Functional Encryption for Bounded Collusions, Revisited

Shweta Agrawal[1(✉)] and Alon Rosen[2]

[1] IIT Madras, Chennai, India
shweta.a@cse.iitm.ac.in
[2] Efi Arazi School of Computer Science, IDC Herzliya, Herzliya, Israel
alon.rosen@idc.ac.il

Abstract. We provide a new construction of functional encryption (FE) for circuits in the bounded collusion model. In this model, security of the scheme is guaranteed as long as the number of colluding adversaries can be a-priori bounded by some polynomial Q. Our construction supports *arithmetic* circuits in contrast to all prior work which support Boolean circuits. The ciphertext of our scheme is sublinear in the circuit size for the circuit class NC_1; this implies the first construction of arithmetic reusable garbled circuits for NC_1.

Additionally, our construction achieves several desirable features:

- Our construction for reusable garbled circuits for NC_1 achieves the optimal "full" simulation based security.
- When generalised to handle Q queries for any fixed polynomial Q, our ciphertext size grows additively with Q^2. In contrast, previous works that achieve full security [5,39] suffered a multiplicative growth of Q^4.
- The ciphertext of our scheme can be divided into a succinct data dependent component and a non-succinct data independent component. This makes it well suited for optimization in an *online-offline model* that allows a majority of the computation to be performed in an offline phase, before the data becomes available.

Security of our reusable garbled circuits construction for NC_1 is based on the Ring Learning With Errors assumption (RLWE), while the bounded collusion construction (with non-succinct ciphertext) may also be based on the standard Learning with Errors (LWE) assumption. To achieve our result, we provide new public key and ciphertext evaluation algorithms. These algorithms are general, and may find application elsewhere.

1 Introduction

Functional encryption (FE) [52,53] generalizes public key encryption to allow fine grained access control on encrypted data. In functional encryption, a secret key SK_f corresponds to a function f, and a ciphertext $\mathsf{CT}_{\mathbf{x}}$ corresponds to some input $\mathbf{x}$ from the domain of f. Given SK_f and $\mathsf{CT}_{\mathbf{x}}$, functionality posits that the

Y. Kalai and L. Reyzin (Eds.): TCC 2017, Part I, LNCS 10677, pp. 173–205, 2017.
https://doi.org/10.1007/978-3-319-70500-2_7

user may run the decryption procedure to learn the value $f(\mathbf{x})$, while security guarantees that nothing about $\mathbf{x}$ beyond $f(\mathbf{x})$ can be learned.

Recent years have witnessed significant progress towards constructing functional encryption for advanced functionalities [3,4,11,13,15,16,21,25,31,32,35,40–42,45,46,54]. However, for the most general notion of functional encryption – one that allows the evaluation of arbitrary efficient functions and is secure against general adversaries, the only known constructions rely on indistinguishability obfuscation (iO) [31] or on the existence of multilinear maps [33]. For full-fledged functional encryption, reliance on such strong primitives is not a co-incidence, since functional encryption has been shown to imply indistinguishability obfuscation [7,8,12].

Unfortunately, all known candidate multi-linear map constructions [27,30,34] as well as some candidates of indistinguishability obfuscation have recently been broken [22–24,26,43,49]. To support general functionalities and base hardness on standard assumptions, a prudent approach is to consider principled relaxations of the security definition, as studied in [37,39,41].

The notion of *bounded collusion functional encryption*, inspired from the domain of secure multiparty computation (MPC), was introduced by Gorbunov, Vaikuntanathan and Wee [39]. This notion assumes that the number of colluding adversaries against a scheme can be upper bounded by some polynomial Q, which is known at the time of system design. It is important to note that Q-bounded security does not impose any restriction on the functionality of FE – in particular, it does not disallow the system from issuing an arbitrary number of keys. It only posits, à la MPC, that security is guaranteed as long as any collusion of attackers obtains at most Q keys. Note that multiple independent collusions of size at most Q are supported.

The notion of Q-bounded FE is appealing – proving security under the assumption that not too many parties are dishonest is widely accepted as reasonable in protocol design. Even in the context of FE, for the special case of Identity Based Encryption (IBE), bounded collusion security has been considered in a number of works [28,29,38].

Structure versus Generality. Gorbunov et al. [39] showed that Q-bounded FE can be constructed generically from *any* public key encryption (PKE) scheme by leveraging ideas from multiparty computation. Considering that most constructions of FE for general functionalities rely on the existence of sophisticated objects such as multilinear maps or indistinguishability obfuscation, basing a meaningful relaxation of FE on an assumption as generic and mild as PKE is both surprising, and aesthetically appealing. However, this generality comes at the cost of efficiency and useful structural properties. The ciphertext of the scheme is large and grows multiplicatively as $O(Q^4)$ to support collusions of size Q. Additionally, the entire ciphertext is data dependent, making the scheme unsuitable for several natural applications of FE, as discussed below.

1.1 Our Results

In this work, we provide a new construction of bounded key functional encryption. Our construction makes use of the recently developed Functional Encryption for Linear Functions [1,5], denoted by LinFE, and combines this with techniques developed in the context of Fully Homomorphic Encryption (FHE)[1] [18,19]. Since LinFE and FHE can be based on LWE/Ring LWE, our construction inherits the same hardness assumption. Our construction offers several advantages:

1. Our construction supports *arithmetic* circuits as against Boolean circuits.
2. The ciphertext of our scheme is succinct for circuits in NC_1 under Ring LWE and any constant depth under standard LWE. This gives the first construction of arithmetic reusable garbled circuits. We note that even single use arithmetic garbled circuits have only been constructed recently [10].
3. Our construction achieves the optimal "full" simulation based security.
4. When generalised to handle Q queries for any fixed polynomial Q, our ciphertext size grows additively with Q^2. In contrast, previous works that achieve full security [5,39] suffered a multiplicative growth of Q^4.
5. The ciphertext of our scheme can be divided into a succinct data dependent component and a non-succinct data independent component. This makes it well suited for optimization in an *online-offline model* that allows a majority of the computation to be performed in an offline phase, before the data becomes available. This is followed by an efficient online phase which is performed after the data is available.

1.2 Related Work

The first functional encryption scheme for circuits was provided by Gorbunov, Vaikuntanathan and Wee [39]. Surprisingly, the security of this construction may be based only on the existence of public key encryption. However, the ciphertext size of this construction is large and does not enjoy the online-offline property described above. The online component of [39] depends on the circuit size and the number of queries in addition to the message size, whereas that of our scheme depends only on the message size. Additionally, the overall ciphertext size of [39] grows multiplicatively with Q^4, whereas that in our scheme grows *additively* with Q^2. More recently, Agrawal et al. [5] provided a construction for bounded collusion FE. However, their ciphertext size grows as $O(Q^6)$ and does not support online-offline computation.

Concurrent and Subsequent Work. Subsequent to our work, Agrawal [2] also constructed Q collusion Functional Encryption where the ciphertext size grows additively with $O(Q^2)$. However, this construction only achieves semi-adaptive rather than full security in a weak security game where the attacker must announce all

[1] We emphasise that we do not rely on FHE in a black box way, but rather adapt techniques developed in this domain to our setting.

Q queries "in one shot". Additionally, it supports Boolean rather than arithmetic circuits and makes black box use of "heavy machinery" such fully homomorphic encryption and attribute based encryption.

In another recent work, Canetti and Chen [20] provide a new construction for single key FE for NC_1 achieving full security. However, their construction supports Boolean rather than arithmetic circuits, which is the main focus of this work. Moreover, to generalise this construction to support Q queries, one must rely on the [39] compiler, which incurs a multiplicative blowup of $O(Q^4)$ in the ciphertext size. For more details about related work, please see Appendix A.

1.3 Techniques

In this section, we describe our techniques. We begin by outlining the approach taken by previous work. [39] begin with a single key FE scheme for circuits [51] and generalize this to a Q query scheme for NC_1 circuits. This is the most sophisticated part of the construction, and leverages techniques from multiparty computation. Then, the Q query FE for NC_1 is bootstrapped to Q query FE for all circuits by replacing the circuit in the key by a tuple of low degree polynomials admitted by computational randomized encodings [9].

Recently, Agrawal et al. [5] observe that a different construction for bounded collusion FE can be obtained by replacing the single key FE [51] and its generalisation to Q query FE for NC_1, with an FE that computes inner products modulo some prime p. Such a scheme, which we denote by LinFE, was constructed by [1,5] and computes the following functionality: the encryptor provides a ciphertext $\mathsf{CT}_{\mathbf{x}}$ for some vector $\mathbf{x} \in F_p^\ell$, the key generator provides a key $\mathsf{SK}_{\mathbf{v}}$ for some vector $\mathbf{v} \in F_p^\ell$, and the decryptor, given $\mathsf{CT}_{\mathbf{x}}$ and $\mathsf{SK}_{\mathbf{v}}$ can compute $\langle \mathbf{x}, \mathbf{v} \rangle \mod p$[2]. Since the bootstrapping theorem in [39] only requires FE for degree 3 polynomials, and FE for linear functions trivially implies FE for bounded degree polynomials simply by linearizing the message terms $\mathbf{x}$ and encrypting each monomial $x_i x_j x_k$ separately, LinFE may be used to compute degree 3 polynomials.

Thus, in [5], the challenge of supporting multiplication is "brute-forced" by merely having the encryptor encrypt each monomial separately so that the FE must only support linear functions in order to achieve bounded degree polynomials. This brute force approach has several disadvantages: the ciphertext is not online-offline and its size grows as $O(Q^6)$. See Appendix A for more details.

Our Approach. In this work, we observe that viewing functional encryption through the lens of fully homomorphic encryption (FHE) enables a more sophisticated application of the Linear FE scheme LinFE, resulting in a bounded collusion FE scheme for circuits that is decomposable, online-succinct as well as achieves ciphertext dependence of $O(Q^2)$ additively on Q.

[2] We note that the FE scheme by Abdalla et al. [1] also supports linear functions but only over $\mathbb{Z}$, while the bounded collusion FE of [5] requires an FE scheme that supports $\mathbb{Z}_p$. Also note the difference from Inner Product *orthogonality testing* schemes [4,45] which test whether $\langle \mathbf{x}, \mathbf{v} \rangle = 0 \mod p$ or not.

We begin on FE for quadratic polynomials for ease of exposition. Additionally, here and in the rest of the paper, we present our construction from Ring-LWE rather than standard LWE, for notational convenience and clarity. Our construction can be ported to the standard LWE setting, by performing standard transformations such as replacing ring products by vector tensor products. Details are provided in the full version [6].

Consider the ring LWE based symmetric key FHE scheme by Brakerski and Vaikuntanathan [19]. Recall that the ciphertext of this scheme, as in [50], is structured as (u, c) where $c = u \cdot s + 2 \cdot \mu + x$. Here, s is the symmetric key chosen randomly over an appropriate ring R, u is an element chosen by the encryptor randomly over R, x is a message bit and μ is an error term chosen by the encryptor from an appropriate distribution over R. Given secret key s, the decryptor may compute $c - u \cdot s \mod 2$ to recover the bit x.

The main observation in [19] was that if:

$$c_i = u_i \cdot s + 2 \cdot \mu_i + x_i$$
$$c_j = u_j \cdot s + 2 \cdot \mu_j + x_j$$

then the decryption equation can be written as

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

Thus, the 3 tuple $(c_i c_j,\ u_i c_j + u_j c_i,\ u_i u_j)$ is a legitimate level 2 FHE ciphertext, decryptable by the secret key s. [19] observed that it is sufficient to add one ciphertext element per level of the circuit to propagate the computation.

