Composable Functional Encryption: Secure and Flexible Encrypted Computation

Seungwan Hong[1], Jiseung Kim[2], Changmin Lee[3], and Minhye Seo[4]

[1] Yale University
seungwan.hong@yale.edu
[2] Jeonbuk National University
jiseungkim@jbnu.ac.kr
[3] Korea University
changminlee@korea.ac.kr
[4] Duksung Women's University
mhseo@duksung.ac.kr

Abstract. Functional encryption (FE) enables fine-grained access control over encrypted data by allowing decryption keys to reveal only a specific function of the underlying plaintext. However, existing FE schemes remain limited in practicality for complex workflows due to their single-shot evaluation model: each functional key permits only one evaluation, and the output is typically revealed in plaintext making secure reuse of intermediate results impossible.

We introduce Composable Functional Encryption (CFE), a new paradigm that addresses this limitation by enabling encrypted outputs to be reused as inputs for subsequent computations. This reusability property allows complex functionalities (e.g., $f \circ g(x)$) to be realized through the efficient composition of simpler functions, with ciphertext and key sizes scaling linearly with the composition depth. Our construction, based on standard lattice assumptions (RLWE), avoids reliance on heavy cryptographic primitives such as indistinguishability obfuscations or multilinear maps.

By making ciphertext-level reusability a native capability, CFE provides a powerful foundation for real-world applications including multi-stage data analytics and privacy-preserving machine learning, where intermediate outputs must be securely retained and processed across dynamic or iterative pipelines.

1 Introduction

Functional encryption (FE) [17, 46] is a cryptographic primitive that enables fine-grained access control over encrypted data. Unlike traditional encryption schemes that only allow decryption of entire plaintexts, FE grants the ability to compute specific functions on encrypted data while keeping all other information hidden. Given a functional key fk_F and a ciphertext $\mathsf{Enc}(\boldsymbol{m})$, the decryption algorithm, when provided with fk_F and $\mathsf{Enc}(\boldsymbol{m})$, returns the value $F(\boldsymbol{m})$.

FE research has developed along two primary directions. One focuses on practicality by developing schemes for specific function classes such as linear [1, 6, 36] and quadratic polynomials [13, 28]. The other explores general-purpose FE for complex functions like NC^1 circuits, ofen relying on strong primitives such as indistinguishability obfuscations and multilinear maps [10,11,16,27,35,40–42], and frequently operating under the bounded-collusion (or Q-bounded) security model [3, 4, 7, 8, 12, 15, 21, 25].

Despite significant progress, existing FE constructions suffer from a fundamental limitation: their *single-shot* nature. Each functional key reveals its output in plaintext, prohibiting secure reuse of intermediate values. This lack of ciphertext-level reusability breaks the modularity essential for many real-world applications.

While fully homomorphic encryption (FHE) [19, 22, 29] supports arbitrary computation over ciphertexts, it lacks the ability to enforce fine-grained access control. Conversely, FE provides such control but fails to support workflows requiring multiple composed evaluations over intermediate ciphertexts. This makes FE fundamentally non-suited for layered, iterative, or branching computations.

The implications are clear in settings like privacy-preserving machine learning (PPML), where encrypted models undergo iterative updates, and intermediate states (e.g., activations, gradients) must remain encrypted throughout. Existing FE schemes force decryption and re-encryption between stages, undermining efficiency and modularity. While prior work has proposed simplified PPML pipelines [24, 30, 38, 39, 47, 48, 51, 52] or partially encrypted computation [49], these approaches either limit flexibility or weaken security [20].

Although general-purpose FE schemes theoretically support complex functions, most constructions rely on bootstrapping from indistinguishability obfuscation [35, 40, 41], resulting in prohibitive costs and assumptions. This leaves a persistent gap between theory and practice: although FE promises expressive and secure computation, its non-reusable design fundamentally obstructs integration into modular and adaptive real-world pipelines.

To bridge this gap, we argue that *reusability* must become a native capability of FE for various applications. A new paradigm is needed; one that allows encrypted intermediate values to be securely reused, composed, and propagated across multiple stages of computation.

1.1 Our Contribution

We address a fundamental limitation of existing FE schemes: their inability to securely reuse encrypted outputs in downstream computations. To overcome this, we introduce a new paradigm, *Composable Functional Encryption (CFE)*, that brings reusability as a native capability to FE.

We instantiate CFE under the standard RLWE assumption, avoiding heavy cryptographic primitives such as indistinguishability obfuscation or multilinear maps. Our CFE supports secure ciphertext-level reusability while preserving simulation-based security under the Q-bounded collusion model.

Moreover, CFE integrates the strengths of both homomorphic encryption (HE) and functional encryption (FE), striking a practical balance between fine-grained access control and modular encrypted computation.

The main characteristics of our CFE are summarized as follows:

- **Ciphertext Reusability:** We propose the first FE framework that allows encrypted outputs to be securely reused as inputs for further function evaluations. This capability transforms FE from a single-shot tool into a modular computation primitive.
- **Composable Evaluation with Flexible Output Control:** Our CFE construction supports encrypted function chaining, allowing each evaluation to choose between returning a plaintext or ciphertext output. This flexibility enables both terminal and composable computations within the same system. Concretely, with ciphertext $C_{\boldsymbol{x}} \leftarrow \mathsf{Enc}(\mathbf{x})$ and keys $\mathsf{fk}_{\mathbf{Enc}\circ g}, \mathsf{fk}_f$,

$$\mathbf{Dec}(C_{\boldsymbol{x}}, \mathsf{fk}_{\mathbf{Enc}\circ g}) = C_{g(\boldsymbol{x})}, \quad \mathbf{Dec}(C_{g(\boldsymbol{x})}, \mathsf{fk}_f) = f \circ g(\boldsymbol{x}),$$

  which realizes an encryption of $f \circ g(\mathbf{x})$ while preserving FE-style authorization without heavy cryptographic primitives such as multilinear maps or indistinguishability obfuscation (iO).
- **Efficiency with Linear Overhead:** Our scheme enables encrypted function composition with key and ciphertext sizes growing linearly in the composition depth, making secure reusability practical for real-world applications.
- **New Applications through Reusability:** By enabling encrypted intermediates to persist across stages, CFE unlocks capabilities for modular secure analytics, privacy-preserving machine learning, and dynamic branching workflows-none of which are feasible with existing FE schemes.

1.2 Applications

The property of CFE gives the following applications.

Privacy-Preserving Fine-Tuning of Pre-trained Models. Fine-tuning is the process of adapting a pre-trained model to a specific task or dataset by continuing training on new, often sensitive data. While this approach effectively improves task-specific performance, it introduces serious privacy risks when the fine-tuning dataset contains confidential information such as medical records or proprietary business assets. Let the model consist of L layers $\{f_1, f_2, \ldots, f_L\}$, with network output $f(x) = f_L \circ \cdots \circ f_1(x)$ for an input x. CFE enables privacy-preserving fine-tuning by allowing encrypted data to be used throughout the training process while providing both flexible output control and layer-by-layer composable function evaluation.

- **Flexible output control.** Only selected values are revealed in plaintext. For example, intermediate metrics like loss or accuracy can be selectively decrypted in plaintext to guide optimization, while model parameters and gradients remain encrypted. The final fine-tuned model can also be optionally released in plaintext for downstream use.

- **Layer-by-layer composable function evaluation.** Training consists of two phases: forward and backward propagations. In the *forward propagation*, the input x is processed layer by layer to produce the prediction $f(x)$. CFE enables sequential evaluation of model layers in the encrypted domain via function composition:

$$\mathsf{Dec}_{fk_{\mathsf{Enc}\circ f_i}}(\mathsf{Enc}(f_{i-1}\circ\cdots\circ f_1(x))) = \mathsf{Enc}(f_i\circ\cdots\circ f_1(x)),$$

  for $i = 1, \ldots, L-1$, while flexible output control allows the final layer to be decrypted,

$$\mathsf{Dec}_{fk_{f_L}}(\mathsf{Enc}(f_{L-1}\circ\cdots\circ f_1(x))) = f_L\circ\cdots\circ f_1(x) = f(x),$$

  enabling plaintext loss computation while keeping intermediate activations encrypted. During the *backward propagation*, gradients are computed and propagated using these encrypted activations, allowing parameter updates without exposing sensitive intermediate information.

This fine-tuning scenario highlights inherent limitations of existing cryptographic approaches. Fully homomorphic encryption (FHE) enables arbitrary computation over encrypted data but forces all outputs to remain encrypted, whereas fine-tuning fundamentally requires selective plaintext access to values such as the loss and final-layer outputs to guide optimization. Functional encryption (FE) relaxes this restriction by allowing specific outputs to be revealed, yet it cannot maintain privacy across iterative, layer-by-layer computations: once an intermediate value is decrypted, it cannot be securely reused as input to subsequent encrypted evaluations. As a result, neither FHE nor FE can naturally support the privacy-preserving forward and backward passes required for neural network fine-tuning, motivating the need for a primitive that simultaneously enables composable encrypted computation and flexible output release.

Multi-Level Access Control for Medical and Genomic Data Analysis. CFE is highly applicable in scenarios involving sensitive data sharing among multiple stakeholders, such as hospitals and research institutions handling encrypted patient records. Using CFE, institutions can control the format of computation results based on access privileges. Two main properties of CFE are particularly valuable here.

- **Composable function evaluation.** Medical and genomic analytics naturally form a multi-stage pipeline – e.g., variant calling f_1, quality control/filtering f_2, and risk scoring f_3. CFE enables this pipeline to run entirely over encrypted inputs while enforcing role-specific visibility.
- **Flexible output control.** CFE ensures that each stakeholder sees only the level of detail permitted by their access policy, ranging from encrypted ciphertexts to plaintext summaries or full results. For instance, doctors or researchers may be granted access to plaintext analytical results (e.g., statistical summaries) and given a restricted key for an aggregate/reporting function h: $\mathsf{Dec}_{fk_h}(\mathsf{Enc}(f_3\circ f_2\circ f_1(x))) = h(f_3\circ f_2\circ f_1(x))$, releasing only

h-filtered plaintext. Cloud servers are only permitted to store and process encrypted results without learning the underlying data. When updates to encrypted user data $\mathsf{Enc}(x)$ are required, the cloud server can perform the necessary computations and directly obtain the updated result in encrypted form for secure storage: $\mathsf{Dec}_{fk_{\mathsf{Enc}\circ u}}(\mathsf{Enc}(x)) = \mathsf{Enc}(u(x))$, where u is an update function. Full decryption privileges remain exclusively with authorized administrators, enabling collaborative analysis without compromising patient privacy or institutional security policies.

2 Related work

In this section, we discuss prior works related to composable computations in functional encryption. We compare our CFE model to three main paradigms: Nesting Functional Encryption (Nesting FE), Hierarchical Functional Encryption (HFE), and Streaming Functional Encryption (sFE), and also discuss constructions in dynamic bounded-collusion models. These schemes differ in whether and how they support composition: Nesting FE and HFE allow only static, key-based forms of composition and ultimately output plaintext, while streaming FE addresses dynamic data streams but not compositional computation. Consequently, none of these approaches achieves full ciphertext-level composition. Out CFE scheme, by contrast, supports reusable ciphertext outputs, and we also compare it to generic and gabling-based alternatives.

2.1 Nesting Functional Encryption

Nesting FE, introduced by Agrawal and Rosen [8], enables the evaluation of composite functions over encrypted data, specifically targeting NC^1 circuits. However, the notion of 'composition' in this setting is static: the authority must generate a single key for the entire composite function at setup time. The evaluator cannot dynamically compute functions and later decide to apply another function to the encrypted result. Moreover, as in standard FE, the output of Nesting FE is always revealed in plaintext, so intermediate results cannot be cryptographically reused.

While Nesting FE and CFE both aim to enable layered function evaluations, our CFE framework offers distinct advantages in terms of output flexibility, efficiency, and ciphertext adaptability:

- **Output Flexibility:** Nesting FE always outputs plaintexts, limiting cryptographic reuse. In contrast, CFE enables selective output control—supporting both plaintext and ciphertext outputs.
- **Key Efficiency:** Nesting FE requires separate keys for functions even if they share common sub-computations. CFE allows partial key reuse, reducing redundancy in key generation and storage.

2.2 Hierarchical Functional Encryption

Hierarchical Functional Encryption (HFE), introduced by Ananth et al. [9] and extended by Brakerski et al. [18], augments standard FE with a key delegation mechanism. Given a functional key fk_f for a function g, a user can derive a new key $\mathsf{fk}_{g \circ f}$ for the composed function $g \circ f$, enabling function chaining without access to the master secret key. This supports hierarchical access control and delegation across distributed systems.

While both HFE and Composable Functional Encryption (CFE) support function composition, they differ in key capabilities:

- **Output Control:** HFE returns only plaintext outputs and lacks built-in support for encrypted intermediate results. In contrast, CFE introduces selective output control using distinct key types: one for obtaining plaintext results (fk_f) and another for producing encrypted outputs ($\mathsf{fk}_{\mathrm{Enc} \circ f}$), enabling further cryptographic computation on the result.
- **Underlying Assumptions and Function Classes:** HFE constructions rely on strong cryptographic primitives such as indistinguishability obfuscation [9] or general-purpose public-key FE [18]. While efficient HFE schemes exist for restricted classes such as linear functions (e.g., inner products [2, 14, 37, 45]), support for more general or high-degree functions remains costly. In contrast, our CFE construction is based on the standard lattice assumption, supports quadratic polynomials efficiently, and is readily extensible to higher-degree polynomials within the same algebraic framework.

2.3 Streaming Functional Encryption

Streaming Functional Encryption (sFE) [15, 34] is designed for dynamic data environments where inputs arrive sequentially over time. In the sFE model, a fixed, stateful function is evaluated incrementally on each prefix of the input stream, producing a series of plaintext outputs y_i while maintaining internal state st_i in encrypted form. This paradigm is well-suited for applications such as secure telemetry, private monitoring, or encrypted sensor data processing.

In contrast, CFE operates on a fixed ciphertext input and supports layered function composition across computational depth. The primary distinction lies in output and programmability:

- **Data vs. Function Composition:** sFE extends FE along the axis of data availability over time, focusing on streaming and stateful computation. CFE extends FE along the axis of computational programmability, supporting complex function pipelines on static inputs.
- **Output Control:** sFE inherently produces plaintext outputs at each step and does not support encrypted intermediate results. CFE, by contrast, provides selective control over output type—allowing users to obtain either plaintext or ciphertext after each function evaluation, facilitating further cryptographic operations.

Table 1. Comparison of FE paradigms. Nesting FE and HFE only enable partial composition, since their outputs remain plaintext and cannot be reused. sFE does not support composition at all. here

Feature	Nesting FE [8]	HFE [18]	sFE [15]	CFE (ours)
Output Control	Plaintext	Plaintext	Plaintext	Flexible
Composition	Yes	Yes	No (Stream data)	Yes
Reusability	No	No	No	Yes
Assumptions	Standard	Strong	Standard	Standard
Bounded-collusion	Static	-	Dynamic	Static

2.4 Static vs. Dynamic Bounded-Collusion Models

Functional encryption schemes are often designed under a *bounded collusion* model, which limits the number of functional keys an adversary can obtain. This model can be instantiated in either a static or dynamic form:

- In the static model, the global collusion bound Q is fixed at system setup, and security holds as long as no more than Q keys are ever issued.
- In the dynamic model [7, 26], the collusion bound can be specified *per ciphertext* at encryption time, allowing more flexible key distribution and finer-grained control over security.

