFE}}, B_{\mathsf{NFE}})$-$\mathsf{NLinFE}$ where the parameters $(\mathcal{D}_{\mathsf{NFE}}, \mathcal{F}_{\mathsf{NFE}}, B_{\mathsf{NFE}})$ are computed as:

1. $\mathbf{y} \leftarrow \mathcal{D}_{\mathsf{NFE}}$ if $\mathbf{y} = \Big(\mathcal{M}^1, \mathsf{Nse}(\mathcal{C}^1), \ldots, \mathcal{M}^d, \mathsf{Nse}(\mathcal{C}^d), \boldsymbol{\beta}\Big)$ where each of the components is sampled as above.
2. $h(\mathbf{y}) = \hat{f}\big((\mathcal{M}^1, \mathsf{Nse}(\mathcal{C}^1), \ldots, \mathcal{M}^d, \mathsf{Nse}(\mathcal{C}^d)\big) + G_f(\boldsymbol{\beta})$ for $h \in \mathcal{F}_{\mathsf{NFE}}$.
3. B_{NFE} is bounded below $p_d/5$ to allow decryption and is chosen superpolynomially larger than any $|h(\mathbf{y})|$ above, for reasons discussed in Section 7. We set the parameters in Section 9.

7 Constructing Noisy Linear Functional Encryption

In this section, we provide a construction of noisy linear functional encryption NLinFE as defined in Section 3. The ciphertext space for our construction is a ring R_q for some prime q where $R = \mathbb{Z}[x]/(x^d + 1)$ and $R_q = R/qR$ and the plaintext space is R_p for some prime $p << q$.

Our construction of NLinFE relies on:

1. In the public key setting, an L blockwise local PRG with input size n, block size b and stretch $\Omega(n \cdot 2^{b(1+\epsilon)})$ and a compact FE scheme supporting polynomials of degree L.

2. In the private key setting, an L blockwise local correlated noise swallowing generator CNG with input size n, block size b and stretch $\Omega(n \cdot 2^{b(1+\epsilon)})$ and a compact function hiding FE scheme supporting polynomials of degree L.

Intuition. Recall that NLinFE requires that if two challenge messages $\mathbf{x}_0, \mathbf{x}_1$ evaluate to only approximately the same value for a given function key, their ciphertexts should be indistinguishable. Evidently, to prevent functionality itself from nullifying security, the decryption values must be suitably modified so as to be indistinguishable.

In our applications of NLinFE, the difference in decryption values is a random variable of fixed distribution and bounded norm. For instance, we see in Section 6 that the challenge message decryption values differ by the term $\big(\mu_{f(\mathbf{x})} + G_f(\boldsymbol{\beta})\big)$[11] (please see Equation 6.5) where $\mu_{f(\mathbf{x})}$ and $G_f(\boldsymbol{\beta})$ are deterministic functions of random variables from known distributions. To make the decryption values indistinguishable, all our constructions will add a carefully constructed noise term, $\mathsf{noise_{fld}}$ (say), generated either using PRG or CNG so as to smudge the above term and make the resultant term simulatable. In more detail, we require:

$$\big(\mu_{f(\mathbf{x})} + G_f(\boldsymbol{\beta})\big) + \mathsf{noise_{fld}} \overset{c}{\approx} \mathsf{noise_{fld}}$$

Now, the decryptor recovers either $\mathsf{val} + \big(\mu_{f(\mathbf{x})} + G_f(\boldsymbol{\beta})\big) + \mathsf{noise_{fld}}$ or $\mathsf{val} + \mathsf{noise_{fld}}$ for some value $\mathsf{val} \in R_q$, which are indistinguishable by the choice of added noise.

In order to preserve functionality of the system, the added noise must not be so large as to completely randomize the decrypted value val – to achieve this, the added noise term will still have norm that is suitably bounded below the modulus size q, and live in a fixed ideal that is modded out to recover val exactly as in fully homomorphic encryption (FHE). Indeed, this is not a coincidence since we will be using NLinFE to compute an FHE decryption equation, as discussed in Section 4.

In summary, we will use an FE scheme to construct a PRG or CNG noise term which must be large enough to smudge the leaky distribution of the aforementioned noise term, but small enough to be removable in the final decryption process. Since our FE must construct a linear equation plus noise[12], the degree of the polynomial that FE must support is equal to the degree required to construct the noise term.

7.1 PRG with non-Boolean output.

As discussed above, we will require our PRG or CNG to generate elements in R_q of bounded size (please see Section 6 and Section 4 to confirm this). Here, we discuss two approaches to achieve the same.

Using Boolean PRG to construct non-Boolean PRG. In our first approach, we will use a standard binary PRG to generate coefficients of the ring element, so that each coefficient is bounded by some value $\mathsf{B_{coef}} << q$ (say) as follows. Let r_i be pseudorandom bits output by the PRG,

[11] In the symmetric key setting, the challenge message decryption values need differ only by term $\mu_{f(\mathbf{x})}$

[12] We emphasize that we are not constructing LWE samples, our usage of noisy linear equations is much more general and does not follow the LWE distribution.

then $\sum_{i \in [\log\lceil \mathsf{B_{coef}} \rceil]} 2^i \cdot r_i$ generates a pseudorandom element bounded by $\mathsf{B_{coef}}$. Thus, to generate ℓ output ring elements of bounded norm, the binary PRG must output $m = \ell \cdot \log\lceil \mathsf{B_{coef}} \rceil \cdot d$ bits. Since the powers of 2 expansion is a linear function, the degree of the function that computes the pseudorandom ring element we require retains degree equal to that of the binary PRG. Let $R_{\mathsf{Bd}} \subset R_q$ be the set of elements in R_q with bounded norm as we require. We will instantiate this in Section 7.2.3.

Below, we will let $G^q : \{-1,1\}^n \to R_{\mathsf{Bd}}^{\ell}$ be a PRG that outputs ℓ ring elements of bounded coefficients directly. G_i^q will denote the function that outputs the i^{th} element for $i \in [\ell]$. Our construction of NLinFE will support ℓ function queries, and have ciphertext size sublinear in ℓ.

Direct construction of non-Boolean PRG. As discussed in Section 1, using a Boolean PRG to construct non-Boolean output is wasteful, since Boolean PRGs are more restricted than what we need, and indeed have the undesirable side-effect of requiring large arithmetic degree to compute.

As discussed in Section 5.2, we propose that the MQ (Multivariate Quadratic) assumption (please see [DY09] and references therein) be used in order to generate the term $\mathsf{noise_{fld}}$ via quadratic polynomials.

7.2 Public Key Noisy Linear Functional Encryption

Let PrgFE be an FE scheme supporting degree L polynomials, where L is the degree required to compute a pseudorandom generator as discussed above. Let Sym be a symmetric key encryption scheme. We show how to construct NLinFE from these. Our proof follows the strategy of embedding a hidden thread in the functionality which is only active during simulation [CIJ+13, ABSV15]. To convey intuition, we will describe the simplest scheme which does not leverage the blockwise locality of the PRG; we discuss in Section 7.2.1 how to pre-process the input and massage the functionality as in [LT17] so that the degree of the functionality is reduced to the block locality of the PRG.

$\mathsf{NLinFE.Setup}(1^\kappa, 1^w)$: Upon input the security parameter and length of input message, do the following:

1. Choose PRG. Let G^q be a PRG with input size n and output length ℓ as described above.
2. Invoke $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{PrgFE.Setup}(1^\kappa, 1^{w+n+\kappa+1})$ and output $(\mathsf{PK}, \mathsf{MSK})$.

$\mathsf{NLinFE.Enc}(\mathsf{PK}, \mathbf{z})$: Upon input the master secret key MSK and a vector $\mathbf{z} \in R_p^w$, do the following:

1. Sample a seed $\mathbf{s} \leftarrow \{-1,1\}^n$ for the PRG.
2. Compute $\mathsf{PrgFE.Enc}\big(\mathsf{PK}, (\mathbf{z}, \mathbf{s}, \perp, \mathsf{mode} = 0)\big)$.

$\mathsf{NLinFE.KeyGen}(\mathsf{MSK}, \mathbf{v}_i, \ldots \mathbf{v}_\ell)$: Upon input the master secret key MSK and ℓ vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\ell \in R_p^w$, do the following:

1. For $i \in [\ell]$, choose CT_i randomly from the space of Sym ciphertexts.
2. For $i \in [\ell]$, define functionality $g_{i,\mathbf{v}_i,\mathsf{CT}_i}$ in Figure 7.2
3. Output $\mathsf{PrgFE.KeyGen}(\mathsf{MSK}, g_i)$ for $i \in [\ell]$.

$\mathsf{NLinFE.Dec}(\mathsf{PK}, \mathsf{CT}_\mathbf{z}, \mathsf{SK}_\mathbf{v})$: Upon input the public key PK, a ciphertext $\mathsf{CT}_\mathbf{z}$ and a function key $\mathsf{SK}_\mathbf{v}$, compute $\mathsf{PrgFE.Dec}(\mathsf{PK}, \mathsf{CT}_\mathbf{z}, \mathsf{SK}_\mathbf{v})$ and output it.

Functionality $g_{i,\mathbf{v}_i,\mathsf{CT}_i}(\mathbf{z}, \mathsf{s}, \mathsf{Sym.K}, \mathsf{mode})$

If $\mathsf{mode} = 0$, output $y = \langle \mathbf{z},\ \mathbf{v}_i\rangle + G_i^q(\mathbf{s})\ \in R_q$ else output $y = \mathsf{Sym.Dec}(\mathsf{K}, \mathsf{CT})$.

Figure 7.1: Functionality $G_{i,\mathbf{v}_i,\mathsf{CT}_i}$

Correctness. We have by correctness of PrgFE that decryption recovers $\langle \mathbf{z},\ \mathbf{v}_i\rangle + G_i^q(\mathbf{s})$ as desired.

7.2.1 Shrinking Degree of Functionality.

The case of the Boolean PRG. In this section, we show how to pre-process the input and modify the function description so that the output can be computed by a degree L polynomial where L is the blockwise locality of the Boolean PRG.

To begin, we choose the encryption scheme Sym to simply be a one time pad with the mask computed using a PRG. In more detail, we let $\mathbf{s}'$ be a PRG seed, and encrypt a value y_i as $y_i \oplus G_i^q(\mathbf{s}')$. We set $\mathsf{Sym.K} = \mathbf{s}'$.

Then, the function $g_{i,\mathbf{v}_i,\mathsf{CT}_i}$ from Figure 7.2 may be written as the polynomial

$$g_{i,\mathbf{v}_i,\mathsf{CT}_i} = (1 - \mathsf{mode})\Big(\mathbf{z},\ \mathbf{v}_i\rangle + G_i^q(\mathbf{s})\Big) + \mathsf{mode} \cdot \Big(\mathsf{CT} \oplus G_i^q(\mathbf{s}')\Big) \tag{7.1}$$

The largest degree term in the above polynomial is $\mathsf{mode} \cdot G_i^q(\mathbf{s})$ which, if computed naively, would require degree $L' + 1$[13] where L' is the locality of G_i^q. However, by choosing G^q to be a block local PRG and pre-processing the input in a manner inspired from [LT17], we may express the above functionality as a degree L polynomial, where L is the blockwise locality of G_i^q. Since PRG may have much smaller block locality as compared to locality, this significantly reduces the degree required to be supported by the FE scheme.

We note that our pre-processing step is significantly simpler than that in [Lin17, LT17] since our functionality has a native degree of $L'+1$ as against the degree $3L'+2$ incurred by the bootstrapping of [LV16, Lin17, LT17].

In more detail, we define a function $\mathsf{PreProcess}$ as follows.

1. Sample two seeds $\mathbf{s}, \mathbf{s}' \leftarrow \{-1, 1\}^{n\cdot b}$ for the PRG. We view each seed as a matrix with n columns and b rows. We denote the n^{th} column of the matrices as s_γ and s'_γ respectively.

2. Compute all multilinear monomials in every block of size b. Define

$$\mathsf{Mnml}(A) \triangleq \{a_{i_1}a_{i_2}\ldots a_{i_q}\ |q \le |A|\ \text{ and } \forall\ j,k,\ a_{i_j} \ne a_{i_k} \in A\ \}$$

3. Define

$$V_1 = \{\mathsf{Mnml}(s_\gamma)\}_{\gamma\in[n]} \otimes (\mathsf{mode}||1), \quad V_2 = \{\mathsf{Mnml}(s'_\gamma)\}_{\gamma\in[n]} \otimes (\mathsf{mode}||1)$$

   Return $(V_1\|V_2)$.

[13]Note that multiplication with CT does not incur any additional degree (as in the case of [Lin17]) since this is a constant hardwired in the function.

Since all the monomials in a given column of the seed matrix are pre-computed, and since by definition of block locality the output of the PRG may depend on inputs from only L columns, it holds that the output of the PRG can be computed by a degree L polynomial. By further computing the product of these monomials with mode, we conclude that the polynomial that computes the function $g_{i,\mathbf{v}_i,\mathsf{CT}_i}$ defined above has degree L in the input $(\mathbf{z}, \|V_1\|V_2)$.

Thus, if instead of computing PrgFE encryption of message $(\mathbf{z}, \mathbf{s}, \perp, \mathsf{mode} = 0)$, we compute PrgFE encryption of input $(\mathbf{z}, \|V_1\|V_2)$, and if instead of computing the key corresponding to function $g_{i,\mathbf{v}_i,\mathsf{CT}_i}$ we compute the keys corresponding to the degree L polynomials that compute $g_{i,\mathbf{v}_i,\mathsf{CT}_i}$ on input $(\mathbf{z}, \|V_1\|V_2)$, we obtain that FE supporting evaluation of degree L polynomials suffices.

The case of the non-Boolean PRG. Note that if there exists a secure quadratic instantiation of non-Boolean PRG that suffices for our purpose, then we do not require the pre-processing above (except the pre-multiplication with mode), simplifying the pre-processing even further.

Ciphertext Size. It remains to argue that the ciphertext size is sublinear in the circuit size. This is implied by [Lin17, LT17] since our pre-processed input size is strictly smaller than in these works. In more detail, note that $|V_1| = |V_2| = |\mathsf{Mnml}(s_\gamma)| \cdot n \cdot 2$. By choosing block size $\log \kappa$, we get that $|\mathsf{Mnml}(s_\gamma)| = \kappa$. Moreover, as in [LT17], we set the seed size $n = S^{\frac{1}{1+\alpha}}$ where S is the size of the circuit and $\alpha > 0$. Hence the length of the message encrypted by PrgFE is $O(\kappa \cdot S^{\frac{1}{1+\alpha}})$ which is clearly sublinear in S. Additionally, since we assumed that PrgFE enjoys compact ciphertext, we get that the ciphertext size is sublinear in S.

7.2.2 Proof of Security.

Next, we argue that the NLinFE scheme constructed above is secure.

Theorem 7.1. *Assume that* PrgFE *is an* FE *scheme that supports evaluation of degree L polynomials and satisfies* IND *security as in Definition 2.4. Assume that G^q is a secure* PRG *with block locality L and polynomial expansion. Then, the* NLinFE *scheme constructed in Section 7.2 satisfies* Noisy-IND *security as in Definition 3.4.*

Proof. The proof proceeds via a sequence of hybrids. Let us say the adversary provides Q ciphertext challenges, denoted by message pairs $-(\mathbf{z}_1^0, \mathbf{z}_1^1) \ldots, (\mathbf{z}_Q^0, \mathbf{z}_Q^1)$. For a given key vector $\mathbf{v}$, let us say we have $\langle(\mathbf{z}_i^0 - \mathbf{z}_i^1), \mathbf{v}\rangle = \epsilon_i$ for $i \in [Q]$.

We construct Q hybrids, one corresponding to each challenge message pair. In hybrid 0, all messages are encrypted as in the real world with bit $b = 0$. In hybrid i for $i \in [Q]$, the first $i-1$ challenge messages correspond to bit $b = 1$, namely $\{(\mathbf{z}_j^1, \mathbf{s}_j, \perp, \mathsf{mode} = 0)\}$ for $j \in [i-1]$ and $\{(\mathbf{z}_j^0, \mathbf{s}_j, \perp, \mathsf{mode} = 0)\}$ for $j \in [i, Q]$. Thus, hybrid Q corresponds to the real world with bit $b = 1$. Between Hybrids i and $i+1$ for $i \in [Q]$, we define 4 sub-Hybrids as follows.

Hybrid $(i, 0)$: In hybrid i for $i \in [Q]$, the first $i-1$ challenge ciphertexts correspond to bit $b = 1$, namely $\{(\mathbf{z}_j^1, \mathbf{s}_j, \perp, \mathsf{mode} = 0)\}$ for $j \in [i]$ and $\{(\mathbf{z}_j^0, \mathbf{s}_j, \perp, \mathsf{mode} = 0)\}$ for $j \in [i+1, Q]$.

Hybrid $(i, 1)$: In this world, we hardwire the output of the function on the i^{th} message i.e. $y_i = \langle \mathbf{z}_i^0,\ \mathbf{v}\rangle + G_{\mathbf{v}}^q(\mathbf{s}_i)$ into the function key using symmetric key encryption. That is, let $\mathsf{CT} = \mathsf{Sym.Enc}(\mathsf{Sym.K}, y_i)$.

Hybrid $(i, 2)$: In this world, change the mode of the message in the i^{th} ciphertext, i.e. message encoded is $(\perp, \perp, \mathsf{Sym.K}, \mathsf{mode} = 1)$.

Hybrid $(i, 3)$: In this world, we change the value of y_i to $y_i = \langle \mathbf{z}_i^1,\ \mathbf{v} \rangle + G_{\mathbf{v}}^q(\mathbf{s}_i)$.

Hybrid $(i, 4)$: In this world, we change the message in the i^{th} ciphertext to $(\mathbf{z}_i^1, \mathbf{s}, \perp, \mathsf{mode} = 0)$.

Hybrid $(i, 5)$: In this world, we change the value of CT hardwired in the key back to random. Note that this world corresponds to Hybrid $(i + 1, 0)$, with the first i challenge ciphertexts corresponding to bit $b = 1$.

Thus, Hybrid $(1, 0)$ corresponds to the real world with $b = 0$ and Hybrid $(Q, 5)$ corresponds to the real world with $b = 1$. Next, we argue that consecutive hybrids are indistinguishable.

Lemma 7.2. *Hybrids $(i, 0)$ and $(i, 1)$ are indistinguishable assuming the security of* Sym.

Proof. The only thing that changes between Hybrid $(i, 0)$ and $(i, 1)$ is the choice of CT, so that in the former it is chosen randomly and in the latter case it is an honest encryption of the scheme Sym. Given an adversary $\mathcal{A}$ who distinguishes between Hybrid 0 and Hybrid 1, we construct an adversary $\mathcal{B}$ who breaks the semantic security of Sym.

$\mathcal{B}$ generates the public key honestly and returns it to $\mathcal{A}$. When $\mathcal{A}$ outputs two challenge message pairs $(\mathbf{z}_1^0, \mathbf{z}_1^1) \ldots, (\mathbf{z}_Q^0, \mathbf{z}_Q^1)$, $\mathcal{B}$ samples $\mathbf{s}_j$ and honestly computes ciphertexts for $(\mathbf{z}_j^b, \mathbf{s}_j, \perp, \mathsf{mode} = 0)$ where $b = 1$ for $j \in [i - 1]$ and $b = 0$ for $j \in [i, Q]$ and returns these to $\mathcal{A}$. When $\mathcal{A}$ requests a function key $\mathbf{v}$, $\mathcal{B}$ computes the value $y = \langle \mathbf{z}_i^0,\ \mathbf{v} \rangle + G_{\mathbf{v}}(\mathbf{s}_i)$, and sends y to the Sym challenger. The Sym challenger responds with CT which is either an honest encryption of y or an element chosen randomly from the ciphertext space. $\mathcal{B}$ uses CT in constructing the function key and returns this to $\mathcal{A}$. Now, if CT is random, $\mathcal{A}$ sees the view of Hybrid 0 and if it is an encryption of y, it sees the view of Hybrid 1. $\square$

Lemma 7.3. *Hybrids $(i, 1)$ and $(i, 2)$ are indistinguishable assuming the security of* PrgFE.

Proof. The only difference between Hybrids 1 and 2 is that in the former the message in the i^{th} ciphertext is $(\mathbf{z}_i^0, \mathbf{s}_i, \perp, \mathsf{mode} = 0)$ and in the latter it is $(\perp, \perp, \mathsf{Sym.K}, \mathsf{mode} = 1)$. Assume there is an adversary $\mathcal{A}$ who distinguishes between Hybrid 1 and Hybrid 2, we construct an adversary $\mathcal{B}$ who can break the security of PrgFE.

$\mathcal{B}$ does the following:

1. It obtains the public key from the PrgFE challenger and returns this to $\mathcal{A}$.

2. When $\mathcal{A}$ outputs two message pairs $(\mathbf{z}_1^0, \mathbf{z}_1^1) \ldots, (\mathbf{z}_Q^0, \mathbf{z}_Q^1)$, it samples $\mathbf{s}_i$ for $i \in [Q]$, $\mathsf{Sym.K}$ and returns challenges $(\mathbf{z}_i^0, \mathbf{s}_i, \perp, \mathsf{mode} = 0)$ and $(\perp, \perp, \mathsf{Sym.K}, \mathsf{mode} = 1)$ to the PrgFE challenger. It obtains an encryption of one of them chosen at random. The rest of the ciphertexts it generates honestly and returns these to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a function $\mathbf{v}$, $\mathcal{B}$ constructs the function $G_{\mathbf{v}}$ as described in Figure 7.2 and sends this to the PrgFE challenger. Here, CT is computed as $\mathsf{Sym.Enc}(\mathsf{Sym.K}, y)$ where $y = \langle \mathbf{z}_i^0,\ \mathbf{v} \rangle + G_{\mathbf{v}}(\mathbf{s})$. It returns the obtained key to $\mathcal{A}$.

4. When $\mathcal{A}$ outputs a guess bit, it outputs the same.

When the PrgFE challenger returns an encryption of $(\mathbf{z}_i^0, \mathbf{s}_i, \perp, \mathsf{mode} = 0)$, $\mathcal{A}$ sees the view of Hybrid 1, and when it returns an encryption of $(\perp, \perp, \mathsf{Sym.K}, \mathsf{mode} = 1)$, it sees the view of Hybrid 2. Note that in either case the decrypted value is the same. Thus, the advantage of $\mathcal{A}$ translates to the advantage of $\mathcal{B}$. □

Lemma 7.4. *Hybrids $(i, 2)$ and $(i, 3)$ are indistinguishable assuming the security of* PRG.

Proof. The only difference between hybrids $(i, 2)$ and $(i, 3)$ is that in the former case, we have $y = \langle \mathbf{z}_i^0,\ \mathbf{v}\rangle + G_{\mathbf{v}}^q(\mathbf{s}_i)$ and in the latter, we have $y = \langle \mathbf{z}_i^1,\ \mathbf{v}\rangle + G_{\mathbf{v}}^q(\mathbf{s}_i)$. If $G_{\mathbf{v}}$ is a PRG, then the two values of y are indistinguishable by the security of the PRG. To see this, note that the output of the PRG is uniform over a set R_{Bd} of elements with norm bounded by Bd, which is super-polynomially larger than $\|\epsilon_i\|$ where $\epsilon_i = \langle \mathbf{z}_i^0 - \mathbf{z}_i^1,\ \mathbf{v}\rangle$. In more detail, let Unif_i be distributed uniformly over R_{Bd}. Then, we have:

$$\langle \mathbf{z}_i^0 - \mathbf{z}_i^1,\ \mathbf{v}\rangle + G_{\mathbf{v}}(\mathbf{s}_i) \overset{c}{\approx} \epsilon_i + \mathsf{Unif}_i \overset{s}{\approx} \mathsf{Unif}_i$$

□

Lemma 7.5. *Hybrids $(i, 3)$ and $(i, 4)$ are indistinguishable assuming the security of* PrgFE.

Proof. The only difference between Hybrids $(i, 3)$ and $(i, 4)$ is that in the former, the message encoded in the i^{th} ciphertext is $(\perp, \perp, \mathsf{Sym.K}, \mathsf{mode} = 1)$ and in the latter the message encrypted is $(\mathbf{z}_i^1, \mathbf{s}_i, \perp, \mathsf{mode} = 0)$. Note that in both cases, we have the same output of decryption hence the two ciphertexts are indistinguishable by security of PrgFE. □

Hybrids $(i, 4)$ and $(i, 5)$ are are indistinguishable assuming the security of Sym as in Lemma 7.2. Thus, we have that Hybrids $(1, 0)$ and $(Q, 5)$ are indistinguishable, which proves the theorem.

□

7.2.3 Putting it all together.

For the proof of Lemma 7.4, we require that the bound Bd on the norm of elements in R_{Bd} must be super-polynomially larger than $\|\epsilon_i\|$ where $\epsilon_i = \langle \mathbf{z}_i^0 - \mathbf{z}_i^1,\ \mathbf{v}\rangle$. Recall that R_{Bd}^{ℓ} is the range of the PRG in our construction. In our application of NLinFE in the construction of $\mathsf{FeNC_1}$ (Section 6), we have that $\|\epsilon_i\| = O(B_{\mathsf{init}}^{2^d})$ where B_{init} is the noise bound in level 1 encodings, and d is the depth of the circuit. Hence, setting $B_{\mathsf{init}} = O(\kappa)$ and $\mathsf{Bd} = \kappa^{2(2^d)}$ suffices for security. To ensure correctness, we must set the size of the modulus a constant factor larger than the largest noise, say $5 \cdot \kappa^{2(2^d)}$. Note that the NLinFE construction from PrgFE is oblivious to the distribution of the noise term ϵ_i since it uses a brute force flooding argument.

7.2.4 Interfacing with known constructions of quadratic functional encryption

Existing constructions of quadratic functional encryption (QuadFE)c suffer from output size restriction, namely, decryption only works if the output is restricted to a bounded polynomial size interval. Instantiating PrgFE in our construction using the QuadFE of [Lin17, BCFG17] and conjectured 2 block local PRG then poses a challenge, since our PrgFE is required to compute a quadratic function of super-polynomial size. This is because, as discussed above, we require that the bound Bd on the norm of elements in R_{Bd} must be super-polynomially larger than $\|\epsilon_i\|$ where $\epsilon_i = \langle \mathbf{z}_i^0 - \mathbf{z}_i^1,\ \mathbf{v}\rangle$.

To overcome this issue, we rely on the Chinese Remainder Theorem (CRT) to split the computation of the above large output (of size $O(2^\kappa)$, say) into κ computations of bounded size, each of which are performed in parallel using a fresh QuadFE scheme. In more detail, recall that we must compute a function $\langle \mathbf{z},\ \mathbf{v}_i\rangle + G_i^q(\mathbf{s}) \in R_p$. For our application, we may choose the modulus p composite as a product of κ distinct small primes $p_1, \ldots, p_\kappa$, where each $|p_i| = O(1)$. Then, by CRT, we have that the map $x \mod p \to (x \mod p_1, \ldots, x \mod p_\kappa)$ defines a ring homomorphism. Hence, any computation modulo p may be performed independently in each "slot" modulo p_i, and the outputs of the computation along each slot may be combined using the isomorphism from right to left (i.e. in "reverse") to obtain the output modulo p.

Thus, our computation may be decomposed along the CRT basis into κ components each of bounded size, each of which is implemented using a fresh QuadFE scheme. The encryptor as well as the key generator decompose their inputs $\mathbf{z}, \mathbf{v}$ into vectors of constant sized elements according to the CRT representation and compute the noisy inner product for each CRT slot using a new QuadFE scheme. Since the input sizes are bound by a constant and the computation is low degree, one may bound the output size in each CRT slot by a small polynomial, and recover it using the output restricted QuadFE scheme. The outputs of the κ QuadFE schemes are then combined using the isomorphism from right to left to obtain the final (large) output as desired. We defer further details to the full version of the paper.

We also remark that our construction may be instantiated using either ring LWE or 1 dimensional LWE[14]. In the case of ring LWE, the underlying ring is a polynomial ring with integer coefficients, i.e. of the form $R = \mathbb{Z}[x]/f(x)$, whereas the QuadFE constructions [Lin17, BCFG17] support message space $\mathbb{Z}$. However, this does not create an incompatibility, since a product of two polynomials in R (and R_p) can be represented as a matrix vector product in $\mathbb{Z}$ (and $\mathbb{Z}_q$). This is standard, see for instance [LPR10].

7.3 Symmetric Key Noisy Linear Functional Encryption

In this section, we rely on function hiding symmetric key FE for degree L polynomials, where L is the blockwise locality of a CNG. The construction is almost the same as above, except that we do not require Sym to encrypt the value of y in the function key – we may program the value of y in the function key directly, and use function hiding of the FE scheme to hide it.

$\mathsf{NLinFE.Setup}(1^\kappa, 1^w)$: Upon input the security parameter and length of input message, choose the CNG family $G : R^n \to R^m$. Invoke $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{CngFE.Setup}(1^\kappa, 1^{w+n+1})$ and output $(\mathsf{PK}, \mathsf{MSK})$.

$\mathsf{NLinFE.Enc}(\mathsf{MSK}, \mathbf{z})$: Upon input the master secret key MSK and a vector $\mathbf{z} \in R_q^w$, do the following:

1. Sample a seed $\mathbf{s} \leftarrow \mathcal{D}_{\mathsf{seed}}^n$ for the CNG.
2. Compute $\mathsf{CngFE.Enc}\big(\mathsf{MSK}, (\mathbf{z}, \mathbf{s}, \mathsf{mode} = 0)\big)$.

[14]It is also possible to instantiate it using the more standard polynomial dimensional LWE, but in this case we cannot support bootstrapping to NC_1 directly, we must bootstrap to NC_0 and then rely on randomized encodings to bootstrap further to NC_1.

$\mathsf{NLinFE.KeyGen}(\mathsf{MSK}, \mathbf{v})$: Upon input the master secret key MSK and a vector $\mathbf{v} \in R_q^w$, do the following:

1. Let $y^* = 0$.
2. Choose the CNG smudging polynomial $G_\mathbf{v}$ depending on $\mathbf{v}$ as described in Definition 5.1. We define functionality $G_{\mathbf{v},y^*}$ in Figure 7.3.
3. Output $\mathsf{CngFE.KeyGen}(\mathsf{MSK}, G_{\mathbf{v},y^*})$

Functionality $G_{\mathbf{v},y^*}(\mathbf{z}, \mathbf{s}, \mathsf{mode})$

If $\mathsf{mode} = 0$, output $\langle \mathbf{z},\ \mathbf{v}\rangle + G_\mathbf{v}(\mathbf{s})$ else output y^*.