In the context of FE, things are significantly more complex even for quadratic polynomials, since we must return a key that allows the decryptor to learn $x_i x_j$ and nothing else. Hence, providing s to the decryptor is disastrous for FE security. Here we use our first trick: observe that in the above equation, the parenthesis can be shifted so that:

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

Now, if we use the Linear FE scheme to encrypt the terms in parenthesis, then we can have the decryptor recover the term $u_i u_j(s^2) - u_j(c_i s) - u_i(c_j s)$. More formally, let $|\mathbf{x}| = w$. Now if,

$$\mathsf{CT} = \mathsf{LinFE.Enc}(s^2, c_1 s, \ldots, c_w s)$$
$$\mathsf{SK}_{ij} = \mathsf{LinFE.KeyGen}(u_i u_j, -0-, u_i, -0-, u_j, -0-)$$

then, $\mathsf{LinFE.Dec}(\mathsf{SK}_{ij}, \mathsf{CT})$ should yield the above term by correctness. Since $c_1, \ldots, c_w$ may be provided directly in the ciphertext, the decryptor may itself compute the term $c_i c_j$. Now, LinFE decryption yields $u_i u_j(s^2) - u_j(c_i s) - u_i(c_j s)$, so the decryptor may recover (approximately) $x_i x_j$ as desired[3].

A bit more abstractly, we observe that a quadratic plaintext $x_i x_j$ can be represented as a quadratic polynomial which is quadratic in *public* terms c_i, c_j,

[3] As in FHE, approximate recovery is enough since the noise can be modded out.

and only *linear* in secret terms $c_i s$. In particular, since the number of secret terms $c_i s$ which must be encrypted is only linear in $|\mathbf{x}|$, we appear to avoid the quadratic blowup caused by linearization.

This intuition, while appealing, is very misleading. To begin, note that if we permit the decryptor to learn the term $u_i u_j s^2 - u_j c_i s - u_i c_j s$ exactly, then he can recover exact quadratic equations in the secret s, completely breaking the security of the scheme. To handle this, we resort to our second trick: add noise *artificially* to the decryption equation. This takes care of the above attack, but to handle Q queries, we need Q fresh noise terms to be encrypted in the ciphertext. This step introduces the dependence of the ciphertext size on Q. Providing a proof of security requires crossing several additional hurdles. The details of the proof are provided in Sect. 3.

New Public Key and Ciphertext Evaluation Algorithms. To generalize our construction to NC_1, we develop new algorithms to compute on the public key and ciphertext. Designing these algorithms is the most challenging part of our work. Intuitively, the ciphertext evaluation algorithm enables the decryptor to compute a "functional" ciphertext $\mathsf{CT}_{f(\mathbf{x})}$ encoding $f(\mathbf{x})$ on the fly, using the function description f, and encodings of $\mathbf{x}$ provided by the encryptor obliviously of f. The public key evaluation algorithm enables the key generator to compute the "functional" public key PK_f given the public key PK and the function f, obliviously of $\mathbf{x}$ so that the functional public key PK_f *matches* the functional ciphertext $\mathsf{CT}_{f(\mathbf{x})}$ enabling the key generator to provide a functional secret key which allows decryption of $\mathsf{CT}_{f(\mathbf{x})}$.

We note that a previous work by Boneh et al. [14] also provided a ciphertext evaluation algorithm which enables computing $\mathsf{CT}_{f(\mathbf{x})}$ given $\mathsf{CT}_{\mathbf{x}}$ and f, but this algorithm crucially requires the evaluator to have some knowledge of $\mathbf{x}$ in order to support multiplications. In more detail, the evaluator must know at least one of encoded values x_1, x_2 in the clear in order to compute an encoding of $x_1 \cdot x_2$. In contrast, our ciphertext evaluation algorithm is completely oblivious of $\mathbf{x}$ even for multiplication gates.

We give a brief description of our approach below. Recall that the "level 1" encodings $\mathbf{c}$ of message $\mathbf{x}$ along with "level 2" encodings of message $\mathbf{c} \cdot s$ in the LinFE ciphertext were sufficient to compute encodings of degree two polynomials in $\mathbf{x}$. Generalizing, we get that at any level k in the circuit, given an encoding $\mathbf{c}^{k-1}$ of message $f^{k-1}(\mathbf{x})$ where f^{k-1} is the output of the circuit at level $k-1$, as well as encodings of $\mathbf{c}^{k-1} \cdot s$, we would be in a position to compute encodings $\mathbf{c}^k$ of level k output of the circuit using the method to evaluate quadratic polynomials described above.

This intuition is complicated by the fact that the encryptor may not provide $\mathbf{c}^{k-1}$ directly as this depends on f which it does not know. Thus, the encryptor must provide *advice* which enables the decryptor to compute $\mathbf{c}^{k-1}$ on the fly. Moreover, this advice must be sublinear in the size of the circuit. We design advice encodings that enable a decryptor to compute functional ciphertexts dynamically via *nested FHE decryptions*. Please see Sect. 4 for more details.

Organization of the paper. We provide preliminaries in Sect. 2. Our bounded collusion functional encryption scheme for quadratic polynomials is described in Sect. 3. To generalize our method beyond quadratic polynomials, we describe our public key and ciphertext evaluation procedures in Sect. 4. The succinct single key FE using these procedures is constructed in Sect. 5. The bounded collusion scheme is provided in Sect. 6, and parameters in Appendix B.

2 Preliminaries

In this section, we define the preliminaries we require for our constructions.

2.1 Functional Encryption

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

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

- $\mathsf{FE.Setup}(1^\lambda)$ is a p.p.t. algorithm takes as input the unary representation of the security parameter and outputs the master public and secret keys $(\mathsf{PK}, \mathsf{MSK})$.
- $\mathsf{FE.Keygen}(\mathsf{MSK}, C)$ is a p.p.t. algorithm that takes as input the master secret key MSK and a circuit $C \in \mathcal{C}_\lambda$ and outputs a corresponding secret key SK_C.
- $\mathsf{FE.Encrypt}(\mathsf{PK}, \mathbf{x})$ is a p.p.t. algorithm that takes as input the master public key PK and an input message $\mathbf{x} \in \mathcal{X}_\lambda$ and outputs a ciphertext $\mathsf{CT}_\mathbf{x}$.
- $\mathsf{FE.Decrypt}(\mathsf{SK}_C, \mathsf{CT}_\mathbf{x})$ is a deterministic algorithm that takes as input the secret key SK_C and a ciphertext $\mathsf{CT}_\mathbf{x}$ and outputs $C(\mathbf{x})$.

Definition 2.1 (Correctness). *A functional encryption scheme $\mathcal{F}$ is correct if for all $C \in \mathcal{C}_\lambda$ and all $x \in \mathcal{X}_\lambda$,*

$$\Pr\left[\begin{array}{l}(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\lambda);\\ \mathsf{FE.Decrypt}\Big(\mathsf{FE.Keygen}(\mathsf{MSK}, C), \mathsf{FE.Encrypt}(\mathsf{PK}, \mathbf{x})\Big) \neq C(\mathbf{x})\end{array}\right] = \mathrm{negl}(\lambda)$$

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

2.2 Simulation Based Security for Single Key FE

In this section, we define simulation based security for single key FE, as in [37, Definition 4.1].

Definition 2.2 (FULL-SIM Security). *Let $\mathcal{F}$ be a functional encryption scheme for a circuit family $\mathcal{C}$. For every stateful p.p.t. adversary* Adv *and a stateful p.p.t. simulator* Sim*, consider the following two experiments:*

$\mathsf{Exp}^{\mathsf{real}}_{\mathcal{F},\mathsf{Adv}}(1^\lambda)$:	$\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{F},\mathrm{Sim}}(1^\lambda)$:
1: $(\mathsf{PK},\mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$	1: $(\mathsf{PK},\mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$
2: $C \leftarrow \mathsf{Adv}(1^\lambda, \mathsf{PK})$	2: $C \leftarrow \mathsf{Adv}(1^\lambda, \mathsf{PK})$
3: $\mathsf{SK}_C \leftarrow \mathsf{FE.Keygen}(\mathsf{MSK}, C)$	3: $\mathsf{SK}_C \leftarrow \mathsf{FE.Keygen}(\mathsf{MSK}, C)$
4: $\mathbf{x} \leftarrow \mathsf{Adv}(\mathsf{SK}_C)$	4: $\mathbf{x} \leftarrow \mathsf{Adv}(\mathsf{SK}_C)$
5: $\mathsf{CT}_\mathbf{x} \leftarrow \mathsf{FE.Encrypt}(\mathsf{PK}, \mathbf{x})$	5: $\mathsf{CT}_\mathbf{x} \leftarrow \mathrm{Sim}(1^\lambda, 1^{\lvert\mathbf{x}\rvert}, \mathsf{PK}, C, \mathsf{SK}_C, C(\mathbf{x}))$
6: $\alpha \leftarrow \mathsf{Adv}(\mathsf{CT}_\mathbf{x})$	6: $\alpha \leftarrow \mathsf{Adv}(\mathsf{CT}_\mathbf{x})$
7: *Output* $(\mathbf{x}, \alpha)$	7: *Output* $(\mathbf{x}, \alpha)$

The functional encryption scheme $\mathcal{F}$ is then said to be FULL-SIM*-secure if there is an admissible stateful p.p.t. simulator* Sim *such that for every stateful p.p.t. adversary* Adv*, the following two distributions are computationally indistinguishable.*

$$\left\{\mathsf{Exp}^{\mathsf{real}}_{\mathcal{F},\mathsf{Adv}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{F},\mathrm{Sim}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}}$$

In the bounded collusion variant of the above definition, the adversary is permitted an a-priori fixed Q queries in Step 2, and Q is input to the FE.Setup algorithm.

2.3 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. Let L be a discrete subset of $\mathbb{Z}^n$. 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 $\rho_{\sigma,\mathbf{c}}(L) := \sum_{\mathbf{x}\in L}\rho_{\sigma,\mathbf{c}}(\mathbf{x})$ be the discrete integral of $\rho_{\sigma,\mathbf{c}}$ over L, and let $\mathcal{D}_{L,\sigma,\mathbf{c}}$ be the discrete Gaussian distribution over L with center $\mathbf{c}$ and parameter σ. Specifically, for all $\mathbf{y} \in L$, we have $\mathcal{D}_{L,\sigma,\mathbf{c}}(\mathbf{y}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{y})}{\rho_{\sigma,\mathbf{c}}(L)}$. For notational convenience, $\rho_{\sigma,\mathbf{0}}$ and $\mathcal{D}_{L,\sigma,\mathbf{0}}$ are abbreviated as ρ_σ and $\mathcal{D}_{L,\sigma}$, respectively.

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

Lemma 2.3 ([48, Lemma 4.4]). *Let Λ be an n-dimensional lattice, let $\mathbf{T}$ be a basis for Λ, and suppose $\sigma \geq \|\mathbf{T}\|_{\mathsf{GS}} \cdot \omega(\sqrt{\log n})$. Then for any $\mathbf{c} \in \mathbb{R}^n$ we have*

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

Lemma 2.4 (Flooding Lemma). [36] *Let $n \in \mathbb{N}$. For any real $\sigma = \omega(\sqrt{\log n})$, and any $\mathbf{c} \in \mathbb{Z}^n$,*

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

2.4 Hardness Assumptions

Our main construction of arithmetic reusable garbled circuits for NC_1 is based on the hardness of Ring Learning with Errors, defined below. Our bounded collusion construction for circuits may also be based on the standard Learning with Errors problem, but we defer this discussion to the full version [6].

Ring Learning with Errors. 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 $q = 1 \mod 2n$. Let χ be a probability distribution on R_q. 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 $(a, a \cdot s + e)$.