Our CFE construction is presented in the static bounded-collusion model. This choice allows us to focus on establishing the feasibility and core functionality of composable computation with selective output control, while keeping the analysis simple and transparent.

Importantly, recent work by Garg, Goyal, and Lu [25] introduces a generic compiler that transforms any statically secure FE scheme into a dynamically secure one, assuming the existence of identity-based encryption (IBE).

Thus, while our construction is currently formulated in the static model, it admits a clear and well-established pathway to dynamic security without requiring changes to the underlying cryptographic core.

2.5 Generic and Garbling-Based Constructions

Composable functionalities similar to CFE can be achieved using generic cryptographic transformations. For example, one could employ a general-purpose FE scheme to evaluate a function of the form $H(x) := \mathrm{Enc}(F(G(x)))$, or use circuit garbling techniques to embed encryption logic within functional evaluations.

However, such approaches typically rely on extremely strong assumptions—such as indistinguishability obfuscation (iO)—and suffer from severe inefficiencies. Even recent efforts that avoid iO often require complex circuit-based delegation mechanisms and are not well-suited for practical use.

Recent works demonstrate this trend: Bhushan et al. [15] construct streaming FE using garbled circuits, and Garg et al. [26] employ garbling-based techniques in compilers that upgrade FE schemes to dynamic collusion settings.

In contrast, our CFE construction is the first to realize composable functional encryption directly and efficiently under standard lattice-based assumptions. By leveraging the algebraic structure of RLWE-based FE for quadratic (and extendable polynomial) functions, we avoid the need for heavy generic transformations and offer a concrete, post-quantum-secure primitive that supports layered function computation with selective output encryption.

3 Technical Overview

This section provides a high-level overview of our CFE scheme, with a focus on how composable functional keys are generated and how encrypted outputs are enabled for reuse.

A core technique of our construction is the ability to generate a functional key $\mathsf{fk}_{\mathsf{Enc} \circ f}$ that produces an *encrypted* output rather than revealing $f(\boldsymbol{x})$ in the clear. To achieve this, we design a key generation procedure that composes encryption with the function, allowing the output to remain encrypted and reusable.

Crucially, this requires restricting the structure of the underlying ciphertext in a way that supports composable decryption. Our scheme modifies the simplified Quadratic Functional Encryption (QFE) scheme of Agrawal and Rosen [8] to support such functionality. In particular, we adapt the encryption and key generation algorithms to ensure that decrypted outputs can serve as valid inputs for subsequent function evaluations.

3.1 Previous QFE Scheme

Let $R = \mathbb{Z}[x]/\langle x^n + 1 \rangle$, where n is a power of two. For any $\boldsymbol{m} = (m_i) \in \mathbb{Z}^\ell$, we let $F(\boldsymbol{m}) = \sum_{j \le k} F_{j,k} \cdot m_j \cdot m_k \in R$ be a quadratic polynomial to be evaluated. For simplicity, we omit noise flooding terms, which serve to hide secret information.

The **QFE** algorithm is designed to ensure that, at the end of the decryption procedure, the decryptor obtains

$$\sum_{j,k} F_{j,k} \cdot (P \cdot m_j + e_j) \cdot (P \cdot m_k + e_k) \approx P^2 \cdot F(\boldsymbol{m}).$$

Here, two message-related components $P \cdot m_j + e_j$ and $P \cdot m_k + e_k$ are derived from a ciphertext. To hide messages, RLWE encodings are used: $c_i = u_i \cdot s + P \cdot m_i + e_i$ for properly chosen u_i, s and e_i.

During decryption, the following computations are required:

$$\begin{aligned}(P \cdot m_k + e_k) \cdot (P \cdot m_j + e_j) &= (c_k - u_k \cdot s)(c_j - u_j \cdot s) \bmod q \\ &= c_k \cdot c_j - u_j \cdot c_k \cdot s - u_k \cdot c_j \cdot s + u_k \cdot u_j \cdot s^2 \bmod q\end{aligned}$$

Here, u_i are public parameters in pp, and c_i are included in the ciphertext of $\boldsymbol{m}$. Additionally, one should also hide information of s, so the authors generate a ciphertext $\mathbf{LFE.Enc}(\mathbf{LFE}.\mathsf{pp}, (s^2, \boldsymbol{c}\cdot s))$, where $(s^2, \boldsymbol{c}\cdot s) = (s^2, c_1\cdot s, \cdots, c_\ell \cdot s)$ and $\mathbf{LFE}$ is a FE scheme for linear polynomials like [1, 6]. The syntax and detailed description of $\mathbf{LFE}$ are recalled in Appendix A.2. For ease of exposition, we assume that the message and ciphertext spaces are well defined.

As a consequence, $\mathbf{Enc}, \mathbf{KGen}$ and $\mathbf{Dec}$ algorithms are defined as follows:

- $\mathbf{QFE.Enc}(\boldsymbol{m})$: Outputs $\mathsf{ct} = (\boldsymbol{c}, \boldsymbol{b})$, where

$$\boldsymbol{c} = \boldsymbol{u}\cdot s + P\cdot \boldsymbol{m} + \boldsymbol{e} \in R_q^{\ell+1} \text{ and } \boldsymbol{b} = \mathbf{LFE.Enc}(\mathbf{LFE}.\mathsf{pp}, (s^2, \boldsymbol{c}\cdot s))$$

- $\mathbf{QFE.KGen}(F)$: Computes two terms (F, fk_F) of the form:

$$\mathsf{fk}_F \leftarrow \mathbf{LFE.KGen}(\mathbf{LFE}.\mathsf{msk}, \mathsf{fk}'_F),$$

  where fk'_F is a vector in $R^{\ell+1}$ defined as:

$$\mathsf{fk}'_F = \left(\sum_{1\le j\le k\le \ell} F_{j,k}(u_j \cdot u_k \cdot \boldsymbol{\delta}_1^{\ell+1} - u_j \cdot \boldsymbol{\delta}_{1+k}^{\ell+1} - u_k \cdot \boldsymbol{\delta}_{1+j}^{\ell+1}) \right).$$

  Here, $\boldsymbol{\delta}_j^{\ell+1} \in \mathbb{Z}^\ell$ denotes j-th standard basis vector. Since the scheme does not provide a function-hiding property for F, we denote by fk_F the output of $\mathbf{QFE.KGen}(F)$.

- $\mathbf{QFE.Dec}(\mathsf{ct}, \mathsf{fk}_F)$:
  1. Given $\mathsf{ct} = (\boldsymbol{c}, \boldsymbol{b})$, compute $F(\boldsymbol{c}) + \mathbf{LFE.Dec}(\mathbf{LFE}.\mathsf{pp}, \boldsymbol{b}, \mathsf{fk}_F)$, which is identical to $\mu' = \sum_{j,k} F_{j,k}(P\cdot m_j + e_j)\cdot(P\cdot m_k + e_k)$.
  2. Return $\lfloor \mu'/P^2 \rceil$.

The correctness of the QFE scheme is established in the main text (see Section 4.1). As mentioned earlier, our scheme treats QFE as a subroutine. For brevity, we denote $\mathbf{Enc}, \mathbf{KGen}$ and $\mathbf{Dec}$ omitting the $\mathbf{QFE}$ qualifier.

3.2 High-level Overview of CFE Construction

In CFE, decryption can return either $F(\boldsymbol{m})$ or $\mathbf{Enc}(F(\boldsymbol{m}))$, depending on the functional key. The functional key generation of fk_F that returns $F(\boldsymbol{m})$ during the decryption procedure is essentially the same as the previous $\mathbf{QFE}$ of [8].

Thus, this section focuses only on techniques for generating $\mathsf{fk}_{\mathbf{Enc}\circ F}$ that outputs $\mathbf{Enc}(F(\boldsymbol{m}))$, based on the simplified $\mathbf{QFE}$ scheme in Section 3.1. In other words, $\mathbf{Dec}(\mathsf{ct}, \mathsf{fk}_{\mathbf{Enc}\circ F})$ returns a pair $(\boldsymbol{c}^1, \boldsymbol{b}^1)$ satisfying the desired properties: $\boldsymbol{c}^1$ would be a tuple of RLWE encodings with some ephemeral secret s_1, and $\boldsymbol{b}^1$ must be expressed as an output of $\mathbf{LFE.Enc}$ with inputs $(\mathbf{LFE}.\mathsf{pp}, (s_1^2, \boldsymbol{c}^1\cdot s_1))$. To streamline the exposition, we temporarily decompose the computation of $\mathsf{fk}_{\mathbf{Enc}\circ F}$ into two components, $\mathsf{fk}^{\boldsymbol{c}_1}_{\mathbf{Enc}\circ F}$ and $\mathsf{fk}^{\boldsymbol{b}_1}_{\mathbf{Enc}\circ F}$, which we present separately.

The description is oversimplified, but it is very helpful to explain our idea for building CFE.

3.2.1 Computation for $\boldsymbol{c}^1$

This step begins with the following observation: Given RLWE encodings of zeros $\boldsymbol{d}$, the expression $\boldsymbol{d} + P \cdot F(\boldsymbol{m})$ can be RLWE encodings.

In fact, we need to generate $\mathsf{fk}_{\mathbf{Enc} \circ F}$ such that $\mathbf{Dec}(\mathsf{ct}, \mathsf{fk}_{\mathbf{Enc} \circ F})$ for $\mathsf{ct} = (\boldsymbol{c}, \boldsymbol{b})$ returns RLWE encodings of the form $\boldsymbol{d} + P \cdot F(\boldsymbol{m})$. To achieve this, we slightly modify the encryption and Keygeneration algorithms of **QFE** scheme as follows:

- $\mathbf{QFE.Enc}(\boldsymbol{m})$: Outputs $\mathsf{ct} = (\boldsymbol{c}, \boldsymbol{b})$ such that

$$\boldsymbol{c} = \boldsymbol{u} \cdot s + P \cdot \boldsymbol{m} + \boldsymbol{e}' \bmod q \text{ and } \boldsymbol{b} = \mathbf{LFE.Enc}(\mathbf{LFE}.\mathsf{pp}, (s^2, \boldsymbol{c} \cdot s, \mathbf{1}))$$

  where $\mathbf{1}$ is a vector of ones.

Then, the key-generation procedure producing $\mathsf{fk}^{\boldsymbol{c}_1}_{\mathbf{Enc} \circ F}$ whose decryption yields $\boldsymbol{c}^1$ is given below:

- Computation for $\mathsf{fk}^{\boldsymbol{c}_1}_{\mathbf{Enc} \circ F}$:
  1. Sample s_1, and $\boldsymbol{e}$.
  2. Compute $\boldsymbol{d} = (d_i) = \boldsymbol{u} \cdot s_1 + \boldsymbol{e}$ and

$$\mathsf{fk}_F \leftarrow \mathbf{LFE.KGen}(\mathbf{LFE}.\mathsf{msk}, (\mathsf{fk}'_F, P \cdot \boldsymbol{d})),$$

     where fk'_F is a vector in $R^{\ell+1}$ as above.

- $\mathbf{QFE.Dec}(\mathsf{ct}, \mathsf{fk}^{\boldsymbol{c}_1}_{\mathbf{Enc} \circ F})$:
  1. Compute $\mu' = F(\boldsymbol{c}) + \mathbf{LFE.Dec}(\mathbf{LFE}.\mathsf{pp}, \boldsymbol{b}, \mathsf{fk}^{\boldsymbol{c}_1}_{\mathbf{Enc} \circ F})$
  2. Returen $\lfloor \mu' / P^2 \rceil$.

Compared with the original decryption in Section 3.1, the modified procedure introduces the additional term $\langle P \cdot \boldsymbol{d}, \mathbf{1} \rangle$. Equivalently,

$$\mu' = F(\boldsymbol{c}) + \mathbf{LFE.Dec}(\mathbf{LFE}.\mathsf{pp}, \boldsymbol{b}, \mathsf{fk}_F) = P \cdot \sum_i d_i + P^2 \cdot F(\boldsymbol{m}) + e^*,$$

where e^* denotes the noise term. Rounding after division by P gives

$$\lfloor \frac{\mu'}{P} \rceil = \boldsymbol{c}^1 = \sum_i d_i + P \cdot F(\boldsymbol{m}),$$

so the output is an RLWE-style encoding.

Multiple operation:. We can simultaneously evaluate multiple quadratic functions $\{F_i\}_{i=1}^t$. In this case, $\boldsymbol{c}^1$ becomes a vector whose length scales with the number t of evaluated functions. More precisely, the length of $\boldsymbol{c}^1$ is determined by the length of the message vector $\big(F_i(\boldsymbol{m})\big)_{i=1}^t$, and the length of $\boldsymbol{b}^1$ likewise depends on the message length.

Consequently, our construction must be *leveled*: the length of the original ciphertext $(\boldsymbol{c}, \boldsymbol{b})$ may differ from that of $(\boldsymbol{c}^1, \boldsymbol{b}^1)$ depending on how many polynomials are evaluated.

Finally, although this modification appears simple, its concrete instantiation is nontrivial due to security considerations.

3.2.2 Computation for $\boldsymbol{b}^1$

For ease of description, suppose that $\boldsymbol{c}^1 = d + P \cdot F(\boldsymbol{m})$, where $d = u' \cdot s_1 + e$. That is, $F(\boldsymbol{m})$ is a vector of a proper length. To compute $\boldsymbol{b}^1$ associated with $\boldsymbol{c}^1$, we will generate a term of the form:

$$\boldsymbol{b}^1 := \mathsf{LFE.Enc}(\mathsf{LFE.pp}, (s_1^2, \boldsymbol{c}^1 \cdot s_1, \mathbf{1})) = \mathsf{LFE.Enc}\big(\mathsf{LFE.pp}, \big(s_1^2, (d + P \cdot F(\boldsymbol{m})) \cdot s_1, \mathbf{1}\big)\big)$$

By the linearity of the map $\mathsf{LFE.Enc}$ in its message argument, we can decompose

$$\boldsymbol{b}^1 = \mathsf{LFE.Enc}(\mathsf{LFE.pp}, (s_1^2, s_1 \cdot \boldsymbol{d}, \mathbf{1})) + \mathsf{LFE.Enc}(\mathsf{LFE.pp}, (0, s_1 \cdot P, \mathbf{0})) \cdot F(\boldsymbol{m}).$$

This representation makes explicit that each coordinate of $\boldsymbol{b}^1$ is a quadratic function of $\boldsymbol{m}$. Equivalently, there exists a vector-valued quadratic map G such that $\boldsymbol{b}^1 = G(\boldsymbol{m})$. Writing $G(\boldsymbol{x}) = (G_i(\boldsymbol{x}))_i$ with each G_i a quadratic polynomial, we can recover each coordinate via

$$\mathsf{QFE.Dec}\big((\boldsymbol{c}, \boldsymbol{b}), \mathsf{fk}_{G_i}\big),$$

and thus obtain the desired vector $\boldsymbol{b}^1$.

