# Functional Encryption for Bounded Collusions, Revisited

Shweta Agrawal* Alon Rosen†

**Abstract**

We provide a new construction of functional encryption 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 $q$. Compared to the previous best construction in this model, by Gorbunov, Vaikuntanathan and Wee (CRYPTO'12), our scheme has the following advantages:

- The ciphertext of our scheme grows as $O(q^2)$, whereas that in [GVW12] grows as $O(q^4)$.
- 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 for preparation in an offline phase, where a majority of the computation is done before the data becomes available. This is followed by an efficient online phase, which is performed when the data becomes known. The online component of our scheme significantly outperforms the online component of [GVW12], and depends linearly only on the message size, whereas that of [GVW12] additionally depends on the circuit size and the number of queries.
- The ciphertext of our scheme is decomposable – namely, the data dependent component of the ciphertext $\mathsf{CT}_{\mathbf{x}}$ may be decomposed as $\left(\mathsf{CT}_1, \ldots, \mathsf{CT}_{|\mathbf{x}|}\right)$, where $\mathsf{CT}_i$ depends only on the bit $x_i$. Since our ciphertext enjoys both decomposability as well as succinctness of the online component, our scheme is very suitable for distributed data applications where the data to be encrypted is held by multiple, separated parties that share only a PRF seed. Existing schemes do not enjoy this feature to the best of our knowledge.

Security of our scheme is based on the standard Learning With Errors assumption (LWE) or its ring variant (Ring-LWE). To prove security of our scheme, we introduce a new proof technique, called *noisy functional encryption*, which may be of independent interest.

*IIT Delhi, India. Email: shweta.a@gmail.com. Work done (in part) while visiting IDC Herzliya, supported by the ERC under the EU's Seventh Framework Programme (FP/2007-2013) ERC Grant Agreement n. 307952. Work done (in part) while visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant CNS-1523467.

†Efi Arazi School of Computer Science, IDC Herzliya, Israel. Email: alon.rosen@idc.ac.il. Supported by ISF grant no. 1255/12, NSF-BSF Cyber Security and Privacy grant no. 2014/632, and by the ERC under the EU's Seventh Framework Programme (FP/2007-2013) ERC Grant Agreement n. 307952. Work done (in part) while visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant CNS-1523467.

# 1 Introduction

Functional encryption (FE) [SW05, SW] generalizes public key encryption to allow fine grained access control on encrypted data. In functional encryption, a secret key $\mathsf{SK}_g$ corresponds to a function $g$, and a ciphertext $\mathsf{CT}_{\mathbf{x}}$ corresponds to some input $\mathbf{x}$ from the domain of $g$. Given $\mathsf{SK}_g$ and $\mathsf{CT}_{\mathbf{x}}$, functionality posits that the user may run the decryption procedure to learn the value $g(\mathbf{x})$, while security guarantees that nothing about $\mathbf{x}$ beyond $g(\mathbf{x})$ can be learned.

Recent years have witnessed significant progress towards constructing functional encryption for advanced functionalities [BF01, Coc01, BW06, BW07, GPV08, CHKP10, ABB10, GPSW06, BSW07, KSW08, LOS+10, AFV11, Wat12, GVW13, GGH+13c, GGH+13b, GVW15]. 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) [GGH+13b] or on the existence of multilinear maps [GGHZ14]. 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 [AJ15, BV15, AJS15].

Unfortunately, all known candidate multi-linear map constructions [GGH13a, CLT13, GGH15] as well as some candidates of indistinguishability obfuscation have been recently broken [CHL+15, CGH+15, HJ15, CJL, CFL+, MSZ], rendering all constructions of general FE uninstantiable. 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 [GVW12, GKP+13, GVW15].

The notion of *bounded collusion functional encryption*, inspired from the domain of secure multiparty computation (MPC), was introduced by Gorbunov, Vaikuntanathan and Wee [GVW12]. 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 [DKXY02, CHH+07, GLW12].

**Structure versus Generality.** Gorbunov et al. [GVW12] 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 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 black box use of the recently developed Functional Encryption for Linear Functions [ALS16], denoted by $\mathsf{LinFE}$, and combines this with techniques developed in the context of Fully Homomorphic Encryption ($\mathsf{FHE}$)[1] [BV11b, BV11a]. Since $\mathsf{LinFE}$ and $\mathsf{FHE}$ can be based on LWE/Ring LWE, our construction inherits the same hardness assumption.

Our construction offers several advantages:

1. The ciphertext size of our scheme grows *quadratically* with the number of queries $q$, improving over the previous best construction [GVW12], where it grows as $O(q^4)$.

2. Our scheme is highly suitable for the *online-offline* model, which allows for an expensive offline encryption phase, where a majority of the computation is done before the data becomes available, followed by an efficient online phase, which is performed when the data becomes known. The online component of our ciphertext is *succinct*, i.e. depends *only on the message size*. By contrast, [GVW12] has an online component that depends on both the size of the circuit as well as the number of queries.

3. The ciphertext in our construction is *decomposable* – namely, the data dependent "online" component, denoted by $\mathsf{CT}_{\mathsf{online}}$, may be decomposed as $\left(\mathsf{CT}_1, \ldots, \mathsf{CT}_{|\mathbf{x}|}\right)$, where $\mathsf{CT}_i$ depends only on the bit $x_i$. Since our ciphertext enjoys both decomposability as well as succinctness of the online component, our scheme is very suitable for *distributed data applications* as discussed below. Existing schemes do not imply these to the best of our knowledge.

**Distributed Data Applications.** Our construction is very suitable for applications where data belonging to a single party is distributed across multiple locations. As an example, consider a large healthcare organization with $k > 1$ branches across multiple cities, which desires to store data from all centres in a central repository and compute functions on this. Ideally, we would like each centre to upload encryptions of its data to the server independently of other centres. Our construction provides a natural solution – all $k$ centres may share a PRF seed, using which centre $i$ can construct ciphertext component $\mathsf{CT}_i$ that depends only on its data $\mathbf{x}_i$ and upload this. The size of the ciphertext component $\mathsf{CT}_i$ is $\mathrm{poly}(\lambda, |\mathbf{x}_i|)$, where $\lambda$ is the security parameter, hence the upload operation consumes optimal bandwidth. The ciphertext components $\mathsf{CT}_1, \ldots, \mathsf{CT}_k$ can then be combined to produce a ciphertext $\mathsf{CT}_{\mathbf{x}}$ which may be decrypted by a function key $\mathsf{SK}_g$ to obtain $g(\mathbf{x})$. See Section 7 for more details.

As another example consider the following scenario described by Applebaum et al. [AIK14]: a computationally weak device is sent to the field in order to perform some expensive computation $C$ on some sensitive data $\mathbf{x}$. The computation is too expensive for the device to perform it on its own, but since the data is sensitive, the device cannot send the data back in the clear. As noted by [AIK14], garbled circuits provide the only feasible non-interactive solution to this question: the weak device can compute the garbled circuit for circuit $C$ in an offline phase before going to the field, and keep the garbled keys $\{K_i^0, K_i^1\}_{i \in [|\mathbf{x}|]}$ with itself. Once it learns the input $\mathbf{x}$, it performs the lightweight operation of selecting the appropriate keys $K_i^{x_i}$ and sends these back. Note that

[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.

this application crucially relies on succinctness of the "online" or data-dependent component of the encoding $K_i^{x_i}$, so that transmission cost is minimized. Indeed, reducing the online complexity of randomized encodings, for both single and multiple executions of the circuit, is an active area of research – see [LR14, HKK+14, AIKW15] and references therein. Also, note that *decomposability* of garbled circuits enables supporting multiple sensor devices: since the encoding $K_i^{x_i}$ depends only on bit $x_i$ and none other, device $i$ may be given keys $(K_i^0, K_i^1)$ before it is sent to the field, and the function may be computed on data collected by multiple devices.

Since functional encryption may be viewed as a generalization of randomized encodings, many applications of randomized encodings benefit immediately from the additional power offered by FE. For instance, by replacing the decomposable, online-succinct garbled circuit in the above example by a decomposable, online-succinct functional encryption scheme, we may: 1) choose the function to be computed on the data *after* the devices are sent to the field, 2) a given function may be computed on an unbounded number of ciphertexts, and 3) we may compute multiple (i.e. $q$) functions on a given set of data. It is evident that these capabilities are not enjoyed by the garbled circuit protocol described above. Also note that since computing $\mathsf{CT}_i$ is the online part of the operation and is efficient, it is within the capabilities of the weak device.

See Section 7 for more details about how our scheme achieves these properties and Appendix A for a discussion on why previous schemes could not.

### 1.2 Techniques

In this section, we describe our techniques. We begin by outlining the approach taken by previous work. [GVW12] begin with a single key FE scheme for circuits [SS10] and generalize this to a $q$ query scheme for $\mathsf{NC}_1$ circuits. This is the most sophisticated part of the construction, and leverages techniques from multiparty computation. Then, the $q$ query FE for $\mathsf{NC}_1$ 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 [AIK06].

Recently, Agrawal et al. [ALS16] observe that a different construction for bounded collusion FE can be obtained by replacing [SS10] and its generalisation to $q$ query FE for $\mathsf{NC}_1$, with an FE that computes inner products modulo some prime $p$. Such a scheme, which we denote by $\mathsf{LinFE}$, was constructed by [ALS16] 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 [GVW12] 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, $\mathsf{LinFE}$ may be used to compute degree 3 polynomials.

Thus, in [ALS16], 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 decomposable as the influence of bit $x_i$ cannot be contained to $\mathsf{CT}_i$, is not online-succinct as the entire ciphertext is data dependent, and its size grows as $O(q^6)$. See Appendix A for more details.

[2]We note that the FE scheme by Abdalla et al. [ABCP15] also supports linear functions but only over $\mathbb{Z}$, while [ALS16] requires an FE scheme that supports $\mathbb{Z}_p$. Also note the difference from Inner Product *orthogonality testing* schemes [KSW08, AFV11] which test whether $\langle \mathbf{x}, \mathbf{v} \rangle = 0 \mod p$ or not.

**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 $\mathsf{LinFE}$, resulting in a bounded collusion FE scheme for circuits that is decomposable, online-succinct as well as achieves ciphertext growth of $O(q^2)$.

In what follows, we focus 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 Appendix C.

Consider the ring LWE based symmetric key FHE scheme of [BV11b]. Recall that the ciphertext of this scheme, as in [Reg09], 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 $\mu$ 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 [BV11b] 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 $s$. [BV11b] 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, $\mathsf{LinFE}$ 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].

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

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 $c_i, c_j$, 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 $w$, we appear to avoid the quadratic blowup caused by linearization.

This intuition, while appealing, is very misleading. First off, 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. Thus, instead of using Linear FE, we will use a *noisy linear functional encryption scheme* which trades some correctness for security. This notion is formally defined in Section 3, and constructed in Section 5. 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. For example, the simulator must "program" the challenge ciphertext and function keys to decrypt correctly, but to do this, the simulator must satisfy $w^2$ decryption equations (corresponding to $w^2$ monomial keys) in only $w$ variables (corresponding to the terms $\{c_i s\}_{i \in [w]}$ in the ciphertext). To handle this, we introduce additional terms in the key so as to create sufficient degrees of freedom in the simulation. This requires care, since in the real world these terms are chosen independently but in the simulation, they must depend on the challenge ciphertext. Hence, arguing indistinguishability is not straightforward. The details of the proof are provided in Section 4. Generalizing quadratic to ternary polynomials requires additional tricks, since the online-succinct and decomposability properties of the scheme must be maintained. We describe how this is accomplished in Section 6.

**Towards Unbounded Query FE.** As discussed above, the ciphertext of our scheme can be divided into two components: a succinct data dependent component, which we refer to as $\mathsf{CT}_{\mathsf{online}}(\mathbf{x})$ and a non-succinct data independent component, which we call $\mathsf{CT}_{\mathsf{indpt}}(\mathbf{x})$. Aside from the aforementioned benefits of this structure, it is very intriguing to us that $\mathsf{CT}_{\mathsf{online}}(\mathbf{x})$ is *sufficient by itself* to propagate computation on $\mathbf{x}$ efficiently down an arithmetic circuit. To see this for the case of quadratic monomials, observe that the ciphertext $c_i = u_i \cdot s + 2 \cdot \mu_i + x_i$ for $i \in [w]$ along with $\mathbf{b} = \mathsf{LinFE.Enc}(s^2, c_1 s, \ldots, c_w s)$ is sufficient to compute a quadratic monomial

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

as described above. Since $\mathsf{LinFE}$ has succinct ciphertext [ALS16], $\mathbf{b}$ is succinct, hence $\mathsf{CT}_{\mathsf{online}} = (c_1, \ldots, c_w, \mathbf{b})$ is also succinct. Section 4 presents the scheme for quadratic polynomials, while Section 6 extends the construction to degree 4[4].