Definition 2.5 (Ring Learning With Errors- $\mathsf{RLWE}_{\phi,q,\chi}$). [47,50] *The decision R-$\mathsf{LWE}_{\phi,q,\chi}$ problem is: for $s \leftarrow R_q$, given a $\mathrm{poly}(n)$ number of samples that are either (all) from $A_{s,\chi}$ or (all) uniformly random in $R_q \times R_q$, output 0 if the former holds and 1 if the latter holds.*

The hardness of the ring LWE problem was studied in [47] and is summarised in the following theorem.

Theorem 2.6 ([47]). *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 $\mathsf{RLWE}_{\phi,q,\chi}$ where χ is the discrete Gaussian distribution with parameter r. The reduction runs in time $\mathrm{poly}(n, q)$.*

3 Warm-Up: Bounded Query Functional Encryption for Quadratic Polynomials

As a warm-up, we present our bounded key FE for the special case of quadratic functions, which we denote by QuadFE. Our construction will make use of the linear functional encryption scheme, denoted by LinFE, constructed by [1,5].

Our construction makes use of two prime moduli $p_0 < p_1$ where p_0 serves as the message space for QuadFE, and p_1 serves as the message space for LinFE. Let $L = |1 \leq j \leq i \leq w|$. Below, let distributions $\mathcal{D}_0, \mathcal{D}_1$ be discrete Gaussians with width σ_0, σ_1 respectively. Please see Appendix B for parameters.

For ease of exposition, our key generation algorithm receives the index of the requested key as input. This restriction can be removed using standard tricks, see the full version [6] for details. Additionally, we present our construction using Ring-LWE. This is both for efficiency and ease of exposition. The transformation to standard LWE follows standard machinery, please see the full version [6] for details.

$\mathsf{FE.Setup}(1^\lambda, 1^w, 1^Q)$: On input a security parameter λ, a parameter w denoting the length of message vectors and a parameter Q denoting the number of keys supported, do:

1. Invoke $\mathsf{LinFE.Setup}(1^\lambda, 1^{w+1+Q})$ to obtain $\mathsf{LinFE.PK}$ and $\mathsf{LinFE.MSK}$.
2. Sample $\mathbf{u} \leftarrow R_{p_1}^w$.
3. Output $\mathsf{PK} = (\mathsf{LinFE.PK}, \mathbf{u})$, $\mathsf{MSK} = (\mathsf{LinFE.MSK})$.

$\mathsf{FE.Enc}(\mathsf{PK}, \mathbf{x})$: 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 an encoding of the message as:

$$\mathbf{c} = \mathbf{u} \cdot s_1 + p_0 \cdot \boldsymbol{\mu} + \mathbf{x} \in R_{p_1}^w.$$

2. For $i \in [Q]$, sample $\eta_i \leftarrow \mathcal{D}_1$ and let $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_Q)$.
3. Let $\mathbf{b} = \mathsf{LinFE.Enc}\ (s_1^2, c_1 s_1, \ldots, c_w s_1, p_0 \cdot \boldsymbol{\eta})$.
4. Output $\mathsf{CT} = (\mathbf{c}, \mathbf{b})$.

$\mathsf{FE.KeyGen}(\mathsf{PK}, \mathsf{MSK}, k, \mathbf{g})$: On input the public parameters PK, the master secret key MSK, a counter $k \in [Q]$ denoting the index of the requested function key and a function $\mathbf{g} = \sum_{1 \le j \le i \le w} g_{ij} x_i x_j$, represented as a coefficient vector $(g_{ij}) \in \mathbb{Z}_{p_0}^L$ do:

1. Let $\mathbf{e}_k$ denote the binary unit vector with a 1 in the k^{th} position and 0 elsewhere. Compute

$$\mathbf{u_g} = \Big(\sum_{1 \le j \le i \le w} g_{ij}\, (u_i u_j, 0....0, -u_i, 0...0, -u_j, 0...0) \Big) \in R_{p_1}^{w+1}.$$

2. Compute $\mathsf{SK_g} = \mathsf{LinFE.KeyGen}\big(\mathsf{LinFE.PK}, \mathsf{LinFE.MSK}, (\mathbf{u_g} \| \mathbf{e}_k)\big)$ and output it.

$\mathsf{FE.Dec}(\mathsf{PK}, \mathsf{SK_g}, \mathsf{CT_x})$: On input the public parameters PK, a secret key $\mathsf{SK_g}$ for polynomial $\sum_{1 \le j \le i \le w} g_{ij} x_i x_j$, and a ciphertext $\mathsf{CT_x} = (\mathbf{c}, \mathbf{b})$, compute

$$\sum_{1 \le j \le i \le w} g_{ij} c_i c_j + \mathsf{LinFE.Dec}(\mathbf{b}, \mathsf{SK_g}) \bmod p_1 \bmod p_0$$

and output it.

3.1 Correctness

We establish correctness of the above scheme. Let $1 \le j \le i \le w$. Let us assume $\mathbf{g}$ is the k^{th} key constructed by KeyGen, where $k \in [Q]$. By definition

$$x_i + p_0 \cdot \mu_i = c_i - u_i s_1 \mod p_1, \quad x_j + p_0 \cdot \mu_j = c_j - u_j s_1 \mod p_1$$

Letting $\mu_{ij} = x_i \mu_j + x_j \mu_i + p_0 \mu_i \mu_j$, we have

$$x_i x_j + p_0 \cdot \mu_{ij} = c_i c_j - c_i u_j s_1 - c_j u_i s_1 + u_i u_j s_1^2 \mod p_1 \tag{3.1}$$

By correctness of the linear scheme LinFE, we have that

$$\mathsf{LinFE.Dec}(\mathbf{b}, \mathsf{SK_g}) = \sum_{1 \le j \le i \le w} g_{ij} \big(- c_i u_j s_1 - c_j u_i s_1 + u_i u_j s_1^2 \big) + p_0 \cdot \eta_k$$

$$\begin{aligned}
&\text{Therefore we have, } \sum_{1 \le j \le i \le w} g_{ij} c_i c_j + \mathsf{LinFE.Dec}(\mathbf{b}, \mathsf{SK_g}) \\
&= \sum_{1 \le j \le i \le w} g_{ij} \Big(c_i c_j - c_i u_j s_1 - c_j u_i s_1 + u_i u_j s_1^2 \Big) + p_0 \cdot \eta_k \\
&= \sum_{1 \le j \le i \le w} g_{ij} \big(x_i x_j + p_0 \cdot \mu_{ij} \big) + p_0 \cdot \eta_k \\
&= \sum_{1 \le j \le i \le w} g_{ij}\ x_i x_j \bmod p_1 \bmod p_0 \text{ as desired.}
\end{aligned} \tag{3.2}$$

3.2 Security

Theorem 3.7. *The construction in Sect. 3 achieves full simulation based security as per Definition 2.2.*

Proof. We describe our simulator.

Simulator $\mathrm{Sim}\big(1^\lambda, 1^{|\mathbf{x}|}, \mathsf{PK}, \{\mathbf{g}_k, \mathsf{SK}_{\mathbf{g}_k}, \mathbf{g}_k(\mathbf{x})\}_{k\in[Q]}\big)$. The simulator given input the security parameter, length of message $\mathbf{x}$, the functions $\mathbf{g}_1, \ldots, \mathbf{g}_Q$, the secret keys $\mathsf{SK}_{\mathbf{g}_1}, \ldots, \mathsf{SK}_{\mathbf{g}_Q}$ and the values $\mathbf{g}_1(\mathbf{x}), \ldots, \mathbf{g}_Q(\mathbf{x})$ does the following:

1. It picks the ciphertext $\mathbf{c} \leftarrow R_{p_1}^w$ randomly.
2. It parses $\mathbf{g}_k = \sum_{1\le j\le i\le w} g_{k,ij}\, x_i x_j$ for some $g_{k,ij} \in R_{p_0}$. For $k \in [Q]$, it samples $\eta_k \leftarrow \mathcal{D}_1$ and computes $d_k = \sum_{1\le j\le i\le w} g_{k,ij}\,\big(x_i x_j - c_i c_j\big) + p_0 \cdot \eta_k$.
3. It invokes the Q key LinFE simulator with input $\mathbf{d} = (d_1, \ldots, d_Q)$. It sets as $\mathbf{b}$ the output received by the LinFE simulator.
4. It outputs $\mathsf{CT}_{\mathbf{x}} = (\mathbf{c}, \mathbf{b})$.

We will prove that the output of the simulator is indistinguishable from the real world via a sequence of hybrids.

The Hybrids. Our Hybrids are described below.

Hybrid 0. This is the real world.

Hybrid 1. In this hybrid, the only thing that is different is that $\mathbf{b}$ is computed using the LinFE simulator as $\mathbf{b} = \mathsf{LinFE}.\mathrm{Sim}\big(1^\lambda, 1^{w+1+Q}, \{\mathbf{g}_k, \mathsf{SK}_{\mathbf{g}_k}, d_k\}_{k\in[Q]}\big)$ where

$$d_k = \sum_{1\le j\le i\le w} g_{k,ij}\big(x_i x_j - c_i c_j\big) + p_0 \cdot \Big(\sum_{1\le j\le i\le w} g_{k,ij}\mu_{ij} + \eta_k\Big) \;\; \forall \;\; k \in [Q]$$

Above, μ_{ij} is as defined in Eq. 3.1.

Hybrid 2. In this hybrid, let $d_k = \sum_{1\le j\le i\le w} g_{k,ij}\big(x_i x_j - c_i c_j\big) + p_0 \cdot \eta_k$ for $k \in [Q]$.

Hybrid 3. In this hybrid, sample $\mathbf{c}$ at random. This is the simulated world.

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

Claim. Hybrid 0 is indistinguishable from Hybrid 1 assuming that LinFE is secure.

Proof. Recall that for $j \leq i \leq w$, we have:

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

$$\therefore \sum_{j \leq i \leq w} g_{k,ij}\big(x_i x_j + p_0 \cdot \mu_{ij}\big) = \sum_{j \leq i \leq w} g_{k,ij}\Big(c_i c_j + u_i u_j s_1^2 - u_j c_i s_1 - u_i c_j s_1\Big)$$

$$\text{This implies, } \sum_{j \leq i \leq w} g_{k,ij}\big(x_i x_j - c_i c_j\big) + p_0 \cdot \Big(\sum_{j \leq i \leq w} g_{k,ij}\mu_{ij} + \eta_k\Big)$$
$$= \sum_{j \leq i \leq w} g_{k,ij}\big(u_i u_j s_1^2 - u_j c_i s_1 - u_i c_j s_1\big) + p_0 \cdot \eta_k$$

In Hybrid 0, we have by Eq. 3.2 that the output of LinFE decryption is:

$$\sum_{1 \leq j \leq i \leq w} g_{ij}\big(-c_i u_j s_1 - c_j u_i s_1 + u_i u_j s_1^2\big) + p_0 \cdot \eta_k$$
$$= \sum_{j \leq i \leq w} g_{k,ij}\big(x_i x_j - c_i c_j\big) + p_0 \cdot \Big(\sum_{j \leq i \leq w} g_{k,ij}\mu_{ij} + \eta_k\Big)$$

In Hybrid 1, the LinFE simulator is invoked with the above value, hence by security of LinFE, Hybrids 0 and 1 are indistinguishable.

Claim. Hybrid 1 and Hybrid 2 are statistically indistinguishable.