By combining these observations, a CFE scheme can be constructed by appropriately modifying the QFE scheme to support the desired functionality.

Higher-level Computation. We name $(\boldsymbol{c}, \boldsymbol{b})$ a level-0 ciphertext and $(\boldsymbol{c}^1, \boldsymbol{b}^1)$ a level-1 ciphertext, respectively. The previous discussion explains how to transform a level-0 ciphertext into a level-1 ciphertext using an encrypted functional key. While it is possible to extend this process to higher levels, the procedure becomes significantly more complex. Thus, we defer its detailed explanation to the main section.

4 Preliminaries

Notation. We use bold lowercase letters (e.g., $\boldsymbol{v}, \boldsymbol{w}$) to denote vectors. For a positive integer a, $[a]$ stands for the set $\{1, \ldots, a\}$. The notation $\|\boldsymbol{v}\|_\infty$ denotes the infinity norm of a vector $\boldsymbol{v}$. Given two vectors $\boldsymbol{v}$ and $\boldsymbol{w}$, we use $\boldsymbol{v} \| \boldsymbol{w}$ to denote their concatenation and $\langle \boldsymbol{v}, \boldsymbol{w} \rangle$ for their inner product. For any $x \in \mathbb{R}$, we denote by $\lfloor \cdot \rceil$ the closest integer to x. For every set S, $x \leftarrow S$ indicates that x is chosen uniformly at random from S.

Let $R = \mathbb{Z}[x]/(x^n + 1)$ be a polynomial ring where n is a power of two. For any $q \geq 2$, we define $R_q = R/qR = \mathbb{Z}_q[x]/(x^n + 1)$. The arrow notation is used to denote sampling elements from some distribution. In particular, for any vector $\boldsymbol{v}$ of length ℓ, we write $\boldsymbol{v} \leftarrow \mathcal{D}^\ell$ to indicate that each element of $\boldsymbol{v}$ is sampled from some distribution $\mathcal{D}$. We use a notation $\boldsymbol{\delta}_i^n$ to denote the i-th standard vector of $\mathbb{Z}^n$. $\mathbf{0}^n$ is a vector in $\mathbb{Z}^n$ consisting entirely of zeros.

4.1 Lattice-based Quadratic Functional Encryption

This section presents a lattice-based quadratic functional encryption scheme in [8]. This construction is built from a lattice-based linear functional encryption, denoted **LFE** [1, 6]. The detailed construction of **LFE** is in Appendix A.2.

We represent a quadratic polynomial F as a vector and vice versa. More precisely, for any quadratic polynomial F, there exists a set of coefficients $\{F'_i\}$ and $\{F'_{i,j}\}$ for $1 \leq i, j \leq \ell$ such that

$$F(\boldsymbol{x}) = \sum_{i,j} F'_{i,j} \cdot x_i \cdot x_j + \sum_i F'_i \cdot x_i + F_0.$$

Moreover, there exists a different representation of F, denoted by $\boldsymbol{F} = (F_{i,j})_{1 \leq i \leq j \leq \ell+1}$, satisfying

$$\langle \boldsymbol{F}, \boldsymbol{x}' \otimes \boldsymbol{x}' \rangle = \sum_{1 \leq i \leq j \leq \ell+1} F_{i,j} \cdot \boldsymbol{x}'_i \cdot \boldsymbol{x}'_j = F(\boldsymbol{x}),$$

where $\boldsymbol{x}' = (\boldsymbol{x}, 1) = (x_1, \ldots, x_\ell, 1) \in \mathbb{Z}^{\ell+1}$. Based on this representation, an ℓ-variate quadratic polynomial $F(\boldsymbol{x})$ can be expressed as another quadratic polynomial $F'(\boldsymbol{x}, 1)$ such that $F(\boldsymbol{x}) = F'(\boldsymbol{x}, 1)$ for all $\boldsymbol{x}$. This ensures that evaluating F at $\boldsymbol{m}$ is equivalent to evaluating F' at the extended vector $\boldsymbol{m}'$, i.e., $F(\boldsymbol{m}) = F'(\boldsymbol{m}')$.

In the ring setting, the quadratic polynomials we work with are represented in this vectorized form. We now provide a description of a variant of the quadratic FE [8], denoted by **QFE**. It is composed of four algorithms and achieves SIM-CPA secure given the priori bounded number of functional keys.

Let λ be the security parameter and M, ℓ, P and K be integers. Let $\mathcal{M} = \{0, 1, \ldots, M-1\}^\ell \subset R^\ell$ be a message space, and $\mathcal{F}^\ell_K \subset R^\ell$ be a set of quadratic functions of which input-length is ℓ, and each coefficient of quadratic polynomials is smaller than K.

QFE.Setup(λ, ℓ, Q, K, M) : On input the secret security parameter λ, a parameter ℓ denoting the length of message vectors and a parameter Q denoting the number of keys supported, do:

1. Choose real numbers α_0, α_1 and an integer P satisfying the security and the correctness. We defer the details later.
2. Invoke **LFE.Setup**$(\lambda, \ell + 2 + Q, P, K)$ to **LFE**.pp and **LFE**.msk.
3. Sample $\boldsymbol{u} \leftarrow R_q^{\ell+1}$, where q is a public parameter of **LFE**.pp.
4. Output $\mathsf{pp} = (\textbf{LFE}.\mathsf{pp}, \boldsymbol{u} = (u_i), \{\alpha_0, \alpha_1\}, P)$ and $\mathsf{msk} = \textbf{LFE}.\mathsf{msk}$.

QFE.KGen$(\mathsf{pp}, \mathsf{msk}, F \in \mathcal{F}^\ell_K, \mathsf{cnt} \in [Q])$: On input the public parameters pp, the master secret key msk, a function $F(\boldsymbol{x}) = \langle \boldsymbol{F}, \boldsymbol{x}' \otimes \boldsymbol{x}' \rangle$ with a counter $\mathsf{cnt} \in [Q]$, do:

1. Compute fk'_F defined as

$$\left(\sum_{1 \leq j \leq k \leq \ell+1} F_{j,k} (u_j \cdot u_k \cdot \boldsymbol{\delta}_1^{\ell+2} - u_j \cdot \boldsymbol{\delta}_{1+k}^{\ell+2} - u_k \cdot \boldsymbol{\delta}_{1+j}^{\ell+2}) \right).$$

2. Compute
$$\mathsf{fk}_F \leftarrow \mathbf{LFE.KGen}(\mathbf{LFE}.\mathsf{msk}, (\mathsf{fk}'_F, \boldsymbol{\delta}^Q_{\mathsf{cnt}})).$$
Then, it outputs $\mathsf{fk}_F \in R_q^{d_F}$, where d_F is the length of functional key.

Later, we will call an algorithm of the form
$$\mathbf{LFE.KGen}(\mathbf{LFE}.\mathsf{msk}, (\mathsf{fk}'_F, \boldsymbol{v}))$$
for some vector $\boldsymbol{v}$. We will call the algorithm $\overline{\mathbf{QFE.KGen}}(\mathsf{pp}, \mathsf{msk}, F, \boldsymbol{v})$.

$\mathbf{QFE.Enc}(\mathsf{pp}, \boldsymbol{m})$: On input public parameters pp and message vector $\boldsymbol{m} \in \mathcal{M}$, do:

1. Set $\boldsymbol{m}' = (\boldsymbol{m}, 1)$ and compute $\boldsymbol{c} = (c_i)_{i\in[\ell+1]}$ such that
$$\boldsymbol{c} = \boldsymbol{u} \cdot s_1 + P \cdot \boldsymbol{m}' + \boldsymbol{e} \in R_q^{\ell+1},$$
where $s_1 \leftarrow R_q$ and $\boldsymbol{e} \leftarrow \mathcal{D}^{\ell+1}_{R,\alpha_0 q}$.
2. Sample $\boldsymbol{\eta} = (\eta_i) \leftarrow \mathcal{D}^Q_{R,\alpha_1 q}$.
3. Sample $\boldsymbol{b} \leftarrow \mathbf{LFE.Enc}(\mathbf{LFE}.\mathsf{pp}, (s_1^2, \boldsymbol{c} \cdot s_1, \boldsymbol{\eta}))$.
4. Output $\mathsf{ct} = (\boldsymbol{c}, \boldsymbol{b})$

For a later use, we define a subroutine algorithm, called $\overline{\mathbf{QFE.Enc}}$, which takes input $(\mathsf{pp}, P \cdot \boldsymbol{m}')$ and return $\boldsymbol{c}$ in the first step.

$\mathbf{QFE.Dec}(\mathsf{pp}, \mathsf{fk}, \mathsf{ct})$: On input the public parameter pp, a function key $\mathsf{fk}_F \in R_P^{d_f}$ for polynomial $\sum_{1\le i\le j\le \ell+1} F_{i,j} \cdot x'_i \cdot x'_j$, and a ciphertext $\mathsf{ct} = (\boldsymbol{c}, \boldsymbol{b})$ with $\boldsymbol{c} = (c_i) \in R_q^{\ell+1}$, compute
$$\sum_{1\le i\le j\le \ell+1} F_{i,j} \cdot c_i \cdot c_j + \mathbf{LFE.Dec}(\mathbf{LFE}.\mathsf{pp}, \boldsymbol{b}, \mathsf{fk}_F) \tag{1}$$
and output $\mu = \left\lfloor \frac{\sum_{1\le i\le j\le \ell+1} F_{i,j}\cdot c_i\cdot c_j + \mathbf{LFE.Dec}(\mathbf{LFE}.\mathsf{pp},\boldsymbol{b},\mathsf{fk}_F)}{P^2} \right\rceil$. As in the above, we additionally define a subroutine algorithm for later use. Let $\overline{\mathbf{QFE.Dec}}(\mathsf{pp}, \mathsf{fk}, \mathsf{ct})$ be an algorithm that returns a value $\left\lfloor \frac{\sum_{1\le i\le j\le \ell+1} F_{i,j}\cdot c_i\cdot c_j + \mathbf{LFE.Dec}(\mathbf{LFE}.\mathsf{pp},\boldsymbol{b},\mathsf{fk}_F)}{P} \right\rceil$, not μ.

Correctness of the QFE. By the definition of **QFE.Dec**, we directly get
$$\sum_{1\le i\le j\le \ell+1} F_{i,j} \cdot c_i \cdot c_j + \mathbf{LFE.Dec}(\mathbf{LFE}.\mathsf{pp}, \boldsymbol{b}, \mathsf{fk}_F).$$
The remaining part depends on the correctness of $\mathbf{LFE.Dec}(\mathbf{LFE}.\mathsf{pp}, \boldsymbol{b}, \mathsf{fk}_F)$. Due to the correctness of **LFE.Dec**, we know $\mathbf{LFE.Dec}(\mathbf{LFE}.\mathsf{pp}, \boldsymbol{b}, \mathsf{fk}_F)$ returns a value
$$\sum_{1\le j\le k\le \ell+1} F_{j,k} \cdot (u_j \cdot u_k \cdot s_1^2 - u_k \cdot c_j \cdot s_1 - u_j \cdot c_k \cdot s_1) + \eta_{\mathsf{cnt}}.$$

Thus, $\sum_{1\le j\le k\le \ell+1} F_{j,k}\cdot c_j\cdot c_k + \mathbf{LFE.Dec}(\mathbf{LFE}.\mathsf{pp}, \boldsymbol{b}, \mathsf{fk}_F)$ is written as

$$\sum_{1\le j\le k\le \ell+1} F_{j,k}\cdot (c_k\cdot c_j + u_k\cdot u_j\cdot s_1^2 - u_k\cdot c_j\cdot s_1 - u_j\cdot c_k\cdot s_1) + \eta_{\mathsf{cnt}}.$$

On the other hand, due to the definition of $\boldsymbol{c}$, it holds that

$$\begin{aligned} c_k\cdot c_j - u_j\cdot c_k\cdot s_1 - u_k\cdot c_j\cdot s_1 + u_k\cdot u_j\cdot s_1^2 &= (c_k - u_k\cdot s_1)(c_j - u_j\cdot s_1)\\ &= (P\cdot m'_k + e_k)\cdot(P\cdot m'_j + e_j) \end{aligned}$$

Therefore, we get

$$\sum_{j,k} F_{j,k}\cdot (P\cdot m'_k + e_k)\cdot(P\cdot m'_j + e_j) + \eta_{\mathsf{cnt}}$$

and it can be written as $F'(P\cdot \boldsymbol{m} + \boldsymbol{e}) + \eta_i$ due to the definition of F'. Hence, the right above equation is expressed as $P^2\cdot F(\boldsymbol{m}) + e$, where $e = F'(P\cdot \boldsymbol{m}' + \boldsymbol{e}) - P^2\cdot F(\boldsymbol{m}) + \eta_{\mathsf{cnt}}$. If $\|e\|_\infty < P^2/2$, then μ equals to $F(\boldsymbol{m})$. More precisely, we set the size of $\alpha_1\cdot q$ so that η_{cnt} smudges the other terms. Hence, we set $P\cdot\alpha_1\cdot q \le P^2/2$.

Parameter Constraint. First of all, due to the correctness, we require $P\cdot\alpha_1\cdot q \le P^2/2$ and $P < q$. Next, for the noise flooding, $\alpha_1\cdot q$ satisfies that

$$\frac{\alpha_0 q\cdot L\cdot K\cdot P\cdot M}{\alpha_1 q} = \mathsf{negl}(\lambda).$$

This equation is obtained as follows: Recall $e = F'(P\cdot \boldsymbol{m}' + \boldsymbol{e}) - P^2\cdot F(\boldsymbol{m}) + \eta_{\mathsf{cnt}}$. The size of η_i should smudge the $F'(P\cdot \boldsymbol{m}' + \boldsymbol{e}) - P^2\cdot F(\boldsymbol{m})$ to hide the message relevant information. Letting $L = |\{(i,j) \ : \ 1\le j\le i\le \ell\}|$ and $M = \|\boldsymbol{m}\|_\infty$, We can give the size bound of $F'(P\cdot \boldsymbol{m}' + \boldsymbol{e}) - P^2\cdot F(\boldsymbol{m})$ as $\alpha_0 q\cdot L\cdot K\cdot P\cdot M$.

Remark 1. Although the decryption algorithm on the functional key for $F(\boldsymbol{x})$ in the scheme of [8] should take as input F for the complete description, it seems that the functional key implicitly has the function F. As the previous scheme, we omit that the decryption algorithm takes as input F as well.

5 Composable Functional Encryption

In this section, we formally introduce the notion of composable functional encryption (CFE), which generalizes functional encryption to support sequential composition of function evaluations over encrypted data. A CFE scheme enables a ciphertext to be iteratively transformed by applying encrypted functions without requiring decryption of intermediate results. We define the syntax of CFE, its correctness condition, and a full simulation-based security notion adapted from [8]. We omit explicit inputs pp and msk in the algorithms when clear from context.