Thus, *functionally*, our scheme implies succinct FE for degree 4 polynomials: the encryptor can provide a succinct ciphertext for $\mathbf{x}$, namely $\mathsf{CT}_{\mathsf{online}}(\mathbf{x})$, the key generator can provide a key $\mathsf{SK}_P$ for polynomial $P$, and the decryptor can compute a ciphertext corresponding to $\mathsf{CT}(P(\mathbf{x}))$, apply to it the key $\mathsf{SK}_P$ and learn $P(\mathbf{x})$.

Unfortunately, the succinct scheme described above is insecure, and to achieve security, we must artificially add noise to the decryption equation computed by every key. This forces the encryptor to encode a separate noise term for each possible key, which results in a large data independent

[4]We do not extend to higher degree polynomials in this work, but conjecture that constant depth circuits can be supported.

component $\mathsf{CT}_{\mathsf{indpt}}$ and creates the dependence of the ciphertext size on the number of requested keys. This artificial addition of noise to the decryption equation is captured by the primitive of noisy functional encryption, which allows for a proof of security at the cost of succinctness.

Whether the noise can be added to the ciphertext in a more efficient way is a pressing open question, and one with important consequences, as recent work [LV16] has shown that functional encryption for constant degree polynomials implies indistinguishability obfuscation.

### 1.3 Organization of the paper

The paper is organized as follows. In Section 2, we describe preliminary definitions and notation used throughout the paper. In Section 3, we define the notion of noisy functional encryption which we rely on crucially for our construction. In Section 4, we describe a construction for FE for quadratic forms for the online-offline model, which relies on a a construction for noisy functional encryption for linear functions, secure against $q$ queries, provided in Section 5. In Section 6, we construct FE for degree 4 polynomials and in Section 7 we bootstrap it to bounded collusion FE for all circuits. In Appendix A, we discuss additional related work. We conclude in Section 8.

## 2 Preliminaries

In this section, we define the preliminaries we require for our constructions. Some additional preliminaries are provided in Appendix B.

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

A functional encryption scheme $\mathcal{F}$ for $\mathcal{G}$ consists of four algorithms $\mathcal{F} = (\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ 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 $(\mathsf{PK}, \mathsf{MSK})$.
- $\mathsf{FE.Keygen}(\mathsf{MSK}, g)$ is a p.p.t. algorithm that takes as input the master secret key $\mathsf{MSK}$ and a circuit $g \in \mathcal{G}_\lambda$ and outputs a corresponding secret key $\mathsf{SK}_g$.
- $\mathsf{FE.Enc}(\mathsf{PK}, x)$ is a p.p.t. algorithm that takes as input the master public key $\mathsf{PK}$ and an input message $x \in \mathcal{X}_\lambda$ and outputs a ciphertext $\mathsf{CT}$.
- $\mathsf{FE.Dec}(\mathsf{SK}_g, \mathsf{CT}_x)$ is a deterministic algorithm that takes as input the secret key $\mathsf{SK}_g$ and a ciphertext $\mathsf{CT}_x$ and outputs $g(x)$.

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

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

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

## 2.2 Indistinguishability-based Definition of Security

We define the standard indistinguishability based security definition for functional encryption.

**Definition 2.2.** A functional encryption scheme $\mathcal{F}$ for a function family $\mathcal{G}$ is secure in the adaptive indistinguishability game, denoted as $\mathsf{AD\text{-}IND}$ secure, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following experiment is negligible in the security parameter $\lambda$:

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

The *advantage* of $\mathcal{A}$ is the absolute value of the difference between its success probability and $1/2$. In the *selective* game, the adversary is required to declare the challenge messages in the very first step, without seeing the public key/ In the *semi-adaptive* game the adversary is required to declare the challenge messages after it sees the public parameters (before making any key queries).

## 2.3 Linear Functional Encryption

Our construction will make use of the linear functional encryption scheme, denoted by $\mathsf{LinFE}$, constructed by [ALS16]. Recall the functionality of $\mathsf{LinFE}$: the encryptor provides a ciphertext $\mathsf{CT}_{\mathbf{x}}$ for some vector $\mathbf{x} \in F_p^k$, the key generator provides a key $\mathsf{SK}_{\mathbf{v}}$ for some vector $\mathbf{v} \in F_p^k$, and the decryptor, given $\mathsf{CT}_{\mathbf{x}}$ and $\mathsf{SK}_{\mathbf{v}}$ can compute $\langle \mathbf{x}, \mathbf{v} \rangle \mod p$. We note that the FE scheme by Abdalla et al. [ABCP15] also supports linear functions but only over $\mathbb{Z}$, while [ALS16] requires an FE scheme that supports $\mathbb{Z}_p$.

The construction $\mathsf{LinFE}$ has several useful structural properties which will be very useful for our construction. These are:

1. **Decomposability:** Say that $\mathsf{LinFE}$ supports messages of length $k$. Then, the public key in $\mathsf{LinFE}$ may be interpreted as $\mathsf{PK} = (\mathsf{PK}_1, \ldots, \mathsf{PK}_k, \mathsf{PK}_{\mathsf{indpt}})$ and the ciphertext may be interpreted as $\mathsf{CT}_{\mathbf{x}} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_k, \mathsf{CT}_{\mathsf{indpt}})$. Here the data dependent components of the

public key and the ciphertext are $(\mathsf{PK}_1, \ldots, \mathsf{PK}_k)$ and $(\mathsf{CT}_1, \ldots, \mathsf{CT}_k)$ and the data independent components are $(\mathsf{PK}_{\mathsf{indpt}}, \mathsf{CT}_{\mathsf{indpt}})$. Additionally, we may write:

$$\mathsf{CT}_i = \mathcal{E}\left(\mathsf{PK}_i, x_i, r, \hat{r}_i\right) \forall i \in [k] \quad \text{and} \quad \mathsf{CT}_{\mathsf{indpt}} = \mathcal{E}\left(\mathsf{PK}_{\mathsf{indpt}}, r, \hat{r}\right)$$

Here, $\mathcal{E}$ is a deterministic encoding algorithm and $r$ is common randomness used by all components of the encryption. Apart from the common randomness $r$, each $\mathsf{CT}_i$ may additionally make use of independent randomness $\hat{r}_i$. The data dependent part of the ciphertext $(\mathsf{CT}_1, \ldots, \mathsf{CT}_k)$ is referred to as $\mathsf{CT}_{\mathsf{online}}$.

2. **Malleability:** The ciphertext components of $\mathsf{LinFE}$ are malleable so that if $\mathsf{CT}_i$ encodes $x_i$ then $\mathsf{CT}_i + x_j$ is a valid encoding for $x_i + x_j$ (as long as $x_i + x_j$ belong to the message space of $\mathcal{E}$). More formally, if $\mathsf{CT}_i = \mathcal{E}\left(\mathsf{PK}_i, x_i, r, \hat{r}_i\right)$ then $\mathcal{E}\left(\mathsf{PK}_i, x_i + x_j, r, \hat{r}_i\right) = \mathsf{CT}_i + x_j$.

3. **Additive homomorphism of ciphertext components:** The ciphertext components of $\mathsf{LinFE}$ enjoy additive homomorphism, i.e. if $\mathsf{CT}_i$ encodes $x_i$ and $\mathsf{CT}_j$ encodes $x_j$ then $v_i\ \mathsf{CT}_i + v_j\ \mathsf{CT}_j$ encodes $v_i x_i + v_j x_j$, as long as $v_i, v_j$ belong to the valid function space of $\mathsf{LinFE}$.

4. **Ciphertext Succinctness:** The ciphertext of $\mathsf{LinFE}$ is succinct, i.e. the size of $\mathsf{CT}(\mathbf{x})$ is $O(\mathrm{poly}(\lambda), k)$ where $\lambda$ is the security parameter and $k = |\mathbf{x}|$.

# 3 Noisy Functional Encryption

In this section, we define the notion of noisy functional encryption that we will use in our construction. Intuitively, noisy FE is like regular FE, except that the function value is recovered only up to some noise. This property turns out to be surprisingly useful, as we shall see in subsequent sections. We believe this primitive may be of independent interest, for example in differential privacy applications, where it is desirable to not reveal the function value exactly to safeguard privacy. We define the notion formally below.

An $(\epsilon, \delta, Q)$-Noisy functional encryption scheme $\mathcal{F}$ for some circuit family $\mathcal{G}$ consists of four algorithms $\mathcal{F} = (\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ as in the case of regular functional encryption. However, now we require the correctness condition to hold only up to some error $\delta > 0$, define security assuming that the number of queries asked is less than $Q$, and require that the challenge messages evaluated on all the keys requested by the adversary differ by at most $\epsilon$. These properties are described formally next.

**Definition 3.1** ($\delta$-Correctness)**.** Let $\delta > 0$. A functional encryption scheme $\mathcal{F}$ is $\delta$-correct if for all $g \in \mathcal{G}_\lambda$ and all $\mathbf{x} \in \mathcal{X}_\lambda$,

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

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

Security is defined in the standard indistinguishability setting and should require that for challenge messages $\mathbf{x}_0, \mathbf{x}_1 \in \mathcal{X}_\lambda$, if it holds that:

- For every queried key $g_i$, we have $|g_i(\mathbf{x}_0) - g_i(\mathbf{x}_1)| \leq \epsilon$.

- The total number of queries requested is less than $Q$, for some polynomial $Q$. We note that it may be that $Q >> |\mathbf{x}|$.

Then, we require that the adversary cannot distinguish $\mathsf{CT}(\mathbf{x}_0)$ from $\mathsf{CT}(\mathbf{x}_1)$ with non negligible advantage. Formally, we define the Noisy-IND game as follows.

**Definition 3.2** ($(\epsilon, Q)$ Noisy-AD-IND security.)**.** Let $\epsilon, \delta > 0$ such that $\epsilon/\delta < \text{negl}(\lambda)$. A $\delta$-correct functional encryption scheme $\mathcal{F}$ for a function family $\mathcal{G}$ is $(\epsilon, Q)$-secure in the noisy indistinguishability game if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following experiment is negligible in the security parameter $\lambda$:

1. **Public Key:** Challenger returns $\mathsf{PK}$ to the adversary.
2. **Pre-Challenge Key Queries:** $\mathcal{A}$ may adaptively request keys for any functions $g_1, \ldots, g_{\ell_1} \in \mathcal{G}$ for $i \in [\ell_1]$ and $\ell_1 \leq Q$. In response, $\mathcal{A}$ is given the corresponding keys $\mathsf{SK}(g_i)$.
3. **Challenge Ciphertext:** $\mathcal{A}(1^\lambda)$ outputs the challenges $(\mathbf{x}_0, \mathbf{x}_1) \in \mathcal{X}$ to the challenger. The challenger checks that $|g_i(\mathbf{x}_0) - g_i(\mathbf{x}_1)| \leq \epsilon$ for all functions $g_i$ requested so far. If not, it outputs $\perp$ and aborts. Otherwise, the challenger chooses a random bit $b$, and returns the ciphertext $\mathsf{CT}(\mathbf{x}_b)$.
4. **Post-Challenge Key Queries:** The adversary may request $\ell_2 \in [Q - \ell_1]$ keys for functions of its choice, subject to the same restrictions as above. In response, $\mathcal{A}$ is given the corresponding keys.
5. **Guess.** $\mathcal{A}$ outputs a bit $b'$, and succeeds if $b' = b$.

The *advantage* of $\mathcal{A}$ is the absolute value of the difference between its success probability and $1/2$.

# 4 $Q$ bounded FE for Quadratic forms

Since the intuition behind the construction has been discussed in Section 1, we proceed to the formal description. Our construction uses as a black box a noisy linear FE scheme, denoted by $(\epsilon, \delta, Q)$-NLinFE that computes noisy inner products modulo $p_1$ and supports $Q$ queries upto correctness fudge factor $\delta$ as long as the legitimate decryption values differ upto $\epsilon$. A construction for the same is provided in Section 5.

For the construction below supporting quadratic polynomials, we will require two prime moduli $p_0 < p_1$ where $p_0$ serves as the message space for the final quadratic scheme (think of $p_0 = 2$), and $p_1$ serves as the message space for the NLinFE scheme.

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

1. Invoke $\mathsf{NLinFE.Setup}(1^\lambda, 1^{w+2})$ to obtain NLinFE.PK and NLinFE.MSK.
2. Sample $\mathbf{u} \leftarrow R_{p_1}^w$.
3. For $1 \leq j \leq i \leq w$, compute $f_{ij}$ as follows:
   - Sample $\tilde{\mu}_{ij} \leftarrow \mathcal{D}_1$ for $1 \leq j \leq i \leq w$.

- Sample $t_i \in R_{p_1}$ for $i \in [0, w]$.
- Define

$$f_{ij} = u_i u_j t_0 - u_j t_i - u_i t_j + p_0 \cdot \tilde{\mu}_{ij}$$

Set $\mathbf{f} = (f_{ij})_{1 \le j \le i \le w} \in R^L$ where $L = |\{i, j : 1 \le j \le i \le w\}|$.