Proof. This follows by our choice of parameters since for $k \in [Q]$, we have

$$\mathsf{SD}\Big(\sum_{1 \leq j \leq i \leq w} g_{k,ij}\mu_{ij} + \eta_k, \eta_k\Big) = \mathrm{negl}(\lambda)$$

Hybrid 2 and Hybrid 3 are indistinguishable assuming the hardness of ring LWE. In more detail, we show:

Claim. 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 $\mathbf{c} = \mathbf{u} \cdot s_1 + p_0 \cdot \boldsymbol{\mu} + \mathbf{x}$ is indistinguishable from random, 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 [35] 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 $\mathsf{PK} = \mathbf{u}$ and does the following:

- Run $\mathsf{LinFE.Setup}$ to obtain $\mathsf{LinFE.PK}$ and $\mathsf{LinFE.MSK}$. Return $\mathsf{PK} = (\mathsf{LinFE.PK}, \mathbf{u})$ to $\mathcal{B}$.
- When $\mathcal{B}$ requests a key $\mathbf{g}_k$ for $k \in [Q]$, construct it honestly as in Hybrid 0.
- When $\mathcal{B}$ outputs challenge $\mathbf{x}$, $\mathcal{A}$ outputs the same.

- $\mathcal{A}$ receives $\mathbf{c}$ where $\mathbf{c} = \mathbf{u} \cdot s_1 + p_0 \cdot \boldsymbol{\mu} + \mathbf{x}$ or random.
- $\mathcal{A}$ samples $\eta_1, \ldots, \eta_Q$ as in Hybrid 2 and computes $d_k = \sum_{1 \le j \le i \le w} g_{k,ij}\big(x_i x_j - c_i c_j\big) + p_0 \cdot \eta_k$. It invokes $\mathsf{LinFE.Sim}\big(1^\lambda, 1^{w+1+Q}, \{\mathbf{g}_k, \mathsf{SK}_{\mathbf{g}_k}, d_k\}_{k \in [Q]}\big)$ and receives LinFE ciphertext $\mathbf{b}$. It returns $(\mathbf{c}, \mathbf{b})$ to $\mathcal{B}$.
- $\mathcal{B}$ may request more keys (bounded above by Q) which are handled as before. 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. Hence the claim follows.

4 Public Key and Ciphertext Evaluation Algorithms

In this section, we provide the tools to extend our construction for quadratic polynomials to circuits in NC_1. 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 Sect. 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 4.1 (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 4.2. *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* $\mathsf{CT}_{(f(\mathbf{x}))} = \mathsf{Eval}_{\mathsf{CT}}(\bigcup_{i\in[d]} \mathcal{C}^i, f)$*, 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:*

$$\mathsf{CT}_{(f(\mathbf{x}))} = \mathsf{PK}_f \cdot s + p_{d-1}\cdot \eta_f^{d-1} + \mu_{f(\mathbf{x})} + f(\mathbf{x}) \tag{4.1}$$

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* $\mu_{f(\mathbf{x})}$[4]*. 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:*

$$\mathsf{CT}_{f(\mathbf{x})} = \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) + \langle \mathsf{Lin}_f, \mathcal{C}^d\rangle \tag{4.2}$$

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*

$$f(\mathbf{x}) + \mu_{f(\mathbf{x})} = \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) + \langle \mathsf{Lin}_f, \mathcal{M}^d\rangle \tag{4.3}$$

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:*

$$\mathsf{PK}\big(\mathsf{CT}_{f(\mathbf{x})}\big) = \Big\langle \mathsf{Lin}_f,\ \big(\mathsf{PK}(\mathcal{C}_1^d), \ldots, \mathsf{PK}(\mathcal{C}_{L_d}^d)\big)\Big\rangle \tag{4.4}$$

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

We prove that:

Lemma 4.3. *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:

$$\mathcal{E}^k(c^{k-1} \cdot s), \mathcal{E}^k(c^{k-1}) = \mathsf{LinComb}\big(\mathcal{C}^k, \mathcal{E}^{k-1}(y_i)\mathcal{E}^{k-1}(y_j)\big) \;\; \forall\, i, j \tag{4.5}$$

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 Sect. 3 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 [44], 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:

$$\begin{aligned} c^2_{i,j} = \mathcal{E}^2(x_i x_j + p_0 \cdot \mu_{ij}) &= \mathcal{E}^2\big(c^1_i c^1_j + \; u^1_i u^1_j(s^2) - u^1_j(c^1_i s) - u^1_i(c^1_j s)\;\big) \\ &= \mathcal{E}^2(x_i x_j + p_0 \cdot \mu_{ij}) = c^1_i c^1_j + \mathcal{E}^2\big(\; u^1_i u^1_j(s^2) - u^1_j(c^1_i s) - u^1_i(c^1_j s)\;\big) \\ &= c^1_i c^1_j + u^1_i u^1_j\, \mathcal{E}^2(s^2) - u^1_j\, \mathcal{E}^2(c^1_i s) - u^1_i\, \mathcal{E}^2(c^1_j s) \quad \in R_{p_2} \end{aligned} \tag{4.6}$$

In the above, the first equality follows by additive malleability of RLWE: here, $c^1_i c^1_j \in R_{p_1}$ is a message added to the encoding $\mathcal{E}^2(u^1_i u^1_j(s^2) - u^1_j(c^1_i s) - u^1_i(c^1_j s))$.

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:

$$u_\ell^2 \triangleq \mathsf{PK}\left(\mathcal{E}^2(x_i x_j + p_0 \cdot \mu_{ij})\right) = u_i^1 u_j^1 \,\mathsf{PK}\left(\mathcal{E}^2(s^2)\right) - u_j^1 \,\mathsf{PK}\left(\mathcal{E}^2(c_i^1 s)\right) - u_i^1 \,\mathsf{PK}\left(\mathcal{E}^2(c_j^1 s)\right)$$
$$\mathsf{Nse}_\ell^2 \triangleq \mathsf{Nse}\left(\mathcal{E}^2(x_i x_j + p_0 \cdot \mu_{ij})\right) = u_i^1 u_j^1 \,\mathsf{Nse}\left(\mathcal{E}^2(s^2)\right) - u_j^1 \,\mathsf{Nse}\left(\mathcal{E}^2(c_i^1 s)\right) - u_i^1 \,\mathsf{Nse}\left(\mathcal{E}^2(c_j^1 s)\right)$$

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 [17,18] – we do not do this here for ease of exposition.

The Quadratic Method. Thus, we may compute a level 2 encoding as:

$$\mathcal{E}^2(x_i x_j + p_0 \cdot \mu_{ij}) = \mathcal{E}^1(x_i)\mathcal{E}^1(x_j) + u_i^1 u_j^1 \,\mathcal{E}^2(s^2) - u_j^1 \,\mathcal{E}^2(\mathcal{E}^1(x_i) \cdot s) - u_i^1 \,\mathcal{E}^2(\mathcal{E}^1(x_j) \cdot s) \quad (4.7)$$

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 Eq. 4.7. Thus, letting $y_\ell = x_i x_j$, we may write

$$\mathcal{E}^2(y_\ell) = u_\ell^2 \cdot s + \mathsf{Nse}_\ell^2 + y_\ell \quad \in R_{p_2} \quad (4.8)$$

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:

$$c_\ell^3 = c_i^2 + c_j^2 \quad (4.9)$$

and

$$\mathsf{PK}(c_\ell^3) = \mathsf{PK}(c_i^2) + \mathsf{PK}(c_j^2) \quad (4.10)$$

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:

$$\begin{aligned} c_t^4 &= \mathcal{E}^4(y_\ell y_{\ell'}) = c_\ell^3 c_{\ell'}^3 + \mathcal{E}^4\big(u_\ell^3 u_{\ell'}^3 (s^2) - u_{\ell'}^3 (c_\ell^3 s) - u_\ell^3 (c_{\ell'}^3 s)\big) \in R_{p_4} \\ &= c_\ell^3 c_{\ell'}^3 + u_\ell^3 u_{\ell'}^3 \,\mathcal{E}^4(s^2) - u_{\ell'}^3 \,\mathcal{E}^4(c_\ell^3 s) - u_\ell^3 \,\mathcal{E}^4(c_{\ell'}^3 s) \end{aligned} \quad (4.11)$$

By correctness of first three levels of evaluation as described above, the decryptor can compute the encoding of y_ℓ, namely c^3_ℓ correctly, hence the quadratic term $c^3_\ell c^3_{\ell'}$ may be computed. It remains to compute the terms $\mathcal{E}^4(c^3_\ell s)$. Note that the encryptor may not provide the encodings $\mathcal{E}^4(c^3_\ell s)$ directly and preserve succinctness because $c^3_\ell = \mathcal{E}^2(x_i\ x_j + p_0 \cdot \mu_{ij}) + \mathcal{E}^2(x_m\ x_t + p_0 \cdot \mu_{mt})$ and $\mathcal{E}^2(x_i\ x_j + p_0 \cdot \mu_{ij})$ contains the cross term $c^1_i c^1_j$ as shown by Eq. 4.6.

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 $p_1 \cdot \mu^3_\ell$.

$$
\begin{aligned}
\mathcal{E}^4(c^3_\ell s) &= \mathcal{E}^4\big((\mathcal{E}^2(x_i\ x_j + p_0 \cdot \mu_{ij}) + \mathcal{E}^2(x_m\ x_t + p_0 \cdot \mu_{mt})) \cdot s\big) \\
&= \mathcal{E}^4\Big(\big(c^1_i c^1_j + u^1_i u^1_j\ \mathcal{E}^2(s^2) - u^1_j\ \mathcal{E}^2(c^1_i s) - u^1_i\ \mathcal{E}^2(c^1_j s)\big)\ \cdot s\Big) \\
&+ \mathcal{E}^4\Big(\big(c^1_m c^1_t + u^1_m u^1_t\ \mathcal{E}^2(s^2) - u^1_t\ \mathcal{E}^2(c^1_m s) - u^1_m\ \mathcal{E}^2(c^1_t s)\big)\ \cdot s\Big) \\
&= \mathcal{E}^4(c^1_i c^1_j s)\ + \mathcal{E}^4\big(u^1_i u^1_j\ \mathcal{E}^2(s^2)\ s\big)\ - \mathcal{E}^4\big(u^1_j\ \mathcal{E}^2(c^1_i s)\ s\big) - \mathcal{E}^4\big(u^1_i\ \mathcal{E}^2(c^1_j s)\ s\big) \\
&+ \mathcal{E}^4(c^1_m c^1_t s)\ + \mathcal{E}^4\big(u^1_m u^1_t\ \mathcal{E}^2(s^2)\ s\big) - \mathcal{E}^4\big(u^1_t\ \mathcal{E}^2(c^1_m s) s\big) - \mathcal{E}^4\big(u^1_m\ \mathcal{E}^2(c^1_t s)\ s\big) \\
&= \mathcal{E}^4(c^1_i c^1_j s)\ + u^1_i u^1_j\ \mathcal{E}^4\big(\mathcal{E}^2(s^2)\ s\big)\ - u^1_j\ \mathcal{E}^4\big(\mathcal{E}^2(c^1_i s)\ s\big) - u^1_i\ \mathcal{E}^4\big(\mathcal{E}^2(c^1_j s)\ s\big) \\
&+ \mathcal{E}^4(c^1_m c^1_t s)\ + u^1_m u^1_t\ \mathcal{E}^4\big(\mathcal{E}^2(s^2)\ s\big) - u^1_t\ \mathcal{E}^4\big(\mathcal{E}^2(c^1_m s) s\big) - u^1_m\ \mathcal{E}^4\big(\mathcal{E}^2(c^1_t s)\ s\big) \quad (4.12)
\end{aligned}
$$

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 $m \in [w]$,

$$
\Big\{\mathcal{E}^4\big(\mathcal{E}^2(s^2)\ s\big),\quad \mathcal{E}^4\big(\mathcal{E}^2(c^1_m s) s\big)\Big\} \subseteq \mathcal{C}^4 \tag{4.13}
$$

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 Eq. 4.7. 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 (Eq. 4.7): 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

$$
d_i \triangleq \mathcal{E}^2(c^1_i)\ ,\ \ h_j \triangleq \mathcal{E}^2(c^1_j \cdot s)\ \in R_{p_2}\ \text{ and }\ \hat{d}_i \triangleq \mathcal{E}^4\big(\mathcal{E}^2(c^1_i)\ s\big)\ ,\ \hat{h}_j \triangleq \mathcal{E}^4\big(\mathcal{E}^2(c^1_j \cdot s)\ s\big)\ \in R_{p_4}
$$