$\mathsf{NLinFE.Dec}(\mathsf{PK}, \mathsf{CT}_\mathbf{z}, \mathsf{SK}_\mathbf{v})$: Upon input the public key PK, a ciphertext $\mathsf{CT}_\mathbf{z}$ and a function key $\mathsf{SK}_\mathbf{v}$, compute $\mathsf{CngFE.Dec}(\mathsf{PK}, \mathsf{CT}_\mathbf{z}, \mathsf{SK}_\mathbf{v})$ and output it.

Correctness and Efficiency. We have by correctness of CngFE that decryption recovers $\langle \mathbf{z},\ \mathbf{v}\rangle + G_\mathbf{v}(\mathbf{s})$ as desired. By pre-processing the seed exactly as in Section 7.2.1, we obtain a polynomial of degree equal to the blockwise locality of the CNG that computes functionality $G_{\mathbf{v},y^*}(\cdot)$.

7.3.1 Proof of Security

Next, we argue that the NLinFE scheme constructed above is secure.

Theorem 7.6. *Assume that* CngFE *is an* FE *scheme that supports evaluation of degree L polynomials and satisfies* IND *based security as in Definition 2.4. Assume that $G_\mathbf{v}$ is a secure* CNG *of blockwise locality L as in Definition 5.1. Then the* NLinFE *scheme described in Section 7.3 satisfies* $\mathsf{Noisy\text{-}IND}$ *security as in Definition 3.4.*

Proof. The proof proceeds via a sequence of hybrids. Let us say the adversary submits Q challenge message pairs $(\mathbf{z}_i^0, \mathbf{z}_i^1)$ for $i \in [Q]$. We construct Q hybrids, one corresponding to each challenge message pair. In hybrid 0, all messages are encrypted as in the real world with bit $b = 0$. In hybrid i for $i \in [Q]$, the first i challenge messages correspond to bit $b = 1$, namely $\{(\mathbf{z}_j^1, \mathbf{s}_j, \mathsf{mode} = 0)\}$ for $j \in [i]$ and $\{(\mathbf{z}_j^0, \mathbf{s}_j, \mathsf{mode} = 0)\}$ for $j \in [i+1, Q]$. Thus, hybrid Q corresponds to the real world with bit $b = 1$. Between Hybrids i and $i + 1$, we define 4 sub-Hybrids as follows.

Hybrid $(i, 0)$: In this world, the encoded message in the first i challenge ciphertexts is $(\mathbf{z}_j^1, \mathbf{s}_j, \mathsf{mode} = 0)$ for $j \in [i]$, the value of y^* in the key $\mathsf{SK}_\mathbf{v}$ is 0. The remaining $Q - i$ ciphertexts encode $(\mathbf{z}_j^0, \mathbf{s}_j, \mathsf{mode} = 0)$ for $j \in [i, Q]$.

Hybrid $(i, 1)$: In this world, we change the value of y^* hardwired in the key $\mathsf{SK}_\mathbf{v}$ to $y^* = \langle \mathbf{z}_{i+1}^0,\ \mathbf{v}\rangle + G_\mathbf{v}(\mathbf{s}_{i+1})$.

Hybrid $(i, 2)$: In this world, we change the mode of the $i + 1^{st}$ ciphertext, i.e. message encoded is $(\perp, \perp, \mathsf{mode} = 1)$.

Hybrid $(i, 3)$: In this world, we change the value of y^* in the key to $y^* = \langle \mathbf{z}_{i+1}^1,\ \mathbf{v}\rangle + G_\mathbf{v}(\mathbf{s}_{i+1})$.

Hybrid $(i, 4)$: In this world, we change the mode of the $i + 1^{st}$ ciphertext, message encoded is $(\mathbf{z}^1_{i+1}, \mathbf{s}_{i+1}, \mathsf{mode} = 0)$.

Hybrid $(i, 5)$: In this world, we change the value of y^* in the key back to 0. In this world, the encoded message in the first $i + 1$ challenge ciphertexts is $(\mathbf{z}^1_j, \mathbf{s}_j, \mathsf{mode} = 0)$, for $j \in [i + 1]$.

Note that hybrid $(i, 5)$ is the same as hybrid $(i + 1, 0)$. Clearly, hybrid $(0, 0)$ corresponds to the real world with bit $b = 0$ and hybrid $(Q - 1, 5) = (Q, 0)$ corresponds to the real world with bit $b = 1$.

Now, we argue that consecutive hybrids are indistinguishable.

Claim 7.7. *Hybrids* $(i, 0)$ *and* $(i, 1)$ *are indistinguishable due to function hiding of* CngFE.

Proof. Note that the only difference between hybrids $(i, 0)$ and $(i, 1)$ is that the value of y^* that is hardwired in the function key is different. However, note that this value is hidden by the function hiding property of CngFE. To see this, consider an adversary $\mathcal{A}$ who can distinguish between Hybrids 0 and 1. Then we construct an adversary $\mathcal{B}$ against the function hiding of CngFE. $\mathcal{B}$ does the following:

1. **Ciphertext Queries.** When $\mathcal{A}$ requests a ciphertext corresponding to a message $\mathbf{z}$, $\mathcal{B}$ samples seed $\mathbf{s}$, obtains a CngFE encryption of $(\mathbf{z}, \mathbf{s}, \mathsf{mode} = 0)$ from the CngFE challenger and returns this to $\mathcal{A}$.

2. **Challenge Ciphertexts.** When $\mathcal{A}$ outputs the challenge messages $(\mathbf{z}^0_j, \mathbf{z}^1_j)$ for $j \in [Q]$, $\mathcal{B}$ samples $\mathbf{s}_j$ for $j \in [Q]$, obtains CngFE encryptions of $(\mathbf{z}^1_j, \mathbf{s}_j, \mathsf{mode} = 0)$ from the CngFE challenger for $j \in [i]$, and of $(\mathbf{z}^0_j, \mathbf{s}_j, \mathsf{mode} = 0)$ for $j \in [i + 1, Q]$ and returns these to $\mathcal{A}$.

3. **Key Requests.** When $\mathcal{A}$ outputs a key request for $\mathbf{v}$, $\mathcal{B}$ submits the circuit pair $(G_{\mathbf{v},y_0},\ G_{\mathbf{v},y_1})$ where $y_0 = 0$ and $y_1 = \langle \mathbf{z}^0_{i+1},\ \mathbf{v}\rangle + G_{\mathbf{v}}(\mathbf{s}_{i+1})$ to the CngFE challenger. It receives a key for $G_{\mathbf{v},y_b}$ where b is a randomly chosen bit. It returns this to $\mathcal{A}$.

4. **Guess.** When $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ outputs the same.

Note that $\mathcal{B}$ is an admissible CngFE adversary because $G_{\mathbf{v},y_0}(\mathbf{z}^0_{i+1}, \mathbf{s}_{i+1}, 0) = G_{\mathbf{v},y_1}(\mathbf{z}^0_{i+1}, \mathbf{s}_{i+1}, 0)$. This is because even though the programmed value for y^* is different for the two keys, this value does not participate in decryption since $\mathsf{mode} = 0$. Now, if $b = 0$, $\mathcal{A}$ sees the distribution in Hybrid 0, else in Hybrid 1. The advantage of $\mathcal{A}$ translates directly to an advantage of $\mathcal{B}$. $\square$

The indistinguishability of remaining hybrids is argued as in the proof of Theorem 7.1. In more detail, we have:

Lemma 7.8. *Hybrids* $(i, 1)$ *and* $(i, 2)$ *are indistinguishable by security of* CngFE.

Proof. Given an adversary $\mathcal{A}$ that distinguishes between hybrids $(i, 1)$ and $(i, 2)$, we may construct an adversary $\mathcal{B}$ to break the security of CngFE as follows.

1. **Ciphertext Queries.** To answer a ciphertext query for $\mathbf{z}$, it samples $\mathbf{s}$, obtains a CngFE encryption of $(\mathbf{z}, \mathbf{s}, \mathsf{mode} = 0)$ from the CngFE challenger and returns this to $\mathcal{A}$.

2. **Challenge Ciphertexts.** When $\mathcal{A}$ submits challenge message pairs $(\mathbf{z}_i^0, \mathbf{z}_i^1)$ for $i \in [Q]$, $\mathcal{B}$ samples $\mathbf{s}_j$ for $j \in [Q] \setminus (i+1)$, obtains CngFE encryptions of $(\mathbf{z}_j^1, \mathbf{s}_j, \mathsf{mode} = 0)$ from the CngFE challenger for $j \in [i]$, of $(\mathbf{z}_j^0, \mathbf{s}_j, \mathsf{mode} = 0)$ for $j \in [i+2, Q]$, and of $(\perp, \perp, \mathsf{mode} = 1)$ for $j = i+1$, and returns these to $\mathcal{A}$.

3. **Key Requests.** To answer key requests, $\mathcal{B}$ computes y^* as described, forwards key requests to CngFE challenger and obtains function keys which it relays to $\mathcal{A}$.

4. **Guess.** When $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ outputs the same.

Note that $\mathcal{B}$ is an admissible adversary in the CngFE game, hence the advantage of $\mathcal{A}$ translates to the advantage of $\mathcal{B}$. □

Lemma 7.9. *Hybrids* $(i, 2)$ *and* $(i, 3)$ *are indistinguishable by security of* CNG.

Proof. The only difference between hybrids $(i, 2)$ and $(i, 3)$ is that the in the former $y^* = \langle \mathbf{z}_{i+1}^0,\ \mathbf{v} \rangle + G_\mathbf{v}(\mathbf{s}_{i+1})$, and in the latter $y^* = \langle \mathbf{z}_{i+1}^1,\ \mathbf{v} \rangle + G_\mathbf{v}(\mathbf{s}_{i+1})$. By security of CNG, these are indistinguishable.

In more detail, given an adversary $\mathcal{A}$ who can distinguish between hybrids $(i, 2)$ and $(i, 3)$, we construct an adversary $\mathcal{B}$ against CNG as follows:

1. $\mathcal{B}$ invokes $\mathsf{NLinFE.Setup}$ and returns the public key to $\mathcal{A}$.

2. $\mathcal{A}$ outputs challenge message pairs $(\mathbf{z}_j^0, \mathbf{z}_j^1)$ for $j \in [Q]$. $\mathcal{B}$ constructs the challenge ciphertexts honestly for $(\mathbf{z}_j^1, \mathbf{s}_j, \mathsf{mode} = 0)$ when $j \in [i]$, for message $(\perp, \perp, \mathsf{mode} = 1)$ when $j = i+1$ and for $(\mathbf{z}_j^0, \mathbf{s}_j, \mathsf{mode} = 0)$ when $j \in [i+2, Q]$ It returns these to $\mathcal{A}$.

3. $\mathcal{A}$ makes a function query $\mathbf{v}$ upon which $\mathcal{B}$ does the following:

   - Note that by admissibility of the NLinFE adversary, $\langle \mathbf{z}_{i+1}^0,\ \mathbf{v} \rangle - \langle \mathbf{z}_{i+1}^1,\ \mathbf{v} \rangle = f_\mathbf{v}(\mathcal{D}_{\mathsf{NFE}})$ for some fixed distribution $\mathcal{D}_{\mathsf{NFE}}$ and function $f_\mathbf{v} \in \mathcal{F}_{\mathsf{NFE}}$. We denote $\epsilon_{i+1,\mathbf{v}} = (\langle \mathbf{z}_{i+1}^0,\ \mathbf{v} \rangle - \langle \mathbf{z}_{i+1}^1,\ \mathbf{v} \rangle)$.
   - $\mathcal{B}$ sends $\big(f_\mathbf{v},\ \ \epsilon_{i+1,\mathbf{v}}\big)$ to CNG challenger. The challenger chooses a corresponding "smudging polynomial" $G_\mathbf{v}$ and a random bit b. If $b = 0$, it returns $\epsilon_{i+1,\mathbf{v}} + G_\mathbf{v}(\mathbf{s}_{i+1})$ else it returns $G_\mathbf{v}(\mathbf{s}_{i+1})$. Set $y_b = \langle \mathbf{z}_{i+1}^1,\ \mathbf{v} \rangle + G_\mathbf{v}(\mathbf{s}_{i+1}) + (1-b) \cdot \epsilon_{i+1,\mathbf{v}}$.
   - $\mathcal{B}$ constructs the function key honestly using y_b. This key is returned to $\mathcal{A}$.
   - When the adversary $\mathcal{A}$ outputs a guess bit, $\mathcal{B}$ does the same.

   We note that when $b = 0$, we are in Hybrid $(i, 2)$, and when $b = 1$ we are in Hybrid $(i, 3)$. Also, note that $\mathcal{B}$ is a valid adversary against CNG. Hence, if $\mathcal{A}$ succeeds in distinguishing between hybrids $(i, 2)$ and $(i, 3)$, then $\mathcal{B}$ succeeds in breaking the security of CNG.

□

Indistinguishability between hybrids $(i, 3)$ and $(i, 4)$ is argued as in Lemma 7.8 and between hybrids $(i, 4)$ and $(i, 5)$ as in Lemma 7.7. This completes the proof. □

7.3.2 Putting it all together.

Our construction of NLinFE from CngFE is instantiated similarly to the PrgFE based construction discussed in Section 7.2.3. The only difference is that since we use CNG rather than PRG for the noise smudging, the distribution of ϵ_i becomes relevant. This distribution is analysed in Section 6.4. Again, we must choose the range of CNG so that it smudges the functional noise term but is small relative to the modulus so that decryption is possible. Since CNG is a new primitive that generalises PRG, we do not yet have constructions (other than those of PRG), so we do not instantiate the CNG concretely here.

8 Direct Construction: Noisy Linear Functional Encryption

In this section we will provide a construction for succinct noisy functional encryption, denoted by NLinFE. Our construction is inspired from the construction for linear functional encryption (denoted by LinFE) by [ABCP15, ALS16]. Below, we outline the LinFE construction which is our starting point.

Recap: Linear Functional Encryption. Recall that in LinFE, the encryptor wishes to encrypt a message $\mathbf{z} \in R_{p_1}^w$, the key generator provides a key for vector $\mathbf{v} \in R_{p_1}^w$ and the decryptor must recover $\langle \mathbf{z}, \mathbf{v} \rangle \in R_{p_1}$. [ABCP15, ALS16] provide a construction of LinFE based on the LWE assumption. This construction can be readily adapted to rely on RLWE, which may be described as follows:

1. **Public Key.** Let us choose $\mathbf{w} \in R_{p_2}^m$ and $\mathbf{a} \in R_{p_2}^w$ as the public key and a short Gaussian matrix $\mathbf{E} \in R^{m \times w}$ such that $\mathbf{E}^\top \mathbf{w} = \mathbf{a}$ as the master secret key.
2. **Encryption.**The encryptor given a vector $\mathbf{z} \in R_{p_1}^w$, samples a small secret $s \in R_{p_2}$ as well as small noise terms $\boldsymbol{\nu} \in R^m$, $\boldsymbol{\mu} \in R^w$ according to some fixed distribution, and computes:
$$\mathbf{c} = \mathbf{w} \cdot s + p_1 \cdot \boldsymbol{\nu} \in R_{p_2}^m, \quad \mathbf{b} = \mathbf{a} \cdot s + p_1 \cdot \boldsymbol{\mu} + \mathbf{z} \in R_{p_2}^w$$
3. **Key Generation.** The key for a vector $\mathbf{v} \in R_{p_1}^w$ is computed as $\mathbf{e_v} = \mathbf{E} \cdot \mathbf{v}$.
4. **Decryption.** Decryption proceeds as:
$$\begin{aligned} \mathbf{v}^\top \mathbf{b} - \mathbf{e}_\mathbf{v}^\top \mathbf{c} &= \mathbf{v}^\top (\mathbf{a} \cdot s + p_1 \cdot \boldsymbol{\mu} + \mathbf{z}) - \mathbf{v}^\top \mathbf{E}^\top (\mathbf{w} \cdot s + p_1 \cdot \boldsymbol{\nu}) \\ &= \mathbf{v}^\top \mathbf{a} \cdot s + p_1 \cdot (\mathbf{v}^\top \boldsymbol{\mu}) + \mathbf{v}^\top \mathbf{z} - \mathbf{v}^\top \mathbf{a} \cdot s - p_1 \cdot (\mathbf{v}^\top \mathbf{E}^\top \boldsymbol{\nu}) \\ &= \mathbf{v}^\top \mathbf{z} \mod p_1 \end{aligned}$$

Generalizing to Noisy Linear Functional Encryption. The above construction already supports computing linear equations, hence provides a natural starting point for computing noisy linear equations. Recall that our bootstrapping theorem (see Section 6 and Section 7.3) shows that compact FE supporting the following functionality suffices for constructing compact FE for NC_1:

$$\langle \mathbf{z}, \mathbf{v} \rangle + G(\boldsymbol{\beta}) \in R_{p_1}$$

where the encryptor has input message $\mathbf{z}$, samples $\boldsymbol{\beta}$ as the seed of a correlated noise generator G, and encrypts $(\mathbf{z} \| \boldsymbol{\beta}) \in R_{p_1}^{w+n}$, and the key generator given input $\mathbf{v} \in R_{p_1}^w$ provides a key for the above polynomial.

Our Approach. Our approach is to follow the LinFE blueprint provided above to compute $\langle \mathbf{z}, \mathbf{v} \rangle$, augmenting it with a method that allows computing an encoding of the term $G(\boldsymbol{\beta})$ on the fly. In more detail, suppose we had a method of enabling the decryptor to compute

$$d = h \cdot s + \eta + G(\boldsymbol{\beta})$$

where h is known to the key generator, then $(\mathbf{b} \| d)$ can be viewed as an encryption of $(\mathbf{z} \| G(\boldsymbol{\beta}))$ and the key generator could provide a key for the vector $(\mathbf{v} \| 1)$ so as to enable the decryptor to recover $\langle \mathbf{z}, \mathbf{v} \rangle + G(\boldsymbol{\beta})$ as desired[15].

Of course, the primary difficulty is in designing a method to compute a fresh RLWE encoding of $G(\boldsymbol{\beta})$ given only encodings of $\boldsymbol{\beta}$, in a way that the RLWE label h is computable by the key generator. At present, we do not know how to overcome the above difficulty from RLWE (or other standard assumptions).

In this work, we turn the question around and ask: if we cannot compute what we would like to, can we make do with what we *can* compute? Since designing compact FE to support evaluation of G appears difficult, our approach is to introduce the minimal principled strengthening of the NTRU and RLWE assumptions that allow us to compute some structured high degree noise terms which we shall use as an approximation of $G(\boldsymbol{\beta})$. Here, we crucially leverage the fact that G is a CNG whose only purpose is to smudge leaky, correlated noise terms; this permits us to be relaxed about correctness of the computation.

For ease of exposition, we outline a simpler variant of our construction than described in Section 1: this does not include the trick of adding noise terms jointly generated by the key generator and encryptor. The general construction is provided in Appendix B.

Construction. We proceed to describe the construction NLinFE. For ease of exposition, we let $G(\boldsymbol{\beta}) = \sum_{1 \le j \le i \le n} v'_{ij}\beta_i\beta_j$ be a quadratic polynomial, generalizing the idea to higher degree is straightforward and discussed in Section 8.2. The construction exactly mimics the LinFE construction described above for computing the inner product. To illustrate the new ideas, parts of the scheme below which *differ* from LinFE are highlighted in purple.

Notation. Our construction uses two prime moduli p_1 and p_2 with $p_1 << p_2$. Our message and function vectors will be chosen from R_{p_1} while the public key and ciphertext are from R_{p_2}. The function G used to compute the noise term is represented by a vector $\mathbf{v}^{\times} = (v'_{ij}) \in R_{p_1}^L$ for $L = |1 \le j \le i \le n|$. As in most lattice based cryptographic constructions, we will collect noise terms as the computation proceeds, and will be required to bound the final noise term relative to the modulus in order for decryption to succeed. For better readability, we club together all low norm elements under the general term small where their precise value is not important. Our construction will make use of the fact that elements in R_{p_1} are small in R_{p_2} and that elements sampled from the discrete Gaussian distribution, denoted by $\mathcal{D}$, are small in R_{p_2}.

NLinFE.Setup$(1^\kappa, 1^w)$: On input a security parameter κ, a parameter w denoting the length of the function and message vectors, do the following:

[15]Note that a "brute force" approach would be to hard code a fresh noise term for each key into the ciphertext and force the decryptor to add this into the decryption equation. This intuition is formalized in Appendix D. However, this approach leads to non-compact ciphertext, and does not suffice for our purpose.

1. Sample $\mathbf{w} \leftarrow R_{p_2}^m$ with a trapdoor $\mathbf{T_w}$ using the algorithm TrapGen as defined in Section 2.7.
2. Sample $\mathbf{E} \in \mathcal{D}^{m \times w}$ and set $\mathbf{a} = \mathbf{E}^\intercal \mathbf{w} \in R_{p_2}^w$.
3. For $i \in \{1, \ldots, w\}$, $\ell \in \{1, \ldots, k\}$, sample $f_{1i}^\ell, f_{2i}^\ell \leftarrow \mathcal{D}$ and $g_1^\ell, g_2^\ell \leftarrow \mathcal{D}$. If g_1^ℓ, g_2^ℓ are not invertible over R_{p_2}, resample. Set
$$h_{1i}^\ell = \frac{f_{1i}^\ell}{g_1^\ell}, \quad h_{2i}^\ell = \frac{f_{2i}^\ell}{g_2^\ell} \in R_{p_2}$$
4. Sample a PRF seed, denoted as seed.

Output
$$\mathsf{MSK} = \Big(\mathbf{w}, \mathbf{T_w}, \mathbf{a}, \mathbf{E}, \{f_{1i}^\ell, f_{2i}^\ell\}_{i \in [w], \ell \in [k]}, \{g_1^\ell, g_2^\ell\}_{\ell \in [k]} \}, \mathsf{seed} \Big)$$

$\mathsf{NLinFE.Enc}(\mathsf{MSK}, \mathbf{z})$: On input public key MSK, a message vector $\mathbf{z} \in R_{p_1}^w$, do:

1. **Construct Message Encodings.** Sample $\boldsymbol{\nu} \leftarrow \mathcal{D}^m$, $\boldsymbol{\eta} \leftarrow \mathcal{D}^w$ and $t_1, t_2 \leftarrow \mathcal{D}$. Set $s = t_1 \cdot t_2$. Compute:
$$\mathbf{c} = \mathbf{w} \cdot s + p_1 \cdot \boldsymbol{\nu} \in R_{p_2}^m, \quad \mathbf{b} = \mathbf{a} \cdot s + p_1 \cdot \boldsymbol{\eta} + \mathbf{z} \in R_{p_2}^w$$
2. **Sample Structured Noise.** To compute encodings of noise, do the following:
   (a) Define lattices:
   $$\Lambda_1^\ell \stackrel{\text{def}}{=} g_1^\ell \cdot R, \quad \Lambda_2^\ell \stackrel{\text{def}}{=} g_2^\ell \cdot R$$
   (b) Sample noise terms from the above lattices as:
   $$e_{1i}^\ell \leftarrow \widehat{\mathcal{D}}(\Lambda_2^\ell), \tilde{e}_{1i}^\ell \leftarrow \widehat{\mathcal{D}}'(\Lambda_2^\ell), \quad e_{2i}^\ell \leftarrow \widehat{\mathcal{D}}(\Lambda_1^\ell), \tilde{e}_{2i}^\ell \leftarrow \widehat{\mathcal{D}'}(\Lambda_1^\ell) \quad \forall i \in [w], \ell \in [k]$$
   Here $\widehat{\mathcal{D}}(\Lambda_1^\ell), \widehat{\mathcal{D}}'(\Lambda_1^\ell)$ are discrete Gaussian distributions on Λ_1^ℓ and $\widehat{\mathcal{D}}(\Lambda_2^\ell), \widehat{\mathcal{D}}'(\Lambda_2^\ell)$ are discrete Gaussian distributions on Λ_2^ℓ.
3. **Compute Encodings of Noise.**
   (a) Let
   $$d_{1i}^\ell = h_{1i}^\ell \cdot t_1 + p_1 \cdot \tilde{e}_{1i}^\ell + p_0 \cdot e_{1i}^\ell \in R_{p_2}$$
   Here, $p_1 \cdot \tilde{e}_{1i}^\ell$ behaves as noise and $p_0 \cdot e_{1i}^\ell$ behaves as the message. Let $\mathbf{d}_1^\ell = (d_{1i}^\ell)$.
   (b) Similarly, let
   $$d_{2i}^\ell = h_{2i}^\ell \cdot t_2 + p_1 \cdot \tilde{e}_{2i}^\ell + p_0 \cdot e_{2i}^\ell \in R_{p_2}$$
   Here, $p_1 \cdot \tilde{e}_{2i}^\ell$ behaves as noise and $p_0 \cdot e_{2i}^\ell$ behaves as the message. Let $\mathbf{d}_2^\ell = (d_{2i}^\ell)$.
4. **Output Ciphertext.** Output message encodings $(\mathbf{c}, \mathbf{b})$ and noise encodings $(\mathbf{d}_1^\ell, \mathbf{d}_2^\ell)$ for $\ell \in [k]$.

$\mathsf{NLinFE.KeyGen}(\mathsf{MSK}, \mathbf{v})$: On input the master secret key MSK, and a NLinFE function vector $\mathbf{v} \in R_{p_1}^w$, do the following.

1. **Sampling Basis Preimage vectors.**
   (a) Sample short $\mathbf{e}_{ij} \in R^m$ using $\mathsf{SamplePre}$ (please see Section 2.7) with randomness $\mathsf{PRF}(\mathsf{seed}, ij)$ such that

$$\langle \mathbf{w};\ \mathbf{e}_{ij} \rangle = h_{ij}, \text{ where } h_{ij} \overset{\text{def}}{=} \sum_{\ell \in [k]} h_{1i}^{\ell} h_{2j}^{\ell} + p_0 \cdot \Delta_{ij} + p_1 \cdot \tilde{\Delta}_{ij}$$

   Above $\Delta_{ij}, \tilde{\Delta}_{ij} \leftarrow \mathcal{D} \in R$ for $1 \le j \le i \le n$.

$$\text{Let } \mathbf{E}^{\times} = (\mathbf{e}_{ij}) \in R^{m \times L}, \quad \mathbf{h}^{\times} = (h_{ij}) \in R_{p_2}^{L}$$

   where $L = |1 \le j \le i \le n|$.

2. **Combining Basis Preimages to Functional Preimage.** Define

$$\mathbf{k}_{\mathbf{v}} = \mathbf{E} \cdot \mathbf{v} + \mathbf{E}^{\times} \cdot \mathbf{v}^{\times} \quad \in R^m \tag{8.1}$$

3. Output $(\mathbf{k}_{\mathbf{v}}, \mathbf{v})$.

$\mathsf{NLinFE.Dec}(\mathsf{CT}_{\mathbf{z}}, \mathsf{SK}_{\mathbf{v}})$: On input a a ciphertext $\mathsf{CT}_{\mathbf{z}} = \big(\mathbf{c}, \mathbf{b}, \{\mathbf{d}_1^{\ell}, \mathbf{d}_2^{\ell}\}_{\ell \in [k]} \big)$ and a secret key $\mathbf{k}_{\mathbf{v}}$ for function $\mathbf{v}$, do the following

1. Compute encoding of noise term on the fly as:

$$\mathbf{d}^{\times} \overset{\text{def}}{=} \Big(\sum_{\ell \in [k]} d_{1i}^{\ell} \cdot d_{2j}^{\ell}\Big) \in R_{p_2}^{L}$$

2. Compute functional ciphertext as:

$$b_{\mathbf{v}} = \mathbf{v}^{\mathsf{T}} \mathbf{b} + (\mathbf{v}^{\times})^{\mathsf{T}} \mathbf{d}^{\times} \in R_{p_2}$$

3. Compute $b_{\mathbf{v}} - \mathbf{k}_{\mathbf{v}}^{\mathsf{T}} \mathbf{c} \mod p_1$ and output it.

Correctness. In this section, we establish that the above scheme is correct. We walk through the steps performed by the decrypt algorithm:

1. We compute an encoding of a correlated noise term on the fly as described in Figure 8.1.
   Thus, we get that $\mathbf{d}^{\times} = \mathbf{h}^{\times} \cdot s + p_1 \cdot \mathsf{small} + p_0 \cdot \mathsf{small}$

2. The decryption equation is:

$$b_{\mathbf{v}} - \mathbf{k}_{\mathbf{v}}^{\mathsf{T}} \mathbf{c} = (\mathbf{v}^{\mathsf{T}} \mathbf{b} + (\mathbf{v}^{\times})^{\mathsf{T}} \mathbf{d}^{\times}) - \mathbf{k}_{\mathbf{v}}^{\mathsf{T}} \mathbf{c}$$

3. Recall that $\mathbf{b} = \mathbf{a} \cdot s + p_1 \cdot \boldsymbol{\eta} + \mathbf{z} \in R_{p_2}^{w}$. Hence,

$$\mathbf{v}^{\mathsf{T}} \mathbf{b} = \mathbf{v}^{\mathsf{T}} \mathbf{a} \cdot s + p_1 \cdot \mathsf{small} + \mathbf{v}^{\mathsf{T}} \mathbf{z}$$

4. Since $\mathbf{d}^\times = \mathbf{h}^\times \cdot s + p_1 \cdot \mathsf{small} + p_0 \cdot \mathsf{small}$ and $\mathbf{v}^\times \in R_{p_1}^L$ is small in R_{p_2}, we have

$$(\mathbf{v}^\times)^\mathsf{T}\mathbf{d}^\times = (\mathbf{v}^\times)^\mathsf{T}\mathbf{h}^\times \cdot s + p_1 \cdot \mathsf{small} + p_0 \cdot \mathsf{small}$$

   Hence we have

$$\mathbf{v}^\mathsf{T}\mathbf{b} + (\mathbf{v}^\times)^\mathsf{T}\mathbf{d}^\times = (\mathbf{v}^\mathsf{T}\mathbf{a} + (\mathbf{v}^\times)^\mathsf{T}\mathbf{h}^\times) \cdot s + p_1 \cdot \mathsf{small} + p_0 \cdot \mathsf{small} + \mathbf{v}^\mathsf{T}\mathbf{z}$$

5. Next, note that

$$\mathbf{k}_\mathbf{v}^\mathsf{T}\mathbf{w} = \mathbf{v}^\mathsf{T}\mathbf{a} + (\mathbf{v}^\times)^\mathsf{T}\mathbf{h} = a_\mathbf{v} \in R_{p_2}$$

6. Recall that $\mathbf{c} = \mathbf{w} \cdot s + p_1 \cdot \boldsymbol{\nu}$ hence,

$$\begin{aligned}\mathbf{k}_\mathbf{v}^\mathsf{T}\mathbf{c} &= a_\mathbf{v} \cdot s + p_1 \cdot \langle \boldsymbol{\nu}, \mathbf{k}_\mathbf{v} \rangle \\ &= (\mathbf{v}^\mathsf{T}\mathbf{a} + (\mathbf{v}^\times)^\mathsf{T}\mathbf{h}) \cdot s + p_1 \cdot \mathsf{small}\end{aligned}$$

7. Hence, $b_\mathbf{v} - \mathbf{k}_\mathbf{v}^\mathsf{T}\mathbf{c} \mod p_1 = \mathbf{v}^\mathsf{T}\mathbf{z} + p_0 \cdot \mathsf{small}$ as long we set the modulus p_2 to be large enough so that the cumulative noise terms $p_1 \cdot \mathsf{small}$ may be bounded below $\frac{p_2}{5}$ (say). This is a standard requirement in lattice based constructions and can be ensured by appropriate choice of parameters, please see Section 9.