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

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

1. Sample $s_1 \xleftarrow{\mathrm{R}} R_{p_1}$ and $\boldsymbol{\mu} \leftarrow \mathcal{D}_0^w$, and compute "message carrier"

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

2. Let $\mathbf{b} = \mathsf{NLinFE.Enc}\ (s_1^2, c_1 s_1, \ldots, c_w s_1, 0)$.
3. Output $\mathsf{CT} = (\mathbf{c}, \mathbf{b})$

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

1. Compute

$$\mathbf{u_g} = \sum_{1 \le j \le i \le w} g_{ij}\, (u_i u_j, 0....0, -u_i, 0...0, -u_j, 0...0, f_{ij}) \in R_{p_1}^{w+2}.$$

2. Compute $\mathsf{SK_g} = \mathsf{NLinFE.KeyGen}(\mathbf{u_g})$ and output it.

$\mathsf{FE.Dec}(\mathsf{PK}, \mathsf{SK_g}, \mathsf{CT_x})$: On input the public parameters $\mathsf{PK}$, 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

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

and output it.

### 4.1 Correctness

We establish correctness of the above scheme.

Let $1 \le j \le i \le w$. By definition

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

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

$$x_i x_j + p_0 \cdot \mu_{ij} = c_i c_j - c_i u_j s_1 - c_j u_i s_1 + u_i u_j s_1^2 \tag{4.1}$$

We denote the noise added by the scheme $\mathsf{NLinFE}$ by $p_0 \cdot \rho_{ij}$. We note that the noise added by $\mathsf{NLinFE}$ can easily be chosen to be a multiple of $p_0$, since this is chosen by the encryptor of the $\mathsf{NLinFE}$ scheme. See Section 5 for more details.

By correctness of the linear scheme $\mathsf{NLinFE}$, we have that

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

**Analysis of Ciphertext Structure.** The ciphertext in the above scheme is comprised of:

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

It is evident that $\mathbf{c}$ is decomposable as well as succinct, since the $i^{th}$ component $c_i$ of $\mathbf{c}$ depends on the $i^{th}$ component $x_i$ of $\mathbf{x}$ and none other. Next, $\mathbf{b}$ is decomposable and online-succinct, since $\mathsf{NLinFE}$ is decomposable and online-succinct, as we shall see in Section 5.

Note that $\mathbf{c}$ is succinct, hence succinctness of the ciphertext relies on the succinctness of $\mathbf{b}$. Unfortunately, $\mathbf{b}$ is only online-succinct and not fully succinct (see Section 5 for details). Thus, a succinct version of $\mathsf{NLinFE}$ would lead to succinct ciphertext in the above construction.

### 4.2 Security

For ease of exposition, we will consider the special case where $\mathbf{x}_0$ and $\mathbf{x}_1$ differ only in a single coordinate, namely the $w^{th}$ one. This restriction can be removed with a careful change of basis operation in $R_{p_0}^w$, and is specified in Appendix G. For the special case in consideration, we note the following properties about admissible key requests and some implications that will be used crucially in the proof.

- $\mathbf{x}_0[i]\mathbf{x}_0[j] = \mathbf{x}_1[i]\mathbf{x}_1[j]$ for all $i, j \neq w$ by assumption and $\mathbf{x}_0[i]\mathbf{x}_0[w] = \mathbf{x}_1[i]\mathbf{x}_1[w]$ iff it holds that $\mathbf{x}_0[i]\mathbf{x}_0[w] = \mathbf{x}_1[i]\mathbf{x}_1[w] = 0$. We will refer to these monomials as admissible monomials. Note that since $\mathbf{x}_0[i] = \mathbf{x}_1[i]$ and $\mathbf{x}_0[w] \neq \mathbf{x}_1[w]$ by assumption, $\mathbf{x}_0[i]\mathbf{x}_0[w] = \mathbf{x}_1[i]\mathbf{x}_1[w] = 0$ can only occur when $\mathbf{x}_0[i] = \mathbf{x}_1[i] = 0$.
- It follows that an admissible quadratic polynomial query may only contain a monomial $x_i x_w$ for some $i$, if it holds that $\mathbf{x}_0[i]\mathbf{x}_0[w] = \mathbf{x}_1[i]\mathbf{x}_1[w] = 0$.
- Hence it suffices to ensure that the simulated keys for $x_i x_j$, where $i, j \neq w$ decrypt correctly and the simulated key for $x_i x_w$ decrypts correctly as long as $\mathbf{x}_0[i]\mathbf{x}_0[w] = \mathbf{x}_1[i]\mathbf{x}_1[w] = 0$. In particular note that the simulator is not required to provide a correctly working key for $x_i x_w$ when $\mathbf{x}_0[i]\mathbf{x}_0[w] \neq \mathbf{x}_1[i]\mathbf{x}_1[w]$.

- In the proof below, the keys and challenge ciphertext will be programmed so that decryption provides the honest answer for all admissible monomials. The simulator will be designed to construct keys even for non-admissible monomials but these will not decrypt correctly, since the simulator cannot know the value $\mathbf{x}_b[w]$ and hence cannot know the value of $\mathbf{x}_b[i]\mathbf{x}_b[w]$ unless $\mathbf{x}_b[i] = 0$. However, the adversary will never notice this, since he may not request these keys.

**Theorem 4.1.** *The construction in Section 4 achieves semi-adaptive indistinguishability based security defined in Section 2.2.*

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

**The Hybrids.** Our Hybrids are described below.

**Hybrid 0.** This is the real world with message $\mathbf{x}_0$. In hybrid 0, $f_{ij}$ is picked as follows:

1. Sample $t_0, \ldots, t_w \leftarrow R_{p_1}$.
2. Set $f_{ij} = u_i u_j t_0 - u_j t_i - u_i t_j + p_0 \cdot \tilde{\mu}_{ij} \mod p_1$. Here, $\tilde{\mu}_{ij} \leftarrow \mathcal{D}_1$ is noise chosen to flood $\mu_{ij}$.

**Hybrid 1.** In this hybrid, the only thing that is different is that the challenger picks $f_{ij}$ to depend on the challenge ciphertext. Specifically,

1. Sample $t_0, \ldots, t_w \leftarrow R_{p_1}$.
2. Set

$$f_{ij} = \left(x_i x_j - c_i c_j\right) - \left(u_i u_j t_0 - u_j t_i - u_i t_j - p_0 \cdot \tilde{\mu}_{ij}\right) \mod p_1$$

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

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

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

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

**Claim 4.2.** *Hybrid 0 is statistically indistinguishable from Hybrid 1.*

**Proof**. The only difference between Hybrid 0 and Hybrid 1 is in the way $f_{ij}$ is sampled.
In hybrid 0, we have:

1. Sample $t_0, \ldots, t_w \leftarrow R_{p_1}$.
2. Set $f_{ij} = u_i u_j t_0 - u_j t_i - u_i t_j + p_0 \cdot \tilde{\mu}_{ij}$.

In Hybrid 1, $f_{ij}$ is picked as follows:

1. Sample $t_0, \ldots, t_w \leftarrow R_{p_1}$.
2. Set $f_{ij} = \big((c_i - \hat{c}_i)(c_j - \hat{c}_j) - c_i c_j\big) - \big(u_i u_j t_0 - u_j t_i - u_i t_j + p_0 \cdot \tilde{\mu}_{ij}\big) \mod p_1$.

In Hybrid 1, we have:

$$\begin{aligned}
f_{ij} &= \big(x_i x_j - c_i c_j\big) - \big(u_i u_j t_0 - u_j t_i - u_i t_j - p_0 \cdot \tilde{\mu}_{ij}\big) \\
&= \big(u_i u_j s_1^2 - u_i c_j s_1 - u_j c_i s_1 - p_0 \cdot \mu_{ij}\big) - \big(u_i u_j t_0 - u_j t_i - u_i t_j - p_0 \cdot \tilde{\mu}_{ij}\big) \qquad \text{by equation 2.2} \\
&= u_i u_j (s_1^2 - t_0) - u_i (c_j s_1 - t_j) - u_j (c_i s_1 - t_i) - p_0 \cdot \mu_{ij} + p_0 \cdot \tilde{\mu}_{ij} \\
&= u_i u_j t'_0 - u_i t'_j - u_j t'_i - p_0 \cdot \mu_{ij} + p_0 \cdot \tilde{\mu}_{ij} \\
&\approx u_i u_j t'_0 - u_i t'_j - u_j t'_i + p_0 \cdot \tilde{\mu}_{ij} \quad \text{as long as } \tilde{\mu}_{ij} \text{ floods } \mu_{ij}.
\end{aligned}$$

Here, we define $t'_0 = s_1^2 - t_0$ and $t'_i = c_i s_1 - t_i$ for $i \in [w]$. Note that $t'_0, \ldots, t'_w$ are i.i.d due to addition of i.i.d $t_0, \ldots, t_w$. Thus, we get that the distributions of $f_{ij}$ in Hybrid 0 and Hybrid 1 are statistically indistinguishable. Since the only thing that changed between Hybrid 0 and Hybrid 1 is the manner of sampling $f_{ij}$, we get that Hybrid 0 and Hybrid 1 are statistically indistinguishable. □

**Claim 4.3.** *Assuming adaptive IND security of the noisy linear FE scheme* NLinFE*, Hybrid 1 and Hybrid 2 are indistinguishable.*

**Proof**. Given an adversary $\mathcal{B}$ who distinguishes between Hybrid 0 and 1, we construct an adversary $\mathcal{A}$ who breaks the adaptive IND security of the noisy linear FE scheme NLinFE.

$\mathcal{A}$ runs as follows:

1. The challenger of the NLinFE scheme provides NLinFE.PK to $\mathcal{A}$. $\mathcal{A}$ picks $\mathbf{u} \in R_{p_1}^w$, and sends $\mathsf{PK} = (\mathsf{NLinFE.PK}, \mathbf{u})$ to $\mathcal{B}$.
2. $\mathcal{A}$ picks $s_1 \leftarrow R_{p_1}$, and $\mathbf{t} \leftarrow R_{p_1}^{w+1}$.
3. $\mathcal{A}$ computes
$$c_i = u_i s_1 + p_0 \cdot \mu_i + \mathbf{x}_0[i] \quad \forall i \in [w].$$
4. When $\mathcal{B}$ requests a key for monomial $x_i x_j$, $\mathcal{A}$ does the following:
   - $\mathcal{A}$ computes $\mathbf{t}'$ as:
   $$t'_0 = s_1^2 - t_0, \quad \{t'_i = c_i s_1 - t_i\}_{i \in [w]}$$
   - $\mathcal{A}$ computes
   $$f_{ij} = u_i u_j t'_0 - u_j t'_i - u_i t'_j + p_0 \cdot \tilde{\mu}_{ij}$$

- $\mathcal{A}$ parses this as a linear key with coefficients

$$(u_i u_j, 0....0, u_i, 0...0, u_j, 0...0, f_{ij}) \in R_{p_1}^{w+2}.$$

  Here, $u_i$ appears at the $j^{th}$ position, $u_j$ appears at the $i^{th}$ position.

- $\mathcal{A}$ forwards this key request to the linear FE challenger and returns its response $\mathsf{NLinFE.SK}_{ij}$ to $\mathcal{B}$.

5. $\mathcal{B}$ outputs challenges $\mathbf{x}_0, \mathbf{x}_1 \in R_{p_0}^w$. $\mathcal{A}$ chooses challenge messages $\mathbf{z}_0, \mathbf{z}_1$ as:

$$\begin{aligned} \mathbf{z}_0 &= \left(s_1^2, c_1 s_1, \ldots, c_w s_1,\ 0\right) &\in R_{p_1}^{w+2} \\ \mathbf{z}_1 &= (t_0,\ t_1,\quad \ldots\quad ,t_w,\ 1) &\in R_{p_1}^{w+2} \end{aligned}$$

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

6. When the linear FE challenger sends challenge $\mathsf{NLinFE.CT}(\mathbf{z}_b)$ to $\mathcal{A}$, it sends $\mathsf{CT} = \left(\mathsf{NLinFE.CT}(\mathbf{z}_b), \mathbf{c}\right)$ to $\mathcal{B}$. Recall that it computed $\mathbf{c}$ itself in Step 3.

7. $\mathcal{B}$ may request more keys which are answered as before.

8. When $\mathcal{B}$ outputs a guess bit $b$, $\mathcal{A}$ forwards this guess to the linear FE challenger.

Now, we argue that for all the requested keys, the difference between the decryption values on the challenge messages $\mathbf{z}_0$ and $\mathbf{z}_1$ is very small. By setting the parameter $\epsilon$ of the $(\epsilon, \delta, Q)$-$\mathsf{NLinFE}$ scheme to be an upper bound on this difference, we obtain by the security of the noisy linear FE scheme that $\mathsf{NLinFE.CT}(\mathbf{z}_0)$ and $\mathsf{NLinFE.CT}(\mathbf{z}_1)$ are indistinguishable. We proceed to show this formally. We will let $Q = w^2$. Consider a key request for monomial $x_i x_j$.