Then, we have:

$$
\begin{aligned}
\mathcal{E}^4(c^1_i c^1_j s + p_1 \cdot \mu^2_{ij}) = d_i h_j + \mathsf{PK}\big(\mathcal{E}^2(c^1_i)\big)\ \mathsf{PK}\big(\mathcal{E}^2(c^1_j \cdot s)\big)\ \mathcal{E}^4(s^2) \\
- \mathsf{PK}\big(\mathcal{E}^2(c^1_i)\big)\ \hat{h}_j - \mathsf{PK}\big(\mathcal{E}^2(c^1_j \cdot s)\big)\ \hat{d}_i\ \in R_{p_4}
\end{aligned} \tag{4.14}
$$

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 $\mathcal{E}^4(c_i^1 c_j^1 s + p_1 \cdot \mu_{ij}^2)$ may be computed as:

$$\begin{aligned}\mathsf{PK}\big(\mathcal{E}^4(c_i^1 c_j^1 s + p_1 \cdot \mu_{ij}^2)\big) = \mathsf{PK}\big(\mathcal{E}^2(c_i^1)\big)\ \mathsf{PK}\big(\mathcal{E}^2(c_j^1 \cdot s)\big)\ \mathsf{PK}\big(\mathcal{E}^4(s^2)\big) \\ - \mathsf{PK}\big(\mathcal{E}^2(c_i^1)\big)\ \mathsf{PK}(\hat{h}_j) - \mathsf{PK}\big(\mathcal{E}^2(c_j^1 \cdot s)\big)\ \mathsf{PK}(\hat{d}_i)\end{aligned} \tag{4.15}$$

Thus we have, $\mathcal{E}^4(c_\ell^3 s + p_1 \cdot \mu_\ell^3)$ is a Regev encoding with public key

$$\begin{aligned}&\mathsf{PK}(\mathcal{E}^4(c_\ell^3 s + p_1 \cdot \mu_\ell^3)) \\ &= \mathsf{PK}\Big(\mathcal{E}^4(c_i^1 c_j^1 s + p_1 \cdot \mu_{ij}^2)\ + u_i^1 u_j^1\ \mathcal{E}^4(\mathcal{E}^2(s^2)\ s)\ - u_j^1\ \mathcal{E}^4(\mathcal{E}^2(c_i^1 s)\ s) - u_i^1\ \mathcal{E}^4(\mathcal{E}^2(c_j^1 s)\ s) \\ &+ \mathcal{E}^4((c_m^1 c_t^1 s + p_1 \cdot \mu_{mt}^2)\ + u_m^1 u_t^1\ \mathcal{E}^4(\mathcal{E}^2(s^2)\ s) - u_t^1\ \mathcal{E}^4(\mathcal{E}^2(c_m^1 s) s) - u_m^1\ \mathcal{E}^4(\mathcal{E}^2(c_t^1 s)\ s)\Big) \\ &= \mathsf{PK}(\mathcal{E}^4(c_i^1 c_j^1 s + p_1 \cdot \mu_{ij}^2))\ + u_i^1 u_j^1\ \mathsf{PK}(\mathcal{E}^4(\mathcal{E}^2(s^2)\ s))\ - u_j^1\ \mathsf{PK}(\mathcal{E}^4(\mathcal{E}^2(c_i^1 s)\ s)) \\ &- u_i^1\ \mathsf{PK}(\mathcal{E}^4(\mathcal{E}^2(c_j^1 s)\ s)) + \mathsf{PK}(\mathcal{E}^4((c_m^1 c_t^1 s + p_1 \cdot \mu_{mt}^2))\ + u_m^1 u_t^1\ \mathsf{PK}(\mathcal{E}^4(\mathcal{E}^2(s^2)\ s)) \\ &- u_t^1\ \mathsf{PK}(\mathcal{E}^4(\mathcal{E}^2(c_m^1 s) s)) - u_m^1\ \mathsf{PK}(\mathcal{E}^4(\mathcal{E}^2(c_t^1 s)\ s))\end{aligned} \tag{4.16}$$

Above $\mathsf{PK}\big(\mathcal{E}^4(c_i^1 c_j^1 s + p_1 \cdot \mu_{ij}^2)\big)$ may be computed by Eq. 4.15 and the remaining public keys are provided in $\mathcal{C}^4$ as described in Eq. 4.13. Also, we have $\mu_\ell^3 = \mu_{ij}^2 + \mu_{mt}^2$.

By Eqs. 4.12, 4.13 and 4.14, we may compute $\mathcal{E}^4(c_\ell^3 s + p_1 \cdot \mu_\ell^3)$ for any ℓ.

Note that,

$$\begin{aligned}\mathcal{E}^4(c_\ell^3 s + p_1 \cdot \mu_\ell^3) &= \mathsf{LinComb}\Big(\mathcal{E}^2(c_i^1) \cdot \mathcal{E}^2(c_j^1 \cdot s), \mathcal{E}^4\big(\mathcal{E}^2(c_i^1)\ s\big),\ \mathcal{E}^4\big(\mathcal{E}^2(c_j^1 \cdot s)\ s\big)\Big) \\ &= \langle \mathsf{Lin}_{f^4},\ \mathcal{C}^4 \rangle + \mathsf{Quad}\big(\mathcal{E}^2(c_i^1) \cdot \mathcal{E}^2(c_j^1 \cdot s)\ \big)\end{aligned}$$

for some linear function Lin_{f^4}.

4.1 Ciphertext and Public Key Structure

By Eq. 4.11, we then get that

$$\begin{aligned}c_t^4 &= c_\ell^3\ c_{\ell'}^3 + u_\ell^3\ u_{\ell'}^3 \mathcal{E}^4(s^2) - u_\ell^3\ \Big(\langle \mathsf{Lin'}_{f^4},\ \mathcal{C}^4 \rangle + \mathsf{Quad'}\big(\mathcal{E}^2(c_i^1) \cdot \mathcal{E}^2(c_j^1 \cdot s)\ \big)\Big) \\ &\quad - u_{\ell'}^3\Big(\langle \mathsf{Lin''}_{f^4},\ \mathcal{C}^4 \rangle + \mathsf{Quad''}\big(\mathcal{E}^2(c_i^1) \cdot \mathcal{E}^2(c_j^1 \cdot s)\ \big)\Big) \\ &= \langle \mathsf{Lin'''}_{f^4},\ \mathcal{C}^4 \rangle + \mathsf{Poly}_{f^4}(\mathcal{C}^1, \mathcal{C}^2, \mathcal{C}^3)\end{aligned}$$

for some linear functions $\mathsf{Lin'}_{f^4}, \mathsf{Lin''}_{f^4}, \mathsf{Lin'''}_{f^4}$ and quadratic functions $\mathsf{Quad'}$, $\mathsf{Quad''}$ and polynomial Poly_{f^4}.

Thus, we have computed $\mathcal{E}^4(c_\ell^3 s + p_1 \cdot \mu_\ell^3)$ and hence, c^4 by Eq. 4.11. The final public key for c^4 is given by:

$$\mathsf{PK}(c^4) = u_\ell^3 u_{\ell'}^3\ \mathsf{PK}(\mathcal{E}^4(s^2)) - u_{\ell'}^3\ \mathsf{PK}(\mathcal{E}^4(c_\ell^3 s)) - u_\ell^3\ \mathsf{PK}(\mathcal{E}^4(c_{\ell'}^3 s)) \tag{4.17}$$

$\mathcal{E}^4(c^3)$ and $\mathcal{E}^4(c_i^1 c_j^1)$ 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. 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:

$$
\begin{aligned}
&\mathcal{E}^k(c^{k-1} \cdot s) \\
&= \mathcal{E}^k\Big((c_i^{k-2}\ c_j^{k-2} - u_i^{k-2}\mathcal{E}^{k-1}(c_j^{k-2} \cdot s) - u_j^{k-2}\mathcal{E}^{k-1}(c_i^{k-2} \cdot s) + u_i^{k-2}u_j^{k-2}\mathcal{E}^{k-1}(s^2)\ ) \cdot s \Big) \\
&= \mathcal{E}^k(c_i^{k-2}\ c_j^{k-2}\ s) - u_i^{k-2}\mathcal{E}^k\big(\mathcal{E}^{k-1}(c_j^{k-2} \cdot s) \cdot s\big) \\
&- u_j^{k-2}\mathcal{E}^k\big(\mathcal{E}^{k-1}(c_i^{k-2} \cdot s) \cdot s\big) + u_i^{k-2}u_j^{k-2}\mathcal{E}^k\big(\mathcal{E}^{k-1}(s^2) \cdot s\big) \qquad (4.18)
\end{aligned}
$$

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:

$$
\begin{aligned}
&\mathcal{E}^k(c_i^{k-2}\ c_j^{k-2}\ s) \\
&= \Big(\mathcal{E}^{k-1}(c_i^{k-2}) \cdot \mathcal{E}^{k-1}(c_j^{k-2} \cdot s)\Big) + \mathsf{PK}\big(\mathcal{E}^{k-1}(c_i^{k-2})\big)\mathsf{PK}\big(\mathcal{E}^{k-1}(c_j^{k-2} \cdot s)\big)\ \mathcal{E}^k(s^2) \\
&- \mathsf{PK}\big(\mathcal{E}^{k-1}(c_i^{k-2})\big)\Big(\mathcal{E}^k\big(\mathcal{E}^{k-1}(c_j^{k-2} \cdot s) \cdot s\big)\Big) - \mathsf{PK}\big(\mathcal{E}^{k-1}(c_j^{k-2} \cdot s)\big)\Big(\mathcal{E}^k\big(\mathcal{E}^{k-1}(c_i^{k-2}) \cdot s\big)\Big) \\
&\qquad (4.19)
\end{aligned}
$$

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 Eq. 4.18 also appear in Eq. 4.19 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 Eq. 4.19, 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 4.1) of the former at the cost of adding one more level of nesting and a new multiplication by s. By recursively applying Eq. 4.19, 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:

$$\mathcal{E}^{k-1}(c_j^{k-2} \cdot s) = \mathsf{LinComb}\big(\mathcal{C}^{k-1}, \mathcal{E}^{k-2}(\cdot)\mathcal{E}^{k-2}(\cdot)\big) \tag{4.20}$$

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,

$$\begin{aligned}
&\mathcal{E}^k\big(\mathcal{E}^{k-1}(c_j^{k-2} \cdot s) \cdot s\big) \\
&= \mathcal{E}^k\Big(\mathsf{LinComb}\big(\mathcal{C}^{k-1}, \mathcal{E}^{k-2}(\cdot)\mathcal{E}^{k-2}(\cdot)\big) \cdot s\Big) \\
&= \mathcal{E}^k\Big(\mathsf{LinComb}\big(\mathcal{C}^{k-1} \cdot s\big)\Big) + \mathcal{E}^k\Big(\mathsf{LinComb}\big(\mathcal{E}^{k-2}(y_a)\mathcal{E}^{k-2}(y_b) \cdot s\big)\Big) \\
&= \mathsf{LinComb}\Big(\mathcal{E}^k\big(\mathcal{C}^{k-1} \cdot s\big)\Big) + \mathsf{LinComb}\Big(\mathcal{E}^k\big(\mathcal{E}^{k-2}(y_a)\mathcal{E}^{k-2}(y_b) \cdot s\big)\Big)
\end{aligned} \tag{4.21}$$