Computing Encoding of Correlated Noise Term

We compute $d^\ell_{1i} \cdot d^\ell_{2j}$. Recall that

$$d^\ell_{1i} = h^\ell_{1i} \cdot t_1 + p_1 \cdot \tilde{e}^\ell_{1i} + p_0 \cdot e^\ell_{1i} \in R_{p_2}$$

$$d^\ell_{2j} = h^\ell_{2j} \cdot t_2 + p_1 \cdot \tilde{e}^\ell_{2j} + p_0 \cdot e^\ell_{2j} \in R_{p_2}$$

We claim that for all $i, j \in [w]$, we have :

$$h^\ell_{2j} \cdot e^\ell_{1i} = \mathsf{small}, \quad h^\ell_{2j} \cdot \tilde{e}^\ell_{1i} = \mathsf{small}, \quad h^\ell_{1j} \cdot e^\ell_{2i} = \mathsf{small}, \quad h^\ell_{1j} \cdot \tilde{e}^\ell_{2i} = \mathsf{small}, \quad s = t_1 \cdot t_2$$

To see this, recall that $e^\ell_{1i}, \tilde{e}^\ell_{1i}$ are sampled from lattice Λ^ℓ_2 and $e^\ell_{2i}, \tilde{e}^\ell_{2i}$ are sampled from lattice Λ^ℓ_1.

$$\text{Let} \quad e^\ell_{1i} = g^\ell_2 \cdot \xi^\ell_{1i}, \quad \tilde{e}^\ell_{1i} = g^\ell_2 \cdot \tilde{\xi}^\ell_{1i} \tag{8.2}$$

$$\text{Hence,} \quad h^\ell_{2i} \cdot e^\ell_{1j} = f^\ell_{2i} \cdot \xi^\ell_{1j}, \quad h^\ell_{2i} \cdot \tilde{e}^\ell_{1j} = f^\ell_{2i} \cdot \tilde{\xi}^\ell_{1j} \;\; \forall\, i, j \in [w], \ell \in [k] \tag{8.3}$$

$$\text{Let} \quad e^\ell_{2i} = g^\ell_1 \cdot \xi^\ell_{2i}, \quad \tilde{e}^\ell_{2i} = g^\ell_1 \cdot \tilde{\xi}^\ell_{2i} \tag{8.4}$$

$$\text{Hence,} \quad h^\ell_{1i} \cdot e^\ell_{2j} = f^\ell_{1i} \cdot \xi^\ell_{2j}, \quad h^\ell_{1i} \cdot \tilde{e}^\ell_{2j} = f^\ell_{1i} \cdot \tilde{\xi}^\ell_{2j} \;\; \forall\, i, j \in [w], \ell \in [k] \tag{8.5}$$

Now, we may compute:

$$\begin{aligned}
d^\ell_{1i} \cdot d^\ell_{2j} &= \Big(h^\ell_{1i} \cdot t_1 + p_1 \cdot \tilde{e}^\ell_{1i} + p_0 \cdot e^\ell_{1i}\Big) \cdot \Big(h^\ell_{2j} \cdot t_2 + p_1 \cdot \tilde{e}^\ell_{2j} + p_0 \cdot e^\ell_{2j}\Big) \\
&= h^\ell_{1i} \cdot h^\ell_{2j}(t_1 t_2) + p_1^2 \cdot \big(\tilde{e}^\ell_{1i} \cdot \tilde{e}^\ell_{2j}\big) + p_1 \cdot p_0 \big(\tilde{e}^\ell_{1i} \cdot e^\ell_{2j} + e^\ell_{1i} \cdot \tilde{e}^\ell_{2j}\big) + p_0^2 \big(e^\ell_{1i} \cdot e^\ell_{2j}\big) \\
&\quad + h^\ell_{1i} \cdot t_1 \cdot (p_1 \cdot \tilde{e}^\ell_{2j} + p_0 \cdot e^\ell_{2j}) + h^\ell_{2j} \cdot t_2 \cdot (p_1 \cdot \tilde{e}^\ell_{1i} + p_0 \cdot e^\ell_{1i}) \\
&= h^\ell_{1i} \cdot h^\ell_{2j}(t_1 t_2) + p_1^2 \cdot \big(g^\ell_2 \cdot \tilde{\xi}^\ell_{1i} \cdot g^\ell_1 \cdot \tilde{\xi}^\ell_{2j}\big) \\
&\quad + p_1 \cdot p_0 \big(g^\ell_2 \cdot \tilde{\xi}^\ell_{1i} \cdot g^\ell_1 \cdot \xi^\ell_{2j} + g^\ell_2 \cdot \xi^\ell_{1i} \cdot g^\ell_1 \cdot \tilde{\xi}^\ell_{2j}\big) + p_0^2 \big(g^\ell_2 \cdot \xi^\ell_{1i} \cdot g^\ell_1 \cdot \xi^\ell_{2j}\big) \\
&\quad + p_1 \cdot \big(h^\ell_{1i} \cdot \tilde{e}^\ell_{2j} \cdot t_1 + h^\ell_{2j} \cdot \tilde{e}^\ell_{1i} \cdot t_2\big) + p_0 \cdot \big(h^\ell_{1i} \cdot e^\ell_{2j} \cdot t_1 + h^\ell_{2j} \cdot e^\ell_{1i} \cdot t_2\big) \\
&= h^\ell_{1i} \cdot h^\ell_{2j}(t_1 t_2) + p_1^2 \cdot \big(g^\ell_2 \cdot \tilde{\xi}^\ell_{1i} \cdot g^\ell_1 \cdot \tilde{\xi}^\ell_{2j}\big) \\
&\quad + p_1 \cdot p_0 \big(g^\ell_2 \cdot \tilde{\xi}^\ell_{1i} \cdot g^\ell_1 \cdot \xi^\ell_{2j} + g^\ell_2 \cdot \xi^\ell_{1i} \cdot g^\ell_1 \cdot \tilde{\xi}^\ell_{2j}\big) + p_0^2 \big(g^\ell_2 \cdot \xi^\ell_{1i} \cdot g^\ell_1 \cdot \xi^\ell_{2j}\big) \\
&\quad + p_1 \cdot \big(f^\ell_{1i} \cdot \tilde{\xi}^\ell_{2j} \cdot t_1 + f^\ell_{2j} \cdot \tilde{\xi}^\ell_{1i} \cdot t_2\big) + p_0 \cdot \big(f^\ell_{1i} \cdot \xi^\ell_{2j} \cdot t_1 + f^\ell_{2j} \cdot \xi^\ell_{1i} \cdot t_2\big) \\
&= h^\ell_{1i} \cdot h^\ell_{2j} \underbrace{(t_1 t_2)}_{s} + p_1 \cdot \Big(\underbrace{p_1 \cdot (g^\ell_2 \cdot \tilde{\xi}^\ell_{1i} \cdot g^\ell_1 \cdot \tilde{\xi}^\ell_{2j}) + p_0 \cdot (g^\ell_2 \cdot \tilde{\xi}^\ell_{1i} \cdot g^\ell_1 \cdot \xi^\ell_{2j} + g^\ell_2 \cdot \xi^\ell_{1i} \cdot g^\ell_1 \cdot \tilde{\xi}^\ell_{2j})}_{\mathsf{small}} \\
&\quad + \underbrace{(f^\ell_{1i} \cdot \tilde{\xi}^\ell_{2j} \cdot t_1 + f^\ell_{2j} \cdot \tilde{\xi}^\ell_{1i} \cdot t_2)}_{\mathsf{small}}\Big) + p_0 \cdot \underbrace{\Big(p_0 \cdot (g^\ell_2 \cdot \xi^\ell_{1i} \cdot g^\ell_1 \cdot \xi^\ell_{2j}) + (f^\ell_{1i} \cdot \xi^\ell_{2j} \cdot t_1 + f^\ell_{2j} \cdot \xi^\ell_{1i} \cdot t_2)\Big)}_{\mathsf{small}}
\end{aligned} \tag{8.6}$$

Above, we highlight in blue the terms in the noise that will remain fixed across all ciphertexts.

$$\text{Thus,} \sum_{\ell \in [k]} d^\ell_{1i} \cdot d^\ell_{2j} = \Big(\sum_{\ell \in [k]} h^\ell_{1i} \cdot h^\ell_{2j}\Big) \cdot s + p_1 \cdot \mathsf{small} + p_0 \cdot \mathsf{small} \tag{8.7}$$

Figure 8.1: Computing encoding of noise term as polynomial of encodings.

8.1 Security of Succinct Symmetric Key NLinFE

In this section, we reason about the security of our scheme. Our new construction combines and extends the Ring LWE and NTRU assumptions prevalent in lattice based constructions [LPR10, HPS98].

The following assumptions are necessary for the security of our scheme, in that if violated, will lead to an attack against our scheme. In our opinion, cryptanalysis effort must focus first on these assumptions.

Necessary Assumptions. We assume that for $1 \leq i \leq j \leq w$, $\ell \in [k]$,

1. *NTRU with same denominator.* The terms $\{h^\ell_{1i}\}$, $\{h^\ell_{2i}\}$ rely on the NTRU assumption, with the important distinction that the *denominator in all samples in one set is the same*, namely g^ℓ_1 and g^ℓ_2 respectively. Recall that vanilla NTRU requires the denominator to be chosen afresh for each sample, i.e. h^ℓ_{1i} (resp. h^ℓ_{2i} should be constructed using denominator g^ℓ_{1i} (resp. g^ℓ_{2i}) for $i \in [w]$. As discussed in Section 1 and Section 2, this problem was studied by Peikert [Pei16] as the NTRU learning problem.

2. *Ring LWE with structured noise.* The terms $d^\ell_{1i} = h^\ell_{1i} \cdot t_1 + p_1 \cdot \tilde{e}^\ell_{1i} + p_0 \cdot e^\ell_{1i}$, $d^\ell_{2j} = h^\ell_{2j} \cdot t_2 + p_1 \cdot \tilde{e}^\ell_{2j} + p_0 \cdot e^\ell_{2j}$ which are RLWE samples with *structured noise*, namely noise that lives in a fixed, private ideal instead of the entire ring. We assume that these samples appear pseudorandom.

3. *GPV signatures for correlated messages.* The terms $\mathbf{e}_{ij}$ are sampled in KeyGen to satisfy $\langle \mathbf{w};\ \mathbf{e}_{ij} \rangle = \sum_{\ell \in [k]} h^\ell_{1i} h^\ell_{2j} + p_0 \cdot \Delta_{ij} + p_1 \cdot \tilde{\Delta}_{ij}$. These elements are akin to GPV signatures [GPV08], albeit for correlated as against uniform messages. Note that by [GPV08], it is secure to release many short preimages/signatures $\mathbf{e}_{ij}$ such that $\langle \mathbf{w};\ \mathbf{e}_{ij} \rangle = u_{ij}$ when u_{ij} are uniform. But in our case, the u_{ij} are not uniform but rather correlated as $u_i \cdot u_j$ (in the simplest case). We do not know how to handle this in a proof, but also do not know how to exploit correlated images in an attack.

4. *Semantic Security of $\mathbf{d}^\ell_1, \mathbf{d}^\ell_2$ in presence of noise revealed by decryption.* Most importantly, the elements $\mathbf{e}_{ij}$ are embedded in the secret key and participate in decryption of the challenge ciphertext. Decryption of the challenge ciphertext results in noise that may be analyzed as follows. Grouping equation 8.6 differently, we have:

$$\begin{aligned} d^\ell_{1i} \cdot d^\ell_{2j} = h^\ell_{1i} \cdot h^\ell_{2j} \cdot s_2 &+ p_1 \cdot g^\ell_2 \cdot g^\ell_1 \cdot \Big(p_1 \cdot (\tilde{\xi}^\ell_{1i} \cdot \tilde{\xi}^\ell_{2j}) + p_0 \cdot (\tilde{\xi}^\ell_{1i} \cdot \xi^\ell_{2j} + \xi^\ell_{1i} \cdot \tilde{\xi}^\ell_{2j})\Big) \\ &+ p_0 \cdot g^\ell_2 \cdot g^\ell_1 \cdot \Big(p_0 \cdot (\xi^\ell_{1i} \cdot \xi^\ell_{2j})\Big) + p_1 \cdot (f^\ell_{1i} \cdot \tilde{\xi}^\ell_{2j} \cdot t_1 + f^\ell_{2j} \cdot \tilde{\xi}^\ell_{1i} \cdot t_2) \\ &+ p_0 \cdot (f^\ell_{1i} \cdot \xi^\ell_{2j} \cdot t_1 + f^\ell_{2j} \cdot \xi^\ell_{1i} \cdot t_2) \end{aligned}$$

$$\text{Thus, } \sum_{\ell\in[k]} d_{1i}^{\ell}\cdot d_{2j}^{\ell} = \Big(\sum_{\ell\in[k]} h_{1i}^{\ell}\cdot h_{2j}^{\ell}\Big)\cdot s_2 + p_1\cdot\sum_{\ell\in[k]} g_2^{\ell}\cdot g_1^{\ell}\cdot\Big(p_1\cdot(\tilde{\xi}_{1i}^{\ell}\cdot\tilde{\xi}_{2j}^{\ell}) + p_0\cdot(\tilde{\xi}_{1i}^{\ell}\cdot\xi_{2j}^{\ell} + \xi_{1i}^{\ell}\cdot\tilde{\xi}_{2j}^{\ell})\Big)$$
$$+ p_1\cdot\sum_{\ell\in[k]}(f_{1i}^{\ell}\cdot\tilde{\xi}_{2j}^{\ell}\cdot t_1 + f_{2j}^{\ell}\cdot\tilde{\xi}_{1i}^{\ell}\cdot t_2) + p_0\cdot\sum_{\ell\in[k]} g_2^{\ell}\cdot g_1^{\ell}\cdot\Big(p_0\cdot(\xi_{1i}^{\ell}\cdot\xi_{2j}^{\ell})\Big)$$
$$+ p_0\cdot\sum_{\ell\in[k]}(f_{1i}^{\ell}\cdot\xi_{2j}^{\ell}\cdot t_1 + f_{2j}^{\ell}\cdot\xi_{1i}^{\ell}\cdot t_2)$$

The final p_0 noise term recovered after decryption (i.e. after the $\bmod p_1$ operation) is

$$p_0\cdot\left[\sum_{\ell\in[k]}\left(g_2^{\ell}\cdot g_1^{\ell}\cdot\Big(p_0\cdot(\xi_{1i}^{\ell}\cdot\xi_{2j}^{\ell})\Big) + \Big(f_{1i}^{\ell}\cdot\xi_{2j}^{\ell}\cdot t_1 + f_{2j}^{\ell}\cdot\xi_{1i}^{\ell}\cdot t_2\Big)\right) + \Delta_{ij}\cdot s_1 + \tilde{\Delta}_{ij}\cdot s_2\right] \quad (8.8)$$

Above, the terms in blue are fixed, while the remaining terms are chosen afresh for each computation. For semantic security of $\mathbf{d}_1^{\ell}, \mathbf{d}_2^{\ell}$, we require that the fixed terms $g_1^{\ell},\ g_2^{\ell},\ f_{1i}^{\ell},\ f_{2j}^{\ell}$ are hidden in the above quadratic polynomial.

Recall that

$$e_{1i}^{\ell}\leftarrow\widehat{\mathcal{D}}(\Lambda_2^{\ell}),\ \ \tilde{e}_{1i}^{\ell}\leftarrow\widehat{\mathcal{D}'}(\Lambda_2^{\ell}),\ \ e_{2i}^{\ell}\leftarrow\widehat{\mathcal{D}}(\Lambda_1^{\ell}),\ \ \tilde{e}_{2i}^{\ell}\leftarrow\widehat{\mathcal{D}'}(\Lambda_1^{\ell})$$

where

$$\Lambda_1^{\ell}\overset{\text{def}}{=}g_1^{\ell}\cdot R,\quad \Lambda_2^{\ell}\overset{\text{def}}{=}g_2^{\ell}\cdot R$$
$$e_{1i}^{\ell}=g_2^{\ell}\cdot\xi_{1i}^{\ell},\quad \tilde{e}_{1i}^{\ell}=g_2^{\ell}\cdot\tilde{\xi}_{1i}^{\ell}\quad e_{2i}^{\ell}=g_1^{\ell}\cdot\xi_{2i}^{\ell},\quad \tilde{e}_{2i}^{\ell}=g_1^{\ell}\cdot\tilde{\xi}_{2i}^{\ell}$$

We choose the distributions $\widehat{\mathcal{D}}(\Lambda_2^{\ell})$, $\widehat{\mathcal{D}'}(\Lambda_2^{\ell})$, $\widehat{\mathcal{D}}(\Lambda_1^{\ell})$, $\widehat{\mathcal{D}'}(\Lambda_1^{\ell})$ to output *spherical* discrete Gaussians of size 2^{κ}. For a discussion of relevant attacks, please see Section 8.1.1.

8.1.1 Security against Known Attacks: A Discussion

In this section, we give a brief overview on how our construction sidesteps the attacks that are problematic in the setting of multilinear maps.

Zeroizing Attacks and Annihiliation Attacks. First, we note that the so called "zeroizing attacks" that are the main difficulty in the context of multilinear maps do not apply to our setting, since we do not give out anything analogous to encodings of zero. However, the adversary does recover a high degree noise term in the end that lives over the ring R and we must argue that the more general annihilation attacks [MSZ16] do not apply to this setting.

To begin, we recap the principle of annihilation attacks. The GGH13 encodings [GGH13a] contain two secret elements: a "small" element g, where R/gR forms the plaintext space, and a uniformly random denominator $z\in R_q$. In the setting of annihilation attacks, the attacker gets access to "top-level" encodings of zero, which along with the zero testing parameter yield polynomials of the form

$$f = h(\gamma_1 + g\gamma_2 + \ldots, g^{k-1}\gamma_k)$$

Given multiple polynomials of the above form, say f, f', f'', the idea of annihilation attacks is to construct a polynomial $Q(f, f', f'')$ which evaluates to an element in the ideal $< hg >$. These elements are then exploited to construct attacks, but the details of these are not important. Indeed, in the *weak multilinear map* model [GMM+16], the authors declare that if the adversary succeeds in just generating an element in the above ideal, the adversary has won.

Next, consider how the element in $< hg >$ is constructed. Intuitively, the terms f, f', f'' have only one term that is not a multiple of g, namely the leading term γ_1, and the annihilating polynomial Q is constructed so that $Q(\gamma, \gamma', \gamma'') = 0$. By applying the same polynomial to f, f', f'', one may obtain an element in $< hg >$.

Here, it is crucial to the design of the scheme that g is unique and secret, and recovering elements that are multiples of g leads to attack. On the contrary in our case, the polynomial that the adversary recovers, even in its most oversimplified form, has the shape $\sum p^{\ell}(\cdot)g^{\ell}$ for $\ell \in [k]$. Thus, the multiple secret terms g^{ℓ} in our case are "spread" over the entire ring, and the attacker never recovers any polynomial in a single g^{ℓ} alone.

Secondly, the annihilation attack of [MSZ16] relies crucially on the structure of the γ_1 that it must annihilate. [MSZ16] observe that each polynomial γ_1 will be linear in the entries of a term $\vec{r}$ and potentially non-linear in the entries of terms $\vec{\alpha}$. Here, the $\vec{r}$ variables are totally unstructured and unique to each encoding given out, and therefore present an obstacle to the kind of analysis that will enable them to find an annihilating polynomial. Thus, the linearity of the unique, unstructured variables $\vec{r}$ is important in [MSZ16]. On the contrary, in our setting, there are fresh nonlinear terms in top level encodings, and these are moreover a mixture across many ideals. Finally, [MSZ16] do a "change of variables, so that the resulting set of polynomials has a constant number of variables, for which exhaustive search works". Such a change of variables causing reduction in the number of variables does not seem to apply to our construction.

We observe that while annihilation attacks are significantly more general than zeroizing attacks, they nevertheless exploit certain structure specific to multilinear map based constructions, which do not apply to our setting. Moreover, annihilation attacks can only mount attack on a small class of matrix branching programs and do not apply to the original iO candidate [GGH+13c] due to the extra defenses they apply [MSZ16]. Thus, these attacks can be mitigated even in the setting of multilinear maps, via additional randomization. They do not seem to apply to our construction to begin with, and moreover, our construction already appears to contain significant randomization to render them ineffective.

Perspective. Overall, the resultant noise seen by the adversary is a high degree polynomial mixed across many ideals, with additional randomizers and unstructured linear terms to make it look as unstructured as possible. Moreover, additional tricks are possible to randomize the noise, which we did not include (for instance, the present key vector can itself be made to sit within an LWE encoding with special noise – decryption will still work). Note that PRGs admit low degree constructions, so our overarching design goal is to have the noise seen by the adversary mimic a PRG polynomial so that what the adversary sees is akin to a PRG output.

Basic Attacks: Linearization and Statistical Attacks. The simplest attacks against any system non-linear polynomials is evidently linearization: namely, treat every non-linear term as a fresh linear term and solve the system of linear equations. This attack can be easily mitigated by ensuring that the number of unique monomials used in the system of equations is much larger than the number of equations. Another class of attacks that require care are statistical attacks [DP17]. To prevent these, we choose all small elements in our construction to have size 2^{κ}. Moreover, the noise terms are all sampled using *spherical* discrete Gaussians that do not leak the geometry of the special ideals.

Gröbner Basis Attacks. Since the attacker recovers polynomial equations over the entire ring R, the ability to solve systems of multivariate polynomial equations would lead to recovery of the secret terms. Fortunately, this is a notoriously hard problem in the worst case, and involves computation of Gr´obner Basis in general. The complexity of these problems have received a great deal of attention, please see [MI88, BFP09, BFSS13, BFS03, Wol05, DY09, YDH$^+$15, WP05, TW10, AHKI$^+$17] and this problem has formed the basis of several cryptosystems, please see [DY09] for a survey. While one cannot rule out the possibility that there is some special structure to our particular sets of equations that make these easier to solve than the general case, and serious cryptanalytic effort is required to rule out this possibility, it is reassuring that the view of the adversary captures a well studied hard problem, at least in its general case.

Gentry-Szydlow, Statistical Attacks and the Principal Ideal Problem. Since our noise terms are all sampled using *spherical* discrete Gaussians, the Gentry-Szydlow attack [GS02] does not apply to our setting, please see [GGH13b] for an overview of this attack. A potentially relevant problem is the *principal ideal problem*: given a basis of a principal ideal that is guaranteed to have a "rather short" generator, find such a generator. In our setting, the adversary does recover polynomials that contain elements from principal ideals, however these are "mixed" in the sense that the polynomials are a linear combination of monomials that each lie in different ideals. Hence, the adversary does not recover samples in a single ideal, sidestepping the principal ideal problem [Gen09, CDPR16]. We defer a more detailed description of lattice attacks to the full version of the paper.

In Appendix C, we formulate an assumption under which security of our scheme can be proven. We view this as a first step to provable security. We feel that our assumption does not accurately capture the security of the scheme, as discussed in Section C but hope that it can be improved in future work.

8.2 Propagating Computation on Noise

In this section, we describe how to propagate computation on noise terms. Since we are concerned with constant degree polynomials, it suffices to compute all monomials first, followed by a linear combination at the last step. We leverage the fact that the encodings of quadratic noise at level 2, computed after a single multiplication, look exactly like encodings of linear noise terms at level 1, provided by the encryptor.

Let us recall our basic encodings of noise from Section 8. For ease of exposition, let us assume $k = 1$ to begin with. Let

$$h_{1i} = \frac{f_{1i}}{g_1}, \quad h_{2i} = \frac{f_{2i}}{g_2} \in R_{p_2}$$

Then, we have:

$$d_{1i} = h_{1i} \cdot t_1 + p_1 \cdot g_2 \cdot \tilde{\xi}_{1i}$$
$$d_{2j} = h_{2j} \cdot t_2 + p_1 \cdot g_1 \cdot \tilde{\xi}_{2j}$$

$$\text{Thus, } d_{1i} \cdot d_{2j} = \big(h_{1i} \cdot h_{2j}\,\big) \cdot s + p_1 \cdot \mathsf{small} \tag{8.9}$$

Observe that $h_{1i} \cdot h_{2j} = \frac{f_{1i}\, f_{2j}}{g_1 \cdot g_2}$.

Now, let us say we repeat the above setup so that:

$$d'_{1i} = h'_{1i} \cdot t'_1 + p_1 \cdot g'_2 \cdot \tilde{\xi}'_{1i}$$

$$d'_{2j} = h'_{2j} \cdot t'_2 + p_1 \cdot g'_1 \cdot \tilde{\xi}'_{2j}$$

$$\text{Thus, } d'_{1i} \cdot d'_{2j} = \big(h'_{1i} \cdot h'_{2j}\,\big) \cdot s' + p_1 \cdot \mathsf{small}' \tag{8.10}$$

Again, we have that $h'_{1i} \cdot h'_{2j} = \frac{f'_{1i}\, f'_{2j}}{g'_1 \cdot g'_2}$. Let us denote as $g_1^2 = g_1 \cdot g_2$ and $g_2^2 = g'_1 \cdot g'_2$.

Then, if we now ensure that the noise in $d_{1i} \cdot d_{2j}$ contains factor g_2^2 and noise in $d'_{1i} \cdot d'_{2j}$ contains factor g_1^2, then we have samples that look exactly like level 1 samples, namely:

$$d_{1i} \cdot d_{2j} = \big(\frac{f_{1i}\, f_{2j}}{g_1^2}\,\big) \cdot s + p_1 \cdot g_2^2 \cdot \mathsf{small}$$

$$d'_{1i} \cdot d'_{2j} = \big(\frac{f'_{1i}\, f'_{2j}}{g_2^2}\,\big) \cdot s' + p_1 \cdot g_1^2 \cdot \mathsf{small}'$$

Now, we may repeat the above method to multiply encodings and propagate the computation.

Note that it is straightforward to ensure the above by simply multiplying the requisite terms into the noise of the original samples, as:

$$d_{1i} = h_{1i} \cdot t_1 + p_1 \cdot g_2 \cdot g_2^2 \cdot \tilde{\xi}_{1i}$$
$$d_{2j} = h_{2j} \cdot t_2 + p_1 \cdot g_1 \cdot g_2^2 \cdot \tilde{\xi}_{2j}$$
$$d'_{1i} = h'_{1i} \cdot t'_1 + p_1 \cdot g'_2 \cdot g_1^2 \cdot \tilde{\xi}'_{1i}$$
$$d'_{2j} = h'_{2j} \cdot t'_2 + p_1 \cdot g'_1 \cdot g_1^2 \cdot \tilde{\xi}'_{2j}$$

Thus, by doubling the number of encodings at each level, we can ensure that computation is propagated down the circuit.

9 Parameters

In this section, we discuss the parameters for our constructions. The parameters for the constructions in Sections 4 and 6 are inherited from [AR17]. We denote the magnitude of noise used in the level i encodings by B_i. We require $B_i \leq O(p_i/4)$ at every level for correct decryption. We have that the message space for level 1 encodings $\mathcal{E}^1$ is R_{p_0} and encoding space is R_{p_1}. Then message space for $\mathcal{E}^2$ is $O(p_0^2 + B_1^2) = O(B_1^2)$ since the noise at level 1 is a multiple of p_0. Then, p_2 must be chosen as $O(B_1^2)$. At the next multiplication level, i.e. level 4, we have the message space as $O(p_2^2 + B_2^2) = O(B_1^4)$. In general, for d levels, it suffices to set $p_d = O(B^{2^d})$.

We may set $p_0 = n$ with initial noise level as $B_1 = \text{poly}(n)$ and any $B_i, p_i = O(B_1^{2^i})$. Also, the number of encodings provided at level d is $L_d = O(2^d)$, so in general we may let $d = O(\log n)$, thus supporting the circuit class NC_1.

For the heuristic construction in Section 8, we employ the following basic strategy: choose all distributions $\mathcal{D}$, $\widehat{\mathcal{D}}$ and $\widehat{\mathcal{D}}'$ so that the "small" elements sampled have magnitude 2^κ (i.e. of κ bits each). As mentioned above, $\widehat{\mathcal{D}}$ and $\widehat{\mathcal{D}}'$ are set to be spherical discrete Gaussians on the appropriate ideal lattices, so that the geometry of the samples do not reveal the generators g_1^ℓ, g_2^ℓ. The choice of all small elements to have size 2^κ is to prevent statistical attacks [DP17]. We set $n = \kappa^2$.

Since $p_0 = O(\text{poly}(\kappa))$, and each small element is size $O(2^\kappa)$, the small cummulative noise in the correctness equations 8.6 is of size approximately $O(\text{poly}(\kappa) \cdot 2^{4\kappa})$. For correctness of decryption, we just need to set p_2 sufficiently larger, also $O(\text{poly}(\kappa) \cdot 2^{4\kappa})$.

10 Conclusions

Several questions arise from our work. We summarize some of the most important ones below.

1. **Existence of Degree 2 Block Local PRG.** The first and most important one is: do there exist 2 block local PRG with input size n, block size b and stretch $\Omega(n \cdot 2^{b(1+\epsilon)})$? If so, then we will obtain iO from block local PRG, bilinear maps and LWE.