In Hybrid 1, the function for the monomial $x_i x_j$ evaluated on $\mathbf{z}_0$ yields:

$$u_i u_j s_1^2 - u_i c_j s_1 - u_j c_i s_1$$

In Hybrid 2 the function for the monomial $x_i x_j$ evaluated on $\mathbf{z}_1$ yields:

$$\begin{aligned} &u_i u_j t_0 - u_i t_j - u_j t_i + f_{ij} \\ &= u_i u_j t_0 - u_i t_j - u_j t_i + \left(u_i u_j (s_1^2 - t_0) - u_j (c_i s_1 - t_i) - u_i (c_i s_1 - t_j) + p_0 \cdot (\tilde{\mu}_{ij} - \mu_{ij})\right) \\ &\approx u_i u_j s_1^2 - u_i c_j s_1 - u_j c_i s_1 + p_0 \cdot \tilde{\mu}_{ij} \quad \text{since } \tilde{\mu}_{ij} \text{ floods } \mu_{ij}. \end{aligned}$$

Thus, the decryption values in both worlds are approximately the same upto an additive factor of $p_0 \cdot \tilde{\mu}_{ij}$. We set $\epsilon$ in the $\mathsf{NLinFE}$ scheme to be larger than $p_0 \cdot \tilde{\mu}_{ij}$, we obtain by the guarantee of Noisy Linear FE that Hybrids 1 and 2 are indistinguishable[5]. □

**Claim 4.4.** *Assume Regev public key encryption is semantically secure. Then, Hybrid 2 is indistinguishable from Hybrid 3.*

[5]Recall that in the security game of noisy linear FE, the challenge messages must only evaluate to approximately same values for all functions that are queried. The difference between the decryption values is allowed to differ up to $\epsilon$ which is a parameter to the scheme.

**Proof**. We recall Regev public key encryption. We set $(\mathsf{PK}, \mathsf{SK}) = (\mathbf{u}, s_1)$ as the public and secret key. The ciphertext for message $\mathbf{x}$ is computed $\mathbf{c} = \mathbf{u} \cdot s_1 + p_0 \cdot \boldsymbol{\mu} + \mathbf{x}$, where $\boldsymbol{\mu}$ is suitably chosen noise.

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

- Run $\mathsf{NLinFE.Setup}$ to obtain $\mathsf{NLinFE.PK}$ and $\mathsf{NLinFE.MSK}$. Return $\mathsf{PK} = (\mathsf{NLinFE.PK}, \mathbf{u})$ to $\mathcal{B}$.
- When $\mathcal{B}$ requests a key, construct it honestly as in Hybrid 0.
- When $\mathcal{B}$ outputs challenges $\mathbf{x}_0, \mathbf{x}_1$, $\mathcal{A}$ forwards these to the PKE challenger.
- $\mathcal{A}$ receives $\mathbf{c}$ where $\mathbf{c} = \mathbf{u} \cdot s_1 + p_0 \cdot \boldsymbol{\mu} + \mathbf{x}_b$ for a random bit $b$.
- $\mathcal{A}$ computes $\mathbf{b} = \mathsf{NLinFE.CT}(t_0, \ldots, t_w, 1)$ and returns $(\mathbf{c}, \mathbf{b})$ to $\mathcal{B}$.
- $\mathcal{B}$ may request more keys 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. Also, the advantage of the attacker $\mathcal{B}$ in distinguishing Hybrids 2 and 3 translates directly to the advantage of the attacker $\mathcal{A}$ in breaking the semantic security of Regev public key encryption. Hence the claim follows. □

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

# 5 Noisy Functional Encryption for Linear Functions

In this section, we construct a noisy functional encryption scheme for linear functions which supports $Q$ queries for a fixed polynomial $Q$. Security posits that an adversary cannot distinguish between encryptions of $\mathbf{z}_0$ and $\mathbf{z}_1$ as long as long as $|\mathbf{g}_i(\mathbf{z}_0) - \mathbf{g}_i(\mathbf{z}_1)| \leq \epsilon$ for every key $\mathbf{g}_i$ requested, as long as the number of requested keys is less than $Q$. We emphasize that $Q$ can be greater than the dimension of the message/key vectors, namely $w$ – indeed this is the case we will be interested in.

To support $Q > w$ queries so as to achieve the security definition stated above, we artificially add noise to decryption, so that any two messages whose decryption under a key is equal upto $\epsilon$ noise, will decrypt to only approximately correct values but will have indistinguishable ciphertexts.

For ease of notation, our description below assumes a stateful keygen. We get rid of this restriction using standard tricks as described in Appendix E. We will crucially utilize malleability of the underlying $\mathsf{LinFE}$ scheme, see Section 2.3 for more details.

$\mathsf{FE.Setup}(1^\lambda, 1^Q, 1^w, p_1)$: On input a security parameter $\lambda$, a parameter $w$ denoting the length of the function and message vectors, a parameter $Q$ denoting the number of queries supported

and a modulus $p_1$ denoting the space of the message and function vectors, set $(\mathsf{PK}, \mathsf{MSK}) = \mathsf{LinFE.Setup}(1^{w+1+Q})$.

$\mathsf{FE.KeyGen}(\mathsf{MSK}, \mathbf{g}, i)$: On input $\mathsf{MSK}$, a function vector $\mathbf{g} \in \mathbb{Z}_{p_1}^w$ and an index $i \in [Q]$ denoting query number, do:

1. Sample $\gamma_i \leftarrow \mathcal{D}_1$, where $\mathcal{D}_1$ is a discrete Gaussian of width large enough to flood $\epsilon$.
2. Output $\mathsf{SK}_{\mathbf{g}} = \mathsf{LinFE.KeyGen}(\mathbf{g} \| \gamma_i \| \mathbf{e}_i)$ where $\mathbf{e}_i \in \mathbb{Z}^Q$ is the $i^{th}$ unit vector. Note that the $\mathsf{LinFE}$ key explicitly contains (in the clear) the vector $(\mathbf{g} \| \gamma_i \| \mathbf{e}_i)$.

$\mathsf{FE.Enc}(\mathsf{PK}, \mathbf{z})$: On input public key $\mathsf{PK}$, a message vector $\mathbf{z} \in \mathbb{Z}_{p_1}^w$, do:

1. Sample $\boldsymbol{\delta} \leftarrow \mathcal{D}_2^Q$ and $\mu \leftarrow \mathcal{D}$ subject to the following constraints:
   - $\mathcal{D}_2$ is a discrete Gaussian of width large enough to flood $\mathcal{D}_1$. Thus, it will hold that $\delta_i + \gamma_i \approx \delta_i$, where $\gamma_i$ is chosen by keygen.
   - $\mathcal{D}$ has width large enough so that $\mu$ is distributed indistinguishably from $\mu + 1$.
2. Let $\mathsf{CT}_{\mathbf{z}} = \mathsf{LinFE.Enc}(\mathbf{z} \| \mu \| \boldsymbol{\delta})$

$\mathsf{FE.Dec}(\mathsf{SK}_{\mathbf{g}_i}, \mathsf{CT}_{\mathbf{z}})$: On input a secret key $\mathsf{SK}_{\mathbf{g}_i}$ for function $\mathbf{g}_i$, and a ciphertext $\mathsf{CT}_{\mathbf{z}}$:

1. Compute $\mathsf{LinFE.Dec}(\mathsf{CT}_{\mathbf{z}}, \mathsf{SK}_{\mathbf{g}_i})$ to recover $\mathbf{g}_i^\intercal \mathbf{z} + \mu \cdot \gamma_i + \delta_i$.

Appropximate correctness is evident since we recovered the correct value upto noise $\mu \cdot \gamma_i + \delta_i$.

*Note.* We remark that the keygen algorithm can be made stateless by using standard tricks, as in [GVW12, Section 6]. We refer the reader to Appendix E for more details.

**Analysis of Ciphertext Structure.** We note that the ciphertext of the $\mathsf{NLinFE}$ scheme is a $\mathsf{LinFE}$ ciphertext with message $(\mathbf{z} \| \mu \| \boldsymbol{\delta})$, where $(\mu \| \boldsymbol{\delta})$ are noise terms. Since $\mathsf{LinFE}$ ciphertext is decomposable, as discussed in Section 2, we have that the ciphertext can be split into a pair $(\mathsf{CT}_1, \mathsf{CT}_2)$ where $\mathsf{CT}_1$ encodes $\mathbf{z}$ and $\mathsf{CT}_2$ encodes $(\mu \| \boldsymbol{\delta})$. Note that here, only $\mathsf{CT}_1$ depends on the data, and is the online part of the ciphertext, while $\mathsf{CT}_2$ is the offline part. Since the ciphertext of $\mathsf{LinFE}$ is succinct, we establish that the online part of the $\mathsf{NLinFE}$ ciphertext is succinct.

### 5.1 Security.

Next, we argue that the above scheme is secure. We have that for every key query $\mathbf{g}_i$, $i \in [Q]$, it holds that $\mathbf{g}_i^\intercal (\mathbf{z}_0 - \mathbf{z}_1) = \epsilon_i$. Then, we claim:

**Theorem 5.1.** *The noisy linear FE scheme* $\mathsf{NLinFE}$ *satisfies* $(\epsilon, Q)$ $\mathsf{Noisy\text{-}AD\text{-}IND}$ *security as per Definiton 3.2.*

**Proof**. We argue via a sequence of hybrids.

**Hybrid 0.** This is the real world with message $\mathbf{z}_0$. The challenge CT encrypts $\mathbf{y}_0 = (\mathbf{z}_0 \| \mu \| \boldsymbol{\delta})$.

**Hybrid 1.** In this world, we generate the challenge CT for the message $\mathbf{y}_0 = (\mathbf{z}_0 \| \mu \| \hat{\boldsymbol{\delta}} + \boldsymbol{\delta})$ where $\boldsymbol{\delta}$ floods $\hat{\boldsymbol{\delta}}$. For the keys, we set $\boldsymbol{\gamma} = \hat{\boldsymbol{\delta}}$.

**Hybrid 2.** In this world, the noise in the $Q$ keys changes to $\boldsymbol{\gamma} = \hat{\boldsymbol{\delta}} + \boldsymbol{\epsilon}$.

**Hybrid 3.** In this world, we change the message in the challenge CT to $\mathbf{y}_1 = (\mathbf{z}_1 \| \mu + 1 \| \boldsymbol{\delta})$.

**Hybrid 4.** In this world, we rewrite the message in the challenge CT as $\mathbf{y}_1 = (\mathbf{z}_1 \| \mu \| \boldsymbol{\delta})$. This is the real world with message $\mathbf{z}_1$.

**Indistinguishability of Hybrids.** It is evident that Hybrids 0 and 1 are statistically indistinguishable, since $\boldsymbol{\delta}$ floods $\hat{\boldsymbol{\delta}}$. By the same argument, Hybrid 1 and 2 are statistically indistinguishable since $\hat{\boldsymbol{\delta}}$ floods $\boldsymbol{\epsilon}$ and Hybrid 3 and Hybrid 4 are statistically indistinguishable since $\mu$ is distributed indistinguishably from $\mu + 1$. The chief transition that must be argued is that between Hybrids 2 and 3, which we argue now.

**Claim 5.2.** *If the exact linear scheme is* AD-IND *secure, then Hybrids 2 and 3 are indistinguishable.*

**Proof**. Given an adversary $\mathcal{A}$ who can distinguish between Hybrids 2 and 3, we will construct an adversary $\mathcal{B}$ who will break the security of the exact linear scheme. $\mathcal{B}$ plays the AD-IND game with the LinFE challenger, denoted by $\mathcal{C}$.

1. The LinFE challenger $\mathcal{C}$ outputs the public key PK, which $\mathcal{B}$ forwards to $\mathcal{A}$.
2. $\mathcal{A}$ requests a key $\mathbf{g}_i$ for $i \in [\ell_1]$, where $\ell_1 \leq Q$. $\mathcal{B}$ chooses $\gamma_i$ as in the real world and requests a key for $\hat{\mathbf{g}}_i = (\mathbf{g} \| \gamma_i \| \mathbf{e}_i)$ to the LinFE challenger $\mathcal{C}$. The challenger returns $\mathsf{LinFE.SK}(\hat{\mathbf{g}}_i)$ which $\mathcal{B}$ gives $\mathcal{A}$.
3. $\mathcal{A}$ outputs two challenges $\mathbf{z}_0, \mathbf{z}_1 \in \mathbb{Z}_{p_1}^w$. $\mathcal{B}$ checks that $\mathbf{g}_i^\mathsf{T}(\mathbf{z}_0 - \mathbf{z}_1) = \epsilon_i \leq \epsilon$ for all queries $\mathbf{g}_i$ requested so far. If this condition does not hold, output $\perp$ and abort.
4. $\mathcal{B}$ chooses $\hat{\delta}_i = \gamma_i - \epsilon_i$ for $i \in [\ell_1]$. The remaining $\hat{\delta}_{\ell_1+1} \ldots \hat{\delta}_Q$ it chooses as in the real world. Next, it constructs $\hat{\mathbf{y}}_0 = (\mathbf{z}_0 \| \mu \| \hat{\boldsymbol{\delta}})$ and $\hat{\mathbf{y}}_1 = (\mathbf{z}_1 \| \mu + 1 \| \mathbf{0})$ and returns these to the LinFE challenger $\mathcal{C}$ as the challenge messages.
5. $\mathcal{A}$ may request more keys $\mathbf{g}_i$ so that $\mathbf{g}_i^\mathsf{T}(\mathbf{z}_0 - \mathbf{z}_1) = \epsilon_i$. $\mathcal{B}$ chooses $\gamma_i = \hat{\delta}_i + \epsilon_i$ and requests a key for $\hat{\mathbf{g}}_i = (\mathbf{g} \| \gamma_i \| \mathbf{e}_i)$ to the LinFE challenger. The challenger returns $\mathsf{LinFE.SK}(\hat{\mathbf{g}}_i)$, which $\mathcal{B}$ gives $\mathcal{A}$.
6. $\mathcal{C}$ outputs the challenge ciphertext $\mathsf{CT}(\hat{\mathbf{y}}_b)$. $\mathcal{B}$ adds noise $\boldsymbol{\delta}$ (permitted due to ciphertext malleability, see Section 2.3) to the last $Q$ coordinates and returns this to $\mathcal{A}$ as the challenge CT. Thus, $\mathcal{A}$ sees an encryption of either $\mathbf{y}_0 = (\mathbf{z}_0 \| \mu \| \hat{\boldsymbol{\delta}} + \boldsymbol{\delta})$ or $\mathbf{y}_1 = (\mathbf{z}_1 \| \mu + 1 \| \boldsymbol{\delta})$.
7. When $\mathcal{A}$ outputs a guess for bit $b$, $\mathcal{B}$ outputs the same.