   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,

$$\begin{aligned}
&\mathcal{E}^k\Big(\big(\mathcal{E}^{k-2}(y_a)\mathcal{E}^{k-2}(y_b)\big) \cdot s\Big) \\
&= \mathsf{LinComb}\Big(\mathcal{E}^{k-1}(\mathcal{E}^{k-2}(y_a)) \cdot \mathcal{E}^{k-1}(\mathcal{E}^{k-2}(y_b) \cdot s), \\
&\quad \mathcal{E}^k\big(\mathcal{E}^{k-1}(\mathcal{E}^{k-2}(y_a)) \cdot s\big),\ \mathcal{E}^k\big(\mathcal{E}^{k-1}(\mathcal{E}^{k-2}(y_b) \cdot s) \cdot s\big)\Big)
\end{aligned} \tag{4.22}$$

   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

$$\mathcal{E}^{k-2}(y_b) = \mathsf{LinComb}\big(\mathcal{C}^{k-2}, \mathcal{E}^{k-3}(\cdot)\mathcal{E}^{k-3}(\cdot)\big)$$
$$\therefore \mathcal{E}^k\big(\mathcal{E}^{k-1}(\mathcal{E}^{k-2}(y_b)\cdot s)\cdot s\big) = \mathcal{E}^k\Big(\mathcal{E}^{k-1}\big(\mathsf{LinComb}\big(\mathcal{C}^{k-2}, \mathcal{E}^{k-3}(\cdot)\mathcal{E}^{k-3}(\cdot)\big)\cdot s\big)\cdot s\Big)$$

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:

$$\begin{aligned}
&\mathcal{E}^k\Big(\mathcal{E}^{k-1}(\mathcal{E}^{k-3}(\cdot)\mathcal{E}^{k-3}(\cdot)\cdot s)\cdot s\Big)\\
&= \mathcal{E}^k\Big(\mathsf{LinComb}\Big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot)\cdot s\big)\cdot \mathcal{E}^{k-2}(\mathcal{E}^{k-3}(\cdot)\big),\\
&\qquad \mathcal{E}^{k-1}\big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot)\cdot s\big)\cdot s\big)\Big)\Big)\\
&= \mathsf{LinComb}\Big(\mathcal{E}^{k-1}\big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot)\cdot s\big)\big)\cdot \mathcal{E}^{k-1}\big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot)\cdot s\big)\cdot s\big),\\
&\quad \mathcal{E}^k\Big(\mathcal{E}^{k-1}\big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot)\cdot s\big)\cdot s\big)\cdot s\Big)\cdot s\Big)
\end{aligned}$$

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 $\mathcal{E}^k\Big(\mathcal{E}^{k-1}\big(\mathcal{E}^{k-2}\big(\mathcal{E}^{k-3}(\cdot)\cdot s\big)\cdot s\big)\cdot s\Big)$ which has half the nested message degree of the former at the cost of one additional nesting (and multiplication by s)[5].

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 Fig. 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 Eq. 4.16.

$$\mathsf{PK}(\mathcal{E}^k(c^{k-1}\cdot s)) = \mathsf{LinComb}(\mathsf{PK}(\mathcal{C}^k)) \tag{4.23}$$

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:

$$\begin{aligned}
c_t^k &= c_\ell^{k-1}c_{\ell'}^{k-1} + \mathcal{E}^k\big(u_\ell^{k-1}u_{\ell'}^{k-1}(s^2) - u_{\ell'}^{k-1}(c_\ell^{k-1}s) - u_\ell^{k-1}(c_{\ell'}^{k-1}s)\big)\\
&= c_\ell^{k-1}c_{\ell'}^{k-1} + u_\ell^{k-1}u_{\ell'}^{k-1}\,\mathcal{E}^k(s^2) - u_{\ell'}^{k-1}\,\mathcal{E}^k(c_\ell^{k-1}s) - u_\ell^{k-1}\,\mathcal{E}^k(c_{\ell'}^{k-1}s)
\end{aligned} \tag{4.24}$$

[5] 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.

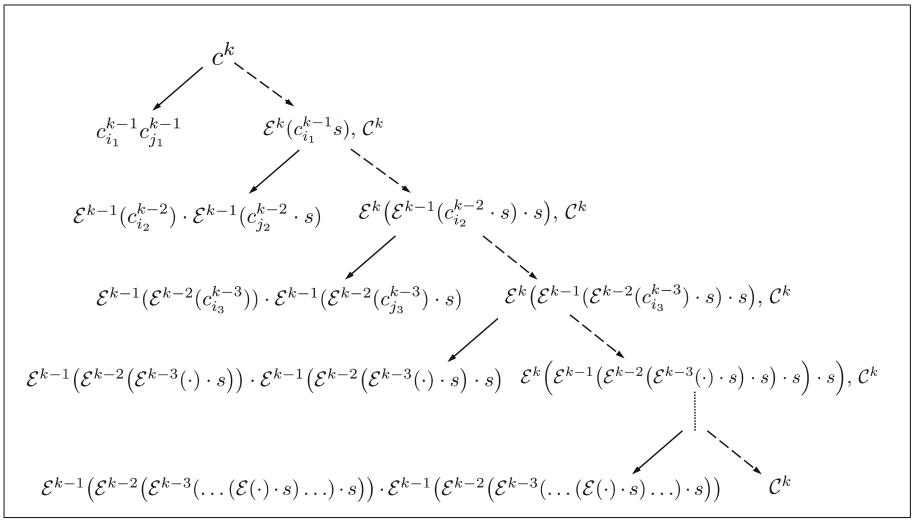

Fig. 1. Computing level k functional ciphertext c^k encoding $f^k(\mathbf{x})$ using induction. A term in any node is implied by a quadratic polynomial in its children, quadratic in the terms of the left child, and linear in the terms of the right child. The solid arrows on the left indicate quadratic 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 $\mathcal{C}^1, \ldots, \mathcal{C}^{k-1}$.

Similarly,

$$\mathsf{PK}(c_t^k) = u_\ell^{k-1} u_{\ell'}^{k-1}\, \mathsf{PK}(\mathcal{E}^k(s^2)) - u_{\ell'}^{k-1}\, \mathsf{PK}\big(\mathcal{E}^k(c_\ell^{k-1} s)\big) - u_\ell^{k-1}\, \mathsf{PK}\big(\mathcal{E}^k(c_{\ell'}^{k-1} s)\big) \tag{4.25}$$

Public Key, Ciphertext and Decryption Structure. From the above, we claim:

Claim. 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:

$$\begin{aligned} c^k = \mathsf{CT}(f^k(\mathbf{x}) + \mu^k_{f(\mathbf{x})}) &= \langle \mathsf{Lin}_{f^k}, \mathcal{C}^k \rangle + \mathsf{LinComb}\big(\mathsf{Quad}(\mathcal{E}^{k-1}(y_i)\ \mathcal{E}^{k-1}(y_j))\big) \\ &= \langle \mathsf{Lin}_{f^k}, \mathcal{C}^k \rangle + \mathsf{Poly}_{f^k}\big(\mathcal{C}^1, \ldots, \mathcal{C}^{k-1}\big) \end{aligned}$$

Moreover, note that the computation of the functional message embedded in a level k ciphertext c^k can be viewed as follows. By Eq. 4.6, 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 Figs. 2 and 3 for the formal descriptions.

Algorithm $\mathsf{Eval}^k_{\mathsf{PK}}(\bigcup_{i\in[k]} \mathsf{PK}(\mathcal{C}^i), \ell)$

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 4.10.
   - Let $\mathsf{PK}^{k-1}_i = \mathsf{Eval}^{k-1}_{\mathsf{PK}}(\bigcup_{j\in[k-1]} \mathsf{PK}(\mathcal{C}^j), i)$ and $\mathsf{PK}^{k-1}_j = \mathsf{Eval}^{k-1}_{\mathsf{PK}}(\bigcup_{i\in[k-1]} \mathsf{PK}(\mathcal{C}^i), j)$,
   - 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 4.17.
   - Let $u^{k-1}_i = \mathsf{Eval}^{k-1}_{\mathsf{PK}}(\bigcup_{j\in[k-1]} \mathsf{PK}(\mathcal{C}^j), i)$ and $u^{k-1}_j = \mathsf{Eval}^{k-1}_{\mathsf{PK}}(\bigcup_{i\in[k-1]} \mathsf{PK}(\mathcal{C}^i), j)$
   - 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 4.25.
     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 4.16, 4.23.

Fig. 2. Algorithm to evaluate on public key.

Algorithm $\mathsf{Eval}^k_{\mathsf{CT}}(\underset{i\in[k]}{\cup}\mathcal{C}^i, \ell)$

To compute the encoding 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 $c^3_\ell = c^2_i + c^2_j$ as in Equation 4.9.
   - 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 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 c^4_ℓ (for any ℓ) using Equations 4.11 and 4.12.
   - 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 4.24. 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 4.1

Fig. 3. Algorithm to evaluate on ciphertext.

5 Succinct Functional Encryption for NC_1

In this section, we extend the construction for quadratic functional encryption provided in Sect. 3 to circuits of depth $O(\log n)$. The construction generalises directly the QuadFE scheme using the public key and ciphertext evaluation algorithms from the previous section. We make black box use of the LinFE scheme [1,5].

We proceed to describe the construction.

$\mathsf{NC}_1.\mathsf{Setup}(1^\lambda, 1^w, 1^d)$: Upon input the security parameter λ, 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 4.2. For $k \in [d-1]$, $i \in [L_k]$, choose uniformly random $u_{i,k} \in R_{p_k}$. Denote $\mathbf{u}_k = (u_{i,k}) \in R^{L_k}_{p_k}$.
2. Invoke $\mathsf{LinFE.Setup}(1^\lambda, 1^{L_d+1}, p_d)$ to obtain $\mathsf{PK} = \mathsf{LinFE.PK}$ and $\mathsf{MSK} = \mathsf{LinFE.MSK}$.
3. Output $\mathsf{PK} = (\mathsf{LinFE.PK}, \mathbf{u}_1, \ldots, \mathbf{u}_{d-1})$ and $\mathsf{MSK} = \mathsf{LinFE.MSK}$.

$\mathsf{NC}_1.\mathsf{KeyGen}(\mathsf{MSK}, f))$: Upon input the master secret key MSK and a circuit f of depth d, do:

1. Let $\mathsf{Lin}_f \in R^{L_d}_{p_d}$ be an f dependent linear function output by the algorithm $\mathsf{Eval}_{\mathsf{PK}}(\mathsf{PK}, f)$. as described in claim 4.1.
2. Compute $\mathsf{SK}_{\mathsf{Lin}} = \mathsf{LinFE.KeyGen}\big(\mathsf{MSK}, (\mathsf{Lin}_f \| 1)\big)$ and output it.

$\mathsf{NC}_1.\mathsf{Enc}(\mathbf{x}, \mathsf{PK})$: 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 4.2.
2. Sample flooding noise η as described in Appendix B.
3. Define $\mathcal{M}^d = \big(\ \mathcal{C}^{d-1},\ \mathcal{C}^{d-1} \cdot s,\ \mathcal{E}^d(s^2)\ \big) \in R^{L_d}_{p_d}$. Compute $\mathsf{CT}_{\mathsf{Lin}} = \mathsf{LinFE.Enc}\ \big(\ \mathsf{PK}, (\mathcal{M}^d \| \eta)\ \big)$
4. Output $\mathsf{CT}_{\mathbf{x}} = (\{\mathcal{C}^k\}_{k\in[d-1]}, \mathsf{CT}_{\mathsf{Lin}})$.