2. **Existence of Degree 2 non-Boolean PRG or CNG.** Second, do non-Boolean PRGs and/or correlated noise generators admit constructions with arithmetic degree 2? These randomness generators are a natural generalization of pseudorandom generators and it would be useful to understand whether new constructions can be found. Additionally, it would be interesting to explore whether they have other applications, particularly in lattice based cryptography.

3. **Security of our new candidate FE.** Next, can we find an attack or a better proof of security for our direct construction for NLinFE? Our construction needs to be subject to thorough cryptanalysis before confidence can be gained in its security.

4. **New constructions following our template.** Are there other constructions that could be inspired by this approach? Our work exploits the structure of NTRU lattices to perform computation on succinct encodings of noise. Are there methods other than those developed in this work to compute on noise? Arguably, this question is much simpler than computing on messages, and we believe other solutions are likely to exist. We believe our work can be seen as a conceptual framework to compute on noise by using nested ideals and combining them randomly to destroy structure. We hope that our methods lead to new solutions which may be shown secure even if ours succumb to attack.

5. **Making it Post-Quantum.** While at the moment, iO from any well understood hardness assumptions would be a big relief, looking ahead, we would like for the construction to be post-quantum secure. iO is a primitive "of the future", in the sense that applications of this primitive may take several years (in the best case scenario) to penetrate the real world. Given its tremendous power and applicability in theory, we are hopeful that a robust and efficient candidate can create significant real world impact. However, the rapid advancements in the development of quantum computers necessitate cryptographic primitives to be based on hardness assumptions that are quantum-safe. To this end, we would like to base iO on well understood hardness assumptions that are known to resist quantum attacks. Several of the tools used in this work, such as LWE and MQ are believed to be quantum safe, and our direct

construction of Section 8 is also based on lattices and may be conjectured quantum secure if proved classically secure. However a significantly better understanding on the post quantum security of our scheme is called for. We also hope other constructions, ideally based entirely on LWE and possibly MQ are discovered.

Acknowledgements. We are grateful to Damien Stehlé for being a generous oracle for any number of arbitrarily distributed queries about lattices. We thank Andrej Bogdanov for discussions about correlated noise generators, Zvika Brakerski for clarifying some aspects of [BBKK17] and Huijia (Rachel) Lin for thought provoking conversations. We are also grateful to Leo Ducas, Alice Pellet-Mary, Gottfried Herold, Bruno Salvy and Damien Stehlé for generously spending their time trying to break our construction and providing insights about the hardness of MQ and Gröbner basis computation. We thank Chris Peikert for telling us about the NTRU learning problem as well as for spending time thinking about our new problem of RLWE with structured noise.

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

[ABCP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. Cryptology ePrint Archive, Report 2015/017, 2015. http://eprint.iacr.org/ To appear in PKC'15.

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In *CRYPTO*, 2015.

[ADGM16] Daniel Apon, Nico Döttling, Sanjam Garg, and Pratyay Mukherjee. Cryptanalysis of indistinguishability obfuscations of circuits over ggh13. eprint 2016, 2016.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Asiacrypt*, 2011.

[Agr17] Shweta Agrawal. Stronger security for reusable garbled circuits, new definitions and attacks. In *Crypto*, 2017.

[AHKI$^+$17] Benny Applebaum, Naama Haramaty-Krasne, Yuval Ishai, Eyal Kushilevitz, and Vinod Vaikuntanathan. Low-Complexity Cryptographic Hash Functions . In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:31, 2017.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.

[AIK11] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 120–129, 2011.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 308–326, 2015.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Achieving compactness generically: Indistinguishability obfuscation from non-compact functional encryption. IACR Cryptology ePrint Archive, 2015:730, 2015.

[Ajt99] Miklos Ajtai. Generating hard instances of the short basis problem. In *ICALP*, volume 1644 of *LNCS*, pages 1–9. Springer, 1999.

[ALS16] Shweta Agrawal, Benoit Libert, and Damien Stehle. Fully secure functional encryption for linear functions from standard assumptions, and applications. In *Crypto*, 2016.

[AP09] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In *STACS*, pages 75–86, 2009.

[AR16] Shweta Agrawal and Alon Rosen. Online offline functional encryption for bounded collusions. Eprint/2016, 2016.

[AR17] Shweta Agrawal and Alon Rosen. Functional encryption for bounded collusions, revisited. In *TCC*, 2017.

[AS17] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In *EUROCRYPT*, 2017.

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

[BCFG17] Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In *Crypto*, 2017.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.

[BFP09] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Hybrid approach for solving multivariate systems over finite fields. *Journal of Mathematical Cryptology*, 3(3), jan 2009.

[BFS03] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. Complexity of Gröbner basis computation for Semi-regular Overdetermined sequences over F_2 with solutions in F_2. Research Report RR-5049, INRIA, 2003.

[BFSS13] Magali Bardet, Jean-Charles Faugre, Bruno Salvy, and Pierre-Jean Spaenlehauer. On the complexity of solving quadratic boolean systems. *Journal of Complexity*, 29(1):53 – 75, 2013.

[BGGS03] Simon R. Blackburn, Domingo Gómez-Pérez, Jaime Gutierrez, and Igor E. Shparlinski. Predicting the inversive generator. In *Cryptography and Coding, 9th IMA International Conference, Cirencester, UK, December 16-18, 2003, Proceedings*, pages 264–275, 2003.

[BGI$^+$01] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001.

[BGL$^+$15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 439–448, 2015.

[BGpGS04] Simon R. Blackburn, Domingo Gomez-perez, Jaime Gutierrez, and Igor E. Shparlinski. Predicting nonlinear pseudorandom number generators. *MATH. COMPUTATION*, 74:2004, 2004.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.

[BLP$^+$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13. ACM, 2013.

[BNPW16] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. In *TCC (B2)*, volume 9986 of *Lecture Notes in Computer Science*, pages 391–418, 2016.

[BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 474–502, 2016.

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 97–106, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 505–524, 2011.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *FOCS*, 2015:163, 2015.

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, pages 290–307, 2006.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007.

[CDPR16] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In *EUROCRYPT*, 2016.

[CFL$^+$] Jung Hee Cheon, Pierre-Alain Fouque, Changmin Lee, Brice Minaud, and Hansol Ryu. Cryptanalysis of the new clt multilinear map over the integers. Eprint 2016/135.

[CGH$^+$15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrede 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 *Advances in Cryptology–CRYPTO 2015*, pages 247–266. Springer, 2015.

[CHJV15] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 429–437, 2015.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552, 2010.

[CHL$^+$15] J.-H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehlé. Cryptanalysis of the multilinear map over the integers. In *Proc. of EUROCRYPT*, volume 9056 of *LNCS*, pages 3–12. Springer, 2015.

[CIJ$^+$13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In *CRYPTO*, 2013.

[CJL] Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low level encoding of zero. Eprint 2016/139.

[CLLT16] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Zeroizing attacks on indistinguishability obfuscation over clt13. Eprint 2016, 2016.

[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.

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC*, 2015.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

[DP17] Léo Ducas and Alice Pellet-Mary. On the statistical leak of the GGH13 multilinear map and some variants. *IACR Cryptology ePrint Archive*, 2017:482, 2017.

[DY09] Jintai Ding and Bo-Yin Yang. *Multivariate Public Key Cryptography*, pages 193–241. Springer Berlin Heidelberg, 2009.

[FHHL18] Pooya Farshim, Julia Hesse, Dennis Hofheinz, and Enrique Larraia. Graded encoding schemes from obfuscation. In *PKC*, 2018.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, 2013.

[GGH13b] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 1–17, 2013.

[GGH+13c] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013. http://eprint.iacr.org/.

[GGH+13d] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO*, 2013.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 498–527, 2015.

[GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure functional encryption without obfuscation. In *IACR Cryptology ePrint Archive*, volume 2014, page 666, 2014.

[GGI05] Domingo Gomez, Jaime Gutierrez, and Alvar Ibeas. Cryptanalysis of the quadratic generator. In Subhamoy Maitra, C. E. Veni Madhavan, and Ramarathnam Venkatesan, editors, *Progress in Cryptology - INDOCRYPT 2005*, pages 118–129, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, pages 555–564, 2013.

[GKPV10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In *ITCS*, 2010.

[GMM+16] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. In Martin Hirt and Adam Smith, editors, *Theory of Cryptography*, pages 241–268, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[Gol00] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, NY, USA, 2000.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[GS02] Craig Gentry and Mike Szydlo. Cryptanalysis of the revised ntru signature scheme. In *EUROCRYPT*. Springer Berlin Heidelberg, 2002.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions from multiparty computation. In *CRYPTO*, 2012.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute based encryption for circuits. In *STOC*, 2013.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from lwe. In *Crypto*, 2015.

[HJ15] Y. Hu and H. Jia. Cryptanalysis of GGH map. Cryptology ePrint Archive: Report 2015/301, 2015.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In Joe P. Buhler, editor, *Algorithmic Number Theory: Third International Symposiun, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Proceedings*, 1998.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 294–304, 2000.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, pages 244–256, 2002.

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 419–428, 2015.

[KMN$^+$14] Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. One-way functions and (im)perfect obfuscation. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, 2014.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, 2012.

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, pages 28–57, 2016.

[Lin17] Huijia Lin. Indistinguishability obfuscation from sxdh on 5-linear maps and locality-5 prgs. In *Crypto*, 2017.

[LOS$^+$10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, volume 6110, 2010.

[LPST16] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Output-compressing randomized encodings and applications. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 96–124, 2016.

[LT17] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local prgs. In *Crypto*, 2017.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In *FOCS*, 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, http://eprint.iacr.org/2017/301, 2017.

[MI88] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *Advances in Cryptology — EUROCRYPT '88*, pages 419–453, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 700–718, 2012.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM Journal on Computing (SICOMP)*, 37(1):267–302, 2007. extended abstract in FOCS 2004.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over ggh13. In *Crypto*, 2016.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342, 2009.

[Pei16] Chris Peikert. *A Decade of Lattice Cryptography*, volume 10, pages 283–424. 03 2016.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J.ACM*, 56(6), 2009. extended abstract in STOC'05.

[SS11] Damien Stehlé and Ron Steinfeld. Making ntru as secure as worst-case problems over ideal lattices. In *Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT'11, 2011.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, pages 617–635, 2009.

[SW] Amit Sahai and Brent Waters. Functional encryption:beyond public key cryptography. Power Point Presentation, 2008. http://userweb.cs.utexas.edu/bwaters/presentations/files/functional.ppt.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. In *STOC*, 2014. `http://eprint.iacr.org/2013/454.pdf`.

[TW10] Enrico Thomae and Christopher Wolf. Solving systems of multivariate quadratic equations over finite fields or: From relinearization to mutantxl, 2010.

[Wat12] Brent Waters. Functional encryption for regular languages. In *Crypto*, 2012.

[Wol05] Christopher Wolf. *Multivariate Quadratic Polynomials In Public Key Cryptography*. PhD thesis, KATHOLIEKE UNIVERSITEIT LEUVEN, 2005.

[WP05] Christopher Wolf and Bart Preneel. Taxonomy of public key schemes based on the problem of multivariate quadratic equations, 2005.

[YDH$^+$15] Takanori Yasuda, Xavier Dahan, Yun-Ju Huang, Tsuyoshi Takagi, and Kouichi Sakurai. Mq challenge: Hardness evaluation of solving multivariate quadratic problems. Cryptology ePrint Archive, Report 2015/275, 2015. `https://eprint.iacr.org/2015/275`.

Appendices

A Quadratic Functional Encryption from CNG and NLinFE

Formally, quadratic polynomials will be represented by a circuit with a multiplication layer, followed by an addition layer. For the construction below, we will require two prime moduli $p_0 < p_1$ where p_0 serves as the message space for the quadratic scheme, and p_1 serves as the message space for the NLinFE scheme. Below, the distribution $\mathcal{D}_0$ is a discrete Gaussian with width σ_0 and $\mathcal{D}_1$ is a discrete Gaussian with width σ_1. Parameters are instantiated in Appendix 9. Our construction will make use of the following ingredients:

1. **Correlated noise generator** CNG. We will require an correlated noise generator CNG as defined in Section 5. We will denote $(\mathbf{x}_0, \boldsymbol{\mu}) \leftarrow \mathcal{D}^1_{\mathsf{CNG}}$ when $\mathbf{x} \in R^w_{p_0}$ is chosen arbitrarily, and $\boldsymbol{\mu} \leftarrow \mathcal{D}^w_0$. Define $\mathcal{F}^1_{\mathsf{CNG}} : R^{2w} \to R$ as $\mathcal{F}^1_{\mathsf{CNG}} = \{f_{ij}\}_{1 \le j \le i \le w}$ where,
$$f_{ij}(\mathbf{x}, \boldsymbol{\mu}) = p_0 \cdot \big(p_0 \cdot \mu_i \mu_j + x_i \mu_j + x_j \mu_i\big) \quad \forall\, 1 \le j \le i \le w$$
Let $(\mathcal{G}, \mathcal{D}_{\mathsf{seed}})$ be an $(\mathcal{F}^1_{\mathsf{CNG}}, \mathcal{D}^1_{\mathsf{CNG}})$-$\mathsf{CNG}$ with seed $\boldsymbol{\beta} \leftarrow \mathcal{D}^n_{\mathsf{seed}}$. We denote by $G_{ij} \in \mathcal{G}$ the function chosen to smudge the function $f_{ij} \in \mathcal{F}^1_{\mathsf{CNG}}$, namely the joint distribution $\{f_{ij}(\mathbf{x}, \boldsymbol{\mu}) + G_{ij}(\boldsymbol{\beta})\}_{i,j}$ will be indistinguishable from $\{G_{ij}(\boldsymbol{\beta})\}_{i,j}$ to any P.P.T adversary for all $1 \le j \le i \le w$. We will informally refer to G_{ij} as *smudging polynomials.*
To support the addition layer following multiplication, we let $\mathcal{D}^2_{\mathsf{CNG}} = \mathcal{D}^1_{\mathsf{CNG}}$ and let $\mathcal{F}^2_{\mathsf{CNG}}$ contain functions of the form $\sum\limits_{1 \le j \le i \le w} g_{ij} f_{ij}$ for some $g_{ij} \in R_{p_0}$. However, linear combinations can be smudged by taking the same linear combinations of the corresponding smudging polynomials, i.e. if $\{f_{ij}(\mathbf{x}, \boldsymbol{\mu}) + G_{ij}(\boldsymbol{\beta})\}_{i,j}$ is indistinguishable from $\{G_{ij}(\boldsymbol{\beta})\}_{i,j}$, then it follows that
$$\Big\{ \sum_{1 \le j \le i \le w} g_{ij}\ \big(\ f_{ij}(\mathbf{x}, \boldsymbol{\mu}) + G_{ij}(\boldsymbol{\beta})\ \big) \Big\}_{i,j} \overset{c}{\approx} \Big\{ \sum_{1 \le j \le i \le w} g_{ij}\ G_{ij}(\boldsymbol{\beta})\ \Big\}_{i,j}$$

2. **Noisy linear functional encryption** NLinFE. We require a $(\mathcal{D}_{\mathsf{NFE}}, \mathcal{F}_{\mathsf{NFE}}, B_{\mathsf{NFE}})$-$\mathsf{NLinFE}$ scheme where (i). $(\mathbf{x}_0, \boldsymbol{\mu}, \boldsymbol{\beta}) \leftarrow \mathcal{D}_{\mathsf{NFE}}$ when $(\mathbf{x}_0, \boldsymbol{\mu}) \leftarrow \mathcal{D}^1_{\mathsf{CNG}}$ and $\boldsymbol{\beta} \leftarrow \mathcal{D}_{\mathsf{seed}}$ (ii). $\mathcal{F}_{\mathsf{NFE}} = \big\{ \sum\limits_{1 \le j \le i \le w} g_{ij}\ \ h_{ij} \big\}_{1 \le j \le i \le w}$ with $h_{ij}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\beta}) = f_{ij}(\mathbf{x}, \boldsymbol{\mu}) + G_{ij}(\boldsymbol{\beta})$ and (iii) B_{NFE} is chosen superpolynomially larger than any $\big|\ h_{ij}(\mathbf{x}_0, \boldsymbol{\mu}, \boldsymbol{\beta})\ \big|$. For parameters, please see Appendix 9.

Construction. We proceed to describe the construction, denoted by QuadFE.

$\mathsf{QuadFE.Setup}(1^\kappa, 1^w)$: On input a security parameter κ and a parameter w denoting the length of message vectors, do:

1. Invoke $\mathsf{NLinFE.Setup}(1^\kappa, 1^{w+2})$ to obtain $\mathsf{NLinFE.PK}$ and $\mathsf{NLinFE.MSK}$.
2. Sample $\mathbf{u} \leftarrow R^w_{p_1}$ and sample the seed of an CNG as $\boldsymbol{\beta} \leftarrow \mathcal{D}^n_{\mathsf{seed}}$.
3. For $1 \le j \le i \le w$, compute key_{ij} as follows:
   - For $1 \le j \le i \le w$, and let $p_0 \cdot \tilde{\mu}_{ij} = G_{ij}(\boldsymbol{\beta})$.
   - Sample uniform $t_i \leftarrow R_{p_1}$ for $i \in [0, w]$.

- Define
$$\mathsf{key}_{ij} = u_i u_j t_0 - u_j t_i - u_i t_j - p_0 \cdot \tilde{\mu}_{ij}$$
Set $\mathbf{key} = (\mathsf{key}_{ij})_{1 \le j \le i \le w} \in R^L$ where $L = |\{i, j : 1 \le j \le i \le w\}|$[16].

4. Output $\mathsf{PK} = (\mathsf{NLinFE.PK}, \mathbf{u})$, $\mathsf{MSK} = (\mathsf{NLinFE.MSK}, \mathbf{key})$.

$\mathsf{QuadFE.Enc}(\mathsf{PK}, \mathbf{x})$: On input public parameters PK, and message vector $\mathbf{x} \in R^w_{p_0}$ do:

1. Sample $s_1 \leftarrow R_{p_1}$ and $\boldsymbol{\mu} \leftarrow \mathcal{D}_0^w$, and compute the encoding of the message
$$\mathbf{c} = \mathbf{u} \cdot s_1 + p_0 \cdot \boldsymbol{\mu} + \mathbf{x} \in R^w_{p_1}.$$
2. Let $\mathbf{b} = \mathsf{NLinFE.Enc}\,(s_1^2, c_1 s_1, \ldots, c_w s_1, 0)$.
3. Output $\mathsf{CT} = (\mathbf{c}, \mathbf{b})$

$\mathsf{QuadFE.KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathbf{g})$: On input the public parameters PK, the master secret key MSK, and a function $\mathbf{g} = \sum_{1 \le j \le i \le w} g_{ij} x_i x_j$, represented as a coefficient vector $(g_{ij}) \in \mathbb{Z}^L_{p_0}$ do:

1. Compute $\mathsf{key}_{\mathbf{g}} = \sum_{1 \le j \le i \le w} g_{ij} \mathsf{key}_{ij} \in R_{p_1}$.
2. Compute
$$\mathbf{u_g} = \sum_{1 \le j \le i \le w} g_{ij}\,(u_i u_j, 0....0, -u_i, 0...0, -u_j, 0...0) \in R^{w+1}_{p_1}.$$
3. Compute $\mathsf{SK}_{\mathbf{g}} = \mathsf{NLinFE.KeyGen}\big(\mathsf{MSK}, (\mathbf{u_g} | \mathsf{key}_{\mathbf{g}})\big)$ and output it.

$\mathsf{QuadFE.Dec}(\mathsf{PK}, \mathsf{SK}_{\mathbf{g}}, \mathsf{CT}_{\mathbf{x}})$: On input the public parameters PK, a secret key $\mathsf{SK}_{\mathbf{g}}$ for polynomial $\sum_{1 \le j \le i \le w} g_{ij} x_i x_j$, and a ciphertext $\mathsf{CT}_{\mathbf{x}} = (\mathbf{c}, \mathbf{b})$, compute
$$\sum_{1 \le j \le i \le w} g_{ij} c_i c_j + \mathsf{NLinFE.Dec}(\mathbf{b}, \mathsf{SK}_{\mathbf{g}}) \bmod p_0$$
and output it.

Ciphertext Size. The ciphertext in the above scheme is comprised of:
$$\mathbf{c} = \mathbf{u} \cdot s_1 + p_0 \cdot \boldsymbol{\mu} + \mathbf{x} \in R^w_{p_1}, \quad \mathbf{b} = \mathsf{NLinFE.Enc}\,(s_1^2, c_1 s_1, \ldots, c_w s_1, 0)$$
Clearly, $\mathbf{c}$ enjoys linear efficiency. Additionally, note that the message size in $\mathbf{b}$ is linear, hence efficiency of QuadFE ciphertext is inherited from that of NLinFE ciphertext.

[16]We remark that **key** plays no role in correctness of the real system and will only be used in the proof.

Correctness. We show correctness of the quadratic FE scheme.

Theorem A.1. *If* NLinFE *is correct, then the* QuadFE *is correct.*

Proof. Let $1 \leq j \leq i \leq w$. By definition

$$x_i + p_0 \cdot \mu_i = c_i - u_i s_1, \quad x_j + p_0 \cdot \mu_j = c_j - u_j s_1$$

Let

$$\mu_{ij} = x_i \mu_j + x_j \mu_i + p_0 \mu_i \mu_j \tag{A.1}$$

$$x_i x_j + p_0 \cdot \mu_{ij} = c_i c_j - c_i u_j s_1 - c_j u_i s_1 + u_i u_j s_1^2 \tag{A.2}$$

We denote the noise corresponding to monomial $x_i x_j$, added by the scheme NLinFE by $p_0 \cdot \rho_{ij}$. We note that the noise added by NLinFE can easily be chosen to be a multiple of p_0, since this is chosen by the encryptor of the NLinFE scheme, please see Section 8 for details.

By correctness of the linear scheme NLinFE, we have that

$$\begin{aligned}
\mathsf{NLinFE.Dec}(\mathbf{b}, \mathsf{SK}_{\mathbf{g}}) &= \sum_{1\leq j\leq i\leq w} g_{ij}\big(- c_i u_j s_1 - c_j u_i s_1 + u_i u_j s_1^2 + p_0 \cdot \rho_{ij}\big) \\
\sum_{1\leq j\leq i\leq w} g_{ij} c_i c_j + \mathsf{NLinFE.Dec}(\mathbf{b}, \mathsf{SK}_{\mathbf{g}}) &= \sum_{1\leq j\leq i\leq w} g_{ij}\Big(c_i c_j - c_i u_j s_1 - c_j u_i s_1 + u_i u_j s_1^2 + p_0 \cdot \rho_{ij}\Big) \\
&= \sum_{1\leq j\leq i\leq w} g_{ij}\big(x_i x_j + p_0 \cdot \mu_{ij} + p_0 \cdot \rho_{ij}\big) \\
&= \sum_{1\leq j\leq i\leq w} g_{ij}\ x_i x_j \bmod p_0 \quad \text{as long as } p_0 \cdot (\mu_{ij} + \rho_{ij}) \leq \frac{p_1}{5}.
\end{aligned}$$

□

Note that the error recovered in decryption is $p_0 \cdot (\mu_{ij} + \rho_{ij})$.

A.1 Indistinguishability Based Security

Theorem A.2. *Assume the noisy linear FE scheme* NLinFE *satisfies semi-adaptive indistinguishability based security as in Definition 3.4. Assume that G is a secure* CNG *as defined in Definition 5.1. Then, the construction* QuadFE *in Section 4 achieves semi-adaptive indistinguishability based security as in Definition 2.4.*

Proof. We will prove the theorem via a sequence of hybrids, where the first hybrid is the real world with challenge $\mathbf{x}_0$ and the last hybrid is the real world with challenge $\mathbf{x}_1$.

The Hybrids. Our Hybrids are described below.

Hybrid 0. This is the real world with message $\mathbf{x}_0$. In hybrid 0, $\mathsf{key}_{\mathbf{g}}$ is picked as follows:

1. Sample $t_0, \ldots, t_w \leftarrow R_{p_1}$ and $p_0 \cdot \tilde{\mu}_{ij} = G_{ij}(\boldsymbol{\beta})$.
2. For $1 \leq j \leq i \leq w$, set $\mathsf{key}_{ij} = \big(u_i u_j t_0 - u_j t_i - u_i t_j - p_0 \cdot G_{ij}(\boldsymbol{\beta})\big) \mod p_1$
3. Let $\mathsf{key}_{\mathbf{g}} = \sum\limits_{1 \leq j \leq i \leq w} g_{ij} \mathsf{key}_{ij} \mod p_1$.

Hybrid 1. In this hybrid, the only thing that is different is that the challenger picks $\mathsf{key}_{\mathbf{g}}$ to depend on the challenge ciphertext $(c_1, \ldots, c_w)$ and the function value $\mathbf{g}(\mathbf{x})$. Specifically,

1. Sample $t_0, \ldots, t_w \leftarrow R_{p_1}$. Let $p_0 \cdot \tilde{\mu}_{ij} \overset{\text{def}}{=} G_{ij}(\boldsymbol{\beta})$.
2. Set

$$\mathsf{key}_{ij} = \big(x_i x_j - c_i c_j\big) - \big(u_i u_j t_0 - u_j t_i - u_i t_j\big) - p_0 \cdot \tilde{\mu}_{ij} \quad \forall\, 1 \leq j \leq i \leq w$$
$$\mathsf{key}_{\mathbf{g}} = \sum_{1 \leq j \leq i \leq w} g_{ij} \mathsf{key}_{ij} \mod p_1$$

Hybrid 2. In this hybrid, we change the input for $\mathsf{NLinFE.Enc}$ to $(t_0, t_1, \ldots, t_w, 1)$ where t_i are chosen uniformly in R_{p_1} for $i \in \{0, \ldots, w\}$.

Hybrid 3. In this hybrid, we change the message vector in $\mathbf{c}$ to $\mathbf{x}_1$.

Hybrids 4 and 5. In Hybrid 4 we change the input to $\mathsf{NLinFE.Enc}$ to $(s_1^2, c_1 s_1, \ldots, c_w s_1, 0)$ as in Hybrid 1. In Hybrid 5, we change $\mathsf{key}_{\mathbf{g}}$ to be chosen independent of the ciphertext as in Hybrid 0. This is the real world with message $\mathbf{x}_1$.

Indistinguishability of Hybrids. Below we establish that consecutive hybrids are indistinguishable.

Claim A.3. *Hybrid 0 and Hybrid 1 are indistinguishable assuming the security of the* CNG G_{ij}.

Proof. The only difference between Hybrid 0 and Hybrid 1 is in the way $\mathsf{key}_{\mathbf{g}}$ is sampled.
In Hybrid 0, we have:

1. Sample $t_0, \ldots, t_w \leftarrow R_{p_1}$ and $\boldsymbol{\beta} \leftarrow \mathcal{D}^n_{\mathsf{seed}}$. Define $p_0 \cdot \tilde{\mu}_{ij} \overset{\text{def}}{=} G_{ij}(\boldsymbol{\beta})$.
2. Set $\mathsf{key}_{ij} = u_i u_j t_0 - u_j t_i - u_i t_j - p_0 \cdot \tilde{\mu}_{ij}$
3. Set $\mathsf{key}_{\mathbf{g}} = \sum\limits_{1 \leq j \leq i \leq w} g_{ij} \mathsf{key}_{ij}$.

In Hybrid 1, $\mathsf{key}_{\mathbf{g}}$ is picked as follows:

1. Sample $t_0, \ldots, t_w \leftarrow R_{p_1}$ and $\boldsymbol{\beta} \leftarrow \mathcal{D}^n_{\mathsf{seed}}$. Let $p_0 \cdot \tilde{\mu}_{ij} \leftarrow G_{ij}(\boldsymbol{\beta})$.
2. Let $\mathsf{key}_{ij} = \big(x_i x_j - c_i c_j\big) - \big(u_i u_j t_0 - u_j t_i - u_i t_j\big) - p_0 \cdot \tilde{\mu}_{ij}$ for $1 \leq j \leq i \leq w$.

3. Set $\mathsf{key}_{\mathbf{g}} = \sum_{1 \le j \le i \le w} g_{ij} \mathsf{key}_{ij}$.

To argue that key_{ij} (and hence $\mathsf{key}_{\mathbf{g}}$) are indistinguishable, we rely on the security of CNG. Assume that an adversary $\mathcal{A}$ distinguishes Hybrid 0 and Hybrid 1. Then we construct an adversary $\mathcal{B}$ to break the security of CNG as follows.

1. $\mathcal{B}$ samples the public key honestly and returns this to $\mathcal{A}$. $\mathcal{A}$ outputs two challenges $\mathbf{x}_0, \mathbf{x}_1$, to which $\mathcal{B}$ encrypts $\mathbf{x}_0$ honestly and returns this to $\mathcal{A}$.

2. $\mathcal{B}$ samples $t_0, \ldots, t_w \leftarrow R_{p_1}$. Say $\mathcal{A}$ requests a key for monomial $x_i x_j$ for some $i, j \le w$. $\mathcal{B}$ does the following. Set
$$y^1_{ij} = \big(x_i x_j - c_i c_j\big) - \big(u_i u_j t_0 - u_j t_i - u_i t_j\big)$$
Now, note that by Equation 4.2,
$$x_i x_j + p_0 \cdot \Big(p_0 \cdot \mu_i \mu_j + x_i \mu_j + x_j \mu_i\Big) = c_i c_j - c_i u_j s_1 - c_j u_i s_1 + u_i u_j s_1^2$$
Hence,
$$\begin{aligned} y^1_{ij} &= \big(u_i u_j s_1^2 - u_i c_j s_1 - u_j c_i s_1 - p_0 \cdot (p_0 \cdot \mu_i \mu_j + x_i \mu_j + x_j \mu_i)\big) - \big(u_i u_j t_0 - u_j t_i - u_i t_j\big) \\ &= u_i u_j (s_1^2 - t_0) - u_i (c_j s_1 - t_j) - u_j (c_i s_1 - t_i) - p_0 \cdot (p_0 \cdot \mu_i \mu_j + x_i \mu_j + x_j \mu_i) \end{aligned}$$
$$\text{Recall } \mu_{ij} = p_0 \cdot \mu_i \mu_j + x_i \mu_j + x_j \mu_i, \quad \text{and let} \quad t'_0 = s_1^2 - t_0, \quad t'_i = c_i s_1 - t_i, \quad \forall\, i \in [w]$$
$$\text{Let } y^0_{ij} = u_i u_j t'_0 - u_i t'_j - u_j t'_i, \quad y^1_{ij} = u_i u_j t'_0 - u_i t'_j - u_j t'_i - p_0 \cdot \mu_{ij}$$
Let $f_{ij}(\mathbf{x}, \boldsymbol{\mu}) = p_0 \cdot \big(p_0 \cdot \mu_i \mu_j + x_i \mu_j + x_j \mu_i\big)$. Return $(f_{ij}, f_{ij}(\mathbf{x}, \boldsymbol{\mu}))_{ij}$ to the CNG challenger.