While a new lower bound for compact lattice-based FE schemes was recently demonstrated in [50], our work just explores Q-bounded FE schemes. This specific focus on Q-boundedness means that our schemes are not directly subject to the lower bound in [50] which targets general compact FE schemes, thereby avoiding a conflict with their results.

Definition 1 (CFE). *Let $\mathcal{F}$ be a function space and $\mathcal{M}$ a message space. For any integer $T > 0$, a (Q-bounded) T-leveled public key composable functional encryption scheme is a tuple of PPT algorithms:*

- $\textbf{\textit{Setup}}(\lambda, \mathcal{M}, \mathcal{F}, Q, T)$*: Outputs* pp *and* msk.
- $\textbf{\textit{Enc}}^0(\mathsf{pp}, m \in \mathcal{M})$*: Outputs a level-0 ciphertext* ct^0.
- $\textbf{\textit{EKGen}}^t(\mathsf{pp}, \mathsf{msk}, F \in \mathcal{F}, \mathsf{cnt}(t, F, E) \in [Q])$*: Outputs a level-*$(t+1)$ *encrypted functional key* $\mathsf{fk}_{\textbf{\textit{Enc}}^{t+1} \circ F}$ *for* $t \in [T-1]$.
- $\textbf{\textit{EDec}}^t(\mathsf{pp}, \mathsf{fk}_{\textbf{\textit{Enc}}^{t+1} \circ F}, \mathsf{ct}^t)$*: Outputs a level-*$(t+1)$ *ciphertext* ct^{t+1} *for* $t \in [T-1]$.
- $\textbf{\textit{KGen}}^t(\mathsf{pp}, \mathsf{msk}, F \in \mathcal{F}, \mathsf{cnt}(t, F, P) \in [Q])$*: Outputs a level-t functional key* fk_F^t *for* $t \in [T]$.
- $\textbf{\textit{Dec}}^t(\mathsf{pp}, \mathsf{fk}_F^t, \mathsf{ct}^t)$*: Outputs a plaintext* m' *for* $t \in [T]$,

where $\mathsf{cnt} : \mathbb{Z} \times \mathcal{F} \times \{E, P\} \mapsto [Q]$ *is an injective function.*

A scheme $\textbf{\textit{CFE}} = (\textbf{\textit{Setup}}, \textbf{\textit{Enc}}^0, \textbf{\textit{EKGen}}^t, \textbf{\textit{EDec}}^t, \textbf{\textit{KGen}}^t, \textbf{\textit{Dec}}^t)$ *is correct if for any $m \in \mathcal{M}$, any function sequence $F_1, F_2, \ldots, F_t \in \mathcal{F}$ with $t \in [T]$,*

$$\mathsf{ct}^0 := \textbf{\textit{Enc}}^0(m),$$

$$\mathsf{ct}^t := \textbf{\textit{EDec}}^{t-1}(\mathsf{fk}_{\textbf{\textit{Enc}}^t \circ F_t}, \mathsf{ct}^{t-1}) = \textbf{\textit{Enc}}^t(F_t \circ \cdots \circ F_1(m)) \quad \text{for } t \in [T],$$

we also have:

$$\textbf{\textit{Dec}}^t(\mathsf{fk}_{F_t}^t, \mathsf{ct}^t) = (F_t \circ \cdots \circ F_1)(m) \quad \text{for } t \in [T]$$

except with negligible probability.

As in leveled homomorphic encryption schemes [19, 22, 23], our leveled construction requires all computations to originate from ciphertexts initially produced by $\textbf{Enc}^0(\cdot)$. A level t ciphertext ct^t (for $t \geq 1$) is then achieved through iterative applications of evaluation functions, denoted as $\textbf{EDec}^i(\cdot)$.

Based upon the security definition presented in [8, 31], we define simulation-based security for our Q-bounded CFE construction. To circumvent the impossibility result for simulation-based security established in [17], our definition—consistent with the approach in [8,31]—prohibits the adversary from making key queries after observing the challenge ciphertext.

Definition 2 (Full simulation-based security of CFE). *Let* $\textbf{\textit{CFE}}$ *be a Q-bound T-level composable functional encryption scheme for a function family $\mathcal{F}$. For every stateful PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a stateful PPT simulator $\mathcal{S}$, consider the following experiments:*

$\mathsf{Exp}^{\mathsf{Real}}_{\mathcal{F},\mathcal{A}}(1^\lambda)$:	$\mathsf{Exp}^{\mathsf{Ideal}}_{\mathcal{F},\mathcal{S}}(1^\lambda)$:
(1) $(\mathsf{pp}, \mathsf{msk}) \leftarrow \textbf{\textit{Setup}}(\lambda, \mathcal{M}, \mathcal{F}, Q, T)$	*(1)* $(\mathsf{pp}^{\mathcal{S}}, \mathsf{msk}^{\mathcal{S}}) \leftarrow \textbf{\textit{Setup}}^{\mathcal{S}}(\lambda, \mathcal{M}, \mathcal{F}, Q, T)$
(2) $\mathbf{F} = \{F_i\}_{i \le Q} \leftarrow \mathcal{A}_1(1^\lambda, \mathsf{pp}, \mathcal{F})$	*(2)* $\mathbf{F} = \{F_i\}_{i \le Q} \leftarrow \mathcal{A}_1(1^\lambda, \mathsf{pp}^S, \mathcal{F})$
(3) $(m^*, st) \leftarrow \mathcal{A}_1^{\textbf{\textit{EKGen}}^t(\mathsf{msk},\cdot), \textbf{\textit{KGen}}^t(\mathsf{msk},\cdot)}(\mathsf{pp}, \mathbf{F})$	*(3)* $(m^*, st) \leftarrow \mathcal{A}_1^{\textbf{\textit{EKGen}}^{t,\mathcal{S}}(\mathsf{msk}^{\mathcal{S}},\cdot), \textbf{\textit{KGen}}^{t,\mathcal{S}}(\mathsf{msk}^{\mathcal{S}},\cdot)}(\mathsf{pp}^{\mathcal{S}}, \mathbf{F})$
(4) $\mathsf{ct} \leftarrow \textbf{\textit{Enc}}^0(\mathsf{pp}, m^*)$	*(4)* $\mathsf{ct}^* \leftarrow \textbf{\textit{Enc}}^{\mathcal{S}}(\mathcal{V}, \lvert m^* \rvert)$, *where* $\mathcal{V}$ *denotes the simulated leakage.*
(5) $\alpha \leftarrow \mathcal{A}_2(\mathsf{pp}, \mathsf{ct}, st)$	*(5)* $\alpha \leftarrow \mathcal{A}_2(\mathsf{pp}^{\mathcal{S}}, \mathsf{ct}^*, st)$
(6) Output (m^*, α)	*(6) Output* (m^*, α)

More specific, $\mathcal{V}$ *contains the following functions, functional keys:*

$$\left\{ \{F_i\}_{\le Q}, \left\{ \{\mathsf{fk}^{t_k}_{F_k}\}_k, \left\{ \mathsf{fk}^{t_j}_{\textbf{\textit{Enc}} \circ F_j} \right\}_j \right\}_{j+k \le Q} \right\}.$$

$\mathcal{V}$ *also contains the results of decrypting composite functions using the keys* $\mathsf{fk}^{t_k}_{F_k}$ *and* $\mathsf{fk}^{t_j}_{\textbf{\textit{Enc}} \circ F_j}$.

We say that ***CFE*** *is fully simulation-secure if for any stateful PPT adversary* $\mathcal{A}$*, there exists an admissible simulator* $\mathcal{S}$ *such that the following two distributions are computationally indistinguishable.*

$$\left\{ \mathsf{Exp}^{\mathsf{Real}}_{\mathcal{F},\mathcal{A}}(1^\lambda) \right\}_{\lambda \in \mathbb{N}} \approx_c \left\{ \mathsf{Exp}^{\mathsf{Ideal}}_{\mathcal{F},\mathcal{S}}(1^\lambda) \right\}_{\lambda \in \mathbb{N}}.$$

The indistinguishability-based (IND) security for Q-bounded **CFE** is defined in Appendix B.

6 Composable FE for Quadratic Polynomials

This section presents our main construction: a collusion-bounded, lattice-based, composable FE scheme for quadratic polynomials. Given a quadratic function F, the scheme generates a functional key $\mathsf{fk}_{\textbf{Enc} \circ F}$ such that $\textbf{Dec}(\mathsf{fk}_{\textbf{Enc} \circ F}, \mathsf{ct})$ returns a ciphertext of $F(\boldsymbol{m})$, provided that ct is a ciphertext of $\boldsymbol{m}$. Our approach is adapted from the lattice-based quadratic scheme due to Agrawal and Rosen (AR) [8]. For the underlying **LFE** scheme, we employ the construction of Abdalla et al. [1] due to the simplicity. However, any LFE scheme satisfying specific properties (as listed in Appendix A.2) can be used to instantiate our CFE scheme.

The proposed composable FE scheme comprises six algorithms: **Setup**, **Enc**, **EKGen**, **EDec**, **KGen**, and **Dec**. Each composition level is denoted by t, and the algorithm pair $((\textbf{E})\textbf{KGen}, (\textbf{E})\textbf{Dec})$ depends on this level, denoted by $((\textbf{E})\textbf{KGen}^t, (\textbf{E})\textbf{Dec}^t)$. We initialize the composition level at $t = 0$ and observe that for $t \ge 2$, the $((\textbf{E})\textbf{KGen}^t, (\textbf{E})\textbf{Dec}^t)$ algorithms remain identical to those at level 1.

Consequently, we describe these algorithms only up to level 1, which is sufficient for levels 1 and 2.

At each level t, we employ a **QFE** scheme, called a level-t **QFE** scheme or simply $\mathbf{QFE}^t$, where the modulus is adjusted as $q_t = q/P^t$. To enable composite operations across different levels, we utilize a reverse-modulus switching technique. For example, to compute encodings over R_q and vectors in R_{q_1}, we multiply vectors in R_{q_1} by P, allowing them to be interpreted as elements in R_q. This technique frequently appears in the **EKGen** algorithm.

We further note that the number of collusion bounds is pre-determined and denoted by Q. Let ℓ_t represent the number of collusion bounds at level-t (with $t \geq 1$), which can be categorized into two types: one for $\mathbf{KGen}^t$ and the other for $\mathbf{EKGen}^t$. We denote these as $\ell_t^{(P)}$ and $\ell_t^{(E)}$, respectively. If T is the last level, these parameters can be freely chosen as long as they satisfy the constraint $\sum_{1 \leq t \leq T} \ell_t^{(E)} + \ell_t^{(P)} \leq Q$ since the security mainly depends on Q. Therefore, we omit the superscripts (E) and (P) for simplicity. We will describe a counter $\ell_t = \ell_t^{(E)} + \ell_t^{(P)}$ of collusion bounds only depending on a level t.

This CFE construction builds upon the AR **QFE** in Section 4.1 and linear FE schemes **LFE** in Appendix A.2. For the instantiation, we employ three QFE schemes, each associated with an LFE scheme, along with an additional LFE scheme. We denote these QFE and LFE schemes as $\mathbf{QFE}^t$ and $\mathbf{LFE}^t$, respectively, for $t \in \{0, 1, 2\}$. The **LFE** scheme without a superscript refers to the additional **LFE** scheme. For simplicity, we omit the pp and msk parameters from algorithm inputs when there is no confusion.

$\mathbf{Setup}(\lambda, \ell, \{\ell_t\}_{t \in [2]}, Q, M, K)$: On input a security parameter λ, a parameter ℓ denoting the length of message vectors, ℓ_t denoting the number of evaluated functions at level t satisfying $\sum_{i=1}^{2} \ell_i = Q$, and a parameter Q denoting the number of functional keys supported[5], do:

- Set real numbers $\{\alpha_t\}_{t \leq 2}$ and integers P and q to satisfy the security and correctness. We defer details later. (See Appendix C).
- Compute $M_{i+1} = L_i \cdot K \cdot M_i$ for $i \in \{0, 1\}$, where $M_0 = M, \ell_0 = \ell$ and $L_i = (\ell_i + 1)^2$.
- For $0 \leq t \leq 2$,

$$(\mathbf{QFE}^t.\mathsf{pp}, \mathbf{QFE}^t.\mathsf{msk}) \leftarrow \mathbf{QFE.Setup}(\lambda, \ell_t, 3Q + 2, M_2, K; q_t, \alpha_0, \alpha_t).^{6}$$

[5] The parameters Q, K, and M are chosen to guarantee the security and correctness of our CFE construction. The details of the parameter selection are provided in Appendix C.

[6] Here, the inputs q_t, α_0, α_t after the semicolon in the invocation of **QFE.Setup** are pre-determined. Specifically, they are selected beforehand to satisfy the conditions in Appendix C. While these could notionally be internal choices of the **QFE.Setup** procedure, we explicitly list them here as inputs to signify their pre-selected nature.

– $(\mathbf{LFE}.\mathsf{pp}, \mathbf{LFE}.\mathsf{msk}) \leftarrow \mathbf{LFE.Setup}(\lambda, d_\zeta, q, q; q^2)$[7] where $d_\zeta = (Q+1)^2 + (Q+1) + 1$.
– Sample $\boldsymbol{u}^0 \leftarrow R_q^{\ell+1}$.
– Sample $u_{i,h}^t \leftarrow R_{q_t}$ for $i \in [\ell_t]$, $h \in [Q+1]$, and $t \in [2]$.
– Output pp and msk as follows.

$$\mathsf{pp} = (\{\mathbf{QFE}^t.\mathsf{pp}\}_t, \mathbf{LFE}.\mathsf{pp}, \boldsymbol{u}^0, \{u_{i,h}^t\}, Q, \{\alpha_t\})$$
$$\mathsf{msk} = (\{\mathbf{QFE}^t.\mathsf{msk}\}_t, \mathbf{LFE}.\mathsf{msk})$$

Note that if the required number of compositions is only one, i.e., $T = 1$, we only need to consider the case $t \in [1]$.
$\mathbf{Enc}^0(\mathsf{pp}, \boldsymbol{m} \in \mathcal{P})$: To encrypt a message vector $\boldsymbol{m} \in \mathcal{P}$, do the followings:

– For $h \in [Q+1]$, sample $\zeta_h \leftarrow \mathcal{D}_{R,\alpha_0 q}$ and let $\boldsymbol{\zeta} = (\zeta_h) \in R^{Q+1}$.
– For $i \in [2\ell_t]$, sample $\eta_i^t \leftarrow \mathcal{D}_{R,\alpha_t q}$ and let $\boldsymbol{\eta} = (\eta_i^t) \in R^{2\ell_1} \times R^{2\ell_2} \simeq R^{2Q}$.
– Construct an extended vector $\boldsymbol{m}' := (\boldsymbol{m}, 1)$
– Sample $s_0 \leftarrow \mathcal{D}_{R,\alpha_0 q}$, $\boldsymbol{e} \leftarrow \mathcal{D}_{R,\alpha_0 q}^{\ell+1}$, and compute

$$\boldsymbol{c} = \boldsymbol{u}^0 \cdot s_0 + \boldsymbol{e} + P \cdot \boldsymbol{m}'$$

– Sample $\boldsymbol{b} \leftarrow \mathbf{LFE}^0.\mathbf{Enc}(s_0^2, \boldsymbol{c} \cdot s_0, \boldsymbol{\zeta}, \boldsymbol{\eta}, 1)$ and $\boldsymbol{a} \leftarrow \mathbf{LFE.Enc}(\boldsymbol{\zeta} \otimes \boldsymbol{\zeta}, \boldsymbol{\zeta}, 1)$.
– Output $\mathsf{ct} = (\boldsymbol{c}, \boldsymbol{b}, \boldsymbol{a})$.