$\mathsf{NC}_1.\mathsf{Dec}(\mathsf{PK}, \mathsf{CT}_{\mathbf{x}}, \mathsf{SK}_f)$: Upon input a ciphertext $\mathsf{CT}_{\mathbf{x}}$ for vector $\mathbf{x}$, and a secret key $\mathsf{SK}_f = \mathbf{k}_f$ for circuit f, do:

1. Compute $\mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1})$ as described in Sect. 4 by running $\mathsf{Eval}_{\mathsf{CT}}(\{\mathcal{C}^k\}_{k\in[d-1]}, f)$.
2. Compute $\mathsf{LinFE.Dec}(\mathsf{CT}_{\mathsf{Lin}}, \mathsf{SK}_{\mathsf{Lin}}) + \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) \mod p_d \mod p_{d-1} \ldots \mod p_0$ and output it.

Correctness follows from correctness of $\mathsf{Eval}_{\mathsf{PK}}$, $\mathsf{Eval}_{\mathsf{CT}}$ and LinFE. In more detail, we have by Theorem 4.2 that,

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

Since $\mathsf{CT}_{\mathsf{Lin}}$ is a LinFE encryption of $(\mathcal{M}^d \| \eta)$ and $\mathsf{SK}_{\mathsf{Lin}}$ is a LinFE functional key for $(\mathsf{Lin}_f \| 1)$, we have by correctness of LinFE that $\mathsf{LinFE.Dec}(\mathsf{CT}_{\mathsf{Lin}}, \mathsf{SK}_{\mathsf{Lin}}) = \langle \mathsf{Lin}_f, \mathcal{M}^d \rangle + \eta \mod p_d$. By correctness of $\mathsf{Eval}_{\mathsf{CT}}$, we have that $\mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) + \langle \mathsf{Lin}_f, \mathcal{M}^d \rangle$ outputs $f(\mathbf{x}) + \mu_{f(\mathbf{x})} + \eta$. Since $\mu_{f(\mathbf{x})}$ as well as η is a linear combination of noise terms which are multiples of moduli p_i for $i \in [0, \ldots, d-1]$, i.e. $\mu_{f(\mathbf{x})} = p_{d-1} \cdot \beta^f_{d-1} + \ldots + p_0 \cdot \beta^f_0$ for some β^f_i, and $f(\mathbf{x}) \in R_{p_0}$, we have that $f(\mathbf{x}) + \mu_{f(\mathbf{x})} + \eta = f(\mathbf{x}) \mod p_d \mod p_{d-1} \ldots \mod p_0$, as desired.

Analysis of Ciphertext Structure. Note that the ciphertext consists of encodings $\mathcal{C}^k$ for $k \in [d-1]$ and LinFE ciphertext for $(\mathcal{M}^d \| \eta)$. Since each message-dependent encoding depends only on a single bit of the message, the ciphertext is decomposable, and enjoys local-updates: if a single bit of the message changes, then only $O(d)$ encodings need updating, not the entire ciphertext. Also, since the LinFE ciphertext is succinct, the message-dependent component of our ciphertext is also succinct. The ciphertext is *not* succinct overall, since we need to encode a fresh noise term per requested key.

Theorem 5.4. *The construction in Sect. 5 achieves full simulation based security as per Definition 2.2.*

Proof. We describe our simulator.

Simulator $\mathsf{NC}_1.\mathsf{Sim}(1^\lambda, 1^{|\mathbf{x}|}, \mathsf{PK}, f, \mathsf{SK}_f, f(\mathbf{x}))$. The simulator given input the security parameter, length of message $\mathbf{x}$, the circuit f, the secret key SK_f and the value $f(\mathbf{x})$ does the following:

1. It computes $\mathsf{Lin}_f = \mathsf{Eval}_{\mathsf{PK}}(\mathsf{PK}, f)$. Note that by claim 4.1 that $\mathsf{Lin}_f \in R^{L_d}_{p_d}$.
2. It samples all encodings upto level $d-1$ randomly, i.e. $\mathcal{C}^k \leftarrow R^{L_k}_{p_k}$ for $k \in [d-1]$.
3. It samples $\eta \leftarrow \mathcal{D}_d$ as described in Appendix B and computes $d' = f(\mathbf{x}) + \eta - \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1})$.
4. It invokes the single key LinFE simulator as

$$\mathsf{CT}_{\mathsf{Lin}} = \mathsf{LinFE.Sim}(1^\lambda, 1^{L_d}, \mathsf{PK}, \mathsf{Lin}_f, \mathsf{SK}(\mathsf{Lin}_f), d')$$

5. It outputs $\mathsf{CT}_{\mathbf{x}} = (\{\mathcal{C}^k\}_{k\in[d-1]}, \mathsf{CT}_{\mathsf{Lin}})$.

We will prove that the output of the simulator is indistinguishable from the real world via a sequence of hybrids.

The Hybrids. Our Hybrids are described below.

Hybrid 0. This is the real world.

Hybrid 1. In this hybrid, the only thing that is different is that $\mathsf{CT}_{\mathsf{Lin}}$ is computed using the LinFE simulator. In more detail,

- It computes $\mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) = \mathsf{Eval}_{\mathsf{CT}}\big(\{\mathcal{C}^k\}_{k\in[d-1]}, f\big)$.
- It computes $f(\mathbf{x}) + \mu_{f(\mathbf{x})} = \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1}) + \langle \mathcal{M}^d, \mathsf{Lin}_f \rangle$
- It samples η such that

$$\mathsf{SD}\big(\eta + \mu_{f(\mathbf{x})}, \eta\big) \leq \mathrm{negl}(\lambda) \tag{5.1}$$

- It invokes the single key LinFE simulator with input $f(\mathbf{x}) + \mu_{f(\mathbf{x})} + \eta - \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1})$.

Hybrid 2. In this hybrid, invoke the LinFE simulator with $f(\mathbf{x}) + \eta - \mathsf{Poly}_f(\mathcal{C}^1, \ldots, \mathcal{C}^{d-1})$.

Hybrid 3. In this hybrid, sample $\mathcal{C}^k$ for $k \in [d-1]$ at random. This is the simulated world.

Indistinguishability of Hybrids proceeds as in Sect. 3. Indistinguishability of Hybrids 0 and 1 follows from security of LinFE. It is easy to see that Hybrids 1 and 2 are statistically indistinguishable by Eq. 5.1. Hybrids 2 and 3 are indistinguishable due to semantic security of Regev encodings $\mathcal{C}^k$ for $k \in [d-1]$.

In the full version [6], we describe how to generalize the above construction to bounded collusion FE scheme for all circuits in P, for any a-priori fixed polynomial bound Q. The approach follows the (by now) standard bootstrapping method of using low depth randomized encodings to represent any polynomial sized circuit [39]. The ciphertext of the final scheme enjoys additive quadratic dependence on the collusion bound Q.

6 Bounded Collusion FE for All Circuits

In this section, we describe how to put together the pieces from the previous sections to build a bounded collusion FE scheme for all circuits in P, denoted by BddFE. The approach follows the (by now) standard bootstrapping method of using low depth randomized encodings to represent any polynomial sized circuit. This approach was first suggested by Gorbunov et al. [39], who show that q query FE for degree three polynomials can be bootstrapped to q query FE for all circuits.

At a high level, their approach can be summarized as follows. Let $\mathcal{C}$ be a family of polynomial sized circuits. Let $C \in \mathcal{C}$ and let $\mathbf{x}$ be some input. Let $\tilde{C}(\mathbf{x}, R)$ be a randomized encoding of C that is computable by a constant depth

circuit with respect to inputs x and R. Then consider a new family of circuits $\mathcal{G}$ defined by:

$$G_{C,\Delta}(\mathbf{x}, R_1, \ldots, R_S) = \tilde{C}\Big(\mathbf{x}; \underset{a\in\Delta}{\oplus} R_a\Big)$$

Note that $G_{C,\Delta}(\cdot,\cdot)$ is computable by a degree three polynomial, one for each output bit. Given an FE scheme for $\mathcal{G}$, one may construct a scheme for $\mathcal{C}$ by having the decryptor first recover the output of $G_{C,\Delta}(\mathbf{x}, R_1, \ldots, R_S)$ and then applying the decoder for the randomized encoding to recover $C(\mathbf{x})$. Since our construction from Sect. 5 is capable of evaluating degree 3 polynomials, it suffices for bootstrapping, to yield q-query FE for all circuits. We will denote this scheme by PolyFE as against $\mathsf{NC_1FE}$ to emphasize that it needs to only compute degree 3 polynomials.

As in [5,39], let (S, v, m) be parameters to the construction. Let Δ_i for $i \in [q]$ be a uniformly random subset of $[S]$ of size v. To support q queries, the key generator identifies the set $\Delta_i \subseteq [S]$ with query i. If $v = O(\lambda)$ and $S = O(\lambda \cdot q^2)$ then the sets Δ_i are cover free with high probability as shown by [39]. Let $L \triangleq (\ell^3 + S \cdot m)$.

$\mathsf{BddFE.Setup}(1^\lambda, 1^\ell)$: Upon input the security parameter λ and the message space $\{0,1\}^\ell$, invoke $(\mathsf{mpk}, \mathsf{msk}) = \mathsf{PolyFE.Setup}(1^\lambda, 1^L)$ and output it.

$\mathsf{BddFE.KeyGen}(\mathsf{msk}, C)$): Upon input the master secret key and a circuit C, do:

1. Choose a uniformly random subset $\Delta \subseteq [S]$ of size v.
2. Express $C(\mathbf{x})$ by $G_{C,\Delta}(\mathbf{x}, R_1, \ldots, R_S)$, which in turn can be expressed as a sequence of degree 3 polynomials $P_1, \ldots, P_k$, where $k \in \mathsf{poly}(\lambda)$.
3. Set $\mathsf{BddFE.SK}_C = \{\mathsf{SK}_i = \mathsf{PolyFE.KeyGen}(\mathsf{PolyFE.msk}, P_i)\}_{i\in[k]}$ and output it.

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

1. Choose $R_1, \ldots, R_S \leftarrow \{0,1\}^m$, where m is the size of the random input in the randomized encoding.
2. Set $\mathsf{CT}_\mathbf{x} = \mathsf{PolyFE.Enc}(\mathsf{PolyFE.mpk}, \mathbf{x}, R_1, \ldots, R_s)$ and output it.

$\mathsf{BddFE.Dec}(\mathsf{mpk}, \mathsf{CT}_\mathbf{x}, \mathsf{SK}_C)$: Upon input a ciphertext $\mathsf{CT}_\mathbf{x}$ for vector $\mathbf{x}$, and a secret key SK_C for circuit C, do the following:

1. Compute $G_{C,\Delta}(\mathbf{x}, R_1, \ldots, R_S) = \mathsf{PolyFE.Dec}(\mathsf{CT}_\mathbf{x}, \mathsf{SK}_C)$.
2. Run the Decoder for the randomized encoding to recover $C(\mathbf{x})$ from $G_{C,\Delta}(\mathbf{x}, R_1, \ldots, R_S)$.

Correctness follows immediately from the correctness of PolyFE and the correctness of randomized encodings. The proof of security follows easily from the security of randomized encodings and of the PolyFE scheme. Please see the full version [6] for details.

Acknowledgements. We thank Damien Stehlé and Chris Peikert for helpful discussions.

Appendix

A Previous Constructions for Bounded Collusion FE

The GVW12 Construction. The scheme of [39] can be summarized as follows.

- The first ingredient they need is a single key FE scheme for all circuits. A construction for this was provided by Sahai and Seyalioglu in [51].
- Next, the single FE scheme is generalized to a q query scheme for NC_1 circuits. This gerenalization is fairly complex, we provide an outline here. At a high level, they run N copies of the single key scheme, where $N = O(q^4)$. The encryptor encrypts the views of the BGW MPC protocol for N parties, computing some functionality related to C. They rely on the fact that BGW is non-interactive when used to compute bounded degree functions. To generate a secret key, KeyGen chooses a random subset of the single query FE keys, where the parameters are set so that the subsets have small pairwise intersections. This subset of keys enables the decryptor to recover sufficiently many shares of $C(x)$ which allows him to recover $C(x)$. [39] argue that an attacker with q keys only learns a share x_i when two subsets of keys intersect, but since the subsets were chosen to have small pairwise intersections, this does not occur often enough to recover enough shares of x. Finally, by the security of secret sharing, x remains hidden.
- As the last step they "bootstrap" the q query FE for NC_1 to q query FE for all circuits using computational randomized encodings [9]. They must additionally use cover free sets to ensure that fresh randomness is used for each randomized encoding.