3. The challenger for CNG sets $p_0 \cdot \tilde{\mu}_{ij} = G_{ij}(\boldsymbol{\beta})$ where $\boldsymbol{\beta} \leftarrow \mathcal{D}^n_{\mathsf{seed}}$ and returns $(1-b) f_{ij}(\mathbf{x}, \boldsymbol{\mu}) + p_0 \cdot \tilde{\mu}_{ij}$ for random bit b. $\mathcal{B}$ sets $\mathsf{key}_{ij} = y^1_{ij} + p_0 \cdot \tilde{\mu}_{ij} + (1-b) f_{ij}(\mathbf{x}, \boldsymbol{\mu})$ and computes the function key for monomial $x_i x_j$ honestly using this value of key_{ij}. This key is returned to $\mathcal{A}$.

4. When the adversary outputs a guess for b, output the same.

Note that the reduction $\mathcal{B}$ is a valid adversary against the CNG. Additionally, if $b = 0$, we are in Hybrid 0, else in Hybrid 1.

□

Indistinguishability of remaining hybrids is argued as in Section 4.3. □

B Direct Construction of NLinFE: The General Case

In this section, we provide the general heuristic construction for noisy linear functional encryption that incorporates all the ideas discussed in Section 1. We will use the special message and function vectors that occur in the construction of QuadFE from NLinFE as described in Section 4.

Construction. We proceed to describe the construction NLinFE.

$\mathsf{Setup}(1^\kappa, 1^{w+2})$: On input a security parameter κ, a parameter $w+2$ denoting the length of the function and message vectors[17], do the following:

1. Choose uniformly random $\mathbf{w} \leftarrow R_{p_2}^m$ with a trapdoor $\mathbf{T_w}$.
2. For $i \in \{0, \ldots, w+1\}$, choose uniformly random $a_i \in R_{p_2}$. Denote $\mathbf{a} = (a_i) \in R_{p_2}^{w+2}$.
3. Sample $\mathbf{u} \leftarrow R_{p_1}^w$ and sample the seed of an CNG as $\boldsymbol{\beta} \leftarrow \mathcal{D}_{\mathsf{seed}}^n$. Recall that $n = O(w, \mathrm{poly}(\kappa))$.
4. For $1 \le j \le i \le w$, compute key_{ij} as follows[18]:
   - For $1 \le j \le i \le w$, and let $p_0 \cdot \tilde{\mu}_{ij} = G_{ij}(\boldsymbol{\beta})$.
   - Sample uniform $t_i \leftarrow R_{p_1}$ for $i \in [0, w]$.
   - Define
   $$\mathsf{key}_{ij} = u_i u_j t_0 - u_j t_i - u_i t_j - p_0 \cdot \tilde{\mu}_{ij}$$

   Set $\mathbf{key} = (\mathsf{key}_{ij})_{1 \le j \le i \le w} \in R^L$ where $L = |\{i, j : 1 \le j \le i \le w\}|$.
5. For $i \in \{1, \ldots, w\}$, $\ell \in \{1, \ldots, k\}$, sample $f_{1i}^\ell, f_{2i}^\ell \leftarrow \mathcal{D}_2$ and $g_1^\ell, g_2^\ell \leftarrow \mathcal{D}_2$. If g_1^ℓ, g_2^ℓ are not invertible over R_{p_2}, resample. Set
$$h_{1i}^\ell = \frac{f_{1i}^\ell}{g_1^\ell}, \quad h_{2i}^\ell = \frac{f_{2i}^\ell}{g_2^\ell} \in R_{p_2}$$
6. Sample a PRF seed, denoted as seed.

Output
$$\mathsf{MSK} = \Big(\mathbf{w}, \mathbf{T_w}, \mathbf{a}, \mathbf{u}, \{f_{1i}^\ell, f_{2i}^\ell\}_{i \in [w], \ell \in [k]}, \{g_1^\ell, g_2^\ell\}_{\ell \in [k]} \}, \mathsf{seed} \Big)$$

$\mathsf{Enc}(\mathsf{MSK}, \mathbf{z})$: On input public key MSK, a message vector $\mathbf{z} = (z_0, z_1, \ldots, z_w, 0) \in R_{p_1}^{w+2}$, do:

1. Sample noise terms $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2 \leftarrow \mathcal{D}_2^m$, $\boldsymbol{\eta} \leftarrow \mathcal{D}_2^{w+2}$ and secret $t_1, t_2 \leftarrow \mathcal{D}_2 \in R_{p_2}$. Set $s = t_1 \cdot t_2$. Choose $s_1 \leftarrow \mathcal{D}_2$ and let $s_2 = s - s_1$.
2. Compute the randomness encodings
$$\mathbf{c}_{00} = \mathbf{w} \cdot s_1 + p_1 \cdot \boldsymbol{\nu}_1 \in R_{p_2}^m.$$
$$\mathbf{c}_{01} = \mathbf{w} \cdot s_2 + p_1 \cdot \boldsymbol{\nu}_2 \in R_{p_2}^m.$$
3. Compute the message encoding
$$\mathbf{b} = \mathbf{a} \cdot s + p_1 \cdot \boldsymbol{\eta} + \mathbf{z} \in R_{p_2}^{w+2}$$

[17]We use $w+2$ to be the length of the function and message vectors to be consistent with the usage of NLinFE in Section 4.

[18]Steps 3 and 4 are to be consistent with function vectors in Section 4, to provide a concrete candidate of QuadFE for cryptanalysis.

4. Let $\widehat{\mathcal{D}}(\Lambda_1^\ell), \widehat{\mathcal{D}}'(\Lambda_1^\ell)$ be spherical discrete Gaussian distributions with parameter σ on $\Lambda_1^\ell \overset{\text{def}}{=} g_1^\ell \cdot R$ and $\widehat{\mathcal{D}}(\Lambda_2^\ell), \widehat{\mathcal{D}}'(\Lambda_2^\ell)$ be spherical discrete Gaussian distributions with parameter σ on $\Lambda_2^\ell \overset{\text{def}}{=} g_2^\ell \cdot R$. Sample the noise terms as:
   (a) For $i \in [w]$, $\ell \in [k]$, sample $e_{1i}^\ell \leftarrow \widehat{\mathcal{D}}(\Lambda_2^\ell)$, $\tilde{e}_{1i}^\ell \leftarrow \widehat{\mathcal{D}}'(\Lambda_2^\ell)$.
$$\text{Let} \quad e_{1i}^\ell = g_2^\ell \cdot \xi_{1i}^\ell, \quad \tilde{e}_{1i}^\ell = g_2^\ell \cdot \tilde{\xi}_{1i}^\ell \tag{B.1}$$
$$\text{Hence,} \quad h_{2i}^\ell \cdot e_{1j}^\ell = f_{2i}^\ell \cdot \xi_{1j}^\ell, \quad h_{2i}^\ell \cdot \tilde{e}_{1j}^\ell = f_{2i}^\ell \cdot \tilde{\xi}_{1j}^\ell \;\; \forall\, i,j \in [w], \ell \in [k] \tag{B.2}$$
   (b) For $i \in [w]$, $\ell \in [k]$, sample $e_{2i}^\ell \leftarrow \widehat{\mathcal{D}}(\Lambda_1^\ell)$, $\tilde{e}_{2i}^\ell \leftarrow \widehat{\mathcal{D}'}(\Lambda_1^\ell)$.
$$\text{Let} \quad e_{2i}^\ell = g_1^\ell \cdot \xi_{2i}^\ell, \quad \tilde{e}_{2i}^\ell = g_1^\ell \cdot \tilde{\xi}_{2i}^\ell \tag{B.3}$$
$$\text{Hence,} \quad h_{1i}^\ell \cdot e_{2j}^\ell = f_{1i}^\ell \cdot \xi_{2j}^\ell, \quad h_{1i}^\ell \cdot \tilde{e}_{2j}^\ell = f_{1i}^\ell \cdot \tilde{\xi}_{2j}^\ell \;\; \forall\, i,j \in [w], \ell \in [k] \tag{B.4}$$

5. Compute the noise encodings as:
   (a) Let
$$d_{1i}^\ell = h_{1i}^\ell \cdot t_1 + p_1 \cdot \tilde{e}_{1i}^\ell + p_0 \cdot e_{1i}^\ell \in R_{p_2}$$
   Here, $p_1 \cdot \tilde{e}_{1i}^\ell$ behaves as noise and $p_0 \cdot e_{1i}^\ell$ behaves as the message. Let $\mathbf{d}_1^\ell = (d_{1i}^\ell)$.
   (b) Similarly, let
$$d_{2i}^\ell = h_{2i}^\ell \cdot t_2 + p_1 \cdot \tilde{e}_{2i}^\ell + p_0 \cdot e_{2i}^\ell \in R_{p_2}$$
   Here, $p_1 \cdot \tilde{e}_{2i}^\ell$ behaves as noise and $p_0 \cdot e_{2i}^\ell$ behaves as the message. Let $\mathbf{d}_2^\ell = (d_{2i}^\ell)$.
6. Output randomness and message encodings $(\mathbf{c}_{00}, \mathbf{c}_{01}, \mathbf{b})$ and noise encodings $(\mathbf{d}_1^\ell, \mathbf{d}_2^\ell)$ for $\ell \in [k]$.

$\mathsf{KeyGen}(\mathsf{MSK}, \mathbf{g})$: On input the master secret key MSK, and a NLinFE function vector (please see Section 4) $\sum\limits_{1\le j\le i\le w} g_{ij}\left(u_iu_j, 0....0, -u_i, 0...0, -u_j, 0...0, \mathsf{key}_{ij}\right) \in R_{p_1}^{w+2}$, do the following.

1. For $1 \le j \le i \le w$, compute
$$u_{ij} = u_iu_ja_0 - u_ia_j - u_ja_i + \mathsf{key}_{ij}a_{w+1} + \sum_{\ell\in[k]} h_{1i}^\ell h_{2j}^\ell \;\; \in R_{p_2} \tag{B.5}$$
2. Using the trapdoor $\mathbf{T_w}$, using randomness $\mathsf{PRF}(\mathsf{seed}, i)$ for $i \in [0, \ldots, w+1]$, $b \in \{0,1\}$, sample short vectors $\mathbf{e}_0^b, \mathbf{e}_i^b, \mathbf{e}_j^b, \mathbf{e}_{w+1}^b \in R^m$ such that
$$\langle \mathbf{w}; \mathbf{e}_0^b\rangle = a_0, \quad \langle \mathbf{w}; \mathbf{e}_i^b\rangle = a_i,$$
3. For $1 \le j \le i \le w$, $b \in \{0,1\}$, sample $\Delta_{ij}^b, \tilde{\Delta}_{ij}^b \leftarrow \mathcal{D}_2 \in R_{p_2}$.
4. Sample short $\mathbf{e}_{ij}^b \in R^m$ using randomness $\mathsf{PRF}(\mathsf{seed}, b, ij)$ such that
$$\langle \mathbf{w};\ \mathbf{e}_{ij}^b\rangle = \sum_{\ell\in[k]} h_{1i}^\ell h_{2j}^\ell + p_0 \cdot \Delta_{ij}^b + p_1 \cdot \tilde{\Delta}_{ij}^b$$

5. Let $\mathbf{k}^b_{ij} = u_i u_j \mathbf{e}^b_0 - u_i \mathbf{e}^b_j - u_j \mathbf{e}^b_i + \mathsf{key}_{ij} \mathbf{e}^b_{w+1} + \mathbf{e}^b_{ij} \quad \in R^m$. Note that

$$\langle \mathbf{w};\ \mathbf{k}^b_{ij} \rangle = u_{ij} + p_0 \cdot \Delta^b_{ij} + p_1 \cdot \tilde{\Delta}^b_{ij} \in R_{p_2} \tag{B.6}$$

6. Compute

$$u_{\mathbf{g}} = \sum_{1 \le j \le i \le w} g_{ij} u_{ij} \quad \in R_{p_2}$$

7. Compute $\mathbf{k}^b_{\mathbf{g}} \in R^m$ as $\mathbf{k}^b_{\mathbf{g}} = \sum_{1 \le j \le i \le w} g_{ij} \mathbf{k}^b_{ij}$. Note that

$$\langle \mathbf{w};\ \mathbf{k}^b_{\mathbf{g}} \rangle = u_{\mathbf{g}} + \sum_{1 \le j \le i \le w} g_{ij} \big(p_0 \cdot \Delta^b_{ij} + p_1 \cdot \tilde{\Delta}^b_{ij} \in R_{p_2}\big) \in R_{p_2}$$

8. Output $(\mathbf{k}^0_{\mathbf{g}}, \mathbf{k}^1_{\mathbf{g}}, \mathbf{g}, \mathbf{u})$ where $\mathbf{g} = (g_{ij})_{1 \le j \le i \le w}$ and $\mathbf{u} = (u_i)_{i \in [w]}$.

$\mathsf{Dec}(\mathsf{CT}_{\mathbf{z}}, \mathsf{SK}_{\mathbf{g}})$: On input a a ciphertext $\mathsf{CT}_{\mathbf{z}} = \big(\ \mathbf{c}_{00}, \mathbf{c}_{01}, \mathbf{b}, \{\mathbf{d}^\ell_1, \mathbf{d}^\ell_2\}_{\ell \in [k]}\ \big)$ and a secret key $(\mathbf{k}^0_{\mathbf{g}}, \mathbf{k}^1_{\mathbf{g}}, \mathbf{g}, \mathbf{u})$ for function $\mathbf{g} = (g_{ij})_{1 \le j \le i \le w}$, do the following

1. Compute

$$b_{ij} = u_i u_j b_0 - u_j b_i - u_i b_j + \mathsf{key}_{ij} b_{w+1} + \sum_{\ell \in [k]} d^\ell_{1i} \cdot d^\ell_{2j} \in R_{p_2}$$

2. Compute $b_{\mathbf{g}} = \sum_{1 \le j \le i \le w} g_{ij} b_{ij}$
3. Compute $b_{\mathbf{g}} - \big(\langle \mathbf{c}_{00},\ \mathbf{k}^0_{\mathbf{g}} \rangle + \langle \mathbf{c}_{01},\ \mathbf{k}^1_{\mathbf{g}} \rangle\big) \mod p_1$ and output it.

Correctness. In this section, we establish that the above scheme is correct. We walk through the steps performed by the decrypt algorithm. In what follows, we will constantly collect small noise terms under the umbrella "small" without keeping track of the precise values. This is because for correctness, the noise terms must merely be maintained below a certain value so that they can be modded out in the end. Their precise shape will be analysed when we discuss security.

1. We compute $d^\ell_{1i} \cdot d^\ell_{2j}$. Recall that

$$d^\ell_{1i} = h^\ell_{1i} \cdot t_1 + p_1 \cdot \tilde{e}^\ell_{1i} + p_0 \cdot e^\ell_{1i} \in R_{p_2}$$

$$d^\ell_{2j} = h^\ell_{2j} \cdot t_2 + p_1 \cdot \tilde{e}^\ell_{2j} + p_0 \cdot e^\ell_{2j} \in R_{p_2}$$

Now, since for all $i, j \in [w]$, we chose :

$$h^\ell_{2j} \cdot e^\ell_{1i} = \mathsf{small}, \quad h^\ell_{2j} \cdot \tilde{e}^\ell_{1i} = \mathsf{small}, \quad h^\ell_{1j} \cdot e^\ell_{2i} = \mathsf{small}, \quad h^\ell_{1j} \cdot \tilde{e}^\ell_{2i} = \mathsf{small}, \quad s = t_1 \cdot t_2$$

we have that $e^\ell_{1i} = g^\ell_2 \cdot \xi^\ell_{1i}, \quad \tilde{e}^\ell_{1i} = g^\ell_2 \cdot \tilde{\xi}^\ell_{1i}, \quad e^\ell_{2j} = g^\ell_1 \cdot \xi^\ell_{2j}, \quad \tilde{e}^\ell_{2j} = g^\ell_1 \cdot \tilde{\xi}^\ell_{2j}$
and $h^\ell_{2i} \cdot e^\ell_{1j} = f^\ell_{2i} \cdot \xi^\ell_{1j}, \quad h^\ell_{2i} \cdot \tilde{e}^\ell_{1j} = f^\ell_{2i} \cdot \tilde{\xi}^\ell_{1j} \quad h^\ell_{1i} \cdot e^\ell_{2j} = f^\ell_{1i} \cdot \xi^\ell_{2j}, \quad h^\ell_{1i} \cdot \tilde{e}^\ell_{2j} = f^\ell_{1i} \cdot \tilde{\xi}^\ell_{2j}$

Now, we may compute:

$$\begin{aligned}
d_{1i}^{\ell} \cdot d_{2j}^{\ell} &= \left(h_{1i}^{\ell} \cdot t_1 + p_1 \cdot \tilde{e}_{1i}^{\ell} + p_0 \cdot e_{1i}^{\ell}\right) \cdot \left(h_{2j}^{\ell} \cdot t_2 + p_1 \cdot \tilde{e}_{2j}^{\ell} + p_0 \cdot e_{2j}^{\ell}\right) \\
&= h_{1i}^{\ell} \cdot h_{2j}^{\ell}(t_1 t_2) + p_1^2 \cdot (\tilde{e}_{1i}^{\ell} \cdot \tilde{e}_{2j}^{\ell}) + p_1 \cdot p_0 (\tilde{e}_{1i}^{\ell} \cdot e_{2j}^{\ell} + e_{1i}^{\ell} \cdot \tilde{e}_{2j}^{\ell}) + p_0^2 (e_{1i}^{\ell} \cdot e_{2j}^{\ell}) \\
&\quad + h_{1i}^{\ell} \cdot t_1 \cdot (p_1 \cdot \tilde{e}_{2j}^{\ell} + p_0 \cdot e_{2j}^{\ell}) + h_{2j}^{\ell} \cdot t_2 \cdot (p_1 \cdot \tilde{e}_{1i}^{\ell} + p_0 \cdot e_{1i}^{\ell}) \\
&= h_{1i}^{\ell} \cdot h_{2j}^{\ell}(t_1 t_2) + p_1^2 \cdot (g_2^{\ell} \cdot \tilde{\xi}_{1i}^{\ell} \cdot g_1^{\ell} \cdot \tilde{\xi}_{2j}^{\ell}) \\
&\quad + p_1 \cdot p_0 (g_2^{\ell} \cdot \tilde{\xi}_{1i}^{\ell} \cdot g_1^{\ell} \cdot \xi_{2j}^{\ell} + g_2^{\ell} \cdot \xi_{1i}^{\ell} \cdot g_1^{\ell} \cdot \tilde{\xi}_{2j}^{\ell}) + p_0^2 (g_2^{\ell} \cdot \xi_{1i}^{\ell} \cdot g_1^{\ell} \cdot \xi_{2j}^{\ell}) \\
&\quad + p_1 \cdot (h_{1i}^{\ell} \cdot \tilde{e}_{2j}^{\ell} \cdot t_1 + h_{2j}^{\ell} \cdot \tilde{e}_{1i}^{\ell} \cdot t_2) + p_0 \cdot (h_{1i}^{\ell} \cdot e_{2j}^{\ell} \cdot t_1 + h_{2j}^{\ell} \cdot e_{1i}^{\ell} \cdot t_2) \\
&= h_{1i}^{\ell} \cdot h_{2j}^{\ell}(t_1 t_2) + p_1^2 \cdot (g_2^{\ell} \cdot \tilde{\xi}_{1i}^{\ell} \cdot g_1^{\ell} \cdot \tilde{\xi}_{2j}^{\ell}) \\
&\quad + p_1 \cdot p_0 (g_2^{\ell} \cdot \tilde{\xi}_{1i}^{\ell} \cdot g_1^{\ell} \cdot \xi_{2j}^{\ell} + g_2^{\ell} \cdot \xi_{1i}^{\ell} \cdot g_1^{\ell} \cdot \tilde{\xi}_{2j}^{\ell}) + p_0^2 (g_2^{\ell} \cdot \xi_{1i}^{\ell} \cdot g_1^{\ell} \cdot \xi_{2j}^{\ell}) \\
&\quad + p_1 \cdot (f_{1i}^{\ell} \cdot \tilde{\xi}_{2j}^{\ell} \cdot t_1 + f_{2j}^{\ell} \cdot \tilde{\xi}_{1i}^{\ell} \cdot t_2) + p_0 \cdot (f_{1i}^{\ell} \cdot \xi_{2j}^{\ell} \cdot t_1 + f_{2j}^{\ell} \cdot \xi_{1i}^{\ell} \cdot t_2) \\
&= h_{1i}^{\ell} \cdot h_{2j}^{\ell} \underbrace{(t_1 t_2)}_{s} + p_1 \cdot \Big(\underbrace{p_1 \cdot (g_2^{\ell} \cdot \tilde{\xi}_{1i}^{\ell} \cdot g_1^{\ell} \cdot \tilde{\xi}_{2j}^{\ell}) + p_0 \cdot (g_2^{\ell} \cdot \tilde{\xi}_{1i}^{\ell} \cdot g_1^{\ell} \cdot \xi_{2j}^{\ell} + g_2^{\ell} \cdot \xi_{1i}^{\ell} \cdot g_1^{\ell} \cdot \tilde{\xi}_{2j}^{\ell})}_{\mathsf{small}} \\
&\quad + \underbrace{(f_{1i}^{\ell} \cdot \tilde{\xi}_{2j}^{\ell} \cdot t_1 + f_{2j}^{\ell} \cdot \tilde{\xi}_{1i}^{\ell} \cdot t_2) \Big)}_{\mathsf{small}} + p_0 \cdot \underbrace{\Big(p_0 \cdot (g_2^{\ell} \cdot \xi_{1i}^{\ell} \cdot g_1^{\ell} \cdot \xi_{2j}^{\ell}) + (f_{1i}^{\ell} \cdot \xi_{2j}^{\ell} \cdot t_1 + f_{2j}^{\ell} \cdot \xi_{1i}^{\ell} \cdot t_2)\Big)}_{\mathsf{small}}
\end{aligned} \tag{B.7}$$

Above, we highlight in blue the terms in the noise that will remain fixed across all ciphertexts.

$$\text{Thus, } \sum_{\ell \in [k]} d_{1i}^{\ell} \cdot d_{2j}^{\ell} = \Big(\sum_{\ell \in [k]} h_{1i}^{\ell} \cdot h_{2j}^{\ell} \Big) \cdot s + p_1 \cdot \mathsf{small} + p_0 \cdot \mathsf{small} \tag{B.8}$$

2. Recall that by definition B.5, we have:

$$u_{ij} = u_i u_j a_0 - u_i a_j - u_j a_i + \mathsf{key}_{ij} a_{w+1} + \sum_{\ell \in [k]} h_{1i}^{\ell} h_{2j}^{\ell} \quad \in R_{p_2}$$

and that $\langle \mathbf{w};\ \mathbf{k}_{\mathbf{g}}^b \rangle = u_{\mathbf{g}} + \sum_{1 \le j \le i \le w} g_{ij} \big(p_0 \cdot \Delta_{ij}^b + p_1 \cdot \tilde{\Delta}_{ij}^b \in R_{p_2}\big) \in R_{p_2}$. Recall that $\mathbf{c}_{0b} = \mathbf{w} \cdot s_b + p_1 \cdot \boldsymbol{\nu}_b$, hence,

$$\begin{aligned}
\langle \mathbf{c}_{0b},\ \mathbf{k}_{\mathbf{g}}^b \rangle &= \Big(u_{\mathbf{g}} + \sum_{1 \le j \le i \le w} g_{ij} \big(p_0 \cdot \Delta_{ij}^b + p_1 \cdot \tilde{\Delta}_{ij}^b\big)\Big) \cdot s_b + p_1 \cdot (\boldsymbol{\nu}_b^{\mathsf{T}} \mathbf{k}_{\mathbf{g}}^b) \\
&= u_{\mathbf{g}} \cdot s_b + p_1 \cdot \mathsf{small} + p_0 \cdot \mathsf{small} \\
\langle \mathbf{c}_{00},\ \mathbf{k}_{\mathbf{g}}^0 \rangle + \langle \mathbf{c}_{01},\ \mathbf{k}_{\mathbf{g}}^1 \rangle &= u_{\mathbf{g}} \cdot (s_0 + s_1) + p_1 \cdot \mathsf{small} + p_0 \cdot \mathsf{small} \\
&= u_{\mathbf{g}} \cdot s + p_1 \cdot \mathsf{small} + p_0 \cdot \mathsf{small}
\end{aligned}$$

Above, the second step follows because the term $\sum_{1 \le j \le i \le w} g_{ij} \big(p_0 \cdot \Delta_{ij}^b + p_1 \cdot \tilde{\Delta}_{ij}^b$, as well as secrets s_1 and s_2 are small by design.

Recall that $\mathbf{b} = \mathbf{a} \cdot s + p_1 \cdot \boldsymbol{\eta} + \mathbf{z} \in R_{p_2}^{w+2}$. Now, let

$$\begin{aligned}
b_{ij} &= u_i u_j b_0 - u_j b_i - u_i b_j + \mathsf{key}_{ij} b_{w+1} + \sum_{\ell \in [k]} d_{1i}^{\ell} \cdot d_{2j}^{\ell} \\
&= \Big(u_i u_j a_0 - u_j a_i - u_i a_j + \mathsf{key}_{ij} a_{w+1} + \sum_{\ell \in [k]} h_{1i}^{\ell} h_{2j}^{\ell} \Big) s + p_1 \cdot \mathsf{small} + \\
&\quad \big(u_i u_j \cdot z_0 - u_j \cdot z_i - u_i \cdot z_j + \mathsf{key}_{ij} \cdot 0 + p_0 \cdot \mathsf{small} \big) \\
&= u_{ij} \cdot s + p_1 \cdot \mathsf{small} + \big(u_i u_j \cdot z_0 - u_j \cdot z_i - u_i \cdot z_j + p_0 \cdot \mathsf{small} \big) \\
\therefore \sum_{1 \le j \le i \le w} g_{ij} b_{ij} &= u_{\mathbf{g}} \cdot s + \Big(\sum_{1 \le j \le i \le w} g_{ij} \big(u_i u_j \cdot z_0 - u_j \cdot z_i - u_i \cdot z_j \big) \Big) \\
&\quad + p_1 \cdot \mathsf{small} + p_0 \cdot \mathsf{small}
\end{aligned}$$

Hence,

$$b_{\mathbf{g}} - \Big(\langle \mathbf{c}_{00},\ \mathbf{k}_{\mathbf{g}}^{0} \rangle + \langle \mathbf{c}_{01},\ \mathbf{k}_{\mathbf{g}}^{1} \rangle \Big) \mod p_1 = \Big(\sum_{1 \le j \le i \le w} g_{ij} \big(u_i u_j \cdot z_0 - u_j \cdot z_i - u_i \cdot z_j \big) \Big) + p_0 \cdot \mathsf{small}$$

Thus, we may recover the required linear function plus noise, as long we set the modulus p_2 to be large enough so that the cumulative noise terms $p_1 \cdot \mathsf{small}$ may be bounded below $\frac{p_2}{5}$ (say).

C Towards a Proof of Security for the construction in Section 8

In this section, we formulate an assumption, under which a proof of security may be provided for our scheme. The resultant assumption does not accurately capture some important aspects of the real system, and therefore, while sufficient, is not necessary for security of our scheme. We emphasize that we view our attempt in this section as a first step, and hope for improvements in future works.

Theorem C.1. *The construction of* NLinFE *provided in Section 8 satisfies* Full-Sel *security (Definition 2.4) under the following two assumptions:*

1. *Distributions* DistrL *and* DistrR *defined in Figure C.1 and C.2 are indistinguishable.*

2. *Distribution seen by the adversary defined in Figure C.3 hides the bit b.*

Proof. We argue the proof via a sequence of hybrids, described below.

Hybrid 0: This is the real world with message $\mathbf{z}_b$ where $b \leftarrow \{0,1\}$.