Here, we note that $\mathbf{LFE}^t$ is automatically derived from $\mathbf{QFE}^t.\mathbf{Setup}$ for every $t \in \{0, 1, 2\}$. Throughout this section, the algorithms $\mathbf{LFE}^t.\mathbf{Enc}$ and $\mathbf{LFE}^t.\mathbf{Dec}$ will frequently appear. For each $t \geq 1$, an encryption algorithm at level t $\mathbf{Enc}^t(\mathsf{pp}, \boldsymbol{m})$ does not exist. Thus, to obtain level-t ciphertext, one should compute $\mathbf{EDec}^{t-1}(\mathsf{fk}_{\mathbf{EKGen}^{t-1} \circ Id}, \mathsf{ct}^{t-1})$, where ct^{t-1} is a level-$(t-1)$ ciphertext, and Id is the identity function.

Furthermore, the notation $\overline{\mathbf{QFE}^t.\mathbf{Enc}}(\cdot)$ and $\overline{\mathbf{QFE}^t.\mathbf{KGen}}(\cdot)$ will also be used frequently. More precisely, $\mathbf{QFE}^t.\mathbf{Enc}(\cdot)$ returns a pair $(\boldsymbol{c}, \boldsymbol{b})$, but $\overline{\mathbf{QFE}^t.\mathbf{Enc}}(\cdot)$ returns a single vector $\boldsymbol{c}$. The $\mathbf{QFE}^t.\mathbf{KGen}$ algorithm consists of two steps: first, computing an intermediate value, denoted as fk'_F; and second, generating the functional key fk_F by running $\mathbf{LFE}^t.\mathbf{KGen}$ with inputs fk'_F and an integer vector $\boldsymbol{\delta}_i^Q$, where it is the i-th standard unit vector in $\mathbb{Z}^Q$. In some cases, we use a different integer vector $\boldsymbol{v}$ instead of $\boldsymbol{\delta}_i^Q$, explicitly emphasizing the choice of $\boldsymbol{v}$. This variant of the key generation algorithm is denoted as $\overline{\mathbf{QFE}^t.\mathbf{KGen}}(\cdot)$. Similarly, for the special purpose, the notation $\overline{\mathbf{QFE.Dec}}(\mathsf{pp}, \mathsf{fk}, \mathsf{ct})$ will be used. The algorithm returns a value

$$\left\lfloor \frac{\sum_{1 \leq i \leq j \leq \ell+1} F_{i,j} \cdot c_i \cdot c_j + \mathbf{LFE.Dec}(\mathbf{LFE}.\mathsf{pp}, \boldsymbol{b}, \mathsf{fk}_F)}{P} \right\rceil,$$

not $\mu = \left\lfloor \frac{\sum_{1 \leq i \leq j \leq \ell+1} F_{i,j} \cdot c_i \cdot c_j + \mathbf{LFE.Dec}(\mathbf{LFE}.\mathsf{pp}, \boldsymbol{b}, \mathsf{fk}_F)}{P^2} \right\rceil$.

[7] Similarly, we pre-fix q^2 as a ciphertext modulus of $\mathbf{LFE}$.

Below, we introduce key generation algorithms that produce either standard functional keys or encrypted functional keys. We describe four key generation algorithms—KGen^0, KGen^1, EKGen^0, and EKGen^1—along with their corresponding decryption algorithms.

For ease of exposition, we describe a stateful key generation algorithm. To achieve a stateless CFE, the key generation algorithm is modified such that the key generator picks a random binary vector of size $(2Q)^2$ and weight λ, replacing the i-th standard unit vector $\boldsymbol{\delta}_i^{2Q}$. Subsequently, the decryption algorithm computes a random subset sum of $(2Q)^2$ noise terms associated with each key. This approach guarantees that each decryption operation benefits from at least one fresh noise term, a feature for stateless CFE. It is well-known technique, called cover-free method [8,33]. For more details, we refer to the previous work [8,33].

6.1 KGen^0: Generate a level-0 functional key fk_F

6.1.1 KGen^0 Intuition

For $F : R^\ell \to R$, KGen^0 generates a functional key fk_F. This process is essentially the same as the previous lattice-based QFE scheme described in Section 4.1. However, the new algorithm requires additional input length due to composability.

For instance, as discussed earlier, the previous key generation algorithm consists of two steps: first, computing an intermediate vector, denoted as fk'_F; and second, generating the functional key fk_F by running $\mathsf{LFE.KGen}$ with inputs fk'_F and an integer vector $\boldsymbol{\delta}_i^Q$, where $\boldsymbol{\delta}_i^Q$ is the i-th standard unit vector in $\mathbb{Z}^Q$.

In contrast, our approach follows a similar two-step process but with a modification in the second step. After computing fk'_F, we generate the functional key fk_F by running $\mathsf{LFE}^0\mathsf{.KGen}$ with inputs fk'_F and an extended integer vector $(\mathbf{0}^{Q+1}, \boldsymbol{\delta}_i^{2Q+1}) \in \mathbb{Z}^{3Q+2}$, where $\mathbf{0}^{Q+1}$ is a zero vector in $\mathbb{Z}^{Q+1}$. The zero vector will be used to preserve the functionality even though masking elements exist.

6.1.2 KGen^0 Construction

Given a quadratic polynomial $F \in \mathcal{F}_K^\ell$ and a counter $\mathsf{cnt} \in [\ell_1]$, KGen^0 is conducted as follows.

- $\mathsf{KGen}^0(F \in \mathcal{F}_K^\ell, \mathsf{cnt} \in [\ell_1])$:
  1. Compute a vector fk'_F is defined as:

$$\left(\sum_{1 \le j \le k \le \ell+1} F_{j,k}(u_j^0 \cdot u_k^0 \cdot \boldsymbol{\delta}_1^{\ell+2} - u_j^0 \cdot \boldsymbol{\delta}_{1+k}^{\ell+2} - u_k^0 \cdot \boldsymbol{\delta}_{1+j}^{\ell+2}) \right)$$

     where $\boldsymbol{\delta}_i^n \in \mathbb{Z}^n$ is the i-th standard unit vector in $\mathbb{Z}^n$.
  2. Compute $\mathsf{fk}_F^0 = \mathsf{LFE}^0\mathsf{.KGen}(\mathsf{fk}'_F, \mathbf{0}^{Q+1}, \boldsymbol{\delta}_{\mathsf{cnt}}^{2Q+1})$.

- $\mathbf{Dec}^0(\mathsf{fk}_F^0, \mathsf{ct}^0)$: On a functional key fk_F^0 for polynomial $\sum_{1\le j\le k\le \ell+1} F_{j,k}\cdot x'_j\cdot x'_k$, and a level-0 ciphertext $\mathsf{ct}^0 = (\boldsymbol{c}, \boldsymbol{b}, \boldsymbol{a})$, output

$$\mu = \left\lfloor \frac{F'(\boldsymbol{c}) + \mathbf{LFE}^0.\mathbf{Dec}(\boldsymbol{b}, \mathsf{fk}_F^0)}{P^2} \right\rceil.$$

6.1.3 Decryption Correctness of KGen^0

This part is straightforward due to the correctness of **QFE**. By the definition of $\mathbf{Dec}^0$, we directly get

$$\sum_{1\le j\le k\le \ell+1} F_{j,k}\cdot c_j\cdot c_k + \mathbf{LFE}^0.\mathbf{Dec}(\boldsymbol{b}, \mathsf{fk}_F^0).$$

The remaining part depends on the correctness of $\mathbf{LFE}^0.\mathbf{Dec}(\boldsymbol{b}, \mathsf{fk}_F^0)$. Due to the correctness of $\mathbf{LFE}^0.\mathbf{Dec}$, we know $\mathbf{LFE}^0.\mathbf{Dec}(\boldsymbol{b}, \mathsf{fk}_F^0)$ returns a value

$$\sum_{1\le j\le k\le \ell+1} F_{j,k}\cdot (u_j^0\cdot u_k^0\cdot s_0^2 - u_k^0\cdot c_j\cdot s_0 - u_j^0\cdot c_k\cdot s_0) + \eta_{\mathsf{cnt}}.$$

The remaining part is exactly the same as a decryption correctness of **QFE**.

Parameter Constraint. The constraint is also the same as **QFE** in Section 4.1. For the noise flooding, $\alpha_1 q$ satisfies

$$\frac{\alpha_1 q\cdot(\ell+1)^2\cdot K\cdot P\cdot M}{\alpha_1 q} = \mathsf{negl}(\lambda).$$

Then, for the correctness, $\alpha_1 q \le P/2$.

6.2 EKGen^0: Generate a level-1 encrypted functional key $\mathsf{fk}_{\mathbf{Enc}^1\circ \boldsymbol{F}}$

This algorithm generates a functional key $\mathsf{fk}_{\mathbf{Enc}^1\circ F}$ so that $\mathbf{Dec}^0(\mathsf{fk}_{\mathbf{Enc}^1\circ F}, \mathbf{Enc}^0(\boldsymbol{m}))$ outputs $\mathbf{Enc}^1(F(\boldsymbol{m}))$. Although this algorithm is primarily designed for a single function $F : R^\ell \to R$, we extend it to handle multiple functions $\boldsymbol{F} = \{F_i\}_{i=1}^{\ell_1}$. This extension enables a vector encryption output in its initial phase.

Thus, the decryption procedure $\mathbf{Dec}^0(\mathsf{fk}_{\mathbf{Enc}^1\circ F}, \mathbf{Enc}^0(\boldsymbol{m}))$ should produce three components as a level-1 ciphertext.

6.2.1 EKGen^0 Intuition

A level-1 ciphertext is of the form $(\boldsymbol{c}^1, \boldsymbol{b}^1, \boldsymbol{a})$: First of all, $\boldsymbol{a}$ is a re-used term from level-0 ciphertext of the form $(\boldsymbol{c}^0, \boldsymbol{b}^0, \boldsymbol{a})$. The other components $\boldsymbol{c}^1 = (c_i^1)$ and $\boldsymbol{b}^1$ satisfy

$$c_i^1 = \sum_{h=1}^{Q+1} c_{i,h}^1\cdot \zeta_h + e_i^1 + P\cdot \hat{F}_i(\boldsymbol{m}) \quad \text{for } i\in[\ell_1+1],$$
$$\boldsymbol{b}^1 = \mathbf{LFE}^1.\mathbf{Enc}\big(s_1^2, \boldsymbol{c}^1\cdot s_1, \boldsymbol{\zeta}, \boldsymbol{\eta}, 1\big),$$

where $e_i^1 \leftarrow \mathcal{D}_{R,\alpha_1 q}$ and $\hat{F}_i(\boldsymbol{m})$ is defined as $\hat{F}_i(\boldsymbol{m}) = F_i(\boldsymbol{m})$ for $i \in [\ell_1]$ and $\hat{F}_{\ell_1+1}(\boldsymbol{m}) = 1$[8]. For each i, by definition of $c_i^1 = \langle (c_{i,1}^1, \ldots, c_{i,Q+1}^1), \boldsymbol{\zeta} \rangle + e_i^1 + P \cdot \hat{F}_i(\boldsymbol{m})$, it could be an encoding of a quadratic evaluation $P \cdot \hat{F}_i(\boldsymbol{m})$. Thus, for each i, it would be generated using $\mathsf{Dec}(\mathsf{QFE.KGen}(\hat{F}_i, \cdot), \cdot)$.

On the other hand, the process of generating $\boldsymbol{b}^1$ is quite complex. The malleability of $\mathsf{LFE}^1.\mathsf{Enc}$ enables us that $\boldsymbol{b}^1$ can be represented as:

$$\mathsf{LFE}^1.\mathsf{Enc}\big(s_1^2, \mathbf{0}^{\ell_1+1}, \boldsymbol{\zeta}, \boldsymbol{\eta}, 1\big) + P \cdot \big(0, 0, \boldsymbol{c}^1 \cdot s_1, \mathbf{0}^{Q+1}, \mathbf{0}^{2Q}, 0\big).^{9}$$

Since c_i^1 contains terms involving ζ_h and η_i, we can decompose $P \cdot \big(0, 0, \boldsymbol{c}^1 \cdot s_1, \mathbf{0}^{Q+1}, \mathbf{0}^{2Q}, 1\big)$ into a few subcomponents based on their purpose.

More precisely, the following decomposition holds:

$$\begin{aligned}
\mathsf{LFE}^1.\mathsf{Enc}\big(s_1^2, \boldsymbol{c}^1 \cdot s_1, \boldsymbol{\zeta}, \boldsymbol{\eta}, 1\big) = &\sum_{h_1=1}^{\ell_1} P \cdot \hat{F}_{h_1}(\boldsymbol{m}) \cdot \big(0, 0, s_1 \cdot \boldsymbol{\delta}_{h_1}^{\ell_1+1}, \mathbf{0}^{3Q+2}\big) \\
&+ \sum_{h_2=1}^{Q+1} \zeta_{h_2} \cdot \mathsf{LFE}^1.\mathsf{Enc}\big(0, s_1 \cdot c_{i,h_2}^1, \boldsymbol{\delta}_{h_2}^{Q+1}, \mathbf{0}^{2Q+1}\big) \\
&+ \sum_{h_3=1}^{\ell_1} \eta_{h_3}^1 \cdot \mathsf{LFE}^1.\mathsf{Enc}\big(0, s_1 \cdot \boldsymbol{\delta}_{h_3}^{\ell_1+1}, \mathbf{0}^{Q+1}, \boldsymbol{\delta}_{h_3}^{2Q}, 0\big) \\
&+ \sum_{h_3=1}^{\ell_1} \eta_{\ell_1+h_3}^1 \cdot \mathsf{LFE}^1.\mathsf{Enc}\big(0, \mathbf{0}^{\ell_1+1}, \mathbf{0}^{Q+1}, \boldsymbol{\delta}_{h_3}^{2Q}, 0\big) \\
&+ \sum_{h_4=1}^{2\ell_2} \eta_{h_4}^2 \cdot \mathsf{LFE}^1.\mathsf{Enc}\big(0, \mathbf{0}^{\ell_1+1}, \mathbf{0}^{Q+1}, \boldsymbol{\delta}_{2\ell_1+h_4}^{2Q}, 0\big) \\
&+ \mathsf{LFE}^1.\mathsf{Enc}\big(s_1^2, \mathbf{0}^{\ell_1+1}, \mathbf{0}^{Q+1}, \mathbf{0}^{2Q}, 1\big).
\end{aligned}$$

Each term in this decomposition serves a distinct purpose:

- The first term (index h_1) is used to evaluate the quadratic polynomial $F(\boldsymbol{m})$.
- The second term (index h_2) corresponds to the masking term $\boldsymbol{\zeta}$.
- The third term (index h_3) represents $\eta_{h_3}^1$, which is crucial for *hiding secret information* via the noise flooding technique. This step is a key component in ensuring the security of our CFE. Notably, only ℓ_1 coordinates are used for this purpose.
- The fourth term (index h_4), along with the final term, ensures that the ciphertext maintains its format.