Thus, to encrypt a message x, the encryptor must secret share it into $N = O(q^4)$ shares, and encrypt each one with the one query FE. Since they use Shamir secret sharing with polynomial of degree t and $t = O(q^2)$, note that at most $O(q^2)$ shares can be generated offline, since $t + 1$ points will determine the polynomial. Hence $O(q^4)$ shares must be generated in the online phase. This results in an online encryption time that degrades as $O(q^4)$.

The ALS16 construction. [5] provide a conceptually simpler way to build q-query Functional Encryption for all circuits. Their construction replaces steps 1 and 2 described above with a inner product modulo p FE scheme, and then uses step 3 as in [39]. Thus, the construction of single key FE in step 1 by Sahai and Seyalioglu, and the nontrivial "MPC in the head" of step 2 can both be replaced by the simple abstraction of an inner product FE scheme. For step 3, observe that the bootstrapping theorem of [39] provides a method to bootstrap an FE for NC_1 that handles q queries to an FE for all polynomial-size circuits that is also secure against q queries. The bootstrapping relies on the result of Applebaum *et al.* [9, Theorem 4.11] which states that every polynomial time computable function f admits a perfectly correct computational randomized encoding of degree 3.

In more detail, let $\mathcal{C}$ be a family of polynomial-size circuits. Let $C \in \mathcal{C}$ and let x be some input. Let $\widetilde{C}(x, R)$ be a randomized encoding of C that is computable by a constant depth circuit with respect to inputs x and R. Then consider a new family of circuits $\mathcal{G}$ defined by:

$$G_{C,\Delta}(x, R_1, \ldots, R_S) = \left\{ \widetilde{C}\Big(x; \bigoplus_{a \in \Delta} R_a\Big) : \ C \in \mathcal{C}, \ \Delta \subseteq [S] \right\},$$

for some sufficiently large S (quadratic in the number of queries q). As observed in [39], circuit $G_{C,\Delta}(\cdot, \cdot)$ is computable by a constant degree polynomial (one for each output bit). Given an FE scheme for $\mathcal{G}$, one may construct a scheme for $\mathcal{C}$ by having the decryptor first recover the output of $G_{C,\Delta}(x, R_1, \ldots, R_S)$ and then applying the decoder for the randomized encoding to recover $C(x)$.

However, to support q queries the decryptor must compute q randomized encodings, each of which needs fresh randomness. This is handled by hardcoding S random elements in the ciphertext and using random subsets $\Delta \subseteq [S]$ (which are cover-free with overwhelming probability) to compute fresh randomness $\bigoplus_{a \in \Delta} R_a$ for every query. [5] observe that bootstrapping only requires support for the particular circuit class $\mathcal{G}$ described above. This circuit class, being computable by degree 3 polynomials, may be supported by a linear FE scheme, via linearization of the degree 3 polynomials.

Putting it together, the encryptor encrypts all degree 3 monomials in the inputs $R_1, \ldots, R_S$ and $x_1, \ldots, x_\ell$. Note that this ciphertext is polynomial in size. Now, for a given circuit C, the keygen algorithm samples some $\Delta \subseteq [S]$ and computes the symbolic degree 3 polynomials which must be released to the decryptor. It then provides the linear FE keys to compute the same. By correctness and security of Linear FE as well as the randomizing polynomial construction, the decryptor learns $C(x)$ and nothing else.

Note that in this construction the challenge of supporting multiplication is sidestepped by merely having the encryptor encrypt each monomial $x_i x_j$ separately so that the FE need only support addition. This "brute force" approach incurs several disadvantages. For instance, decomposability is lost – even though the ciphertext can be decomposed into $|\mathbf{x}|^2$ components, any input bit x_1 (say) must feature in $|\mathbf{x}|$ ciphertext components $x_1 x_2, \ldots, x_1 x_w$, where $w = |\mathbf{x}|$. This makes the scheme inapplicable for all applications involving distributed data, where a centre or a sensor device knows a bit x_i but is oblivious to the other bits. Additionally, the scheme is not online-offline, since all the ciphertext components depend on the data, hence the entire encryption operation must be performed after the data becomes available. For applications where a centre or sensor must transmit data-dependent ciphertext after the data is observed, this incurs a significant cost in terms of bandwidth. Indeed, the work performed by the sensor device in computing the data dependent ciphertext becomes proportional to the size of the function being computed on the data, which may be infeasible for weak devices.

Another approach to obtain bounded collusion FE is to compile the single key FE of Goldwasser et al. [37] with the compiler of [39] to support Q queries. Again, this approach yields succinct CTs but the CT grows as $O(q^4)$ rather than $O(q^2)$ as in our scheme.

B Parameters

In this section, we discuss the parameters for our constructions. 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 require all the distinct moduli to be relatively prime, hence we choose all the moduli to be prime numbers of the aforementioned size.

We must also choose the size of the noise that is added for flooding. As described in Sect. 3, for quadratic polynomials we require $\frac{L \cdot p_0 \cdot \sigma_0^2}{\sigma_1} = \mathrm{negl}(\lambda)$ where σ_1 is the standard deviation for the noise η_i for $i \in [Q]$ encoded in the ciphertext. For depth d (Sect. 5), we require $\frac{L_d \cdot \prod_{i \in [0,d]} p_i \cdot B^{2^d}}{\sigma} = \mathrm{negl}(\lambda)$ where σ is the standard deviation of the noise η encoded in the ciphertext. Since $L_d = \mathrm{poly}(\lambda)$ by definition, we require $\sigma \geq O(\mathrm{poly}(\lambda) B^{2^{d+1}}$.

We may set $p_0 = n$ with initial noise level as $B_1 = \mathrm{poly}(n)$ 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. Note that unlike FHE constructions [18,19], computation in our case proceeds UP a ladder of moduli rather than down, and we may add fresh noise at each level. Hence we never need to rely on subexponential modulus to noise ratio, and may support circuits in NC_1 even without modulus switching tricks.

We note that by the definition of efficiency of reusable garbled circuits [37], it suffices to have ciphertext size that is sublinear in circuit size, which is achieved by our construction.

References

1. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46447-2_33
2. Agrawal, S.: Stronger security for reusable garbled circuits, general definitions and attacks. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 3–35. Springer, Cham (2017). doi:10.1007/978-3-319-63688-7_1
3. Agrawal, S., Boneh, D., Boyen, X.: Efficient Lattice (H)IBE in the Standard Model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_28
4. Agrawal, S., Freeman, D.M., Vaikuntanathan, V.: Functional encryption for inner product predicates from learning with errors. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 21–40. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25385-0_2
5. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for linear functions from standard assumptions, and applications. In: CRYPTO (2016). https://eprint.iacr.org/2015/608

6. Agrawal, S., Rosen, A.: Functional encryption for bounded collusions, revisited. Eprint (2016). https://eprint.iacr.org/2016/361.pdf
7. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 308–326. Springer, Heidelberg (2015). doi:10.1007/978-3-662-47989-6_15
8. Ananth, P., Jain, A., Sahai, A.: Indistinguishability obfuscation from functional encryption for simple functions. Eprint 2015/730 (2015)
9. Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. Comput. Complex. **15**(2), 115–162 (2006)
10. Applebaum, B., Ishai, Y., Kushilevitz, E.: How to garble arithmetic circuits. In: IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, 22–25 October 2011, pp. 120–129 (2011)
11. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334 (2007)
12. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. IACR Cryptology ePrint Archive 2015, p. 163 (2015). http://eprint.iacr.org/2015/163
13. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_13
14. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (2014). doi:10.1007/978-3-642-55220-5_30
15. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007). doi:10.1007/978-3-540-70936-7_29
16. Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006). doi:10.1007/11818175_17
17. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of ITCS, pp. 309–325 (2012)
18. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, 22–25 October 2011, pp. 97–106 (2011)
19. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22792-9_29
20. Canetti, R., Chen, Y.: Constraint-hiding constrained PRFS for NC1 from LWE. In: Eurocrypt (2017)
21. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_27
22. Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 3–12. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46800-5_1

23. Cheon, J.H., Fouque, P.-A., Lee, C., Minaud, B., Ryu, H.: Cryptanalysis of the new CLT multilinear map over the integers. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 509–536. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49890-3_20
24. Cheon, J.H., Jeong, J., Lee, C.: An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low level encoding of zero. Eprint 2016/139
25. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Proceedings of 8th International Conference on IMA, pp. 360–363 (2001)
26. Coron, J.-S., Gentry, C., Halevi, S., Lepoint, T., Maji, H.K., Miles, E., Raykova, M., Sahai, A., Tibouchi, M.: Zeroizing without low-level zeroes: new mmap attacks and their limitations. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 247–266. Springer, Heidelberg (2015). doi:10.1007/978-3-662-47989-6_12
27. Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 476–493. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_26
28. Cramer, R., Hanaoka, G., Hofheinz, D., Imai, H., Kiltz, E., Pass, R., Shelat, A., Vaikuntanathan, V.: Bounded CCA2-secure encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 502–518. Springer, Heidelberg (2007). doi:10.1007/978-3-540-76900-2_31
29. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-insulated public key cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (2002). doi:10.1007/3-540-46035-7_5
30. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38348-9_1
31. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013). http://eprint.iacr.org/
32. Garg, S., Gentry, C., Halevi, S., Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 479–499. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40084-1_27
33. Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Fully secure functional encryption without obfuscation. In: IACR Cryptology ePrint Archive, p. 666 (2014). http://eprint.iacr.org/2014/666
34. Gentry, C., Gorbunov, S., Halevi, S.: Graph-induced multilinear maps from lattices. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 498–527. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46497-7_20
35. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC, pp. 197–206 (2008)
36. Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Robustness of the learning with errors assumption. In: ITCS (2010)
37. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: STOC, pp. 555–564 (2013)
38. Goldwasser, S., Lewko, A., Wilson, D.A.: Bounded-collusion IBE from key homomorphism. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 564–581. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28914-9_32

39. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32009-5_11
40. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute based encryption for circuits. In: STOC (2013)
41. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 503–523. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48000-7_25
42. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security, pp. 89–98 (2006)
43. Hu, Y., Jia, H.: Cryptanalysis of GGH map. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 537–565. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49890-3_21
44. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: Widmayer, P., Eidenbenz, S., Triguero, F., Morales, R., Conejo, R., Hennessy, M. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 244–256. Springer, Heidelberg (2002). doi:10.1007/3-540-45465-9_22
45. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78967-3_9
46. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_4
47. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_1
48. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM J. Comput. **37**(1), 267–302 (2007). Extended abstract in FOCS 2004
49. Miles, E., Sahai, A., Zhandry, M.: Annihilation attacks for multilinear maps: cryptanalysis of indistinguishability obfuscation over GGH13. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 629–658. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53008-5_22
50. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC 2005 (Extended Abstract), J. ACM 56(6) (2009)
51. Sahai, A., Seyalioglu, H.: Worry-free encryption: Functional encryption with public keys. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010 (2010)
52. Sahai, A., Waters, B.: Functional encryption: beyond public key cryptography. Power Point Presentation (2008). http://userweb.cs.utexas.edu/bwaters/presentations/files/functional.ppt
53. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). doi:10.1007/11426639_27
54. Waters, B.: Functional encryption for regular languages. In: Crypto (2012)