Observe that the query $\hat{\mathbf{g}}_i$ is an admissible query for the LinFE challenger because:

$$\begin{aligned}
\hat{\mathbf{g}}_i^\top \hat{\mathbf{y}}_0 &= \mathbf{g}_i^\top \mathbf{z}_0 + \mu \cdot \gamma_i + \hat{\delta}_i \\
\hat{\mathbf{g}}_i^\top \hat{\mathbf{y}}_1 &= \mathbf{g}_i^\top \mathbf{z}_1 + \mu \cdot \gamma_i + \gamma_i \\
&= \mathbf{g}_i^\top \mathbf{z}_0 - \epsilon_i + \mu \cdot \gamma_i + \hat{\delta}_i + \epsilon_i \\
&= \mathbf{g}_i^\top \mathbf{z}_0 + \mu \cdot \gamma_i + \hat{\delta}_i \\
&= \hat{\mathbf{g}}_i^\top \hat{\mathbf{y}}_0
\end{aligned}$$

If the LinFE challenger $\mathcal{C}$ returns an encryption of $\hat{\mathbf{y}}_0$, then $\mathcal{A}$ sees an encryption of $\mathbf{y}_0 = (\mathbf{z}_0 \| \mu \| \hat{\boldsymbol{\delta}} + \boldsymbol{\delta})$, otherwise it sees an encryption of $\mathbf{y}_1 = (\mathbf{z}_1 \| \mu + 1 \| \boldsymbol{\delta})$. In the former case we obtain the distribution of Hybrid 2, in the latter case of Hybrid 3. □

Hence, we argued that for $Q$ (for $Q > w$) queries, our noisy linear FE scheme satisfies $(\epsilon, Q)$ Noisy-AD-IND security as per Definiton 3.2. □

# 6 Functional Encryption for Degree 4 Polynomials

In this section, we extend our construction for quadratic polynomials to degree 4 polynomials. The main observation in this section is that the quadratic scheme presented in Section 4 enables the decryptor to compute level 2 ciphertexts $\mathbf{c}^2$ encoding degree 2 monomials that have the *same* structure as level 1 ciphertexts $\mathbf{c}^1$ that encode the message. Hence, the quadratic scheme can be applied again to level 2 ciphertexts resulting in level 4 ciphertexts that encode monomials of degree 4.

We show that the propagation of computation down the circuit is efficient, so that given $w$ messages $x_1, \ldots, x_w$, the encryptor may provide $O(w)$ ciphertext components which enable the decryptor to construct level 4 ciphertexts $\mathbf{c}^4$ of degree 4 monomials in $x_1, \ldots, x_w$. However, as discussed in Section 4, releasing a key to decrypt these ciphertexts directly is insecure, and the key generator must force addition of a fresh noise term (provided by the encryptor) to each decryption equation. Thus, to support $q$ function keys, the encryptor must provide $q$ noise terms in the encryption. This results in a non-succinct data-independent ciphertext component of size $O(w^4)$. Details follow.

**Technical Overview.** Let us recall the main idea in the quadratic FE construction. We have:

$$\begin{aligned}
c_i^1 &= u_i^1 \cdot s_1 + 2 \cdot \mu_i^1 + x_i \quad \in R_{p_1} \\
c_j^1 &= u_j^1 \cdot s_1 + 2 \cdot \mu_j^1 + x_j \quad \in R_{p_1}
\end{aligned}$$

Above the superscript denotes the level of the circuit and subscript denotes the component within the ciphertext, thus $c_i^1$ denotes the $i^{th}$ component of level 1 ciphertext. We call $c_i^1$ as the "Regev encryption" of message $x_i$, using public label $u_i^1$, secret $s_1$ and noise $2 \cdot \mu_i^1$. In what follows, the precise value of the label and noise in a Regev encryption will be unimportant, so to ease notation we let:

$$\boxed{x_i} = c_i^1, \ , \quad \mathsf{L}\,\boxed{x_i} = u_i^1, \quad \mathsf{N}\,\boxed{x_i} = 2 \cdot \mu_i^1$$

Now, we may write:

$$c_i^1 = \boxed{x_i} = \mathsf{L}\boxed{x_i} \cdot s_1 + \mathsf{N}\boxed{x_i} + x_i \tag{6.1}$$

Note that by additive homomorphism of the components of a given ciphertext, it will hold that

$$\boxed{x_i + x_j} = \boxed{x_i} + \boxed{x_j}$$

As mentioned previously, the decryption equation can be written as

$$x_i x_j \approx c_i c_j + u_i u_j (s_1^2) - u_j (c_i s_1) - u_i (c_j s_1) \quad \in R_{p_1} \tag{6.2}$$

Now, if we use the Linear FE scheme so that:

$$\begin{aligned} \mathsf{CT} &= \mathsf{LinFE.Enc}(s_1^2, c_1 s_1, \ldots, c_w s_1) \\ \mathsf{SK}_{ij} &= \mathsf{LinFE.KeyGen}(u_i u_j, -0-, u_i, -0-, u_j, -0-) \end{aligned}$$

then, $\mathsf{LinFE.Dec}(\mathsf{SK}_{ij}, \mathsf{CT})$ yields the term $u_i u_j (s_1^2) - u_j (c_i s_1) - u_i (c_j s_1)$ by correctness. Since the term $c_i c_j$ can be computed directly by the decryptor, she may recover the right hand side of Equation 6.2, which lets her obtain $x_i x_j$ as desired. Since the scheme $\mathsf{LinFE}$ enables computing linear functions on encrypted messages, it suffices to compute $\mathsf{LinFE}$ ciphertext components that encode monomials, which may be combined to produce arbitrary polynomials.

We are ready to describe the extension to degree 4 polynomials. Let $y_k$ denote level 2 *plaintexts*, namely $y_k = x_i x_j$ for some $i, j \leq w$. Using our new notation and invoking additive homomorphism of $\mathsf{LinFE}$ ciphertext components as well as malleability, our level 2 ciphertext may be written as:

$$c_k^2 = \boxed{y_k} = c_i^1 c_j^1 + \boxed{u_i^1 u_j^1 (s_1^2) - u_j^1 (c_i^1 s_1) - u_i^1 (c_j^1 s_1)} \quad \in R_{p_2} \tag{6.3}$$

$$= c_i^1 c_j^1 + u_i^1 u_j^1 \boxed{s_1^2} - u_j^1 \boxed{c_i^1 s_1} - u_i^1 \boxed{c_j^1 s_1} \tag{6.4}$$

In the above, note that $c_i^1 c_j^1 \in R_{p_1}$ and this term is a message added to the ciphertext $\boxed{u_i^1 u_j^1 (s_1^2) - u_j^1 (c_i^1 s_1) - u_i^1 (c_j^1 s_1)}$. Additionally, by additive homomorphism of the ciphertext components (see Section 2.3),

$$u_k^2 \stackrel{\text{def}}{=} \mathsf{L}\boxed{y_k} = u_i^1 u_j^1 \mathsf{L}\boxed{s_1^2} - u_j^1 \mathsf{L}\boxed{c_i^1 s_1} - u_i^1 \mathsf{L}\boxed{c_j^1 s_1} \quad \in R_{p_2} \tag{6.5}$$

$$\mathsf{N}\boxed{y_k} = u_i^1 u_j^1 \mathsf{N}\boxed{s_1^2} - u_j^1 \mathsf{N}\boxed{c_i^1 s_1} - u_i^1 \mathsf{N}\boxed{c_j^1 s_1} \quad \in R_{p_2} \tag{6.6}$$

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 $R_{p_2}$. Thus, we have established:

**Lemma 6.1.** *Level 2 ciphertext $c_k^2$ may be computed using level 1 ciphertexts $\boxed{x_i} = c_i^1$ as well as advice $\boxed{s_1^2}$ and $\boxed{c_i^1 s_1}$.*

**Preserving ciphertext structure.** The key point is that our level 2 ciphertext has the exact same structure as a level 1 ciphertext, namely it is a Regev ciphertext using some secret $s_2$, some label and noise as computed in equations 6.5. Thus, for $k \in [w^2]$, we may write

$$c_k^2 = \mathsf{L}\boxed{y_k} \cdot s_2 + \mathsf{N}\boxed{y_k} + y_k \tag{6.7}$$

To extend to degree 4 polynomials, we may now apply the quadratic method to level 2 ciphertexts as follows. Let $t \in [w^4]$ index a monomial of degree 4, which is the product of two degree 2 monomials $y_k$ and $y_\ell$. Then,

$$c_t^4 = \boxed{y_k y_\ell} = c_k^2 c_\ell^2 + \boxed{u_k^2 u_\ell^2 (s_2^2) - u_\ell^2 (c_k^2 s_2) - u_k^2 (c_\ell^2 s_2)} \quad \in R_{p_3} \tag{6.8}$$

$$= c_k^2 c_\ell^2 + u_k^2 u_\ell^2 \boxed{s_2^2} - u_\ell^2 \boxed{c_k^2 s_2} - u_k^2 \boxed{c_\ell^2 s_2} \tag{6.9}$$

By correctness of quadratic FE, the decryptor can compute $c_k^2$ correctly for $k \in [w^2]$. By the above equation, it suffices for the encryptor to provide $\boxed{s_2^2}$ and $\boxed{c_k^2 s_2}$ for $k \in [w^2]$. However, the encryptor cannot provide the latter term directly and preserve succinctness of the online component of the ciphertext. This is because level 2 ciphertext $c_k^2$ contains the quadratic term $c_i^1 c_j^1$ as shown by Equation 6.3.

To get around this, note that:

$$\begin{aligned}
\boxed{c_k^2 s_2} &= \boxed{\left(c_i^1 c_j^1 + u_i^1 u_j^1 \boxed{s_1^2} - u_j^1 \boxed{c_i^1 s_1} - u_i^1 \boxed{c_j^1 s_1}\right) \cdot s_2} \\
&= \boxed{c_i^1 c_j^1 s_2} + \boxed{u_i^1 u_j^1 \boxed{s_1^2} s_2} - \boxed{u_j^1 \boxed{c_i^1 s_1} s_2} - \boxed{u_i^1 \boxed{c_j^1 s_1} s_2} \\
&= \boxed{c_i^1 c_j^1 s_2} + u_i^1 u_j^1 \boxed{\boxed{s_1^2} s_2} - u_j^1 \boxed{\boxed{c_i^1 s_1} s_2} - u_i^1 \boxed{\boxed{c_j^1 s_1} s_2}
\end{aligned}$$

Now, the encryptor may provide $\boxed{\boxed{s_1^2} s_2}$, $\boxed{\boxed{c_i^1 s_1} s_2}$ for $i \in [w]$, while maintaining succinctness of the ciphertext, since these terms do not grow quadratically, i.e. these depend only on the depth not the width of the circuit. But the term $\boxed{c_i^1 c_j^1 s_2}$ may not be provided by the encryptor directly, since it would negate succinctness of the ciphertext. Hence, the encryptor is required to provide efficient advice to compute $\boxed{c_i^1 c_j^1 s_2}$.

Fortunately, $c_i^1 c_j^1 s_2$ can be seen as a quadratic term in $z_i = c_i^1$ and $z_j = c_j^1 \cdot s_2$, and to use the quadratic scheme to compute an encryption of the product, we have by Lemma 6.1 that we are required to compute $\boxed{z_i}$, $\boxed{z_j}$ (under some secret $s_3$, say) along with $\boxed{\boxed{z_i}\, s_3}$ and $\boxed{\boxed{z_j}\, s_3}$, which translates to $\boxed{c_i^1}$, $\boxed{c_j^1 \cdot s_2}$ and advice $\boxed{\boxed{c_i^1} \cdot s_3}$, $\boxed{\boxed{c_i^1 \cdot s_2} \cdot s_3}$.