Hybrid 1: This is the same as Hybrid 0, except for the way in which $\mathbf{b}$ is generated.

$$\text{Let } \ \mathbf{b} = \mathbf{E}^{\mathsf{T}} \mathbf{c} + p_1 \cdot \boldsymbol{\eta} + \mathbf{z}_b \in R_{p_2}^{w}$$

Hybrid 2: This is the same as Hybrid 1 except:

1. Sample all relevant elements from DistrR, namely, for $i, j \in [w]$ and $\ell \in [k]$,

$$\begin{pmatrix} \mathbf{c} & \mathbf{d}_1^{\ell}, & \mathbf{d}_2^{\ell}, & \mathbf{E}^{\times} \end{pmatrix} \leftarrow \mathsf{DistrR}$$

Distribution DistrL

1. Sample $\mathbf{w}$ uniformly from $R_{p_2}^m$
2. Sample noise terms $\boldsymbol{\nu} \leftarrow \mathcal{D}^m$ and secret $t_1, t_2 \leftarrow \mathcal{D} \in R_{p_2}$. Set $s = t_1 \cdot t_2$.
3. Set $\mathbf{c} = \mathbf{w} \cdot s + p_1 \cdot \boldsymbol{\nu}$.
4. For $i \in \{1, \ldots, w\}$, $\ell \in \{1, \ldots, k\}$, sample $f_{1i}^\ell, f_{2i}^\ell \leftarrow \mathcal{D}$ and $g_1^\ell, g_2^\ell \leftarrow \mathcal{D}$. If g_1^ℓ, g_2^ℓ are not invertible over R_{p_2}, resample. Set
$$h_{1i}^\ell = \frac{f_{1i}^\ell}{g_1^\ell}, \quad h_{2i}^\ell = \frac{f_{2i}^\ell}{g_2^\ell} \in R_{p_2}$$
5. For $i \in [w]$, $\ell \in [k]$, sample
$$e_{1i}^\ell \leftarrow \widehat{\mathcal{D}}(\Lambda_2^\ell), \quad \tilde{e}_{1i}^\ell \leftarrow \widehat{\mathcal{D}'}(\Lambda_2^\ell) \text{ where } \Lambda_2^\ell \overset{\text{def}}{=} g_2^\ell \cdot R. \text{ Let } \ e_{1i}^\ell = g_2^\ell \cdot \xi_{1i}^\ell, \quad \tilde{e}_{1i}^\ell = g_2^\ell \cdot \tilde{\xi}_{1i}^\ell.$$
$$e_{2i}^\ell \leftarrow \widehat{\mathcal{D}}(\Lambda_1^\ell), \quad \tilde{e}_{2i}^\ell \leftarrow \widehat{\mathcal{D}'}(\Lambda_1^\ell) \text{ where } \Lambda_1^\ell \overset{\text{def}}{=} g_1^\ell \cdot R. \text{ Let } \ e_{2i}^\ell = g_1^\ell \cdot \xi_{2i}^\ell, \quad \tilde{e}_{2i}^\ell = g_1^\ell \cdot \tilde{\xi}_{2i}^\ell.$$
6. For $i \in [w]$, $\ell \in [k]$, compute
$$d_{1i}^\ell = h_{1i}^\ell \cdot t_1 + p_1 \cdot \tilde{e}_{1i}^\ell + p_0 \cdot e_{1i}^\ell \in R_{p_2}$$
$$d_{2i}^\ell = h_{2i}^\ell \cdot t_2 + p_1 \cdot \tilde{e}_{2i}^\ell + p_0 \cdot e_{2i}^\ell \in R_{p_2}$$
Let $\mathbf{d}_1^\ell = (d_{1i}^\ell)$ and $\mathbf{d}_2^\ell = (d_{2i}^\ell)$.
7. Sample short vectors $\mathbf{e}_{ij}$ so that
$$\langle \mathbf{w};\ \mathbf{e}_{ij} \rangle = \sum_{\ell \in [k]} h_{1i}^\ell h_{2j}^\ell. \quad \text{Let } \mathbf{E}^\times = (\mathbf{e}_{ij}) \in R^{m \times L}$$
8. Output
$$\Big(\mathbf{c} \quad \{\mathbf{d}_1^\ell, \quad \mathbf{d}_2^\ell\}_{\ell \in [k]}, \quad \mathbf{E}^\times \Big)$$

Figure C.1: Distribution DistrL

Indistinguishability of Hybrids We now argue that consecutive hybrids are indistinguishable.

Claim C.2. *We claim that Hybrid 0 and 1 are indistinguishable.*

Proof. The only difference between the two hybrids is in how $(\mathbf{a}, \mathbf{b}, \mathbf{E})$ are generated. We argue these are indistinguishable. Note that $\mathbf{a}$ is distributed identically in Hybrids 0 and 1 by [GPV08, Lem 5.2]. Similarly, $\mathbf{E}$ is distributed identically in both Hybrids by [GPV08, Sec 5.3.2]. It remains to argue that $\mathbf{b}$ is indistinguishable in both worlds. This is a standard argument in LWE based

Distribution DistrR

1. Sample $\mathbf{c}$ uniformly from $R_{p_2}^m$ with a trapdoor $\mathbf{T}$ using $\mathsf{TrapGen}$.
2. For $\ell \in [k]$, sample $\tilde{\mathbf{v}}^\ell$ with trapdoor $\mathbf{T}^\ell$ using $\mathsf{TrapGen}$.
3. For $\ell \in [k]$, sample $g_1^\ell, g_2^\ell \leftarrow \mathcal{D}$ (re-sample if any is not invertible).
4. For $\ell \in [k]$, use trapdoor $\mathbf{T}$ to sample $\tilde{\mathbf{E}}^\ell \leftarrow \mathcal{D}^{m\times m}$ such that $\mathbf{c}^\top \tilde{\mathbf{E}}^\ell = (\tilde{\mathbf{v}}^\ell)^\intercal \cdot \frac{1}{g_1^\ell}$.
5. For $\ell \in [k]$, sample $f_{11}^\ell, f_{21}^\ell, \ldots f_{1w}^\ell, f_{2w}^\ell, \leftarrow \mathcal{D}$.
6. Let $\tilde{d}_{1i}^\ell = \frac{f_{1i}^\ell}{g_1^\ell}$ and $\mathbf{E}_i = \sum\limits_{\ell\in[k]} f_{1i}^\ell \cdot \tilde{\mathbf{E}}^\ell$ for $i \in [w]$.
7. Sample $\mathbf{e}_1, \ldots, \mathbf{e}_w \leftarrow \mathcal{D}^m$ using trapdoor $\mathbf{T}^\ell$ such that $(\tilde{\mathbf{v}}^\ell)^\intercal \mathbf{e}_j = \frac{f_{2j}^\ell}{g_2^\ell}$

$$\text{Note that, } \mathbf{c}^\intercal\, \mathbf{E}_i = \sum_{\ell\in[k]} f_{1i}^\ell \cdot \mathbf{c}^\intercal\, \tilde{\mathbf{E}}^\ell = \sum_{\ell\in[k]} \frac{f_{1i}^\ell}{g_1^\ell} \cdot (\tilde{\mathbf{v}}^\ell)^\intercal = \sum_{\ell\in[k]} \tilde{d}_{1i}^\ell \cdot (\tilde{\mathbf{v}}^\ell)^\intercal$$

$$\text{Hence, } \mathbf{c}^\intercal \big(\mathbf{E}_i\, \mathbf{e}_j\big) = \sum_{\ell\in[k]} \Big(\tilde{d}_{1i}^\ell \cdot (\tilde{\mathbf{v}}^\ell)^\intercal\Big) \mathbf{e}_j = \sum_{\ell\in[k]} \tilde{d}_{1i}^\ell\, \tilde{d}_{2j}^\ell \quad \in R_{p_2}$$

8. We let $\mathbf{e}_{ij} = \mathbf{E}_i \mathbf{e}_j$. Let $\mathbf{E}^\times = (\mathbf{e}_{ij}) \in R^{m\times L}$ for $1 \le j \le i \le w$.
9. Let $d_{1i}^\ell = \tilde{d}_{1i}^\ell + p_1 \cdot \tilde{e}_{1i}^\ell + p_0 \cdot e_{1i}^\ell$. Similarly let, $d_{2j}^\ell = \tilde{d}_{2j}^\ell + p_1 \cdot \tilde{e}_{2i}^\ell + p_0 \cdot e_{2i}^\ell$. Here, $\tilde{e}_{1i}^\ell$, e_{1i}^ℓ, $\tilde{e}_{2i}^\ell$, e_{2i}^ℓ are as chosen in Figure C.1.
10. Output

$$\Big(\mathbf{c} \quad \{\mathbf{d}_1^\ell, \quad \mathbf{d}_2^\ell\}_{\ell\in[k]}, \qquad \mathbf{E}^\times\Big)$$

Figure C.2: Distribution DistrR

constructions, and proceeds as follows. In Hybrid 0, we have

$$\begin{aligned}
\mathbf{b} &= \mathbf{E}^\intercal \mathbf{c} + p_1 \cdot \boldsymbol{\eta} + \mathbf{z}_0 \\
&= \mathbf{E}^\intercal \big(\mathbf{w} \cdot s + p_1 \cdot \boldsymbol{\nu}\big) + p_1 \cdot \boldsymbol{\eta} + \mathbf{z}_0 \\
&= \mathbf{a} \cdot s + p_1 \cdot (\mathbf{E}^\intercal \boldsymbol{\nu} + \boldsymbol{\eta}) + \mathbf{z}_0 \\
&= \mathbf{a} \cdot s + p_1 \cdot \boldsymbol{\eta} + \mathbf{z}_0 \;\text{ as long as } \mathsf{SD}(\boldsymbol{\eta}, \mathbf{E}^\intercal \boldsymbol{\nu} + \boldsymbol{\eta}) = \text{negl}(\kappa)
\end{aligned}$$

as in Hybrid 1. □

Claim C.3. *Hybrid 1 and Hybrid 2 are indistinguishable.*

This follows directly from the assumption. Note that answering the key queries and construction of challenge ciphertext remains the same in both hybrids. Note that decryption works correctly in Hybrid 2.

□

Advantage of the Adversary. It remains to argue that the adversary has negligible advantage in winning the game in Hybrid 2. We examine the view of the adversary in Figure C.3.

View of the Adversary

The adversary sees the following:

1. **Ciphertext**:

$$\mathbf{c},\ \mathbf{b} = \mathbf{E}^{\mathsf{T}}\mathbf{c} + p_1 \cdot \boldsymbol{\eta} + \mathbf{z}_b,\ \ d^{\ell}_{1i} = \frac{f^{\ell}_{1i}}{g^{\ell}_1} + p_1 \cdot (g^{\ell}_2 \cdot \tilde{\xi}^{\ell}_{1i}) + p_0 \cdot (g^{\ell}_2 \cdot \xi^{\ell}_{1i})$$

$$d^{\ell}_{2j} = \frac{f^{\ell}_{2j}}{g^{\ell}_2} + p_1 \cdot (g^{\ell}_1 \cdot \tilde{\xi}^{\ell}_{2j}) + p_0 \cdot (g^{\ell}_1 \cdot \xi^{\ell}_{2j}) \ \text{ for } \ell \in [k], 1 \le j \le i \le w.$$

2. **Keys**: $\mathbf{E} \cdot \mathbf{v} + \mathbf{E}^{\times} \cdot \mathbf{v}^{\times}$ where each column of $\mathbf{E}^{\times}$ is $\mathbf{e}_{ij} = \sum_{\ell \in [k]} f^{\ell}_{1i} \cdot \tilde{\mathbf{E}}^{\ell}\mathbf{e}_j$.

   Note that $\mathbf{c}^{\mathsf{T}}(\mathbf{E}_i\mathbf{e}_j) = \sum_{\ell \in [k]} \frac{f^{\ell}_{1i}}{g^{\ell}_1}\frac{f^{\ell}_{2j}}{g^{\ell}_2}$ as required for decryption.

Figure C.3: View of the Adversary

Note that the noise in the product encoding $\sum_{\ell \in [k]} d^{\ell}_{1i} d^{\ell}_{2j}$ is:

$$\beta_{ij} = \sum_{\ell \in [k]} \Bigg(p_1 \cdot g^{\ell}_2 \cdot g^{\ell}_1 \cdot \Big(p_1 \cdot (\tilde{\xi}^{\ell}_{1i} \cdot \tilde{\xi}^{\ell}_{2j}) + p_0 \cdot (\tilde{\xi}^{\ell}_{1i} \cdot \xi^{\ell}_{2j} + \xi^{\ell}_{1i} \cdot \tilde{\xi}^{\ell}_{2j}) \Big)$$
$$+ g^{\ell}_2 \cdot g^{\ell}_1 \cdot p_0^2 \cdot \xi^{\ell}_{1i} \cdot \xi^{\ell}_{2j} + p_0 \cdot (f^{\ell}_{1i}\xi^{\ell}_{2j} + f^{\ell}_{2j}\xi^{\ell}_{1i}) + p_1 \cdot (f^{\ell}_{1i}\tilde{\xi}^{\ell}_{2j} + f^{\ell}_{2j}\tilde{\xi}^{\ell}_{1i}) \Bigg)$$

Letting $\boldsymbol{\beta}^{\times} = (\beta_{ij})_{1 \le j \le i \le w}$, the total noise recovered by the adversary is $(\mathbf{v}^{\times})^{\mathsf{T}}\boldsymbol{\beta}^{\times}$. Thus, decryption reveals a high degree polynomial in noise terms. For security, we require:

1. The secret terms g^{ℓ}_1, g^{ℓ}_2, f^{ℓ}_{1i}, f^{ℓ}_{2j} should not be recoverable from the above polynomial $(\mathbf{v}^{\times})^{\mathsf{T}}\boldsymbol{\beta}^{\times}$. Note that in general this equation can be of degree larger than 2, as discussed in Section 8.2. More broadly, the semantic security of $\mathbf{d}^{\ell}_1$, $\mathbf{d}^{\ell}_2$ must not be compromised by learning these noise terms.

2. The noise recovered above must be sufficient to computationally flood the correlated noise term that occurs in the difference of challenge message decryptions, as described in Section 4.

3. The function keys are of the form $\mathbf{E} \cdot \mathbf{v}_i + \mathbf{E}^{\times} \cdot \mathbf{v}^{\times}_i$ for $i \in [\ell]$. Note that if we could argue that $\mathbf{E}$ was hidden, then we could apply the leftover hash lemma to argue that $\mathbf{b} = \mathbf{E}^{\mathsf{T}}\mathbf{c} + p_1 \cdot \boldsymbol{\eta} + \mathbf{z}_b$ hides $\mathbf{z}_b$. This is indeed the approach taken in [ALS16]. However, since $\mathbf{E}^{\times}$ contains correlated entries, we cannot hope that $\mathbf{E}$ retain entropy after polynomially many keys are seen by the adversary. Instead, one may hope that $\mathbf{E}$ retain sufficient entropy in the view of a

computationally bounded adversary even given the leakage of the function keys, so that $\mathbf{E}^{\mathsf{T}}\mathbf{c}$ appears random to her.[19]

Conjecture C.4. *We conjecture that the distribution in Figure C.3 hides the bit b.*

Under this assumption, we have that the advantage of the adversary in the $\mathsf{Full\text{-}Sel}$ game against our construction is negligible.

Limitations of our assumption.

We remark that the simulated distribution described in Figure C.2 does not accurately capture some important aspects of the real world distribution in Figure C.1. In the real world, the matrix $\mathbf{E}^{\times}$ can be constructed by sampling each column $\mathbf{e}_{ij}$ independently to satisfy $\mathbf{E}^{\times\mathsf{T}}\mathbf{w} = \mathbf{h} + p_0 \cdot \mathbf{\Delta} + p_1 \cdot \tilde{\mathbf{\Delta}}$. This relation ensures that $\mathbf{E}^{\times\mathsf{T}}\mathbf{c} \approx \mathbf{d}^{\times}$ for all ciphertexts, enabling decryption to succeed for all requested ciphertexts.

In the simulation, to be able to support multiple ciphertexts, decryption requires that the relation $\mathbf{E}^{\times\mathsf{T}}\mathbf{c} \approx \mathbf{d}^{\times}$ should hold for any ciphertext $(\mathbf{c}, \{\mathbf{d}_1^{\ell}, \mathbf{d}_2^{\ell}\}_{\ell\in[k]})$ and $\mathbf{d}^{\times}$ constructed from $\{\mathbf{d}_1^{\ell}, \mathbf{d}_2^{\ell}\}_{\ell\in[k]}$. Traditional proofs in LWE based constructions handle this by having the simulator sample the short preimage $\mathbf{E}^{\times}$ first and compute $\mathbf{d}^{\times}$ accordingly. The resultant distribution of $\mathbf{d}^{\times}$ thus obtained is uniform [GPV08].

In our setting, the distribution of the vector $\mathbf{d}^{\times}$ is not uniform, and the above trick does not apply. To get around this, we design the columns $\mathbf{e}_{ij}$ in $\mathbf{E}^{\times}$ to be correlated as $\mathbf{E}_i\mathbf{e}_j$ and design the ciphertext so that the decryption equation holds. This approach results in us re-using some elements chosen while constructing $\mathbf{E}_i$ in the construction of the ciphertext. In particular, this results in multiple ciphertexts re-using the numerators f_{1i}^{ℓ}. Thus, multiple elements in a single ciphertext have the same denominator and all elements in a given position across multiple ciphertexts have the same numerator in the simulated ciphertexts. This does not accurately reflect the real system, which does not require such a re-use, and is a severe limitation of our current assumption and proof technique.

Moreover, since the ciphertext $\mathbf{d}_1^{\ell}, \mathbf{d}_2^{\ell}$ must have a special form to permit decryption, our assumption relies on a trapdoor for $\mathbf{c}$. This is also quite dissatisfying, since a trapdoor for $\mathbf{c}$ is much more powerful than what we require – intuitively we are required to sample a short vector that maps $\mathbf{c}$ to an element of the shape $\frac{\mathsf{small}}{\mathsf{small}}$, which by NTRU is indistinguishable from uniform. Here, it is crucial to note that *any* element from the NTRU distribution would suffice for our purposes. But we do not know any method to achieve this other than using a full trapdoor for $\mathbf{c}$, which enables sampling a short vector to map $\mathbf{c}$ to any arbitrary element. Note that the adversary learning a full trapdoor for $\mathbf{c}$ would surely compromise security but we do not believe the trapdoor is really being leaked since we are only using it for sampling pre-images of random looking images.

Lastly, our construction contains additional randomizers Δ_{ij} and $\tilde{\Delta}_{ij}$ (please see Section 8), which we could not incorporate into our assumption. These randomizers add noise terms into the decryption equation which are jointly generated by the key and the ciphertext and serve to further offset the ideal structure of the constructed noise.

In conclusion, we feel that our assumption does not capture the security of the real system very well, and may be false without compromising the security of the real system. Nevertheless, we view

[19]Note that the correlated values of $\mathbf{E}^{\times}$ cause the system of equations seen by the adversary to be over-defined. However, in general the system of equations is of high degree multivariate polynomials and these are not sufficiently many to yield to linearization attacks.

this as a first step, and hope that more sophisticated trapdoor sampling techniques and more careful "programming" will lead to more accurate assumptions on which future constructions can be based.

D Non-Succinct NLinFE from LinFE

In this section, we construct a noisy functional encryption scheme for linear functions which supports Q queries for a fixed polynomial Q. Security posits that an adversary cannot distinguish between encryptions of $\mathbf{z}_0$ and $\mathbf{z}_1$ as long as $|\mathbf{g}_i(\mathbf{z}_0) - \mathbf{g}_i(\mathbf{z}_1)| \le \epsilon$ for every key $\mathbf{g}_i$ requested, as long as the number of requested keys is less than Q. We emphasize that Q can be greater than the dimension of the message/key vectors, namely w – indeed this is the case we will be interested in.

To support $Q > w$ queries so as to achieve the security definition stated above, we artificially add noise to decryption, so that any two messages whose decryption under a key is equal upto ϵ noise, will decrypt to only approximately correct values but will have indistinguishable ciphertexts.

For ease of notation, our description below assumes a stateful keygen. We may get rid of this restriction using standard tricks as described in [AR17].

D.1 Construction.

The algorithms for the noisy linear FE scheme, denoted by NLinFE are defined as follows.

$\mathsf{Setup}(1^\kappa, 1^Q, 1^w, p_1)$: On input a security parameter κ, a parameter w denoting the length of the function and message vectors, a parameter Q denoting the number of queries supported and a modulus p_1 denoting the space of the message and function vectors, set $(\mathsf{PK}, \mathsf{MSK}) = \mathsf{LinFE.Setup}(1^{w+1+Q})$.

$\mathsf{KeyGen}(\mathsf{MSK}, \mathbf{g}, i)$: On input MSK, a function vector $\mathbf{g} \in \mathbb{Z}_{p_1}^w$ and an index $i \in [Q]$ denoting query number, do:

1. Sample $\gamma_i \leftarrow \mathcal{D}_1$, where $\mathcal{D}_1$ is a discrete Gaussian of width large enough to flood ϵ.
2. Output $\mathsf{SK}_{\mathbf{g}} = \mathsf{LinFE.KeyGen}(\mathbf{g}\|\gamma_i\|\mathbf{e}_i)$ where $\mathbf{e}_i \in \mathbb{Z}^Q$ is the i^{th} unit vector. Note that the LinFE key explicitly contains the vector $(\mathbf{g}\|\gamma_i\|\mathbf{e}_i)$.

$\mathsf{Enc}(\mathsf{PK}, \mathbf{z})$: On input public key PK, a message vector $\mathbf{z} \in \mathbb{Z}_{p_1}^w$, do:

1. Sample $\boldsymbol{\delta} \leftarrow \mathcal{D}_2^Q$ and $\mu \leftarrow \mathcal{D}$ subject to the following constraints:
   - $\mathcal{D}_2$ is a discrete Gaussian of width large enough to flood $\mathcal{D}_1$. Thus, it will hold that $\delta_i + \gamma_i \approx \delta_i$, where γ_i is chosen by keygen.
   - $\mathcal{D}$ has width large enough so that μ is distributed indistinguishably from $\mu + 1$.
2. Let $\mathsf{CT}_{\mathbf{z}} = \mathsf{LinFE.Enc}(\mathbf{z}\|\mu\|\boldsymbol{\delta})$

$\mathsf{Dec}(\mathsf{SK}_{\mathbf{g}_i}, \mathsf{CT}_{\mathbf{z}})$: On input a secret key $\mathsf{SK}_{\mathbf{g}_i}$ for function $\mathbf{g}_i$, and a ciphertext $\mathsf{CT}_{\mathbf{z}}$:

1. Compute $\mathsf{LinFE.Dec}(\mathsf{CT}_{\mathbf{z}}, \mathsf{SK}_{\mathbf{g}_i})$ to recover $\mathbf{g}_i^\mathsf{T}\mathbf{z} + \mu \cdot \gamma_i + \delta_i$.

Appropximate correctness is evident since we recovered the correct value upto noise $\mu \cdot \gamma_i + \delta_i$.

D.2 Security.

Next, we argue that the above scheme is secure. We have that for every key query $\mathbf{g}_i$, $i \in [Q]$, it holds that $\mathbf{g}_i^\mathsf{T}(\mathbf{z}_0 - \mathbf{z}_1) = \epsilon_i$. We argue via a sequence of hybrids.

Hybrid 0. This is the real world with message $\mathbf{z}_0$. The challenge CT encrypts $\mathbf{y}_0 = (\mathbf{z}_0 \| \mu \| \boldsymbol{\delta})$.

Hybrid 1. In this world, we generate the challenge CT for the message $\mathbf{y}_0 = (\mathbf{z}_0 \| \mu \| \hat{\boldsymbol{\delta}} + \boldsymbol{\delta})$ where $\boldsymbol{\delta}$ floods $\hat{\boldsymbol{\delta}}$. For the keys, we set $\boldsymbol{\gamma} = \hat{\boldsymbol{\delta}}$.

Hybrid 2. In this world, the noise in the Q keys changes to $\boldsymbol{\gamma} = \hat{\boldsymbol{\delta}} + \boldsymbol{\epsilon}$.

Hybrid 3. In this world, we change the message in the challenge CT to $\mathbf{y}_1 = (\mathbf{z}_1 \| \mu + 1 \| \boldsymbol{\delta})$.

Hybrid 4. In this world, we rewrite the message in the challenge CT as $\mathbf{y}_1 = (\mathbf{z}_1 \| \mu \| \boldsymbol{\delta})$. This is the real world with message $\mathbf{z}_1$.

Indistinguishability of Hybrids. It is evident that Hybrids 0 and 1 are statistically indistinguishable, since $\boldsymbol{\delta}$ floods $\hat{\boldsymbol{\delta}}$. By the same argument, Hybrid 1 and 2 are statistically indistinguishable since $\hat{\boldsymbol{\delta}}$ floods $\boldsymbol{\epsilon}$ and Hybrid 3 and Hybrid 4 are statistically indistinguishable since μ is distributed indistinguishably from $\mu + 1$. The chief transition that must be argued is that between Hybrids 2 and 3, which we argue now.

Claim D.1. *If the exact linear scheme is* AD-IND *secure, then Hybrids 2 and 3 are indistinguishable.*

Proof. Given an adversary $\mathcal{A}$ who can distinguish between Hybrids 2 and 3, we will construct an adversary $\mathcal{B}$ who will break the security of the exact linear scheme. $\mathcal{B}$ plays the AD-IND game with the LinFE challenger, denoted by $\mathcal{C}$.

1. The LinFE challenger $\mathcal{C}$ outputs the public key PK, which $\mathcal{B}$ forwards to $\mathcal{A}$.

2. $\mathcal{A}$ requests a key $\mathbf{g}_i$ for $i \in [\ell_1]$, where $\ell_1 \le Q$. $\mathcal{B}$ chooses γ_i as in the real world and requests a key for $\hat{\mathbf{g}}_i = (\mathbf{g} \| \gamma_i \| \mathbf{e}_i)$ to the LinFE challenger $\mathcal{C}$. The challenger returns $\mathsf{LinFE.SK}(\hat{\mathbf{g}}_i)$ which $\mathcal{B}$ gives $\mathcal{A}$.

3. $\mathcal{A}$ outputs two challenges $\mathbf{z}_0, \mathbf{z}_1 \in \mathbb{Z}_{p_1}^w$. $\mathcal{B}$ checks that $\mathbf{g}_i^\mathsf{T}(\mathbf{z}_0 - \mathbf{z}_1) = \epsilon_i \le \epsilon$ for all queries $\mathbf{g}_i$ requested so far. If this condition does not hold, output $\perp$ and abort.

4. $\mathcal{B}$ chooses $\hat{\delta}_i = \gamma_i - \epsilon_i$ for $i \in [\ell_1]$. The remaining $\hat{\delta}_{\ell_1+1} \ldots \hat{\delta}_Q$ it chooses as in the real world. Next, it constructs $\hat{\mathbf{y}}_0 = (\mathbf{z}_0 \| \mu \| \hat{\boldsymbol{\delta}})$ and $\hat{\mathbf{y}}_1 = (\mathbf{z}_1 \| \mu + 1 \| \mathbf{0})$ and returns these to the LinFE challenger $\mathcal{C}$ as the challenge messages.

5. $\mathcal{A}$ may request more keys $\mathbf{g}_i$ so that $\mathbf{g}_i^\mathsf{T}(\mathbf{z}_0 - \mathbf{z}_1) = \epsilon_i$. $\mathcal{B}$ chooses $\gamma_i = \hat{\delta}_i + \epsilon_i$ and requests a key for $\hat{\mathbf{g}}_i = (\mathbf{g} \| \gamma_i \| \mathbf{e}_i)$ to the LinFE challenger. The challenger returns $\mathsf{LinFE.SK}(\hat{\mathbf{g}}_i)$, which $\mathcal{B}$ gives $\mathcal{A}$.

6. $\mathcal{C}$ outputs the challenge ciphertext $\mathsf{CT}(\hat{\mathbf{y}}_b)$. $\mathcal{B}$ adds noise $\boldsymbol{\delta}$ to the last Q coordinates and returns this to $\mathcal{A}$ as the challenge CT. Thus, $\mathcal{A}$ sees an encryption of either $\mathbf{y}_0 = (\mathbf{z}_0 \| \mu \| \hat{\boldsymbol{\delta}} + \boldsymbol{\delta})$ or $\mathbf{y}_1 = (\mathbf{z}_1 \| \mu + 1 \| \boldsymbol{\delta})$.

7. When $\mathcal{A}$ outputs a guess for bit b, $\mathcal{B}$ outputs the same.

Observe that the query $\hat{\mathbf{g}}_i$ is an admissible query for the LinFE challenger because:

$$\begin{aligned}
\hat{\mathbf{g}}_i^{\mathsf{T}} \hat{\mathbf{y}}_0 &= \mathbf{g}_i^{\mathsf{T}} \mathbf{z}_0 + \mu \cdot \gamma_i + \hat{\delta}_i \\
\hat{\mathbf{g}}_i^{\mathsf{T}} \hat{\mathbf{y}}_1 &= \mathbf{g}_i^{\mathsf{T}} \mathbf{z}_1 + \mu \cdot \gamma_i + \gamma_i \\
&= \mathbf{g}_i^{\mathsf{T}} \mathbf{z}_0 - \epsilon_i + \mu \cdot \gamma_i + \hat{\delta}_i + \epsilon_i \\
&= \mathbf{g}_i^{\mathsf{T}} \mathbf{z}_0 + \mu \cdot \gamma_i + \hat{\delta}_i \\
&= \hat{\mathbf{g}}_i^{\mathsf{T}} \hat{\mathbf{y}}_0
\end{aligned}$$

If the LinFE challenger $\mathcal{C}$ returns an encryption of $\hat{\mathbf{y}}_0$, then $\mathcal{A}$ sees an encryption of $\mathbf{y}_0 = (\mathbf{z}_0 \| \mu \| \hat{\boldsymbol{\delta}} + \boldsymbol{\delta})$, otherwise it sees an encryption of $\mathbf{y}_1 = (\mathbf{z}_1 \| \mu + 1 \| \boldsymbol{\delta})$. In the former case we obtain the distribution of Hybrid 2, in the latter case of Hybrid 3. □

Hence, we argued that for Q (for $Q > w$) queries, our noisy linear FE scheme satisfies (ϵ, Q) $\mathsf{NsyAD\text{-}IND}$ security as per Definiton 3.4.

E Public Key and Ciphertext Evaluation Algorithms

In this section, we recap the tools provided by [AR17] to extend our construction for quadratic polynomials to NC_1. This section is taken verbatim from [AR17].

Throughout this section, we assume circular security of LWE. This is for ease of exposition as well as efficiency. This assumption can be removed by choosing new randomness s_i for each level i as in levelled fully homomorphic encryption. Since the intuition was discussed in Section 1, we proceed with the technical overview and construction.

Notation. To begin, it will be helpful to set up some notation. We will consider circuits of depth d, consisting of alternate addition and multiplication layers. Each layer of the circuit is associated with a modulus p_k for level k. For an addition layer at level k, the modulus p_k will be the same as the previous modulus p_{k-1}; for a multiplication layer at level k, we require $p_k > p_{k-1}$. This results in a tower of moduli $p_0 < p_1 = p_2 < p_3 = \ldots < p_d$. The smallest modulus p_0 is associated with the message space of the scheme.

We define encoding functions $\mathcal{E}^k$ for $k \in [d]$ such that $\mathcal{E}^k : R_{p_{k-1}} \to R_{p_k}$. At level k, the encryptor will provide L^k encodings $\mathcal{C}^k$ for some $L^k = O(2^k)$. For $i \in [L^k]$ we define

$$\mathcal{E}^k(y_i) = u_i^k \cdot s + p_{k-1} \cdot \eta_i^k + y_i \mod p_k$$

Here $u_i^k \in R_{p_k}$, $\eta_i^k \leftarrow \chi_k$ and $y_i \in R_{p_{k-1}}$. The RLWE secret s is reused across all levels as discussed above, hence is chosen at the first level, i.e. $s \leftarrow R_{p_1}$. We will refer to $\mathcal{E}^k(y_i)$ as the Regev encoding of y_i. At level k, the decryptor will be able to compute a Regev encoding of $f^k(\mathbf{x})$ where f^k is the circuit f restricted to level k.