As a consequence, there is a quadratic polynomial $G(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ such that $G(\boldsymbol{m}, \boldsymbol{\zeta}, \boldsymbol{\eta}) = \boldsymbol{b}^1$, and it can be instantiated using $\mathsf{QFE.KGen}$. We now discuss the detailed construction of EKGen^0.

[8] $\hat{F}$ is used to describe a non-homogeneous quadratic polynomial after the composition.

[9] For any $\boldsymbol{x} \in R^x$ with some $x \in \mathbb{N}$, $\mathsf{LFE}^1.\mathsf{Enc}(\boldsymbol{x})$ is a ring vector in R^{x+1}. To enable this decomposition, we pad 0 at the front of the vector.

6.2.2 EKGen^0 Construction

Given a count $\mathsf{cnt} \in [\ell_2]$, we consider a multivariate function $F_{\mathsf{cnt}} : R^\ell \to R$ as a KGen^0 algorithm. However, for simplicity and efficiency, we set $\mathsf{cnt} = \ell_1$ and describe a method for the simultaneous evaluation of ℓ_1 functions. Consequently, this approach allows us to evaluate $\boldsymbol{F} = \{F_i\}_{i=1}^{\ell_1}$ using a single EKGen^0 algorithm.

Given a set of ℓ_1 quadratic functions $\boldsymbol{F} := \{F_i\}_{i=1}^{\ell_1} \subset \mathcal{F}_K^\ell$, EKGen^0 is conducted as follows.

- $\mathsf{EKGen}^0(\{F_i\}_{i=1}^{\ell_1} \subset \mathcal{F}_K^\ell, \mathsf{cnt} = \ell_1))$:
  1. For $i \in [\ell_1 + 1]$, do:
     - Sample $s_1 \leftarrow \mathcal{D}_{R,\alpha_0 q_1}$ and $e_{i,h} \leftarrow \mathcal{D}_{R,\alpha_0 q_1}$ for each $h \in [Q+1]$. Compute encodings of zero $\boldsymbol{c}_i^1 = (c_{i,h}^1) \in R_{q_1}^{Q+1}$ such that
     $$c_{i,h}^1 = u_{i,h}^1 \cdot s_1 + e_{i,h}^1,$$
     where $u_{i,h}^1$ is an element given in pp.
     - For any $\boldsymbol{x} \in R^\ell$, define a function
     $$\hat{F}_i(\boldsymbol{x}) = \begin{cases} F_i(\boldsymbol{x}) & \text{if } i \in [\ell_1] \\ 1 & \text{if } i = \ell_1 + 1 \end{cases}$$
     - Sample $\hat{\boldsymbol{c}}_i \leftarrow \overline{\mathsf{QFE}^0.\mathsf{KGen}}(\hat{F}_i, P \cdot \boldsymbol{c}_i^1, \boldsymbol{\delta}_i^{2Q}, 0)$ for $i \in [\ell_1]$
     - Sample $\hat{\boldsymbol{c}}_{\ell_1+1} \leftarrow \overline{\mathsf{QFE}^0.\mathsf{KGen}}(\hat{F}_{\ell_1+1}, P \cdot \boldsymbol{c}_{\ell_1+1}^1, \boldsymbol{0}^{2Q}, 0)$.
  2. Let $d^0 = \ell_1 + 5 + 2Q$. For $i \in [d^0]$ do:
     - For $h_1 \in [\ell_1 + 1], h_2 \in [Q+1], h_3, h_4 \in [\ell_1]$ and $h_5 \in [2\ell_2]$, compute $\left\{\boldsymbol{b}^{(0)}, \boldsymbol{b}_{h_i}^{(i)}\right\}_{i \le 5} \subset R_{q_1}^{d^0}$ defined as follows.
     $$\begin{aligned}
     \boldsymbol{b}^{(0)} &\leftarrow P \cdot \mathsf{LFE}^1.\mathsf{Enc}(s_1^2, \boldsymbol{0}^{\ell_1+1}, \boldsymbol{0}^{Q+1}, \boldsymbol{0}^{2Q}, 1) \\
     \boldsymbol{b}_{h_1}^{(1)} &\leftarrow P \cdot \mathsf{LFE}^1.\mathsf{Enc}(0, s_1 \cdot \boldsymbol{\delta}_{h_1}^{\ell_1+1}, \boldsymbol{0}^{Q+1}, \boldsymbol{0}^{2Q}, 0) \\
     \boldsymbol{b}_{h_2}^{(2)} &\leftarrow P \cdot \mathsf{LFE}^1.\mathsf{Enc}(0, s_1 \cdot \boldsymbol{c}_{i,h_2}, \boldsymbol{\delta}_{h_2}^{Q+1}, \boldsymbol{0}^{2Q}, 0) \\
     \boldsymbol{b}_{h_3}^{(3)} &\leftarrow P \cdot \mathsf{LFE}^1.\mathsf{Enc}(0, s_1 \cdot \boldsymbol{\delta}_{h_3}^{\ell_1+1}, \boldsymbol{0}^{Q+1}, \boldsymbol{\delta}_{h_3}^{2Q}, 0) \\
     \boldsymbol{b}_{h_4}^{(4)} &\leftarrow P \cdot \mathsf{LFE}^1.\mathsf{Enc}(0, \boldsymbol{\delta}_{h_4}^{\ell_1+1}/P^2, \boldsymbol{0}^{Q+1}, \boldsymbol{\delta}_{h_4}^{2Q}, 0) \\
     \boldsymbol{b}_{h_5}^{(5)} &\leftarrow P \cdot \mathsf{LFE}^1.\mathsf{Enc}(0, \boldsymbol{0}^{\ell_1+1}, \boldsymbol{0}^{Q+1}, \boldsymbol{\delta}_{2\ell_1+h_5}^{2Q}, 0),
     \end{aligned}$$
     where $\boldsymbol{c}_{i,h_2} = (c_{1,h_2}, \ldots, c_{\ell_1+1,h_2})$. We let $\boldsymbol{b}_{h_1,j}^{(1)}$ denote the binary decomposition of a vector such that $\boldsymbol{b}_{h_1}^{(1)} = \sum_j 2^j \cdot \boldsymbol{b}_{h_1,j}^{(1)}$. According to the correctness constraint of the QE, operations should be performed on each $\boldsymbol{b}_{h_1,j}^{(1)}$. However, for simplicity of exposition, we use the notation $\boldsymbol{b}_{h_1}^{(1)}$.

- For $\boldsymbol{x}_1 \in R^{\ell}, \boldsymbol{y} \in R^{Q+1}, \boldsymbol{z}, \boldsymbol{w} \in R^{\ell_1}$ and $\boldsymbol{v} \in R^{2\ell_2}$, define functions $G_1(\boldsymbol{x}), G_2(\boldsymbol{y}), G_3(\boldsymbol{z}), G_4(\boldsymbol{w}), G_5(\boldsymbol{w})$ and G_0:

$$\begin{aligned} G_1(\boldsymbol{x}) &= \sum_{h_1 \in [\ell_1+1]} \boldsymbol{b}^{(1)}_{h_1} \cdot \hat{F}_{h_1}(\boldsymbol{x}), & G_4(\boldsymbol{w}) &= \sum_{h_3 \in [\ell_1]} \boldsymbol{b}^{(4)}_{h_3} \cdot w_{h_4} \\ G_2(\boldsymbol{y}) &= \sum_{h_2 \in [Q+1]} \boldsymbol{b}^{(2)}_{h_2} \cdot y_{h_2}, & G_5(\boldsymbol{v}) &= \sum_{h_5 \in [2\ell_2]} \boldsymbol{b}^{(5)}_{h_5} \cdot v_{h_5} \\ G_3(\boldsymbol{z}) &= \sum_{h_3 \in [\ell_1]} \boldsymbol{b}^{(3)}_{h_3} \cdot z_{h_3}, & G_0 &= \boldsymbol{b}^{(0)} \end{aligned}$$

- Let $G_{k,j}$ be the j-th entry function of G_k. We also identify linear functions $G_{2,j}, G_{3,j}, G_{4,j}, G_{5,j}$ and $G_{0,j}$ as coefficient vectors $\vec{G}_{k,j}$ for $k \in \{2,3,4,5,0\}$, respectively. For each $i \in [d^0]$, sample

$$\hat{\boldsymbol{b}}_i \leftarrow \overline{\mathsf{QFE}^0.\mathsf{KGen}}(G_{1,i}, \vec{G}_{2,i}, \vec{G}_{3,i}, \vec{G}_{4,i}, \vec{G}_{5,i}, \vec{G}_{0,i}).$$

3. Outputs a vector $\mathsf{fk}_{\mathsf{Enc}^1 \circ F} = (\{\hat{\boldsymbol{c}}_i\}_{i \in [\ell_1+1]}, \{\hat{\boldsymbol{b}}_i\}_{i \in [d^0]})$.

- $\mathsf{EDec}^0(\mathsf{fk}_{\mathsf{Enc}^1 \circ F}, \mathsf{ct}^0)$:
  1. Parse $\mathsf{fk}_{\mathsf{Enc}^1 \circ F}$ into $(\hat{\boldsymbol{c}}_i)$, and $(\hat{\boldsymbol{b}}_i)$, and $\mathsf{ct}^0 = (\boldsymbol{c}^0, \boldsymbol{b}^0, \boldsymbol{a})$
  2. Compute

$$\begin{aligned} c^1_i &\leftarrow \overline{\mathsf{QFE}^0.\mathsf{Dec}}(\hat{\boldsymbol{c}}_i, (\boldsymbol{c}^0, \boldsymbol{b}^0)) \text{ for } i \in [\ell_1 + 1] \\ b^1_i &\leftarrow \overline{\mathsf{QFE}^0.\mathsf{Dec}}(\hat{\boldsymbol{b}}_i, (\boldsymbol{c}^0, \boldsymbol{b}^0)) \text{ for } i \in [d^0]. \end{aligned}$$

  3. Output $\mathsf{ct}^1 = (\boldsymbol{c}^1, \boldsymbol{b}^1, \boldsymbol{a})$, where $\boldsymbol{c}^1 = (c^1_i)_{i \in [\ell_1+1]}$ and $\boldsymbol{b}^1 = (b^1_i)_{i \in [d^0]}$.

6.2.3 Decryption Correctness of EKGen^0

For the decryption correctness, we recall that given

$$\begin{aligned} \boldsymbol{c}^0 &= \overline{\mathsf{QFE}^0.\mathsf{Enc}}(P \cdot \boldsymbol{m}') = \boldsymbol{u}^0 \cdot s_0 + \boldsymbol{e} + P \cdot \boldsymbol{m}' \\ \boldsymbol{b}^0 &= \mathsf{LFE}^0.\mathsf{Enc}(s_0^2, \boldsymbol{c}^0 \cdot s_0, \boldsymbol{\zeta}, \boldsymbol{\eta}, 1) \\ F(\boldsymbol{x}) &= \sum_{j,k} F_{j,k} \cdot x'_j \cdot x'_k, \end{aligned}$$

the following equality holds: For any $\boldsymbol{v}_1 \in \mathbb{Z}^{Q+1}$ and $\boldsymbol{v}_2 \in \mathbb{Z}^{2Q}$,

$$\begin{aligned} &\overline{\mathsf{QFE}^0.\mathsf{Dec}}(\overline{\mathsf{QFE}^0.\mathsf{KGen}}(F, \boldsymbol{v}_1, \boldsymbol{v}_2, 0), (\boldsymbol{c}^0, \boldsymbol{b}^0)) \\ &= \left\lfloor \frac{\sum_{j,k} F_{j,k} \cdot (P \cdot m'_j + e_j) \cdot (P \cdot m'_k + e_k) + \langle \boldsymbol{v}_1, \boldsymbol{\zeta} \rangle + \langle \boldsymbol{v}_2, \boldsymbol{\eta} \rangle}{P} \right\rceil. \end{aligned}$$

By leveraging this identity, we describe it in two cases for $(\hat{\boldsymbol{c}}_i)$ and $(\hat{\boldsymbol{b}}_i)$ for $\mathsf{ct}^0 = \mathbf{Enc}^0(\boldsymbol{m})$. For each $i \in [\ell_1 + 1]$, due to the correctness of the **QFE** scheme, we have

$$\begin{aligned} c_i^1 &= \overline{\mathbf{QFE}^0.\mathbf{Dec}}(\hat{\boldsymbol{c}}_i, (\boldsymbol{c}^0, \boldsymbol{b}^0)) \\ &= \overline{\mathbf{QFE}^0.\mathbf{Dec}}(\overline{\mathbf{QFE}^0.\mathbf{KGen}}(\hat{F}_i, P \cdot \boldsymbol{c}_i^1, \boldsymbol{\delta}_i^{2Q}, 0), (\boldsymbol{c}^0, \boldsymbol{b}^0)) \\ &= \left\lfloor \frac{\sum_{j,k} \hat{F}_{i,j,k} \cdot (P \cdot m'_j + e_j) \cdot (P \cdot m'_k + e_k) + P \cdot \sum_h c_{i,h}^1 \cdot \zeta_h + \eta'_i}{P} \right\rceil \end{aligned}$$

where $\eta'_i = \begin{cases} \eta_i & i \leq \ell_1 \\ 0 & i = \ell_1 + 1 \end{cases}$. Letting $e_i^* = \sum_{j,k} \hat{F}_{i,j,k} \cdot (P \cdot m'_k + e_k) \cdot (P \cdot m'_j + e_j) + \eta'_i - P^2 \cdot \hat{F}_i(\boldsymbol{m})$, it holds that

$$c_i^1 = \sum_{h \in [Q+1]} c_{i,h}^1 \cdot \zeta_h + P \cdot \hat{F}_i(\boldsymbol{m}) + \left\lfloor \frac{e_i^*}{P} \right\rceil .$$

Suppose $\left\lfloor \frac{e_i^*}{P} \right\rceil \leftarrow \mathcal{D}_{\alpha_1 q_1}$. Then, $\boldsymbol{c}^1 = (c_1^1, \ldots, c_{\ell_1+1}^1)$ can be regarded as an output of an algorithm $\overline{\mathbf{QFE}^1.\mathbf{Enc}}(P \cdot \hat{F}_i(\boldsymbol{m}))$.