We let

$$d_i \stackrel{\text{def}}{=} \boxed{c_i^1}, \quad h_j \stackrel{\text{def}}{=} \boxed{c_j^1 \cdot s_2}, \in R_{p_3} \text{ and } \hat{d}_i \stackrel{\text{def}}{=} \boxed{\boxed{c_i^1} \cdot s_3}, \quad \hat{h}_j \stackrel{\text{def}}{=} \boxed{\boxed{c_j^1 \cdot s_2} \cdot s_3} \in R_{p_4}$$

$$\text{Then, } \boxed{c_i^1 c_j^1 \cdot s_2} = d_i h_j + \mathsf{L}\boxed{c_i^1}\, \mathsf{L}\boxed{c_j^1 \cdot s_2}\, \boxed{s_3^2} - \mathsf{L}\boxed{c_i^1}\, \hat{h}_j - \mathsf{L}\boxed{c_j^1 \cdot s_2}\, \hat{d}_i \quad \in R_{p_4} \tag{6.10}$$

Thus, we have:

$$\begin{aligned}\boxed{c_k^2 s_2} &= \boxed{c_i^1 c_j^1 s_2} + u_i^1 u_j^1 \boxed{\boxed{s_1^2} s_2} - u_j^1 \boxed{\boxed{c_i^1 s_1} s_2} - u_i^1 \boxed{\boxed{c_j^1 s_1} s_2} \\ &= \left\{ d_i h_j + \mathsf{L} \boxed{c_i^1} \mathsf{L} \boxed{c_j^1 \cdot s_2} \boxed{s_3^2} - \mathsf{L} \boxed{c_i^1} \hat{h}_j - \mathsf{L} \boxed{c_j^1 \cdot s_2} \hat{d}_i \right\} \\ &+ u_i^1 u_j^1 \boxed{\boxed{s_1^2} s_2} - u_j^1 \boxed{\boxed{c_i^1 s_1} s_2} - u_i^1 \boxed{\boxed{c_j^1 s_1} s_2}\end{aligned} \tag{6.11}$$

$$\text{We may write } \boxed{c_k^2 s_2} = d_i h_j + \sum_{m \in [\mathsf{const} \cdot w]} H_m^3 C_m^4 \tag{6.12}$$

where $H_m^3 \in R_{p_3}$ is a publicly computable linear coefficient, $C_m^4$ is a ciphertext component in $R_{p_4}$ and $d_i, h_j \in R_{p_3}$ serve as *messages* for (the ciphertext constructed via) linear combinations of $C_m^4$. Now, it holds that:

$$\begin{aligned}c_t^4 &= c_k^2 c_\ell^2 + \mathsf{L} \boxed{y_k} \mathsf{L} \boxed{y_\ell} \boxed{s_2^2} - \mathsf{L} \boxed{y_\ell} \boxed{c_k^2 s_2} - \mathsf{L} \boxed{y_k} \boxed{c_\ell^2 s_2} \\ &= c_k^2 c_\ell^2 + \mathsf{L} \boxed{y_k} \mathsf{L} \boxed{y_\ell} \boxed{s_2^2} - \mathsf{L} \boxed{y_\ell} \left\{ d_i h_j + \sum_m H_m^3 C_m^4 \right\} - \mathsf{L} \boxed{y_k} \left\{ d_{i'} h_{j'} + \sum_m {H'}_m^3 C_m^4 \right\}\end{aligned} \tag{6.13}$$

Letting $C_0^4 = \boxed{s_2^2}$, and denoting the terms $d_i h_j, d_{i'} h_{j'} \in R_{p_3}$ by $C_m^3$ we may write:

$$c_t^4 = c_k^2 c_\ell^2 + \sum_m H_m^2 C_m^3 + \sum_{m'} \hat{H}_{m'}^3 C_{m'}^4 \tag{6.14}$$

Note that the terms $C_m^3$ may be computed as a product of $d_i, h_j$ which are provided directly by the encryptor and acts as the message for $\sum_{m'} \hat{H}_{m'}^3 C_{m'}^4$. We invoke $\mathsf{LinFE}$ to compute the decryption of $\sum_{m'} \hat{H}_{m'}^3 C_{m'}^4$, where $m' \in [0, \mathsf{const} \cdot w]$.

**Analyzing Ciphertext Structure.** As discussed, to compute level 4 ciphertexts, the encryptor must provide for $i \in [w]$:

1. To compute $c_k^2 \in R_{p_2}$: Provide Regev ciphertexts $c_i^1 = \boxed{x_i} \in R_{p_1}$ using secret $s_1$ and $\boxed{c_i^1 \cdot s_1}$, $\boxed{s_1^2} \in R_{p_2}$ using secret $s_2$. This is exactly as the quadratic scheme in Section 4.

2. To compute $\boxed{c_k^2 s_2}$ using secret $s_4$:

   (a) $d_i = \boxed{c_i^1}$, $h_j = \boxed{c_j^1 \cdot s_2}$ using secret $s_3$ to avoid circularity since the message contains $s_2$

   (b) $\boxed{c_i^1} \cdot s_3$, $\boxed{c_i^1 \cdot s_2} \cdot s_3$ using secret $s_4$ to avoid circularity since the message contains $s_3$. Together, these are used to compute $\boxed{c_i^1 c_j^1 s_2} \in R_{p_4}$ using secret $s_4$ as in Equation 6.10.

   (c) $\boxed{\boxed{s_1^2} s_2}$ and $\boxed{\boxed{c_i^1 s_1} s_2}$ in $R_{p_3}$ using secret $s_4$.

3. $\boxed{s_2^2}$ and $\boxed{s_3^2}$ in $R_{p_4}$ using secret $s_4$.

By Equations 6.13, 6.14 these suffice to compute level 4 ciphertext $c_\ell^4 = \mathsf{L}\left[\boxed{Z_\ell}\right] \cdot s_4 + \mathsf{N}\left[\boxed{Z_\ell}\right] + Z_\ell$ where $Z_\ell$ is level 4 plaintext. Adding up the above terms, we see that the data dependent component of the ciphertext has size $O(w)$. Thus, the online component of the ciphertext is succinct.

We note that, to ensure security, the linear equation (in secret terms) represented by equation 6.14 must be computed using noisy linear FE exactly as in Section 4. Recall that using noisy FE ensures that fresh, large noise is added to the linear equation so as to maintain security. Following the proof strategy exactly as in Section 4, we achieve the notion of semi-adaptive indistinguishability based security. In Appendix F.1, we discuss how this can be upgraded to adaptive indistinguishability based security, which in turn implies non-adaptive simulation based security for the present functionality [O'N10].

# 7 Putting it together : 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, denoted by $\mathsf{BddFE}$.

## 7.1 Bootstrapping Functional Encryption

Gorbunov et al. [GVW12] 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:

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

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 Section 6 is capable of evaluating degree 3 polynomials, it suffices for bootstrapping, to yield $q$-query online-offline FE for all circuits.

Our construction for bounded collusion FE, $\mathsf{BddFE}$, follows the blueprint described in [ALS16], but we recap it here for completeness. As in [GVW12, ALS16], let $(S, v, m)$ be parameters to the construction. Let $\Delta_i$ 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 $\Delta_i$ are cover free with high probability as shown by [GVW12]. Let $L \stackrel{\text{def}}{=} (\ell^3 + S \cdot m)$.

$\mathsf{BddFE.Setup}(1^\lambda, 1^\ell)$: Upon input the security parameter $\lambda$ and the message space $\{0,1\}^\ell$, invoke $(\mathsf{mpk}, \mathsf{msk}) = \mathsf{PolyFE.Setup}(1^\lambda, 1^{\mathsf{L}})$ 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 $\mathsf{SK}_C$ 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 is analogous to [ALS16], as the only difference between the bounded collusion FE in [ALS16] and our work is the construction of PolyFE. For completeness, we recap the argument in Appendix F. The BddFE ciphertext inherits its decomposability and online succinctness from the PolyFE ciphertext. For details, please see Appendix F.

# 8 Conclusions

We presented a new construction of Functional Encryption for bounded collusions. Apart from enjoying useful structural properties, our ciphertext size degrades quadratically with the number of queries which improves the ciphertext size of [GVW12], which degrades as $O(q^4)$.

We believe that our techniques are even more interesting than our final result, since the online succinctness of the ciphertext has intriguing implications. Our scheme shows that techniques from fully homomorphic encryption may be used to propagate computation efficiently down the circuit even for functional encryption. In more detail, our scheme achieves succinct FE for degree 4 polynomials *functionally* – the encryptor can provide a succinct ciphertext for $\mathbf{x}$, the key generator can provide a key $\mathsf{SK}_P$ for any degree 4 polynomial $P$ in $\mathbf{x}$, and the decryptor can compute on $\mathsf{CT}_{\mathbf{x}}$ to obtain a ciphertext corresponding to $\mathsf{CT}_{P(\mathbf{x})}$, apply the key $\mathsf{SK}_P$ to learn $P(\mathbf{x})$.

As mentioned before, the above strategy is insecure, and to achieve security we must artificially add noise to the decryption equation computed by every key. This forces the encryptor to provide a separate noise term for each possible key, which is added into the decryption equation, creating the dependence of the ciphertext size on the number of requested keys. This is why our scheme ends up with a large ciphertext, divided into a succinct data dependent component and a non-succinct data independent component.

Evidently, the most pressing open question is whether the noise which must be added to each decryption equation may be encoded succinctly. This does not seem impossible, since as noted by [Agr16a], the noise that must be added to each equation is not required to be independent; that is, it can be *correlated* across queries. However, how to exploit this correlation while relying

only on LWE/Ring-LWE is a pressing open question, and one with important consequences, as recent work has shown that functional encryption for constant degree polynomials is sufficient for indistinguishability obfuscation [LV16].

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

## References


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

[ABCP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In *PKC*, 2015.

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

[Agr16a] Shweta Agrawal. Personal Communication, 2016.

[Agr16b] Shweta Agrawal. Interpolating predicate and functional encryption from learning with errors. Eprint 2016, 2016.

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

[AIK14] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. *SIAM J. Comput.*, 43(2):905–929, 2014.

[AIKW15] Benny Applebaum, Yuval Ishai, Eyal Kushilevitz, and Brent Waters. Encoding functions with constant online rate, or how to compress garbled circuit keys. *SIAM Journal on Computing*, 44(2):433–466, 2015.

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

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. Eprint 2015/730, 2015.

[ALS16] Shweta Agrawal, Benoit Libert, and Damien Stehlé. Fully secure functional encryption for linear functions from standard assumptions, and applications. In *CRYPTO*, 2016. https://eprint.iacr.org/2015/608.

[AS16] Shweta Agrawal and Ishaan Preet Singh. Running turing machines on encrypted data from standard assumptions. Manuscript, 2016.

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

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

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011.

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

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

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *IACR Cryptology ePrint Archive*, 2015:163, 2015.

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

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

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

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrede Lepoint, Hemanta K Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New mmap attacks and their limitations. In *Advances in Cryptology–CRYPTO 2015*, pages 247–266. Springer, 2015.

[CHH+07] Ronald Cramer, Goichiro Hanaoka, Dennis Hofheinz, Hideki Imai, Eike Kiltz, Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Bounded cca2-secure encryption. In *Proceedings of the Advances in Crypotology 13th International Conference on Theory and Application of Cryptology and Information Security*, ASIACRYPT'07, 2007.

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

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

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

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 476–493, 2013.

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

[DKXY02] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '02, 2002.

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

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

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

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

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

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

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

[GLW12] Shafi Goldwasser, Allison Lewko, and David Wilson. Bounded-collusion ibe from key homomorphism. In *Theory of Cryptography*, Lecture Notes in Computer Science, 2012.

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

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

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

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

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

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

[HKK+14] Yan Huang, Jonathan Katz, Vladimir Kolesnikov, Ranjit Kumaresan, and Alex J Malozemoff. Amortizing garbled circuits. In *International Cryptology Conference*, pages 458–475. Springer, 2014.

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

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

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

[LR14] Yehuda Lindell and Ben Riva. Cut-and-choose yao-based secure computation in the online/offline and batch settings. In *International Cryptology Conference*, pages 476–494. Springer, 2014.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In *FOCS*, 2016.

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

[MSZ] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over ggh13. Eprint 2016/147.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. http://eprint.iacr.org/.

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

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: Functional encryption with public keys. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, 2010.

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

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

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

# Appendix

## A Previous Constructions for Bounded Collusion FE

In this section, we discuss previous constructions for Bounded Collusion FE.

### A.1 The GVW12 Construction

In this section we sketch the GVW scheme and discuss why it is unsuitable for the applications discussed in Section 1. The scheme 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 [SS10].
- Next, the single FE scheme is generalized to a $q$ query scheme for $\mathsf{NC}_1$ 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, $\mathsf{KeyGen}$ 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 $C(x)$. [GVW12] 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 $\mathsf{NC}_1$ to $q$ query FE for all circuits using computational randomized encodings [AIK06]. 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 $O(q^4)$.