It will be convenient for us to denote encodings of functional values at every level, i.e. $f^k(\mathbf{x})$ by c^k, i.e. $c^k = \mathcal{E}^k\big(f^k(\mathbf{x})\big)$. Here, c^k are encodings computed on the fly by the decryptor whereas $\mathcal{C}^k$ (described above) are a *set* of level k encodings provided by the encryptor to enable the decryptor to compute c^k. We will denote the public key or label of an encoding $\mathcal{E}^k(\cdot)$ (resp. c^k) by $\mathsf{PK}(\mathcal{E}^k(\cdot))$ (resp. $\mathsf{PK}(c^k)$).

In our construction, we will compose encodings, so that encodings at a given level are messages to encodings at the next level. We refer to such encodings as *nested encodings*. In nested encodings at level $k+1$, messages may be level k encodings *or* level k encodings times the RLWE secret s. We define the notions of nesting level and nested message degree as follows.

Definition E.1 (Nesting level and Nested Message Degree.)**.** Given a composition of successive encodings, i.e. a *nested* encoding of the form $\mathcal{E}^k\big(\mathcal{E}^{k-1}\big(\ldots(\mathcal{E}^{\ell+1}(\mathcal{E}^\ell(y)\cdot s)\cdot s)\ldots\cdot s\big)\cdot s\big)$, we will denote as *nesting level* the value $k-\ell$, the *nested message* of the encoding as y, and the *nested message degree* of the encoding as the degree of the innermost polynomial y.

Note that in the above definition of nested message, we consider the message in the innermost encoding and ignore the multiplications by s between the layers.

We prove the following theorem.

Theorem E.2. *There exists a set of encodings $\mathcal{C}^i$ for $i \in [d]$, such that:*

1. **Encodings have size sublinear in circuit.** $\forall i \in [d]\ |\mathcal{C}^i| = O(2^i)$.

2. **Efficient public key and ciphertext evaluation algorithms.** *There exist efficient algorithms $\mathsf{Eval}_{\mathsf{PK}}$ and $\mathsf{Eval}_{\mathsf{CT}}$ so that for any circuit f of depth d, if $\mathsf{PK}_f = \mathsf{Eval}_{\mathsf{PK}}(\mathsf{PK}, f)$ and $\mathsf{CT}_{(f(\mathbf{x}))} = \mathsf{Eval}_{\mathsf{CT}}(\underset{i\in[d]}{\cup}\mathcal{C}^i, f)$, then $\mathsf{CT}_{(f(\mathbf{x}))}$ is a "Regev encoding" of $f(\mathbf{x})$ under public key PK_f. Specifically, for some LWE secret s, we have:*
$$\mathsf{CT}_{(f(\mathbf{x}))} = \mathsf{PK}_f \cdot s + p_{d-1}\cdot \eta_f^{d-1} + \mu_{f(\mathbf{x})} + f(\mathbf{x}) \tag{E.1}$$
where $p_{d-1}\cdot\eta_f^{d-1}$ is RLWE noise and $\mu_{f(\mathbf{x})} + f(\mathbf{x})$ is the desired message $f(\mathbf{x})$ plus some noise $\mu_{f(\mathbf{x})}$[20]*. Here, $\mu_{f(\mathbf{x})} = p_{d-2}\cdot\eta_f^{d-2} + \ldots p_0\cdot\eta_f^0$ for some noise terms $\eta_f^{d-2},\ldots,\eta_f^0$.*

3. **Ciphertext and public key structure.** *The structure of the functional ciphertext is as:*
$$\mathsf{CT}_{f(\mathbf{x})} = \mathsf{Poly}_f(\mathcal{C}^1,\ldots,\mathcal{C}^{d-1}) + \langle \mathsf{Lin}_f, \mathcal{C}^d\rangle \tag{E.2}$$
where $\mathsf{Poly}_f(\mathcal{C}^1,\ldots,\mathcal{C}^{d-1}) \in R_{p_{d-1}}$ is a high degree polynomial value obtained by computing a public f-dependent function on level $k \le d-1$ encodings $\{\mathcal{C}^k\}_{k\in[d-1]}$ and $\mathsf{Lin}_f \in R_{p_d}^{L_d}$ is an f-dependent linear function. We also have
$$f(\mathbf{x}) + \mu_{f(\mathbf{x})} = \mathsf{Poly}_f(\mathcal{C}^1,\ldots,\mathcal{C}^{d-1}) + \langle \mathsf{Lin}_f, \mathcal{M}^d\rangle \tag{E.3}$$
where $\mathcal{M}^d$ are the messages encoded in $\mathcal{C}^d$ and $\mu_{f(\mathbf{x})}$ is functional noise. The public key for the functional ciphertext is structured as:
$$\mathsf{PK}\big(\mathsf{CT}_{f(\mathbf{x})}\big) = \Big\langle \mathsf{Lin}_f,\ \big(\mathsf{PK}(\mathcal{C}_1^d),\ldots,\mathsf{PK}(\mathcal{C}_{L_d}^d)\big)\Big\rangle \tag{E.4}$$

[20]Here $\mu_{f(\mathbf{x})}$ is clubbed with the message $f(\mathbf{x})$ rather than the RLWE noise $p_{d-1}\cdot\eta_f^{d-1}$ since $\mu_{f(\mathbf{x})} + f(\mathbf{x})$ is what will be recovered after decryption of $\mathsf{CT}_{f(\mathbf{x})}$, whereas $p_{d-1}\cdot\eta_f^{d-1}$ will be removed by the decryption procedure. This is merely a matter of notation.

The Encodings. We define $\mathcal{C}^k$ recursively as follows:

1. $\mathcal{C}^1 \stackrel{\text{def}}{=} \{\mathcal{E}^1(x_i), \mathcal{E}^1(s)\}$

2. If k is a multiplication layer, $\mathcal{C}^k = \{\mathcal{E}^k(\mathcal{C}^{k-1}), \mathcal{E}^k(\mathcal{C}^{k-1}\cdot s), \mathcal{E}^k(s^2)\}$. If k is an addition layer, let $\mathcal{C}^k = \mathcal{C}^{k-1}$.

We prove that:

Lemma E.3. *Assume that k is a multiplication layer. Given $\mathcal{C}^k$ for any $2 < k < d$,*

1. *Level k encodings $\mathcal{E}^k(c^{k-1}\cdot s)$ and $\mathcal{E}^k(c^{k-1})$ may be expressed as quadratic polynomials in level $k-1$ encodings and level k advice encodings $\mathcal{C}^k$. In particular, the polynomials are linear in terms $\mathcal{C}^k$ and quadratic in level $k-1$ encodings $\mathcal{E}^{k-1}(y_i)\mathcal{E}^{k-1}(y_j)$. The messages y_i, y_j of the form c_ℓ^{k-3} or $c_\ell^{k-3}\cdot s$ for some level $k-3$ ciphertext c_ℓ^{k-3}.*

   Since the exact value of the coefficients is not important, we express this as:

$$\mathcal{E}^k(c^{k-1}\cdot s), \mathcal{E}^k(c^{k-1}) = \mathsf{LinComb}\big(\,\mathcal{C}^k, \mathcal{E}^{k-1}(y_i)\mathcal{E}^{k-1}(y_j)\big)\ \ \forall\, i,j \tag{E.5}$$

2. *We can compute c^k and c^{k+1} as a linear combination of quadratic terms in level $k-1$ encodings and linear in level k encodings $\mathcal{C}^k$. In particular,*

$$\begin{aligned} c^k = \mathsf{CT}(f^k(\mathbf{x}) + \mu^k_{f(\mathbf{x})}) &= \langle \mathsf{Lin}_{f^k}, \mathcal{C}^k\rangle + \mathsf{LinComb}\big(\mathsf{Quad}(\mathcal{E}^{k-1}(y_i)\ \mathcal{E}^{k-1}(y_j))\big) \\ &= \langle \mathsf{Lin}_{f^k}, \mathcal{C}^k\rangle + \mathsf{Poly}_{f^k}\big(\mathcal{C}^1, \ldots, \mathcal{C}^{k-1}\big) \end{aligned}$$

Proof by induction.

Base Case. While the quadratic scheme described in Section 4 suffices as a base case, we work out an extended base case for level 4 circuits, since this captures the more general case. Moreover polynomials of degree 4 suffice for computing randomized encodings of circuits in P [IK02], which we use in our general construction.

We claim that $\mathcal{C}^4$ defined according to the above rules, permits the evaluator to compute :

1. $\mathcal{E}^4(c^3\cdot s)$ and $\mathcal{E}^4(c^3)$ by taking linear combinations of elements in $\mathcal{C}^4$ and adding to this a quadratic term of the form $\mathcal{E}^3(y_i)\mathcal{E}^3(y_j)$ where $\mathcal{E}^3(y_i)\mathcal{E}^3(y_j) \in \mathcal{C}^3 = \mathcal{C}^2$. We note that since $k-1$ is an addition layer, $\mathcal{C}^3 = \mathcal{C}^2$.

2. Encodings of level 4 functions of $\mathbf{x}$, namely c^4.

Note that our level 2 ciphertext may be written as:

$$\begin{aligned} c^2_{i,j} &= \mathcal{E}^2(x_i x_j + p_0\cdot\mu_{ij}) = \mathcal{E}^2\big(c_i^1 c_j^1 +\ u_i^1 u_j^1(s^2) - u_j^1(c_i^1 s) - u_i^1(c_j^1 s)\ \big) \\ &= \mathcal{E}^2(x_i x_j + p_0\cdot\mu_{ij}) = c_i^1 c_j^1 + \mathcal{E}^2\big(\ u_i^1 u_j^1(s^2) - u_j^1(c_i^1 s) - u_i^1(c_j^1 s)\ \big) \\ &= c_i^1 c_j^1 + u_i^1 u_j^1\ \mathcal{E}^2(s^2) - u_j^1\ \mathcal{E}^2(c_i^1 s) - u_i^1\ \mathcal{E}^2(c_j^1 s) \quad \in R_{p_2} \end{aligned} \tag{E.6}$$

In the above, the first equality follows by additive malleability of RLWE: here, $c_i^1 c_j^1 \in R_{p_1}$ is a message added to the encoding $\mathcal{E}^2(u_i^1 u_j^1(s^2) - u_j^1(c_i^1 s) - u_i^1(c_j^1 s))$. The second equality follows by additive homomorphism of the encodings.

Additionally, the public key and the noise of the resultant encoding may be computed as:

$$\begin{aligned} u_\ell^2 \overset{\text{def}}{=} \mathsf{PK}\left(\mathcal{E}^2(x_i x_j + p_0 \cdot \mu_{ij})\right) &= u_i^1 u_j^1\ \mathsf{PK}\left(\mathcal{E}^2(s^2)\right) - u_j^1\ \mathsf{PK}\left(\mathcal{E}^2(c_i^1 s)\right) - u_i^1\ \mathsf{PK}\left(\mathcal{E}^2(c_j^1 s)\right) \\ \mathsf{Nse}_\ell^2 \overset{\text{def}}{=} \mathsf{Nse}\left(\mathcal{E}^2(x_i x_j + p_0 \cdot \mu_{ij})\right) &= u_i^1 u_j^1\ \mathsf{Nse}\left(\mathcal{E}^2(s^2)\right) - u_j^1\ \mathsf{Nse}\left(\mathcal{E}^2(c_i^1 s)\right) - u_i^1\ \mathsf{Nse}\left(\mathcal{E}^2(c_j^1 s)\right) \end{aligned}$$

Above, $\mathsf{Nse}(\mathcal{E}^2(\cdot))$ refers to the noise level in the relevant encoding. Note that even though u_i^1 are chosen uniformly in R_{p_1}, they do not blow up the noise in the above equation since the above noise is relative to the larger ring R_{p_2}. This noise growth can be controlled further by using the bit decomposition trick [BV11a, BGV12] – we do not do this here for ease of exposition.

The Quadratic Method. Thus, we may compute a level 2 encoding as:

$$\mathcal{E}^2(x_i x_j + p_0 \cdot \mu_{ij}) = \mathcal{E}^1(x_i)\mathcal{E}^1(x_j) + u_i^1 u_j^1\ \mathcal{E}^2(s^2) - u_j^1\ \mathcal{E}^2(\mathcal{E}^1(x_i) \cdot s) - u_i^1\ \mathcal{E}^2(\mathcal{E}^1(x_j) \cdot s) \tag{E.7}$$

Note that the above equation allows us to express the encoding of the desired product at level 2, namely (a noisy version of) $x_i x_j$, as a quadratic polynomial of the following form: level 1 encodings are in the quadratic terms and level 2 encodings are in the linear terms. This equation will be used recursively in our algorithms below, and will be referred to as the "quadratic method".

The key point is that our level 2 ciphertext has the exact same structure as a level 1 encoding, namely it is a Regev encoding using some secret s, some label and noise as computed in equations E.7. Thus, letting $y_\ell = x_i x_j$, we may write

$$\mathcal{E}^2(y_\ell) = u_\ell^2 \cdot s + \mathsf{Nse}_\ell^2 + y_\ell \quad \in R_{p_2} \tag{E.8}$$

Addition (Level 3). To add two encoded messages $y_\ell = x_i x_j + p_0 \cdot \mu_{ij}$ and $y_{\ell'} = x_{i'} x_{j'} + p_0 \cdot \mu_{i'j'}$, it is easy to see that adding their encodings suffices. The resultant public key and noise is just the summation of the individual public keys and noise terms. Thus, if the ℓ^{th} wire is the sum of the i^{th} and j^{th} wires, we have:

$$c_\ell^3 = c_i^2 + c_j^2 \tag{E.9}$$

and

$$\mathsf{PK}(c_\ell^3) = \mathsf{PK}(c_i^2) + \mathsf{PK}(c_j^2) \tag{E.10}$$

Multiplication (Level 4). The nontrivial case is that of multiplication. We next compute an encoding for the product of $y_\ell = x_i x_j + x_m x_t + p_0 \cdot \mu_\ell^4$ and $y_{\ell'} = x_{i'} x_{j'} + x_{m'} x_{t'} + p_0 \cdot \mu_{\ell'}^4$ where $\mu_\ell^4, \mu_{\ell'}^4$ are level 4 noise terms computed as $\mu_\ell^4 = \mu_{ij} + \mu_{mt}$ (analogously for $\mu_{\ell'}^4$). Let c_ℓ^3 and $c_{\ell'}^3$ denote the encodings of y_ℓ and $y_{\ell'}$ computed using the first three levels of evaluation. As before, we have by the quadratic method:

$$\begin{aligned} c_t^4 = \mathcal{E}^4(y_\ell y_{\ell'}) &= c_\ell^3 c_{\ell'}^3 + \mathcal{E}^4\big(u_\ell^3 u_{\ell'}^3 (s^2) - u_{\ell'}^3 (c_\ell^3 s) - u_\ell^3 (c_{\ell'}^3 s)\big) \quad \in R_{p_4} \\ &= c_\ell^3 c_{\ell'}^3 + u_\ell^3 u_{\ell'}^3\ \mathcal{E}^4(s^2) - u_{\ell'}^3\ \mathcal{E}^4(c_\ell^3 s) - u_\ell^3\ \mathcal{E}^4(c_{\ell'}^3 s) \end{aligned} \tag{E.11}$$

By correctness of first three levels of evaluation as described above, the decryptor can compute the encoding of y_ℓ, namely c_ℓ^3 correctly, hence the quadratic term $c_\ell^3 c_{\ell'}^3$ may be computed. It remains to compute the terms $\mathcal{E}^4(c_\ell^3 s)$. Note that the encryptor may not provide the encodings $\mathcal{E}^4(c_\ell^3 s)$ directly and preserve succinctness because $c_\ell^3 = \mathcal{E}^2(x_i\ x_j + p_0 \cdot \mu_{ij}) + \mathcal{E}^2(x_m\ x_t + p_0 \cdot \mu_{mt})$ and $\mathcal{E}^2(x_i\ x_j + p_0 \cdot \mu_{ij})$ contains the cross term $c_i^1 c_j^1$ as shown by Equation E.6.

Consider the term $\mathcal{E}^4(c^3_\ell s)$. In fact, we will only be able to compute a noisy version of this encoding, i.e. $\mathcal{E}^4(c^3_\ell s + p_1 \cdot \mu^3_\ell)$ for some $p_1 \cdot \mu^3_\ell$.

$$\begin{aligned}
\mathcal{E}^4(c^3_\ell s) &= \mathcal{E}^4\big((\mathcal{E}^2(x_i\, x_j + p_0 \cdot \mu_{ij}) + \mathcal{E}^2(x_m\, x_t + p_0 \cdot \mu_{mt})) \cdot s\big) \\
&= \mathcal{E}^4\Big(\big(c^1_i c^1_j + u^1_i u^1_j\, \mathcal{E}^2(s^2) - u^1_j\, \mathcal{E}^2(c^1_i s) - u^1_i\, \mathcal{E}^2(c^1_j s)\,\big)\, \cdot s\Big) \\
&+ \mathcal{E}^4\Big(\big(c^1_m c^1_t + u^1_m u^1_t\, \mathcal{E}^2(s^2) - u^1_t\, \mathcal{E}^2(c^1_m s) - u^1_m\, \mathcal{E}^2(c^1_t s)\,\big)\, \cdot s\Big) \\
&= \mathcal{E}^4\big(c^1_i c^1_j s\big)\ + \mathcal{E}^4\big(u^1_i u^1_j\, \mathcal{E}^2(s^2)\, s\big)\ - \mathcal{E}^4\big(u^1_j\, \mathcal{E}^2(c^1_i s)\, s\big) - \mathcal{E}^4\big(u^1_i\, \mathcal{E}^2(c^1_j s)\, s\big) \\
&+ \mathcal{E}^4\big(c^1_m c^1_t s\big)\ + \mathcal{E}^4\big(u^1_m u^1_t\, \mathcal{E}^2(s^2)\, s\big) - \mathcal{E}^4\big(u^1_t\, \mathcal{E}^2(c^1_m s) s\big) - \mathcal{E}^4\big(u^1_m\, \mathcal{E}^2(c^1_t s)\, s\big) \\
&= \mathcal{E}^4\big(c^1_i c^1_j s\big)\ + u^1_i u^1_j\, \mathcal{E}^4\big(\mathcal{E}^2(s^2)\, s\big)\ - u^1_j\, \mathcal{E}^4\big(\mathcal{E}^2(c^1_i s)\, s\big) - u^1_i\, \mathcal{E}^4\big(\mathcal{E}^2(c^1_j s)\, s\big) \\
&+ \mathcal{E}^4\big(c^1_m c^1_t s\big)\ + u^1_m u^1_t\, \mathcal{E}^4\big(\mathcal{E}^2(s^2)\, s\big) - u^1_t\, \mathcal{E}^4\big(\mathcal{E}^2(c^1_m s) s\big) - u^1_m\, \mathcal{E}^4\big(\mathcal{E}^2(c^1_t s)\, s\big)
\end{aligned} \tag{E.12}$$

Thus, to compute $\mathcal{E}^4(c^3_\ell s)$ by additive homomorphism, it suffices to compute the encodings $\mathcal{E}^4(c^1_i c^1_j s)$, $\mathcal{E}^4\big(\mathcal{E}^2(s^2)\, s\big)$ and $\mathcal{E}^4\big(\mathcal{E}^2(c^1_j s)\, s\big)$ for all i, j. Note that by definition of $\mathcal{C}^4$, we have that for $m \in [w]$,

$$\Big\{\mathcal{E}^4\big(\mathcal{E}^2(s^2)\, s\big), \quad \mathcal{E}^4\big(\mathcal{E}^2(c^1_m s) s\big)\Big\} \subseteq \mathcal{C}^4 \tag{E.13}$$

Note that since level 3 is an addition layer, $\mathcal{E}^3 = \mathcal{E}^2$.

The only terms above not accounted for are $\mathcal{E}^4(c^1_i c^1_j s)$ and $\mathcal{E}^4\big(c^1_m c^1_t s\big)$, which are symmetric. Consider the former. To compute this, we view $c^1_i c^1_j s$ as a quadratic term in c^1_i and $c^1_j \cdot s$ and re-apply the quadratic method given in Equation E.7. This will enable us to compute a noisy version of $\mathcal{E}^4(c^1_i c^1_j s)$, namely $\mathcal{E}^4(c^1_i c^1_j s + p_1 \cdot \mu^2_{ij})$ for some noise μ^2_{ij}.

Applying the Quadratic Method (Equation E.7): Given $\mathcal{E}^2(c^1_i)$, $\mathcal{E}^2(c^1_j \cdot s)$ along with $\mathcal{E}^4\big(\mathcal{E}^2(c^1_i)\, s\big)$ and $\mathcal{E}^4\big(\mathcal{E}^2(c^1_j \cdot s)\, s\big)$ we may compute $\mathcal{E}^4(c^1_i c^1_j s + p_1 \cdot \mu^2_{ij})$ using the quadratic method. In more detail, we let

$$d_i \overset{\text{def}}{=} \mathcal{E}^2(c^1_i)\ ,\ \ h_j \overset{\text{def}}{=} \mathcal{E}^2(c^1_j \cdot s)\ \in R_{p_2}\ \text{ and }\ \hat{d}_i \overset{\text{def}}{=} \mathcal{E}^4\big(\mathcal{E}^2(c^1_i)\, s\big)\ ,\ \hat{h}_j \overset{\text{def}}{=} \mathcal{E}^4\big(\mathcal{E}^2(c^1_j \cdot s)\, s\big)\ \in R_{p_4}$$

Then, we have:

$$\begin{aligned}
\mathcal{E}^4(c^1_i c^1_j s + p_1 \cdot \mu^2_{ij}) = d_i h_j &+ \mathsf{PK}\big(\mathcal{E}^2(c^1_i)\big)\ \mathsf{PK}\big(\mathcal{E}^2(c^1_j \cdot s)\big)\ \mathcal{E}^4(s^2) \\
&- \mathsf{PK}\big(\mathcal{E}^2(c^1_i)\big)\ \hat{h}_j - \mathsf{PK}\big(\mathcal{E}^2(c^1_j \cdot s)\big)\ \hat{d}_i \ \ \in R_{p_4}
\end{aligned} \tag{E.14}$$

where $\mu^2_{ij} = c^1_i \cdot \mathsf{Nse}(\mathcal{E}^2(c^1_j \cdot s)) + c^2_j \cdot s \cdot \mathsf{Nse}(\mathcal{E}^2(c^1_i)) + p_1 \cdot \mathsf{Nse}(\mathcal{E}^2(c^1_j \cdot s)) \cdot \mathsf{Nse}(\mathcal{E}^2(c^1_i))$.
Again, note that though c_i are large in R_{p_1}, they are small in R_{p_2} upwards, and may be clubbed with noise terms as done above.

Also, the public key for $\mathcal{E}^4(c^1_i c^1_j s + p_1 \cdot \mu^2_{ij})$ may be computed as:

$$\begin{aligned}
\mathsf{PK}\big(\mathcal{E}^4(c^1_i c^1_j s + p_1 \cdot \mu^2_{ij})\big) = &\ \mathsf{PK}\big(\mathcal{E}^2(c^1_i)\big)\ \mathsf{PK}\big(\mathcal{E}^2(c^1_j \cdot s)\big)\ \mathsf{PK}\big(\mathcal{E}^4(s^2)\big) \\
&- \mathsf{PK}\big(\mathcal{E}^2(c^1_i)\big)\ \mathsf{PK}(\hat{h}_j) - \mathsf{PK}\big(\mathcal{E}^2(c^1_j \cdot s)\big)\ \mathsf{PK}(\hat{d}_i)
\end{aligned} \tag{E.15}$$

Thus we have, $\mathcal{E}^4(c^3_\ell s + p_1 \cdot \mu^3_\ell)$ is a Regev encoding with public key

$$\begin{aligned}
&\mathsf{PK}\big(\mathcal{E}^4(c^3_\ell s + p_1 \cdot \mu^3_\ell)\big)\\
&= \mathsf{PK}\Big(\mathcal{E}^4(c^1_i c^1_j s + p_1 \cdot \mu^2_{ij}) \ + u^1_i u^1_j \ \mathcal{E}^4\big(\mathcal{E}^2(s^2) \ s\big) \ - u^1_j \ \mathcal{E}^4\big(\mathcal{E}^2(c^1_i s) \ s\big) - u^1_i \ \mathcal{E}^4\big(\mathcal{E}^2(c^1_j s) \ s\big)\\
&+ \mathcal{E}^4(\big(c^1_m c^1_t s + p_1 \cdot \mu^2_{mt}\big) \ + u^1_m u^1_t \ \mathcal{E}^4\big(\mathcal{E}^2(s^2) \ s\big) - u^1_t \ \mathcal{E}^4\big(\mathcal{E}^2(c^1_m s) s\big) - u^1_m \ \mathcal{E}^4\big(\mathcal{E}^2(c^1_t s) \ s\big)\Big)\\
&= \mathsf{PK}\big(\mathcal{E}^4(c^1_i c^1_j s + p_1 \cdot \mu^2_{ij})\big) \ + u^1_i u^1_j \ \mathsf{PK}\big(\mathcal{E}^4\big(\mathcal{E}^2(s^2) \ s\big)\big) \ - u^1_j \ \mathsf{PK}\big(\mathcal{E}^4\big(\mathcal{E}^2(c^1_i s) \ s\big)\big)\\
&- u^1_i \ \mathsf{PK}\big(\mathcal{E}^4\big(\mathcal{E}^2(c^1_j s) \ s\big)\big) + \mathsf{PK}\big(\mathcal{E}^4(\big(c^1_m c^1_t s + p_1 \cdot \mu^2_{mt}\big)\big) \ + u^1_m u^1_t \ \mathsf{PK}\big(\mathcal{E}^4\big(\mathcal{E}^2(s^2) \ s\big)\big)\\
&- u^1_t \ \mathsf{PK}\big(\mathcal{E}^4\big(\mathcal{E}^2(c^1_m s) s\big)\big) - u^1_m \ \mathsf{PK}\big(\mathcal{E}^4\big(\mathcal{E}^2(c^1_t s) \ s\big)\big)
\end{aligned} \tag{E.16}$$

Above $\mathsf{PK}\big(\mathcal{E}^4(c^1_i c^1_j s + p_1 \cdot \mu^2_{ij})\big)$ may be computed by Equation E.15 and the remaining public keys are provided in $\mathcal{C}^4$ as described in Equation E.13. Also, we have $\mu^3_\ell = \mu^2_{ij} + \mu^2_{mt}$.

By equations E.12, E.13 and E.14, we may compute $\mathcal{E}^4(c^3_\ell s + p_1 \cdot \mu^3_\ell)$ for any ℓ.

Note that,

$$\begin{aligned}
\mathcal{E}^4(c^3_\ell s + p_1 \cdot \mu^3_\ell) &= \mathsf{LinComb}\Big(\mathcal{E}^2(c^1_i) \cdot \mathcal{E}^2(c^1_j \cdot s), \mathcal{E}^4\big(\mathcal{E}^2(c^1_i) \ s\big), \ \mathcal{E}^4\big(\mathcal{E}^2(c^1_j \cdot s) \ s\big)\Big)\\
&= \langle \mathsf{Lin}_{f^4}, \ \mathcal{C}^4 \rangle + \mathsf{Quad}\big(\mathcal{E}^2(c^1_i) \cdot \mathcal{E}^2(c^1_j \cdot s) \ \big)
\end{aligned}$$

for some linear function Lin_{f^4}.

E.1 Ciphertext and Public Key Structure.

By Equation E.11, we then get that

$$\begin{aligned}
c^4_t &= c^3_\ell \ c^3_{\ell'} + u^3_\ell \ u^3_{\ell'} \mathcal{E}^4(s^2) - u^3_\ell \ \Big(\langle \mathsf{Lin}'_{f^4}, \ \mathcal{C}^4 \rangle + \mathsf{Quad}'\big(\mathcal{E}^2(c^1_i) \cdot \mathcal{E}^2(c^1_j \cdot s) \ \big)\Big)\\
&- u^3_{\ell'} \Big(\langle \mathsf{Lin}''_{f^4}, \ \mathcal{C}^4 \rangle + \mathsf{Quad}''\big(\mathcal{E}^2(c^1_i) \cdot \mathcal{E}^2(c^1_j \cdot s) \ \big)\Big)\\
&= \langle \mathsf{Lin}'''_{f^4}, \ \mathcal{C}^4 \rangle + \mathsf{Poly}_{f^4}(\mathcal{C}^1, \mathcal{C}^2, \mathcal{C}^3)
\end{aligned}$$

for some linear functions $\mathsf{Lin}'_{f^4}, \mathsf{Lin}''_{f^4}, \mathsf{Lin}'''_{f^4}$ and quadratic functions Quad', Quad'' and polynomial Poly_{f^4}.

Thus, we have computed $\mathcal{E}^4(c^3_\ell s + p_1 \cdot \mu^3_\ell)$ and hence, c^4 by Equation E.11. The final public key for c^4 is given by:

$$\mathsf{PK}(c^4) = u^3_\ell u^3_{\ell'} \ \mathsf{PK}(\mathcal{E}^4(s^2)) - u^3_{\ell'} \ \mathsf{PK}(\mathcal{E}^4(c^3_\ell s)) - u^3_\ell \ \mathsf{PK}(\mathcal{E}^4(c^3_{\ell'} s)) \tag{E.17}$$

$\mathcal{E}^4(c^3)$ and $\mathcal{E}^4(c^1_i c^1_j)$ are computed analogously. Thus, we have established correctness of the base case.

Note. In the base case, we see that each time the quadratic method is applied to compute an encoding of a product of two messages, we get an encoding of the desired product plus noise.

Induction Step. Assume that the claim is true for level $k-1$. Then we establish that it is true for level k.

By the I.H, we have that:

1. We can compute $\mathcal{E}^{k-1}(c^{k-2} \cdot s)$ and $\mathcal{E}^{k-1}(c^{k-2})$ by taking linear combinations of elements in $\mathcal{C}^{k-1}$ and quadratic terms of the form $\mathcal{E}^{k-2}(y_i)\mathcal{E}^{k-2}(y_j)$ for some y_i, y_j of the form $c_i^{k-4},\ c_j^{k-4}\ s$.

2. We can compute c^{k-1}.

To compute c^k using the quadratic method, it suffices to compute $\mathcal{E}^k(c^{k-1} \cdot s)$.