On the other hand, $G_{1,i}$ is identical to a function $s_1 \cdot \hat{F}_i$. Due to the correctness of $\overline{\mathbf{QFE}^0.\mathbf{Dec}}$, we can observe that $\boldsymbol{b}^1 = (b_i^1)_i = \left\lfloor \frac{\bar{b}_i^1}{P} \right\rceil$, where the numerator $\bar{b}_i^1$ can be expressed with:

$$\sum_{h_1,j,k} s_1 \cdot \hat{F}_{h_1,j,k} \cdot (P \cdot m'_j + e_j) \cdot (P \cdot m'_k + e_k) \tag{2}$$

$$+ \sum_{h_2 \in [Q+1]} \boldsymbol{b}_{h_2,i}^{(2)} \cdot \zeta_{h_2} + \sum_{h_3 \in [\ell_1]} \boldsymbol{b}_{h_3,i}^{(3)} \cdot \eta_{h_3}^1 + \sum_{h_4 \in [\ell_1]} \boldsymbol{b}_{h_4,i}^{(4)} \cdot \eta_{h_4+\ell_1}^1 \tag{3}$$

$$+ \sum_{h_5 \in [2\ell_2]} \boldsymbol{b}_{h_5,i}^{(5)} \cdot \eta_{h_5}^2 + \boldsymbol{b}_i^{(0)}, \tag{4}$$

where the sub index i is used to denote the i-th entry of each vector. According to the linear homomorphic property, it holds that

$$\boldsymbol{b}^1 = \mathbf{LFE}^1.\mathbf{Enc}(s_1^2, \boldsymbol{c}^1 \cdot s_1, \boldsymbol{\zeta}, \boldsymbol{\eta}, 1)$$

Consequently, $(\boldsymbol{c}^1, \boldsymbol{b}^1, \boldsymbol{a})$ corresponds to $\mathbf{QFE}^1.\mathbf{Enc}(\boldsymbol{F}(\boldsymbol{m}))$. This completes the correctness.

Parameter Constraint. By the definition of $c_{i,h}$, the equality is rewritten by

$$\left(\sum_{h \in [Q+1]} u_{i,h}^1 \cdot \zeta_h \right) \cdot s_1 + P \cdot \hat{F}_i(\boldsymbol{m}) + \left\lfloor \frac{e_i^* + P \cdot \sum_{h \in [Q+1]} e_{i,h}^1 \cdot \zeta_h}{P} \right\rceil . \tag{5}$$

Suppose that the following equality holds:

$$\frac{\alpha_0 q \cdot P \cdot K \cdot (\ell+1)^2 \cdot M + P \cdot Q \cdot \alpha_0 q_1}{\alpha_1 q} = \mathsf{negl}(\lambda), \tag{6}$$

so that $\eta_i^1 \leftarrow \mathcal{D}_{R,\alpha_1 q}$ smudges other terms. Then, from the Lemma 1, the noise term $\left\lfloor \frac{e_i^*}{P} \right\rceil$ becomes a sample drawn from $\mathcal{D}_{\alpha_1 q_1}$. The similar result is obtained during computations of $\boldsymbol{b}^1$.

6.3 KGen^1: Generate a level-1 functional key $\mathsf{fk}_{\boldsymbol{F}}$

This key generation algorithm is designed to compute the functional key fk_F^1.

6.3.1 KGen^1 Intuition

Recall the structure of $\mathbf{QFE}^0.\mathbf{KGen}(F)$: $\mathsf{fk}_F^0 = \mathbf{LFE}^0.\mathbf{KGen}(\mathsf{fk}'_F, \mathbf{0}^{Q+1}, \boldsymbol{\delta}_{\mathsf{cnt}}^{2Q+1})$, where the vector fk'_F is defined as:

$$\left(\sum_{1 \le j \le k \le \ell+1} F_{j,k}(u_j^0 \cdot u_k^0 \cdot \boldsymbol{\delta}_1^{\ell+2} - u_j^0 \cdot \boldsymbol{\delta}_{1+k}^{\ell+2} - u_k^0 \cdot \boldsymbol{\delta}_{1+j}^{\ell+2}) \right).$$

It is important to emphasize that the functional key shares the u_i^0-term with a ciphertext, which is expressed as:

$$\boldsymbol{c} = \boldsymbol{u}^0 \cdot s + P \cdot \boldsymbol{m}' + \boldsymbol{e}$$

for an initial level encryption $\mathsf{ct}^0 = (\boldsymbol{c}, \boldsymbol{b}, \boldsymbol{a})$, where $\boldsymbol{m}' = \boldsymbol{F}^0(\boldsymbol{m})$.

On the other hand, given a ciphertext at level-1 of the form

$$\mathsf{ct}^1 = \mathbf{Enc}^1(\boldsymbol{F}^0(\boldsymbol{m})) = (\boldsymbol{c}^1, \boldsymbol{b}^1, \boldsymbol{a}),$$

the component $\boldsymbol{c}^1 = (c_i^1)_{i \in [\ell_1+1]}$ can be computed using the encrypted functional key from level-0, takes the following form:

$$c_i^1 = \sum_{h=1}^{Q+1} u_{i,h}^1 \cdot \zeta_h \cdot s_1 + e_i^1 + P \cdot \hat{F}_i^0(\boldsymbol{m}) \text{ for } i \in [\ell_1 + 1].$$

Thus, the u^1-related term $\sum_{h=1}^{Q+1} u_{i,h}^1 \cdot \zeta_h$ in the functional key at level-1 must be updated to align with each ciphertext.

To achieve this, we design the $\mathbf{KGen}^1$ algorithm so that it generates the following structure in the decryption process:

$$\mathsf{fk}_F^1 \leftarrow \mathbf{LFE}^1.\mathbf{KGen}(\mathsf{fk}'_F, \mathbf{0}^{Q+1}, \boldsymbol{\delta}_{\mathsf{cnt}}^{2Q+1}),$$

where the vector fk'_F is defined as:

$$\left(\sum_{1 \le j \le k \le \ell_1 + 1} F_{j,k} \left(u_j^1 \cdot u_k^1 \cdot \boldsymbol{\delta}_1^{\ell_1+2} - u_j^1 \cdot \boldsymbol{\delta}_{1+k}^{\ell_1+2} - u_k^1 \cdot \boldsymbol{\delta}_{1+j}^{\ell_1+2} \right) \right),$$

where $u_j^1 = \sum_{h=1}^{Q+1} u_{j,h}^1$. Once this updated functional key is generated, the pair $\mathsf{ct}^1, \mathsf{fk}_F^1$ mirrors the structure of $\mathsf{ct}^0, \mathsf{fk}_F^0$. As a result, the decryption algorithm applied to this pair should correctly compute $F \circ (\boldsymbol{F}^0(\boldsymbol{m}))$. This observation indicates that the key generation process at level 1 involves updating the functional key to correspond to the u-terms in ct^1.

By exploiting the linear homomorphic property, given a count $\mathsf{cnt} \in [\ell_2]$, the functional key can be represented as a weight sum:

$$\begin{aligned}
\mathsf{fk}_F^1 = & \sum_{h_1, h_2 \in [Q+1]} \mathbf{LFE}^1.\mathbf{KGen} \left(\sum_{1 \le j \le k \le \ell_1 + 1} F_{j,k} \cdot u_{j,h_1}^1 \cdot u_{k,h_2}^1 \cdot \boldsymbol{\delta}_1^{\ell_1+2}, \mathbf{0}^{3Q+2} \right) \cdot \zeta_{h_1} \cdot \zeta_{h_2} \\
& + \sum_{h \in [Q+1]} \mathbf{LFE}^1.\mathbf{KGen} \left(\sum_{1 \le j \le k \le \ell_1 + 1} F_{j,k} \cdot (-u_{j,h}^1 \cdot \boldsymbol{\delta}_{1+k}^{\ell_1+2}), \mathbf{0}^{3Q+2} \right) \cdot \zeta_h \\
& + \sum_{h \in [Q+1]} \mathbf{LFE}^1.\mathbf{KGen} \left(\sum_{1 \le j \le k \le \ell_1 + 1} F_{j,k} \cdot (-u_{k,h}^1 \cdot \boldsymbol{\delta}_{1+j}^{\ell_1+2}), \mathbf{0}^{3Q+2} \right) \cdot \zeta_h \\
& + \mathbf{LFE}^1.\mathbf{KGen} \left(\mathbf{0}^{\ell_1+2}, \mathbf{0}^{Q+1}, \boldsymbol{\delta}_{\mathsf{cnt}}^{2Q+1} \right).
\end{aligned}$$

Thus, fk_F^1 can be expressed as a linear function in terms of variables $\zeta_{h_1} \cdot \zeta_{h_2}$ and ζ_h. Moreover, there exists an index i such that $\zeta_{h_1} \cdot \zeta_{h_2}$ is i-th coordinate of $\boldsymbol{\zeta} \otimes \boldsymbol{\zeta}$. This implies that the functional key can be updated using a linear FE scheme with $\boldsymbol{a} \leftarrow \mathbf{LFE.Enc}(\boldsymbol{\zeta} \otimes \boldsymbol{\zeta}, \boldsymbol{\zeta}, 1)$.

6.3.2 $\mathbf{KGen}^1$ Construction

Given a quadratic polynomial F and a counter $\mathsf{cnt} \in [\ell_2]$, $\mathbf{KGen}^1$ is performed as follows.

- $\mathbf{KGen}^1(F, \mathsf{cnt} \in [\ell_2])$:

1. Given a counter $\mathsf{cnt} \in [\ell_2]$ and $h, h_1, h_2 \in [Q+1]$, compute the followings.

$$\mathsf{sk}^{(0)}_{h_1,h_2} \leftarrow \mathbf{LFE}^1.\mathbf{KGen}\left(\sum_{1\le j\le k\le \ell_1+1} F_{j,k}\cdot u^1_{j,h_1}\cdot u^1_{k,h_2}\cdot \boldsymbol{\delta}_1^{\ell_1+2}, \mathbf{0}^{3Q+2}\right)$$

$$\mathsf{sk}^{(1)}_{h} \leftarrow \mathbf{LFE}^1.\mathbf{KGen}\left(\sum_{1\le j\le k\le \ell_1+1} F_{j,k}\cdot (-u^1_{j,h}\cdot \boldsymbol{\delta}_{1+k}^{\ell_1+2}), \mathbf{0}^{3Q+2}\right)$$
$$+ \mathbf{LFE}^1.\mathbf{KGen}\left(\sum_{1\le j\le k\le \ell_1+1} F_{j,k}\cdot (-u^1_{k,h}\cdot \boldsymbol{\delta}_{1+j}^{\ell_1+2}), \mathbf{0}^{3Q+2}\right)$$

$$\mathsf{sk}^{(2)} \leftarrow \mathbf{LFE}^1.\mathbf{KGen}\left(\mathbf{0}^{\ell_1+2}, \mathbf{0}^{Q+1}, \boldsymbol{\delta}_{\mathsf{cnt}}^{2Q+1}\right)$$

2. For $\boldsymbol{x} \in R^{(Q+1)^2}, \boldsymbol{y} \in R^{Q+1}$, define a vector linear function $H(\boldsymbol{x}, \boldsymbol{y})$ as follows:

$$H(\boldsymbol{x},\boldsymbol{y}) = \sum_{h_1,h_2\in[Q+1]} \mathsf{sk}^{(0)}_{h_1,h_2}\cdot x_{Q\cdot h_1+h_2} + \sum_{h\in[Q+1]} \mathsf{sk}^{(1)}_h\cdot y_h + \mathsf{sk}^{(2)}$$

   Let H_i be the coefficient vector of i-th entry function of $H(\boldsymbol{x}, \boldsymbol{y})$.

3. Output $\mathsf{fk}^1_F = \{\mathbf{LFE.KGen}(P\cdot H_i)\}_{i\in[d_\zeta]}$.[10]

- $\mathbf{Dec}^1(\mathsf{fk}^1_F, \mathsf{ct}^1)$: On a functional key $\mathsf{fk}_F = \{\mathbf{LFE.KGen}(P\cdot H_i)\}_i$ for a quadratic polynomial $\sum_{1\le j\le k\le \ell_1+1} F_{j,k}\cdot x'_j\cdot x'_k$, and a ciphertext $\mathsf{ct}^1 = (\boldsymbol{c}^1, \boldsymbol{b}^1, \boldsymbol{a})$,
  1. Compute a vector $\boldsymbol{f}^1 = (f_i)_i$, where

$$f_i = \left\lfloor \frac{\langle \mathbf{LFE.KGen}(P\cdot H_i), \boldsymbol{a}\rangle}{P\cdot q} \right\rceil \bmod q_1.$$

  2. Output

$$\mu = \left\lfloor \frac{F'(\boldsymbol{c}^1) + \mathbf{LFE}^1.\mathbf{Dec}(\boldsymbol{b}^1, \boldsymbol{f}^1)}{P^2} \right\rceil.$$

6.3.3 Decryption Correctness of $\mathbf{KGen}^1$

Let ct^1 be a level-1 ciphertext of a message $\boldsymbol{m}' = \boldsymbol{F}^0(\boldsymbol{m})$. Our goal is to demonstrate that $\mu = F(\boldsymbol{m}') = (F\circ \boldsymbol{F}^0)(\boldsymbol{m})$. First, we observe that for every i, the following holds:

$$\left\langle \mathbf{LFE}^1.\mathbf{KGen}(P\cdot H_i), \boldsymbol{a}\right\rangle = \left\langle P\cdot H_i, q\cdot(\boldsymbol{\zeta}\otimes\boldsymbol{\zeta}, \boldsymbol{\zeta}, 1)\right\rangle + e^*_i$$

for some noise e^*_i. The constant q is derived that a plaintext space of $\mathbf{LFE}$ is q. Thus, we observe

$$f_i = \left\lfloor \frac{\langle P\cdot H_i, q\cdot(\boldsymbol{\zeta}\otimes\boldsymbol{\zeta}, \boldsymbol{\zeta}, 1)\rangle + e^*_i}{P\cdot q} \right\rceil \bmod q_1.$$

[10] For a high-level t, P^t should be used.