### A.2 The ALS16 construction

[ALS16] 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 [GVW12]. 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 [GVW12] provides a method to bootstrap an FE for $\mathsf{NC}_1$ 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.* [AIK06, 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 [GVW12], 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. [ALS16] 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.

### A.3 Other Related Work

Recently, following our work, [Agr16b] provided a new construction of bounded key functional encryption relying on the subexponential LWE assumption. Our construction relies on the standard (not subexponential) LWE assumption. Additionally, while [Agr16b] achieves overall succinct ciphertexts, the security game for the bounded collusion setting is quite restrictive: the attacker

must announce all the queries $C_i$ where $C_i(\mathbf{x}) = 1$ before receiving a single secret key and before requesting any keys such that $C_i(\mathbf{x}) = 0$. Inspired from our work, [Agr16b] uses a similar technique to force the decryptor to add noise artificially to the decryption equation so as to achieve security. Thus, the results in [Agr16b] and the present work are incomparable.

Another approach to obtain bounded collusion FE is to compile the single key FE of Goldwasser et al [GKP+13] with the compiler of [GVW12] 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, and must rely on subexponential LWE rather than standard LWE.

# B Preliminaries

In this section we describe some preliminaries.

**Notation.** For any integer $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers modulo $q$ and we represent $\mathbb{Z}_q$ as integers in $(-q/2, q/2]$. We let $\mathbb{Z}_q^{n \times m}$ denote the set of $n \times m$ matrices with entries in $\mathbb{Z}_q$. We use bold capital letters (e.g. $\mathbf{A}$) to denote matrices, bold lowercase letters (e.g. $\mathbf{x}$) to denote vectors that are components of our encryption scheme, and arrows (e.g. $\vec{v}$) to denote vectors that represent attributes or predicates. The notation $\mathbf{A}^\intercal$ denotes the transpose of the matrix $\mathbf{A}$. When we say a matrix defined over $\mathbb{Z}_q$ has *full rank*, we mean that it has full rank modulo each prime factor of $q$.

If $\mathbf{A}_1$ is an $n \times m$ matrix and $\mathbf{A}_2$ is an $n \times m'$ matrix, then $[\mathbf{A}_1 \| \mathbf{A}_2]$ denotes the $n \times (m + m')$ matrix formed by concatenating $\mathbf{A}_1$ and $\mathbf{A}_2$. If $\mathbf{x}_1$ is a length $m$ vector and $\mathbf{x}_2$ is a length $m'$ vector, then we let $[\mathbf{x}_1 | \mathbf{x}_2]$ denote the length $(m + m')$ vector formed by concatenating $\mathbf{x}_1$ and $\mathbf{x}_2$. However, when doing matrix-vector multiplication we always view vectors as column vectors.

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

## B.1 Lattice Preliminaries

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

**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 $\sigma$. 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 $\sigma$. 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 $\rho_\sigma$ and $\mathcal{D}_{L,\sigma}$, respectively.

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

**Lemma B.1** ([MR07, Lemma 4.4]). *Let $\Lambda$ be an $n$-dimensional lattice, let $\mathbf{T}$ be a basis for $\Lambda$, and suppose $\sigma \geq \|\mathbf{T}\|_{\mathsf{GS}} \cdot \omega(\sqrt{\log n})$. Then for any $\mathbf{c} \in \mathbb{R}^n$ we have*

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

**Lemma B.2** (Flooding Lemma). *[GKPV10] Let $n \in \mathbb{N}$. For any real $\sigma = \omega(\sqrt{\log n})$, and any $\mathbf{c} \in \mathbb{Z}^n$,*

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

## B.2 Hardness Assumptions

Our constructions can be based on the hardness of LWE or Ring LWE, defined below.

**Learning With Errors.** The *Learning with Errors* problem, or $\mathsf{LWE}$, is the problem of determining a secret vector over $\mathbb{F}_q$ given a polynomial number of "noisy" inner products. The decision variant is to distinguish such samples from random. More formally, we define the (average-case) problem as follows:

**Definition B.3** ([Reg09]). Let $n \geq 1$ and $q \geq 2$ be integers, and let $\chi$ be a probability distribution on $\mathbb{Z}_q$. For $\mathbf{r} \in \mathbb{Z}_q^n$, let $A_{\mathbf{r},\chi}$ be the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing a vector $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \in \mathbb{Z}_q$ according to $\chi$, and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{r} \rangle + e)$.

The decision $\mathsf{LWE}_{q,n,\chi}$ problem is: for uniformly random $\mathbf{r} \in \mathbb{Z}_q^n$, given a $\mathrm{poly}(n)$ number of samples that are either (all) from $A_{\mathbf{r},\chi}$ or (all) uniformly random in $\mathbb{Z}_q^n \times \mathbb{Z}_q$, output 0 if the former holds and 1 if the latter holds.

We say the decision-$\mathsf{LWE}_{q,n,\chi}$ problem is *infeasible* if for all polynomial-time algorithms $\mathcal{A}$, the probability that $\mathcal{A}$ solves the decision-$\mathsf{LWE}$ problem (over $\mathbf{r}$ and $\mathcal{A}$'s random coins) is negligibly close to $1/2$ as a function of $n$.

**Ring Learning with Errors.** Let $R = \mathbb{Z}[x]/(\phi)$ where $\phi = x^n + 1$ and $n$ is a power of 2. Let $R_q \stackrel{\text{def}}{=} R/qR$ where $q$ is a large prime satisfying $q = 1 \mod 2n$. The ring learning with errors assumption, denoted by $\mathsf{RLWE}$, [LPR10] is analogous to the standard $\mathsf{LWE}$ assumption introduced by Regev [Reg09]. Let $\chi$ 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 B.4** (Ring Learning With Errors - $\mathsf{RLWE}_{\phi,q,\chi}$). 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.

**Theorem B.5** ([LPR10]). *Let $r \geq \omega(\sqrt{\log n})$ be a real number and let $R, q$ be as above. Then, there is a randomized reduction from $2^{\omega(\log n)} \cdot (q/r)$ approximate $\mathsf{RSVP}$ to $\mathsf{RLWE}_{\phi,q,\chi}$ where $\chi$ is the discrete Gaussian distribution with parameter $r$. The reduction runs in time $\mathrm{poly}(n, q)$.*

## B.3 Simulation Based Definition of Security against Bounded Collusions

In this section, we define simulation based security for bounded collusions, as in [GVW12, Defn 3.1].

**Definition B.6** ($q$-NA-SIM- and $q$-AD-SIM- Security)**.** Let $\mathcal{F}$ be a functional encryption scheme for a circuit family $\mathcal{C}$. For every p.p.t. adversary $A = (A_1, A_2)$ and a p.p.t. simulator Sim, consider the following two experiments:

| $\underline{\mathsf{Exp}^{\mathsf{real}}_{\mathcal{F},A}(1^\lambda)}$**:** | $\underline{\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{F},\mathrm{Sim}}(1^\lambda)}$**:** |
|---|---|
| 1: $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$ | 1: $\mathsf{PK} \leftarrow \mathsf{FE.Setup}(1^\lambda)$ |
| 2: $(x, st) \leftarrow A_1^{\mathsf{FE.Keygen}(\mathsf{MSK},\cdot)}(\mathsf{PK})$ | 2: $(x, st) \leftarrow A_1^{\mathsf{FE.Keygen}(\mathsf{MSK},\cdot)}(\mathsf{PK})$ |
| | Let $\mathcal{V} \stackrel{\text{def}}{=} (C_i, C_i(x), \mathsf{SK}_i)_{i\in[q]}$ |
| 3: $\mathsf{CT} \leftarrow \mathsf{FE.Enc}(\mathsf{PK}, x)$ | 3: $\mathsf{CT}, st' \leftarrow \mathrm{Sim}(\mathsf{PK}, \mathcal{V}, 1^{\lvert x\rvert})$ |
| 4: $\alpha \leftarrow A_2^{\mathcal{O}(\mathsf{MSK},\cdot)}(\mathsf{PK}, \mathsf{CT}, st)$ | 4: $\alpha \leftarrow A_2^{\mathcal{O}'(\mathsf{MSK},st',\cdot)}(\mathsf{PK}, \mathsf{CT}, st)$ |
| 5: Output $(x, \alpha)$ | 5: Output $(x, \alpha)$ |

Above, $C_i$ denote the queries made by the adversary. We distinguish between two cases of the above experiment:

1. *The adaptive experiment*, where:

   - the oracle $\mathcal{O}(\mathsf{MSK}, \cdot) = \mathsf{FE.Keygen}(\mathsf{MSK}, \cdot)$ and
   - the oracle $\mathcal{O}'(\mathsf{MSK}, st', \cdot)$ is the simulator, namely $\mathrm{Sim}^{U_x(\mathsf{MSK},st',\cdot)}(\cdot)$ and $U_x(C) = C(x)$ for any $C \in \mathcal{C}$.

   The simulator algorithm is stateful in that after each invocation, it updates the state $st'$ which is carried over to its next invocation. We call a stateful simulator algorithm Sim *admissible* if, on each input $C$, Sim makes just a single query to its oracle $U_x(\cdot)$ on $C$ itself.

   The functional encryption scheme $\mathcal{F}$ is then said to be $q$ query *simulation-secure for one message against adaptive adversaries ($q$-*AD-SIM*-secure, for short)* if there is an *admissible* stateful p.p.t. simulator Sim such that for every p.p.t. adversary $A = (A_1, A_2)$ that makes at most $q$ queries, the following two distributions are computationally indistinguishable:

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

2. *The non-adaptive experiment*, where the oracles $\mathcal{O}(\mathsf{MSK}, \cdot)$ and $\mathcal{O}'(\mathsf{MSK}, st, \cdot)$ are both the "empty oracles" that return nothing.

   The functional encryption scheme $\mathcal{F}$ is then said to be $q$ query *simulation-secure for one message against non-adaptive adversaries ($q$-*NA-SIM*-secure, for short)* if there is an *admissible* stateful p.p.t. simulator Sim such that for every p.p.t. adversary $A = (A_1, A_2)$ that makes at most $q$ queries, the two distributions above are computationally indistinguishable.

# C Construction from Standard LWE

In this section, we briefly discuss how the scheme presented in Section 4 can be modified to rely on standard LWE rather than ring LWE. The modification is straightforward: instead of requiring

NLinFE to compute a noisy linear function over a polynomial ring, we now compute a noisy linear function over a field $\mathbb{Z}_q$.

In more detail, the message carrier (for a vector $\mathbf{x} \in \mathbb{Z}_{p_0}^w$ in the encrypt algorithm is now a vector:

$$\mathbf{c} = \mathbf{U}^\intercal \mathbf{s}^1 + p \cdot \boldsymbol{\mu} + \mathbf{x} \in \mathbb{Z}_q^w.$$

Here, $\mathbf{U} = (\mathbf{u}_i) \in \mathbb{Z}_q^{n \times w}$, $\mathbf{s}^1 \leftarrow \mathbb{Z}_q^n$ and $\boldsymbol{\mu} \in \mathbb{Z}^w$ is appropriately sampled noise.

Let us examine what we require for correctness.

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

$$x_i + p \cdot \mu_i = c_i - \langle \mathbf{u}_i; \mathbf{s}^1 \rangle,$$
$$x_j + p \cdot \mu_j = c_j - \langle \mathbf{u}_j; \mathbf{s}^1 \rangle.$$

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

$$x_i x_j + p \cdot \mu_{ij} = c_i c_j - c_i \langle \mathbf{u}_j; \mathbf{s}^1 \rangle - c_j \langle \mathbf{u}_i; \mathbf{s}^1 \rangle + \langle \mathbf{u}_i; \mathbf{s}^1 \rangle \langle \mathbf{u}_j; \mathbf{s}^1 \rangle,$$

which equals

$$c_i c_j - \sum_{k \in [n] \tau \in [T]} \sum u_{jk\tau} (2^\tau c_i s_k^1) - \sum_{k \in [n] \tau \in [T]} \sum u_{ik\tau} (2^\tau c_j s_k^1) + \sum_{k,\ell \in [n] \tau \in [T]} \sum (u_{ik} u_{j\ell})_\tau (2^\tau s_k^1 s_\ell^1). \tag{C.1}$$

Clearly

$$- \sum_{k \in [n] \tau \in [T]} \sum u_{jk\tau} (2^\tau c_i s_k^1) - \sum_{k \in [n] \tau \in [T]} \sum u_{ik\tau} (2^\tau c_j s_k^1) + \sum_{k,\ell \in [n] \tau \in [T]} \sum (u_{ik} u_{j\ell})_\tau (2^\tau s_k^1 s_\ell^1). \tag{C.2}$$

is a linear equation which can be computed by the noisy linear inner product scheme NLinFE.

In more detail, NLinFE encrypt algorithm encrypts messages $\{2^\tau c_i s_k^1 \in \mathbb{Z}_q\}$ and $\{2^\tau s_k^1 s_\ell^1 \in \mathbb{Z}_q\}$ for $k, \ell \in [n], \tau \in [\log q], i \in [w]$ while the key generator provides a key corresponding to vector

$$\big(\mathbf{0},\ (u_{ik} u_{j\ell})_\tau\ , \mathbf{0}, -u_{ik\tau},\ \ \mathbf{0},\ -u_{jk\tau}\ \mathbf{0}\big)$$

so that the equation C.2 may be computed. The remaining details are easily translated from the ring to the standard setting by carefully replacing ring products by field inner products.