Computing $\mathcal{E}^k(c^{k-1} \cdot s)$. We claim that:

Claim E.4. *The term $\mathcal{E}^k(c_\ell^{k-1} s)$ (hence c^k) can be computed as a linear combination of elements in $\mathcal{C}^k$ and quadratic terms of the form $\mathcal{E}^{k-1}(\cdot) \cdot \mathcal{E}^{k-1}(\cdot)$.*

Proof. The term $\mathcal{E}^k(c^{k-1} \cdot s)$ may be written as:

$$\begin{aligned}
&\mathcal{E}^k(c^{k-1} \cdot s) \\
&= \mathcal{E}^k\Big(\big(c_i^{k-2}\ c_j^{k-2} - u_i^{k-2}\mathcal{E}^{k-1}(c_j^{k-2} \cdot s) - u_j^{k-2}\mathcal{E}^{k-1}(c_i^{k-2} \cdot s) + u_i^{k-2}u_j^{k-2}\mathcal{E}^{k-1}(s^2)\ \big) \cdot s\Big) \\
&= \mathcal{E}^k(c_i^{k-2}\ c_j^{k-2}\ s) - u_i^{k-2}\mathcal{E}^k\big(\mathcal{E}^{k-1}(c_j^{k-2} \cdot s) \cdot s\big) \\
&- u_j^{k-2}\mathcal{E}^k\big(\mathcal{E}^{k-1}(c_i^{k-2} \cdot s) \cdot s\big) + u_i^{k-2}u_j^{k-2}\mathcal{E}^k\big(\mathcal{E}^{k-1}(s^2) \cdot s\big) \qquad \text{(E.18)}
\end{aligned}$$

Consider $\mathcal{E}^k\big(\mathcal{E}^{k-1}(s^2) \cdot s\big)$. Since $\mathcal{E}^{k-1}(s^2) \in \mathcal{C}^{k-1}$ and $\mathcal{E}^k\big(\mathcal{C}^{k-1} \cdot s\big)$ is contained in $\mathcal{C}^k$, we have that $\mathcal{E}^k\big(\mathcal{E}^{k-1}(s^2) \cdot s\big) \in \mathcal{C}^k$.

Consider the term $\mathcal{E}^k(c_i^{k-2}\ c_j^{k-2}\ s)$. We may compute $\mathcal{E}^k(c_i^{k-2}\ c_j^{k-2}\ s)$ using the quadratic method with messages c_i^{k-2} and $c_j^{k-2}\ s$ as:

$$\begin{aligned}
&\mathcal{E}^k(c_i^{k-2}\ c_j^{k-2}\ s) \\
&= \Big(\mathcal{E}^{k-1}(c_i^{k-2}) \cdot \mathcal{E}^{k-1}(c_j^{k-2} \cdot s)\Big)\ + \mathsf{PK}\big(\mathcal{E}^{k-1}(c_i^{k-2})\big)\mathsf{PK}\big(\mathcal{E}^{k-1}(c_j^{k-2} \cdot s)\big)\ \mathcal{E}^k(s^2) \\
&- \mathsf{PK}\big(\mathcal{E}^{k-1}(c_i^{k-2})\big)\Big(\mathcal{E}^k\big(\mathcal{E}^{k-1}(c_j^{k-2} \cdot s) \cdot s\big)\Big) - \mathsf{PK}\big(\mathcal{E}^{k-1}(c_j^{k-2} \cdot s)\big)\Big(\mathcal{E}^k\big(\mathcal{E}^{k-1}(c_i^{k-2}) \cdot s\big)\Big) \qquad \text{(E.19)}
\end{aligned}$$

Thus, to compute $\mathcal{E}^k(c^{k-1} \cdot s)$, it suffices to compute the term $\mathcal{E}^k(c_i^{k-2}\ c_j^{k-2}\ s)$ since the additional terms such as $\mathcal{E}^k\big(\mathcal{E}^{k-1}(c_i^{k-2} \cdot s) \cdot s\big)$ that appear in Equation E.18 also appear in Equation E.19 and will be computed in the process of computing $\mathcal{E}^k(c_i^{k-2}\ c_j^{k-2}\ s)$.

Note. We observe that in Equation E.19, by "factoring out" the quadratic term $\mathcal{E}^{k-1}(c_i^{k-2}) \cdot \mathcal{E}^{k-1}(c_j^{k-2} \cdot s)$ (which can be computed by I.H.), we reduce the computation of $\mathcal{E}^k(c^{k-1} \cdot s)$ to $\mathcal{E}^k\big(\mathcal{E}^{k-1}(c_j^{k-2} \cdot s) \cdot s\big)$ where the latter value has half the nested message degree (ref. Definition E.1) of the former at the cost of adding one more level of nesting and a new multiplication by s. By recursively applying Equation E.19, we will obtain $O(k)$ quadratic encodings in level $k-1$ and a linear term in level k advice encodings $\mathcal{C}^k$.

Proceeding, we see that to compute $\mathcal{E}^k(c_i^{k-2}\ c_j^{k-2}\ s)$, we are required to compute the following terms:

1. $\mathcal{E}^{k-1}(c_i^{k-2})$ and $\mathcal{E}^{k-1}(c_j^{k-2} \cdot s)$. These can be computed by the induction hypothesis using linear combinations of elements in $\mathcal{C}^{k-1}$ and quadratic terms of the form $\mathcal{E}^{k-2}(y_i)\mathcal{E}^{k-2}(y_j)$ for

some y_i, y_j. Since the precise linear coefficients are not important, we shall denote:

$$\mathcal{E}^{k-1}(c_j^{k-2} \cdot s) = \mathsf{LinComb}\big(\mathcal{C}^{k-1}, \mathcal{E}^{k-2}(\cdot)\mathcal{E}^{k-2}(\cdot)\big) \tag{E.20}$$

2. $\mathcal{E}^k\big(\mathcal{E}^{k-1}(c_i^{k-2}) \cdot s\big)$ and $\mathcal{E}^k\big(\mathcal{E}^{k-1}(c_j^{k-2} \cdot s) \cdot s\big)$: Consider the latter term (the former can be computed analogously).

   By the induction hypothesis,

$$\begin{aligned}
&\mathcal{E}^k\big(\mathcal{E}^{k-1}(c_j^{k-2} \cdot s) \cdot s\big) \\
&= \mathcal{E}^k\Big(\mathsf{LinComb}\big(\mathcal{C}^{k-1}, \mathcal{E}^{k-2}(\cdot)\mathcal{E}^{k-2}(\cdot)\big) \cdot s\Big) \\
&= \mathcal{E}^k\Big(\mathsf{LinComb}\big(\mathcal{C}^{k-1} \cdot s\big)\Big) + \mathcal{E}^k\Big(\mathsf{LinComb}\big(\mathcal{E}^{k-2}(y_a)\mathcal{E}^{k-2}(y_b) \cdot s\big)\Big) \\
&= \mathsf{LinComb}\Big(\mathcal{E}^k\big(\mathcal{C}^{k-1} \cdot s\big)\Big) + \mathsf{LinComb}\Big(\mathcal{E}^k\big(\mathcal{E}^{k-2}(y_a)\mathcal{E}^{k-2}(y_b) \cdot s\big)\Big)
\end{aligned} \tag{E.21}$$

   Again, we note that the terms $\mathcal{E}^k\big(\mathcal{C}^{k-1} \cdot s\big) \in \mathcal{C}^k$ by definition hence it remains to construct $\mathcal{E}^k\Big(\big(\mathcal{E}^{k-2}(y_a)\mathcal{E}^{k-2}(y_b)\big) \cdot s\Big)$ for some $y_a, y_b \in \{c_a^{k-3}, c_b^{k-3} \cdot s\}$.

   To proceed, again, we will consider $z_a = \mathcal{E}^{k-2}(y_a)$ and $z_b = \mathcal{E}^{k-2}(y_b) \cdot s$ as messages and apply the quadratic method to compute an encoding of their product. In more detail,

$$\begin{aligned}
&\mathcal{E}^k\Big(\big(\mathcal{E}^{k-2}(y_a)\mathcal{E}^{k-2}(y_b)\big) \cdot s\Big) \\
&= \mathsf{LinComb}\Big(\mathcal{E}^{k-1}(\mathcal{E}^{k-2}(y_a)) \cdot \mathcal{E}^{k-1}(\mathcal{E}^{k-2}(y_b) \cdot s), \\
&\quad \mathcal{E}^k\big(\mathcal{E}^{k-1}(\mathcal{E}^{k-2}(y_a)) \cdot s\big),\ \mathcal{E}^k\big(\mathcal{E}^{k-1}(\mathcal{E}^{k-2}(y_b) \cdot s) \cdot s\big)\Big)
\end{aligned} \tag{E.22}$$

   Thus, we are required to compute:

   (a) $\mathcal{E}^{k-1}(\mathcal{E}^{k-2}(y_a))$, $\mathcal{E}^{k-1}(\mathcal{E}^{k-2}(y_b) \cdot s)$: These can be computed via the induction hypothesis.

   (b) $\mathcal{E}^k\Big(\mathcal{E}^{k-1}\big(\mathcal{E}^{k-2}(y_a)\big) \cdot s\Big)$ and $\mathcal{E}^k\big(\mathcal{E}^{k-1}(\mathcal{E}^{k-2}(y_b) \cdot s) \cdot s\big)$: Consider the latter term (the former may be computed analogously). Note that

$$\begin{gathered}
\mathcal{E}^{k-2}(y_b) = \mathsf{LinComb}\big(\mathcal{C}^{k-2}, \mathcal{E}^{k-3}(\cdot)\mathcal{E}^{k-3}(\cdot)\big) \\
\therefore \mathcal{E}^k\big(\mathcal{E}^{k-1}(\mathcal{E}^{k-2}(y_b) \cdot s) \cdot s\big) = \mathcal{E}^k\Big(\mathcal{E}^{k-1}\big(\mathsf{LinComb}\big(\mathcal{C}^{k-2}, \mathcal{E}^{k-3}(\cdot)\mathcal{E}^{k-3}(\cdot)\big) \cdot s\big) \cdot s\Big)
\end{gathered}$$

   Again, $\mathcal{E}^k(\mathcal{E}^{k-1}(\mathcal{C}^{k-2} \cdot s) \cdot s) \in \mathcal{C}^k$ so we are left to compute:

$$\begin{aligned}
&\mathcal{E}^k\Big(\mathcal{E}^{k-1}(\mathcal{E}^{k-3}(\cdot)\mathcal{E}^{k-3}(\cdot) \cdot s) \cdot s\Big) \\
&= \mathcal{E}^k\Big(\mathsf{LinComb}\Big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot) \cdot s\big) \cdot \mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot)\big), \\
&\qquad \mathcal{E}^{k-1}\big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot) \cdot s\big) \cdot s\big)\Big)\Big) \\
&= \mathsf{LinComb}\Big(\mathcal{E}^{k-1}\big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot) \cdot s\big)\big) \cdot \mathcal{E}^{k-1}\big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot) \cdot s\big) \cdot s\big), \\
&\quad \mathcal{E}^k\Big(\mathcal{E}^{k-1}\big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot) \cdot s\big) \cdot s\big) \cdot s\Big) \cdot s\Big)
\end{aligned}$$

Thus, again by "factoring out" quadratic term $\mathcal{E}^{k-1}\big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot)\cdot s\big)\big)\cdot\mathcal{E}^{k-1}\big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot)\cdot s\big)\cdot s\big)$, we have reduced computation of $\mathcal{E}^k\big(\mathcal{E}^{k-1}(\mathcal{E}^{k-2}(y_b)\cdot s)\cdot s\big)$ to $\mathcal{E}^k\Big(\mathcal{E}^{k-1}\big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot)\cdot s\big)\cdot s\big)\cdot s\Big)\cdot s\Big)$ which has half the nested message degree of the former at the cost of one additional nesting (and multiplication by s)[21].

Proceeding recursively, we may factor out a quadratic term for each level, to be left with a term which has half the nested message degree and one additional level of nesting. At the last level, we obtain nested encodings which are contained in $\mathcal{C}^k$ by construction. Hence we may compute $\mathcal{E}^k(c^{k-1}\cdot s)$ as a linear combination of quadratic terms of the form $\mathcal{E}^{k-1}(\cdot)\mathcal{E}^{k-1}(\cdot)$ and linear terms in $\mathcal{C}^k$. Please see Figure E.1 for a graphical illustration. Note that the public key $\mathsf{PK}(\mathcal{E}^k(c^{k-1}\cdot s))$ can be computed as a linear combination of the public keys $\mathsf{PK}(\mathcal{C}^k)$, as in Equation E.16.

$$\mathsf{PK}(\mathcal{E}^k(c^{k-1}\cdot s)) = \mathsf{LinComb}(\mathsf{PK}(\mathcal{C}^k)) \tag{E.23}$$

Note that for the public key computation, the higher degree encoding computations are not relevant as these form the message of the final level k encoding.

□

Computing level k ciphertext. Next, we have that:

$$\begin{aligned} c_t^k &= c_\ell^{k-1}c_{\ell'}^{k-1} + \mathcal{E}^k\big(u_\ell^{k-1}u_{\ell'}^{k-1}(s^2) - u_{\ell'}^{k-1}(c_\ell^{k-1}s) - u_\ell^{k-1}(c_{\ell'}^{k-1}s)\big) \\ &= c_\ell^{k-1}c_{\ell'}^{k-1} + u_\ell^{k-1}u_{\ell'}^{k-1}\ \mathcal{E}^k(s^2) - u_{\ell'}^{k-1}\ \mathcal{E}^k(c_\ell^{k-1}s) - u_\ell^{k-1}\ \mathcal{E}^k(c_{\ell'}^{k-1}s) \end{aligned} \tag{E.24}$$

Similarly,

$$\mathsf{PK}(c_t^k) = u_\ell^{k-1}u_{\ell'}^{k-1}\ \mathsf{PK}(\mathcal{E}^k(s^2)) - u_{\ell'}^{k-1}\ \mathsf{PK}\big(\mathcal{E}^k(c_\ell^{k-1}s)\big) - u_\ell^{k-1}\ \mathsf{PK}\big(\mathcal{E}^k(c_{\ell'}^{k-1}s)\big) \tag{E.25}$$

Public Key, Ciphertext and Decryption Structure. From the above, we claim:

Claim E.5. *The public key for c_t^k (for any t) is a publicly computable linear combination of public keys of level k encodings $\mathsf{PK}(\mathcal{E}^k(s^2))$ and $\mathsf{PK}\big(\mathcal{E}^k(c_\ell^{k-1}s)\big)$ for all ℓ.*

Regarding the ciphertext, since we computed $\mathcal{E}^k(c_\ell^{k-1}s)$ from $\mathcal{C}^k$ above, and c^{k-1} may be computed via the induction hypothesis, we may compute c^k as desired. Moreover, since $\mathcal{E}^k(c_\ell^{k-1}s)$ is linear in level k encodings and has quadratic terms in level $k-1$ encodings, we get by unrolling the recursion that $\mathcal{E}^k(c_\ell^{k-1}s)$ and hence level k ciphertext c^k is linear in level k encodings and polynomial in lower level encodings $\mathcal{C}^1,\ldots,\mathcal{C}^{k-1}$. Hence, we have that:

$$\begin{aligned} c^k = \mathsf{CT}(f^k(\mathbf{x}) + \mu_{f(\mathbf{x})}^k) &= \langle \mathsf{Lin}_{f^k}, \mathcal{C}^k\rangle + \mathsf{LinComb}\big(\mathsf{Quad}(\mathcal{E}^{k-1}(y_i)\ \mathcal{E}^{k-1}(y_j))\big) \\ &= \langle \mathsf{Lin}_{f^k}, \mathcal{C}^k\rangle + \mathsf{Poly}_{f^k}\big(\mathcal{C}^1,\ldots,\mathcal{C}^{k-1}\big) \end{aligned}$$

[21] We note that the multiplication by s does not impact the nested message degree, number of nestings or growth of the expression as we proceed down the circuit.

$$\begin{tikzcd}
 & c^k \arrow[dl] \arrow[dr, dashed] & & & & & \\
c_{i_1}^{k-1}c_{j_1}^{k-1} & & \mathcal{E}^k(c_{i_1}^{k-1}s),\ \mathcal{C}^k \arrow[dl] \arrow[dr, dashed] & & & & \\
 & \mathcal{E}^{k-1}(c_{i_2}^{k-2})\cdot\mathcal{E}^{k-1}(c_{j_2}^{k-2}\cdot s) & & \mathcal{E}^k\big(\mathcal{E}^{k-1}(c_{i_2}^{k-2}\cdot s)\cdot s\big),\ \mathcal{C}^k \arrow[dl] \arrow[dr, dashed] & & & \\
 & & \mathcal{E}^{k-1}(\mathcal{E}^{k-2}(c_{i_3}^{k-3}))\cdot\mathcal{E}^{k-1}(\mathcal{E}^{k-2}(c_{j_3}^{k-3})\cdot s) & & \mathcal{E}^k\big(\mathcal{E}^{k-1}(\mathcal{E}^{k-2}(c_{i_3}^{k-3})\cdot s)\cdot s\big),\ \mathcal{C}^k \arrow[dl] \arrow[dr, dashed] & & \\
 & & & \mathcal{E}^{k-1}\Big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot)\cdot s\big)\Big)\cdot\mathcal{E}^{k-1}\Big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot)\cdot s\big)\cdot s\Big) & & \mathcal{E}^k\Big(\mathcal{E}^{k-1}\big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot)\cdot s\big)\cdot s\big)\cdot s\Big)\cdot s\Big),\ \mathcal{C}^k \arrow[d, dotted, no head] & \\
 & & & & & \vdots \arrow[dl] \arrow[dr, dashed] & \\
 & & & & \mathcal{E}^{k-1}\big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\ldots(\mathcal{E}(\cdot)\cdot s)\ldots)\cdot s\big)\big)\cdot\mathcal{E}^{k-1}\big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\ldots(\mathcal{E}(\cdot)\cdot s)\ldots)\cdot s\big)\big) & & \mathcal{C}^k
\end{tikzcd}$$

Figure E.1: Computing level k functional ciphertext c^k encoding $f^k(\mathbf{x})$ using induction. A term in any node is implied by a quadratic polynomial in its children, quadratic in the terms of the left child, and linear in the terms of the right child. The solid arrows on the left indicate quadratic terms that are computed by the induction hypothesis. The dashed arrows to the right point to terms whose *linear* combination suffices, along with the high degree terms in the left sibling, to compute the parent. The terms in the right child may be further decomposed into quadratic polynomials in its children, quadratic in left child terms and linear in right child terms, until we reach the last level, where the terms in the right child are provided directly by the encryptor as advice encodings $\mathcal{C}^k$. The functional ciphertext at level k, namely the root c^k is thus ultimately linear in $\mathcal{C}^k$, while being high degree in lower level encodings $\mathcal{C}^1,\ldots,\mathcal{C}^{k-1}$.

Moreover, note that the computation of the functional message embedded in a level k ciphertext c^k can be viewed as follows. By equation E.6, we see that the message embedded in c^k equals the *encoding* in the left child plus a linear combination of the *messages* embedded in the right child. At the next level, we see that the message in the right child at level 2 (from the top) again equals the encoding in the left child plus a linear combination of the messages embedded in the right child. At the last level, we get that the message embedded in c^k is a quadratic polynomial in all the left children in the tree, and a linear combination of level k messages $\mathcal{M}^k$. Thus, we have as desired that:

$$f(\mathbf{x}) \approx \mathsf{Poly}_f(\mathcal{C}^1,\ldots,\mathcal{C}^{d-1}) + \langle \mathsf{Lin}_f, \mathcal{M}^d\rangle$$

The Public Key and Ciphertext Evaluation Algorithms. Our evaluation algorithms $\mathsf{Eval}_{\mathsf{PK}}$ and $\mathsf{Eval}_{\mathsf{CT}}$ are defined recursively, so that to compute the functional public key and functional ciphertext at level k, the algorithms require the same for level $k-1$. Please see Figures E.2 and E.3 for the formal descriptions.

Algorithm $\mathsf{Eval}^k_{\mathsf{PK}}(\underset{i\in[k]}{\cup}\mathsf{PK}(\mathcal{C}^i),\ell)$

To compute the label for the ℓ^{th} wire in the level k circuit, do:

1. If the ℓ^{th} wire at level k is the addition of the i^{th} and j^{th} wire at level $k-1$, then do the following:
   - If $k=3$ (base case), then compute $\mathsf{PK}(c^3_\ell)=\mathsf{PK}(c^2_i)+\mathsf{PK}(c^2_j)$ as in Equation E.10.
   - Let $\mathsf{PK}^{k-1}_i=\mathsf{Eval}^{k-1}_{\mathsf{PK}}(\underset{j\in[k-1]}{\cup}\mathsf{PK}(\mathcal{C}^j),i)$ and $\mathsf{PK}^{k-1}_j=\mathsf{Eval}^{k-1}_{\mathsf{PK}}(\underset{i\in[k-1]}{\cup}\mathsf{PK}(\mathcal{C}^i),j)$,
   - Let $\mathsf{PK}^k_\ell=\mathsf{PK}^{k-1}_i+\mathsf{PK}^{k-1}_j$
2. If the ℓ^{th} wire at level k is the multiplication of the i^{th} and j^{th} wire at level $k-1$, then do the following:
   - If $k=4$ (base case), then compute PK^k_ℓ as described in Equation E.17.
   - Let $u^{k-1}_i=\mathsf{Eval}^{k-1}_{\mathsf{PK}}(\underset{j\in[k-1]}{\cup}\mathsf{PK}(\mathcal{C}^j),i)$ and $u^{k-1}_j=\mathsf{Eval}^{k-1}_{\mathsf{PK}}(\underset{i\in[k-1]}{\cup}\mathsf{PK}(\mathcal{C}^i),j)$
   - Let $\mathsf{PK}(c^k_\ell)=u^{k-1}_i u^{k-1}_j\,\mathsf{PK}(\mathcal{E}^k(s^2))-u^{k-1}_j\,\mathsf{PK}\big(\mathcal{E}^k(c^{k-1}_i s)\big)-u^{k-1}_i\,\mathsf{PK}\big(\mathcal{E}^k(c^{k-1}_j s)\big)$ as in Equation E.25.
     Here $\mathsf{PK}(\mathcal{E}^k(s^2))$, $\mathsf{PK}\big(\mathcal{E}^k(c^{k-1}_i s)\big)$ and $\mathsf{PK}\big(\mathcal{E}^k(c^{k-1}_j s)\big)$ are computed using $\mathcal{C}^k$ as described in Equation E.16, E.23.

Figure E.2: Algorithm to evaluate on public key.

E.2 Error Analysis.

As discussed in Section 6, we have the level 2 encoding $\mathcal{E}^2(x_ix_j+p_0\cdot\mu_{ij})$ is computed as

$$\begin{aligned}\mathcal{E}^2(x_ix_j+p_0\cdot\mu_{ij})&=\mathcal{E}^2(c_ic_j-u_ic_js-u_jc_is+u_iu_js^2)\\&=c_ic_j-u_i\mathcal{E}^2(c_js)-u_j\mathcal{E}^2(c_is)+u_iu_j\mathcal{E}^2(s^2)\end{aligned}$$

Thus, computing a level 2 encoding of x_ix_j from level 1 encodings of x_i and level 2 encodings $\mathcal{E}^2(c_i\cdot s)$ entails:

1. Adding noise to the message x_ix_j itself: The level 2 encoding that gets computed is for the message $x_ix_j+p_0\cdot\mu_{ij}$ not purely x_ix_j. This noise μ_{ij} results from the BV FHE evaluation process.
2. Adding noise to protect the encoding: Above, the encoding noise for the noisy message $x_ix_j+p_0\cdot\mu_{ij}$ is the linear combination

$$\mathsf{Nse}\big(\mathcal{E}^2(x_ix_j+p_0\cdot\mu_{ij})\big)=u_i\mathsf{Nse}\big(\mathcal{E}^2(c_js)\big)-u_j\mathsf{Nse}\big(\mathcal{E}^2(c_is)\big)+u_iu_j\mathsf{Nse}\big(\mathcal{E}^2(s^2)\big)$$

Now, we have an encoding of a noisy functional value at level 2, which has the same structure as level 1 and we may propagate the computation up the circuit.

Thus, at each level, there are two types of noise within an encoding:

Algorithm $\mathsf{Eval}^k_{\mathsf{CT}}(\underset{i\in[k]}{\cup}\mathcal{C}^i, \ell)$

```
To compute the encoding for the $\ell^{th}$ wire in the level $k$ circuit, do:
  1. If the $\ell^{th}$ wire at level $k$ is the addition of the $i^{th}$ and $j^{th}$ wire at level $k-1$, then do the following:
      • If $k = 3$ (base case), then compute $c^3_\ell = c^2_i + c^2_j$ as in Equation E.9.
      • Let $\mathsf{CT}^{k-1}_i = \mathsf{Eval}^{k-1}_{\mathsf{CT}}(\underset{j\in[k-1]}{\cup}\mathcal{C}^j, i)$ and $\mathsf{CT}^{k-1}_j = \mathsf{Eval}^{k-1}_{\mathsf{CT}}(\underset{i\in[k-1]}{\cup}\mathcal{C}^i, j)$,
      • Let $\mathsf{CT}^k_\ell = \mathsf{CT}^{k-1}_i + \mathsf{CT}^{k-1}_j$
  2. If the $\ell^{th}$ wire at level $k$ is the multiplication of the $i^{th}$ and $j^{th}$ wire at level $k-1$, then do the following:
      • If $k = 4$ (base case) then compute $c^4_\ell$ (for any $\ell$) using Equations E.11 and E.12.
      • Let $c^{k-1}_i = \mathsf{Eval}^{k-1}_{\mathsf{CT}}(\underset{j\in[k-1]}{\cup}\mathcal{C}^j, i)$ and $c^{k-1}_j = \mathsf{Eval}^{k-1}_{\mathsf{CT}}(\underset{i\in[k-1]}{\cup}\mathcal{C}^i, j)$,
      • Let $c^k_\ell = c^{k-1}_i c^{k-1}_j + u^{k-1}_i u^{k-1}_j\ \mathcal{E}^k(s^2) - u^{k-1}_j\ \mathcal{E}^k(c^{k-1}_i s) - u^{k-1}_i\ \mathcal{E}^k(c^{k-1}_j s)$ as in Equation E.24.
        Here, the terms $\mathcal{E}^k(s^2)$, $\mathcal{E}^k(c^{k-1}_i s)$ and $\mathcal{E}^k(c^{k-1}_j s)$ are computed using $\mathcal{C}^k$ as described in claim E.4
```

Figure E.3: Algorithm to evaluate on ciphertext.

1. The noise that is used to protect the encoding (this may be viewed as the noise within an RLWE sample). For an encoding at level k, this noise is a multiple of p_{k-1}. We refer to this noise as "encoding noise".

2. The noise that is added to the message as part of the FHE evaluation procedure, which takes place within the message space of the encoding. We refer to this noise as "message noise".

We observe that message noise is a quadratic function of the encoding noise terms of the *previous* level. We may compute the resultant noise term in the challenge ciphertext by proceeding bottom-up, computing the message noise created by each quadratic operation.

At the level 3 addition layer, for any gate the noise terms of the input gates are added. To compute the function f^4_g for a multiplication gate g at the 4th layer, which takes as input wires a and b at layer 3, note that we may compute:

1. Encodings of level 3 messages, i.e. encodings c^3 of messages the form $y = x_k x_{k'} + p_0 \cdot \mu_{kk'} + x_\ell x_{\ell'} + p_0 \cdot \mu_{\ell\ell'}$

2. Encodings of advice terms $\mathcal{E}^4(c^3_a \cdot s + p_1 \cdot \mu^3_a)$ and $\mathcal{E}^4(c^3_b \cdot s + p_1 \cdot \mu^3_b)$ for some noise terms μ^3_a, μ^3_b.

When we compute an encoding of product $y_a \cdot y_b$ where y_a, y_b are constructed as y above, we obtain an encoding of the desired value $(x_i x_j + x_{i'} x_{j'})(x_k x_\ell + x_{k'} x_{\ell'})$ plus noise which is computed as the sum of:

1. Noise created by FHE decryption given encodings of y_a and y_b. This noise has the form

$$y_a\ \mathsf{Nse}(\mathcal{E}^3(y_b)) + y_b\ \mathsf{Nse}(\mathcal{E}^3(y_a)) + p_2 \cdot \mathsf{Nse}(\mathcal{E}^3(y_b)) \cdot \mathsf{Nse}(\mathcal{E}^3(y_a))$$

2. Noise created by applying the quadratic method to compute the advice, as discussed in detail in the base case analysis. For instance, the encoding of advice $\mathcal{E}^4(c_\ell^3 s)$ is only computed as $\mathcal{E}^4(c_\ell^3 s + p_1 \cdot \mu_\ell^2)$ where $\mu_\ell^2 = \mu_{ij}^2 + \mu_{mt}^2$. Please see Equations E.15 and E.16 for more details.

3. $y_a \cdot y_b$ is itself only a noisy version of the desired value, since

$$y_a \cdot y_b = \big(x_k x_{k'} + p_0 \cdot \mu_{kk'} + x_\ell x_{\ell'} + p_0 \cdot \mu_{\ell\ell'}\big)\big(x_m x_{m'} + p_0 \cdot \mu_{mm'} + x_t x_{t'} + p_0 \cdot \mu_{tt'}\big)$$

Constructing Message Noise for $\mathcal{E}^d(f(\mathbf{x}))$. We have by Lemma E.3 that the encodings $\mathcal{E}^k(c^{k-1} \cdot s)$ may be expressed as quadratic polynomials in level $k-1$ encodings and the level k advice encodings $\mathcal{C}^k$. Each quadratic $k-1$ encoding induces a quadratic noise term, which is a product of level $k-2$ encoding noise terms (a multiple of p_{k-2}). Since there are at most $k-1$ quadratic encodings of level $k-1$, the noise terms are added together to give the message noise for $c^{k-1} \cdot s$.

When these are combined to construct the level k encoding c^k, the BV FHE decryption equation for $f^k(\mathbf{x})$ adds additional noise to the message, where the noise is a again a quadratic polynomial in the encoding noise terms in level $k-1$ encodings (analogous to noise μ_{ij} at level 2). Thus, the total message noise at level k is $\mathsf{LinComb}(\mathsf{Nse}(\mathcal{E}^{k-1}(\cdot)\mathcal{E}^{k-1}(\cdot))$, where the linear combination depends on the function f being computed.

Starting at level 2, we proceed up the circuit f to compute noise as a function of noise terms in the encodings. The function applied to the noise terms is related to but not exactly the same as the function f, as discussed above.