# D Decomposable Functional Encryption for Circuits

In this section, we recap the notion of *decomposable functional encryption* (DFE) as defined in [AS16]. Decomposable functional encryption is analogous to the notion of decomposable randomized encodings [AIK14]. Intuitively, decomposability requires that the public key $\mathsf{PK}$ and the ciphertext $\mathsf{CT}_\mathbf{x}$ of a functional encryption scheme be decomposable into components $\mathsf{PK}_i$ and $\mathsf{CT}_i$ for $i \in [|\mathbf{x}|]$, where $\mathsf{CT}_i$ depends on a single deterministic bit $x_i$ and the public key component $\mathsf{PK}_i$. In addition, the ciphertext may contain components that are independent of the message and depend only on the randomness.

Formally, let $\mathbf{x} \in \{0,1\}^k$. A functional encryption scheme is said to be decomposable if there exists a deterministic function $\mathcal{E} : \mathcal{P} \times \{0,1\} \times \mathcal{R}_1 \times \mathcal{R}_2 \to \mathcal{C}$ such that:

1. The public key may be interpreted as $\mathsf{PK} = (\mathsf{PK}_1, \ldots, \mathsf{PK}_k, \mathsf{PK}_{\mathsf{indpt}})$ where $\mathsf{PK}_i \in \mathcal{P}$ for $i \in [k]$. The component $\mathsf{PK}_{\mathsf{indpt}} \in \mathcal{P}^j$ for some $j \in \mathbb{N}$.

2. The ciphertext may be interpreted as $\mathsf{CT}_{\mathbf{x}} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_k, \mathsf{CT}_{\mathsf{indpt}})$, where

$$\mathsf{CT}_i = \mathcal{E}\left(\mathsf{PK}_i, x_i, r, \hat{r}_i\right) \forall i \in [k] \quad \text{and} \quad \mathsf{CT}_{\mathsf{indpt}} = \mathcal{E}\left(\mathsf{PK}_{\mathsf{indpt}}, r, \hat{r}\right)$$

   Here $r \in \mathcal{R}_1$ is common randomness used by all components of the encryption. Apart from the common randomness $r$, each $\mathsf{CT}_i$ may additionally make use of independent randomness $\hat{r}_i \in \mathcal{R}_2$.

We note that if a scheme is decomposable "bit by bit", i.e. into $k$ components for inputs of size $k$, it is also decomposable into components corresponding to any partition of the interval $[k]$. Thus, we may decompose the public key and ciphertext into any $i \leq k$ components of length $k_i$ each, such that $\sum k_i = k$. We will sometimes use $\bar{\mathcal{E}}(\mathbf{y})$ to denote the tuple of function values obtained by applying $\mathcal{E}$ to each component of a vector, i.e. $\bar{\mathcal{E}}(\mathsf{PK}, \mathbf{y}, r) \stackrel{\text{def}}{=} \Big(\mathcal{E}(\mathsf{PK}_1, y_1, r, \hat{r}_1), \ldots, \mathcal{E}(\mathsf{PK}_k, y_k, r, \hat{r}_k)\Big)$, where $|\mathbf{y}| = k$. We assume that given the security parameter, the spaces $\mathcal{P}$, $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{C}$ are fixed, and the length of the message $|\mathbf{x}|$ can be any polynomial.

Most known constructions of functional encryption are already decomposable, but this property was formalised and exploited in [AS16].

# E Making KeyGen stateless for Noisy Linear FE

We note that the keygen algorithm can be made stateless by using standard tricks, as in [GVW12, Section 6]. To see this, let us examine how noise is added in the above system at present. The encryptor provides encryptions of $Q$ noise terms $\boldsymbol{\delta}$ and for $i \in [Q]$, the key generator appends the $i^{th}$ key vector $\mathbf{g}_i$ with a $Q$ sized unit vector $\mathbf{e}_i$. As a result, decryption of the $i^{th}$ key with the ciphertext results in an additional term $\boldsymbol{\delta}^\intercal \mathbf{e}_i = \delta_i$, which gets added to the legitimate decryption value $\mathbf{g}_i^{\mathbf{z}}$. Thus, to ensure that a fresh noise term $\delta_i$ is added for each decryption equation, the key generator must keep track of how many keys it has issued in the past (or receive this information as input).

Cover free sets offer a natural tool to deal with this issue, and this technique was used towards a similar end in [GVW12]. The idea is to choose $Q' > Q$ and have the encryptor provide encryptions of $Q'$ noise values in place of $Q$. The key generator is modified to be stateless and randomly pick a $Q'$ sized binary vector of weight $v$ in place of $\mathbf{e}_i$. Now the decryption computes a random subset sum of $Q'$ noise terms for every key, which for appropriate setting of $Q'$ and $v$ guarantees that each decryption equation has at least one fresh noise term. It is shown in [GVW12] that $Q' = Q^2$ and $v = \lambda$ suffices.

The careful reader might notice that the summands in this case are discrete Gaussians and a random subset of discrete Gaussians does not yield the original distribution. However, fortunately this is not an issue, since the cover free property implies that every decryption equation has at least one freshly sampled discrete Gaussian that is not used in any other equation. This suffices for security.

We also note that though the general construction of linear FE for inner products modulo a prime by [ALS16] is stateful, as noted by the authors of [ALS16], this is only for the most general adversary and does not apply to our setting. For our setting, the function queries to the underlying

Linear FE are linearly independent allowing for a stateless keygen, due to the fact that each query vector contains at least one unique index which is nonzero in that vector alone, and zero for all other query vectors. This is ensured by the use of cover free sets in the construction. We refer to [ALS16] for more details, as this issue is discussed at length there.

# F Security of the Construction in Section 7

In this section, we argue that the construction described in Section 7 is secure. First, we argue that scheme achieves semi-adaptive IND based security as defined in Section 2. The proof follows easily from the security of randomized encodings and the semi-adaptive IND security of the PolyFE scheme.

Let us assume that the randomized encodings are secure. Then, given an attacker $\mathcal{A}$ who breaks the security of the BddFE, we construct an attacker $\mathcal{B}$ who breaks the security of PolyFE as follows. $\mathcal{B}$ does the following:

1. $\mathcal{B}$ obtains the public key PK from the PolyFE challenger and returns this to $\mathcal{A}$.
2. $\mathcal{A}$ outputs challenges $\mathbf{x}_0, \mathbf{x}_1$. $\mathcal{B}$ chooses the randomness $\mathbf{R} = R_1, \ldots, R_S$ and returns $(\mathbf{x}_0, \mathbf{R})$ and $(\mathbf{x}_1, \mathbf{R})$ as the challenge messages to the PolyFE challenger.
3. The PolyFE challenger returns the challenge CT, which $\mathcal{B}$ returns to $\mathcal{A}$.
4. $\mathcal{B}$ picks $\Delta_1, \ldots, \Delta_q \subseteq S$ randomly, of size $v$ each. When $\mathcal{A}$ makes a query for a circuit $C_i$, where $i \in [Q]$, $\mathcal{B}$ converts it to a tuple of degree 3 polynomials $G_{C_i,\Delta_i}(\cdot)$ and sends these to the challenger. It forwards the challenger's response to $\mathcal{A}$.
5. When $\mathcal{A}$ outputs a bit, $\mathcal{B}$ does the same.

Note that by security of randomized encodings, $\mathcal{B}$ is an admissible adversary. If $\mathcal{B}$ is admissible, then the advantage of $\mathcal{B}$ is exactly the advantage of $\mathcal{A}$.

**Ciphertext Structure.** We note that the BddFE ciphertext is a PolyFE ciphertext for the message vector $(\mathbf{x}, R_1, \ldots, R_S)$. Since the PolyFE ciphertext is decomposable, and online-succinct, the same features are inherited by the BddFE ciphertext.

**Distributed Data Applications.** Note that since BddFE CT is decomposable and online succinct, $k$ parties holding $k$ pieces of data $x_1, \ldots, x_k$ can share a PRF seed to generate the common randomness required to tie together the components of a decomposable FE scheme (see Appendix D). Given this seed, they can independently encode their individual inputs $x_i$ to construct $\mathsf{CT}_i$ and upload these to a common server. We note that the data independent offline component $\mathsf{CT}_{\mathsf{indpt}}$ may be generated by a designated party with a high bandwidth upload link, or by the server itself.

## F.1 Achieving simulation based security

Above, we argued that the scheme BddFE achieves semi-adaptive IND based security. Other bounded collusion schemes such as [GVW12, ALS16, Agr16b] achieve simulation based security

[6] NA-SIM [O'N10, GVW12], i.e. $(\text{poly}, \text{poly}, 0)$ SIM security, where the adversary can request a polynomial number of pre-challenge keys, ask for polynomially sized challenge ciphertexts but may not request post-challenge keys. The definition is provided in Appendix B.3.

The proof for BddFE inherits semi-adaptive IND security from the semi-adaptive IND security of the quadratic FE scheme provided in Section 4. This is because, for our proof technique, it is necessary for the simulator to program the function keys after seeing the challenge CT. Hence, the proof strategy described in Section 4 does not support pre-challenge CT queries. Agrawal et al. [ALS16] show that adaptive IND of PolyFE implies non-adaptive simulation based security of PolyFE and hence BddFE. Hence, if we could upgrade the security of PolyFE to adaptive rather than semi-adaptive IND, our final scheme BddFE would also achieve NA-SIM.

In this section, we describe how we can achieve adaptive IND in Section 4 by enhancing the proof suitably. For simplicity, consider a single pre-challenge query, say for monomial $x_i x_j$. The pre-challenge key is constructed by the simulator as in the real system and the challenge ciphertext is modified, as described below.

To support one pre-challenge query, the message dimension $w+2$ in the ciphertext is expanded to $w+3$ as follows: the ciphertext is modified to contain $\mathsf{NLinFE.CT}(s_1^2, c_1 s_1, \ldots, c_w s_1, 0, 0)$ (note the additional 0 in the end of the message). In the simulation, we are required to insert the corrective term $f_{ij}$ into the decryption equation. We may do this by switching to $\mathsf{NLinFE.CT}(t_0, \ldots t_w, 0, f_{ij})$ in the simulation. The key for monomial $x_i x_j$ is the NLinFE key for the vector $(u_i u_j, 0....0, -u_i, 0...0, -u_j, 0...0, f_{ij}, 1)$.

By the proof of Claim 4.3, the key for monomial $x_i x_j$ decrypts the two ciphertexts to approximately the same value, as required by NLinFE. However, we introduced an additional slot to support the pre-challenge query, and to support $Q$ pre-challenge key queries, the above ciphertext will contain $Q$ additional slots, increasing the ciphertext size by an additive factor of $Q$.

The above blowup does not change the asymptotic size of the ciphertext. Additionally, this blowup can in fact be avoided: since the construction of NLinFE (see Section 5) anyway inserts $Q$ slots in the LinFE CT to support $Q$ keys, the same slots can be reused for the above mechanism. We defer details to the full version of this work.

Assuming the underlying scheme PolyFE to be non-adaptive simulation secure, i.e. NA-SIM, we get a final security notion of NA-SIM, i.e. $(\text{poly}, \text{poly}, 0)$-SIM for BddFE which means the adversary may make an unbounded number of pre-challenge queries, ask for an unbounded number of challenge ciphertexts but may not make any post-challenge key queries. While the construction of [GVW12] supports the stronger AD-SIM security in addition to NA-SIM, this notion can only be achieved for a scheme that supports a *single* ciphertext. We consider the model of a single ciphertext and bounded number of keys overly restrictive, and do not attempt to achieve this notion. This is also the approach taken by [GKP+13].

# G Handling arbitrary challenge messages

In this section, we discuss how to generalise the proof in Section 4.2 to handle arbitrary challenge messages. This can be done in a manner analogous to the proofs in [ABCP15, ALS16]. At a high

[6] We note that [GVW12] achieve adaptive simulation based security but due to the impossibility of [BSW11], this notion is only achieved for an FE scheme that produces a single ciphertext. Since this setting is quite restrictive, we restrict our attention to non-adaptive simulation security.

level, we design $\mathbf{B}$ to be a low norm basis that contains $\mathbf{x}_0 - \mathbf{x}_1$ as the last basis vector $\mathbf{b}_w$ and the remaining basis vectors orthogonal to it. The challenger constructs the public parameters, and the ciphertext $\mathbf{c}$, $\mathbf{b}$ along this basis. Along this basis, it holds that the adversary only queries for a monomial involving $x_{\mathbf{b}_w}$ iff $\mathbf{x}_0[\mathbf{b}_i]\mathbf{x}_0[\mathbf{b}_w] = \mathbf{x}_1[\mathbf{b}_i]\mathbf{x}_1[\mathbf{b}_w] = 0$. That is, along this basis, the above restriction on the proof holds. To convert this to an arbitrary basis, the challenger applies a change of basis transformation, which, by the construction of [ALS16], is low norm.