Suppose $|e_i^*| < P \cdot q/2$. Then we conclude:

$$f_i = \left\langle H_i, q \cdot (\boldsymbol{\zeta} \otimes \boldsymbol{\zeta}, \boldsymbol{\zeta}, 1) \right\rangle \bmod q_1.$$

Thus, due to the correctness of $\mathbf{LFE}$, the vector $\boldsymbol{f}^1$ can be rewritten as:

$$\boldsymbol{f}^1 = H(\boldsymbol{\zeta} \otimes \boldsymbol{\zeta}, \boldsymbol{\zeta}) = \mathbf{LFE}^1.\mathbf{KGen}\big(\mathsf{fk}'_F, \mathbf{0}^{Q+1}, \boldsymbol{\delta}_{\mathsf{cnt}}^{2Q+1}\big) \quad \text{with}$$

$$\mathsf{fk}'_F = \left(\sum_{1 \le j \le k \le \ell_1+1} F_{j,k}\big(u_j^1 \cdot u_k^1 \cdot \boldsymbol{\delta}_1^{\ell_1+2} - u_j^1 \cdot \boldsymbol{\delta}_{1+k}^{\ell_1+2} - u_k^1 \cdot \boldsymbol{\delta}_{1+j}^{\ell_1+2}\big) \right).$$

In other words, $\boldsymbol{f}^1$ can be regarded as a functional key obtained from $\mathbf{QFE}^1.\mathbf{KGen}$. Hence, by the correctness of $\mathbf{QFE}^1$, we obtain:

$$\mu = (F \circ \boldsymbol{F}^0)(\boldsymbol{m}).$$

Parameter Constraint. To achieve the correctness of $\mathbf{KGen}^1$, a few constraints are required. By definition of $e_i^* = \langle P \cdot H_i, \boldsymbol{e} \rangle$ where $\boldsymbol{e}$ is the noise vector in $\boldsymbol{a}$, we can bound e_i^* by:

$$P \cdot \sigma_{\mathbf{LFE}} \cdot K_1 \cdot (Q+2)^2,$$

where

$$K_1 = \max_{h,h_1,h_2 \in [Q+1]} \big\{ \|\mathsf{sk}_{h_1,h_2}^{(0)}\|_\infty, \|\mathsf{sk}_h^{(1)}\|_\infty, \|\mathsf{sk}^{(2)}\|_\infty \big\}.$$

Thus, for $|e_i^*| < P \cdot q/2$, the following constraint is required:

$$\sigma_{\mathbf{LFE}} \cdot K_1 \cdot (Q+2)^2 < q/2.$$

Similarly to the noise flooding constraint for $\mathbf{QFE}^0$, the correctness constraint for $\mathbf{QFE}^1$ is given by:

$$\frac{\alpha_1 q \cdot (\ell_1+1)^2 \cdot K \cdot P \cdot M_1}{\alpha_2 q} = \mathsf{negl}(\lambda),$$

where M_1 denotes the maximal size of the input message vector.

6.4 $\mathbf{EKGen}^1$: Generate a level-2 encrypted functional key $\mathsf{fk}_{\mathbf{Enc}^2 \circ \boldsymbol{F}}$

This algorithm is designed to compute the functional key $\mathsf{fk}_{\mathbf{Enc}^2 \circ \boldsymbol{F}}$ for a quadratic vector function $\boldsymbol{F} : R^{\ell_1} \to R^{\ell_2}$. As already mentioned, given a count $\mathsf{cnt} \in [\ell_2]$, we can consider a multivariate function $F_{\mathsf{cnt}} : R^{\ell_1} \to R$.

6.4.1 $\mathbf{EKGen}^1$ Intuition

Similar to the previous level, the $\mathbf{EKGen}^1$algorithm must include an update process to align with the ciphertext $\boldsymbol{c}^1$. In other words, this update process

ensures that the generated terms $(\{\hat{\boldsymbol{c}}_i^1\}, \{\hat{\boldsymbol{b}}_i^1\})$ follow a specific structure similar to EKGen^0. Specifically, each term takes the following form:

$$\hat{\boldsymbol{c}}_i^2 = \mathsf{LFE}^1.\mathsf{KGen}(\mathsf{fk}'_{\hat{F}_i}, P \cdot \boldsymbol{c}_i^2, \boldsymbol{\delta}_i^{2Q+1})$$
$$\hat{\boldsymbol{b}}_i^2 = \mathsf{LFE}^1.\mathsf{KGen}(\mathsf{fk}'_{G_{1,i}}, \vec{G}_{2,i}, \vec{G}_{3,i}, \vec{G}_{4,i}, \vec{G}_{5,i}, \vec{G}_{0,i}).$$

As in the KGen^1 step, each term can be decomposed as a linear function in $\zeta_{h_1} \cdot \zeta_{h_2}$. After updating u-associated term, the result is essentially identical to EKGen^0. Based on this intuition, we now provide a formal description of the algorithm.

6.4.2 EKGen^1 Construction

- $\mathsf{EKGen}^1(\{F_i\}_{i=1}^{\ell_2} \subset \mathcal{F}_K^{\ell_2}, \mathsf{cnt} = \ell_2)$:
  1. For $i \in [\ell_2 + 1]$,
     - Sample $s_2 \leftarrow \mathcal{D}_{R,\alpha_0 q}$ and $e_{i,h}^2 \leftarrow \mathcal{D}_{R,\alpha_0 q}$ for each $i \in [\ell_2 + 1]$ and $h \in [Q + 1]$. Compute encodings of zero $\boldsymbol{c}_i^2 = (c_{i,h}^2) \in R_q^{Q+1}$ such that

       $$c_{i,h}^2 = u_{i,h}^2 \cdot s_2 + e_{i,h}^2,$$

       where $u_{i,h}^2$ is employed from pp.

     - For any $\boldsymbol{x} \in R^{\ell_1}$, define a function

       $$\hat{F}_i(\boldsymbol{x}) = \begin{cases} F_i(\boldsymbol{x}) & \text{if } i \in [\ell_2] \\ 1 & \text{if } i = \ell_2 + 1 \end{cases}$$

       We let $\sum_{1 \le j \le k \le \ell_1 + 1} \hat{F}_{i,j,k} \cdot x_j \cdot x_k$ denote the function $\hat{F}_i(\boldsymbol{x})$.

     - Compute $\{\mathsf{sk}_{i,h_1,h_2}^{(c,0)}, \mathsf{sk}_{i,h}^{(c,1)}, \mathsf{sk}_i^{(c,2)}\}$ defined as follows:

       $$\mathsf{sk}_{i,h_1,h_2}^{(c,0)} \leftarrow \mathsf{LFE}^1.\mathsf{KGen}(\sum_{1 \le j \le k \le \ell_2 + 1} \hat{F}_{i,j,k} \cdot u_{j,h_1}^1 \cdot u_{k,h_2}^1 \cdot \boldsymbol{\delta}_1^{\ell_2+2}, \mathbf{0}^{3Q+2})$$
       $$\mathsf{sk}_{i,h}^{(c,1)} \leftarrow \mathsf{LFE}^1.\mathsf{KGen}(\sum_{1 \le j \le k \le \ell_2 + 1} \hat{F}_{i,j,k} \cdot (-u_{j,h}^1 \cdot \boldsymbol{\delta}_{1+k}^{\ell_2+2}), \mathbf{0}^{3Q+2}) + \mathsf{LFE}^1.\mathsf{KGen}(\sum_{1 \le j \le k \le \ell_2 + 1} \hat{F}_{i,j,k} \cdot (-u_{k,h}^1 \cdot \boldsymbol{\delta}_{1+j}^{\ell_2+2}), \mathbf{0}^{3Q+2})$$
       $$\mathsf{sk}_i^{(c,2)} \leftarrow \mathsf{LFE}^1.\mathsf{KGen}(\mathbf{0}^{\ell_2+2}, P \cdot \boldsymbol{c}_i^2, \boldsymbol{\delta}_i^{2Q+1})$$

     - For $\boldsymbol{x} \in R^{(Q+1)^2}, \boldsymbol{y} \in R^{Q+1}$, define a vector linear function $H_i^c(\boldsymbol{x}, \boldsymbol{y})$ as follows:

       $$H_i^c(\boldsymbol{x}, \boldsymbol{y}) = \sum_{h_1, h_2 \in [Q+1]} \mathsf{sk}_{i,h_1,h_2}^{(c,0)} \cdot x_{Q \cdot h_1 + h_2} + \sum_{h \in [Q+1]} \mathsf{sk}_{i,h}^{(c,1)} \cdot y_h + \mathsf{sk}_i^{(c,2)}$$

       Let $H_{i,j}^c$ be the coefficient vector of j-th entry function of $H_i^c(\boldsymbol{x}, \boldsymbol{y})$.

- Output $\mathcal{C}_i^2 = \{\mathbf{LFE.KGen}(H_{i,j}^c)\}_{j\in[d_\zeta]}$.

2. Let $d^1 = \ell_2 + 5 + 2Q$. For $i \in [d^1]$ do:
   - For $h_1 \in [\ell_2+1], h_2 \in [Q+1], h_3, h_4 \in [\ell_2]$ and $h_5 \in [2\ell_1]$, compute $\left\{\boldsymbol{b}^{(0)}, \boldsymbol{b}_{h_i}^{(i)}\right\}_{i\le 5}$ as follows.

$$
\begin{aligned}
\boldsymbol{b}^{(0)} &\leftarrow P \cdot \mathbf{LFE}^2\mathbf{.Enc}(s_1^2, \mathbf{0}^{\ell_2+1}, \mathbf{0}^{Q+1}, \mathbf{0}^{2Q}, 1)\\
\boldsymbol{b}_{h_1}^{(1)} &\leftarrow P \cdot \mathbf{LFE}^2\mathbf{.Enc}(0, s_1 \cdot \boldsymbol{\delta}_{h_1}^{\ell_2+1}, \mathbf{0}^{Q+1}, \mathbf{0}^{2Q}, 0)\\
\boldsymbol{b}_{h_2}^{(2)} &\leftarrow P \cdot \mathbf{LFE}^2\mathbf{.Enc}(0, s_1 \cdot \boldsymbol{c}_{i,h_2}, \boldsymbol{\delta}_{h_2}^{Q+1}, \mathbf{0}^{2Q}, 0)\\
\boldsymbol{b}_{h_3}^{(3)} &\leftarrow P \cdot \mathbf{LFE}^2\mathbf{.Enc}(0, s_1 \cdot \boldsymbol{\delta}_{h_3}^{\ell_2+1}, \mathbf{0}^{Q+1}, \boldsymbol{\delta}_{h_3}^{2Q}, 0),\\
\boldsymbol{b}_{h_4}^{(4)} &\leftarrow P \cdot \mathbf{LFE}^2\mathbf{.Enc}(0, \boldsymbol{\delta}_{h_4}^{\ell_2+1}/P^2, \mathbf{0}^{Q+1}, \boldsymbol{\delta}_{\ell_2+h_4}^{2Q}, 0),\\
\boldsymbol{b}_{h_5}^{(5)} &\leftarrow P \cdot \mathbf{LFE}^2\mathbf{.Enc}(0, \mathbf{0}^{\ell_2+1}, \mathbf{0}^{Q+1}, \boldsymbol{\delta}_{2\ell_2+h_5}^{2Q}, 0)
\end{aligned}
$$

   where $\boldsymbol{c}_{i,h_2}^2 = (c_{1,h_2}^2, \ldots, c_{\ell_2+1,h_2}^2)$. As $\mathbf{EKGen}^1$, we let $\boldsymbol{b}_{h_1,j}^{(1)}$ denote the binary decomposition of a vector such that $\boldsymbol{b}_{h_1}^{(1)} = \sum_j 2^j \cdot \boldsymbol{b}_{h_1,j}^{(1)}$. Ensuring the correctness, operations should be performed on each $\boldsymbol{b}_{h_1,j}^{(1)}$. However, for simplicity of exposition, we use the notation $\boldsymbol{b}_{h_1}^{(1)}$.

   - For $\boldsymbol{x} \in R^{\ell_2}, \boldsymbol{y} \in R^{Q+1}, \boldsymbol{z}, \boldsymbol{w} \in R^{\ell_2}$ and $\boldsymbol{v} \in R^{2\ell_1}$, define functions $G_1(\boldsymbol{x}), G_2(\boldsymbol{y}), G_3(\boldsymbol{z}), G_4(\boldsymbol{w})$ and $G_5(\boldsymbol{v})$:

$$
\begin{aligned}
G_1(\boldsymbol{x}) &= \sum_{h_1\in[\ell_2+1]} \boldsymbol{b}_{h_1}^{(1)} \cdot \hat{F}_{h_1}(\boldsymbol{x}), & G_4(\boldsymbol{w}) &= \sum_{h_4\in[\ell_2]} \boldsymbol{b}_{h_4}^{(4)} \cdot w_{h_4}\\
G_2(\boldsymbol{y}) &= \sum_{h_2\in[Q+1]} \boldsymbol{b}_{h_2}^{(2)} \cdot y_{h_2}, & G_5(\boldsymbol{v}) &= \sum_{h_5\in[2\ell_1]} \boldsymbol{b}_{h_5}^{(5)} \cdot v_{h_5}\\
G_3(\boldsymbol{z}) &= \sum_{h_3\in[\ell_2]} \boldsymbol{b}_{h_3}^{(3)} \cdot z_{h_3}, & G_0 &= \boldsymbol{b}^{(0)}
\end{aligned}
$$

   - Let $G_{k,i}$ be i-th entry function of G_k. We also identify linear functions $G_{2,i}, G_{3,i}$ and $G_{3',i}$ as coefficient vectors $\vec{G}_{2,i}, \vec{G}_{3,i}$ and $\vec{G}_{3',i}$, respectively. For each $i \in [d^1]$, letting $G_{1,i} = \sum_{1\le j\le k\le \ell_1+1} G_{1,i,j,k} x_j \cdot x_k$, we compute $\mathcal{B}_i^2 = \{\mathsf{sk}_{i,h_1,h_2}^{(b,0)}, \mathsf{sk}_{i,h}^{(b,1)}, \mathsf{sk}_i^{(b,2)}\}$ defined as follows:

$$
\begin{aligned}
\mathsf{sk}_{i,h_1,h_2}^{(b,0)} &\leftarrow \mathbf{LFE}^1\mathbf{.KGen}\Big(\sum_{1\le j\le k\le \ell_2+1} G_{1,i,j,k} \cdot u_{j,h_1}^1 \cdot u_{k,h_2}^1 \cdot \boldsymbol{\delta}_1^{\ell_2+2}, \mathbf{0}^{3Q+2}\Big)\\
\mathsf{sk}_{i,h}^{(b,1)} &\leftarrow \mathbf{LFE}^1\mathbf{.KGen}\Big(\sum_{1\le j\le k\le \ell_2+1} G_{1,i,j,k} \cdot (-u_{j,h}^1 \cdot \boldsymbol{\delta}_{1+k}^{\ell_2+2}), \mathbf{0}^{3Q+2}\Big)\\
&\quad + \mathbf{LFE}^1\mathbf{.KGen}\Big(\sum_{1\le j\le k\le \ell_2+1} G_{1,i,j,k} \cdot (-u_{k,h}^1 \cdot \boldsymbol{\delta}_{1+j}^{\ell_2+2}), \mathbf{0}^{3Q+2}\Big)\\
\mathsf{sk}_i^{(b,2)} &\leftarrow \mathbf{LFE}^1\mathbf{.KGen}(\mathbf{0}^{\ell_2+2}, \vec{G}_{2,i}, \vec{G}_{3,i}, \vec{G}_{4,i}, \vec{G}_{5,i}, \vec{G}_{0,i})
\end{aligned}
$$

- For $\boldsymbol{x} \in R^{(Q+1)^2}, \boldsymbol{y} \in R^{Q+1}$, define a vector linear function $H_i^b(\boldsymbol{x}, \boldsymbol{y})$ as follows:

$$H_i^b(\boldsymbol{x}, \boldsymbol{y}) = \sum_{h_1, h_2 \in [Q+1]} \mathsf{sk}_{i,h_1,h_2}^{(b,0)} \cdot x_{Q \cdot h_1 + h_2} + \sum_{h \in [Q+1]} \mathsf{sk}_{i,h}^{(b,1)} \cdot y_h + \mathsf{sk}_i^{(b,2)}$$

  Let $H_{i,j}^b$ be the coefficient vector of j-th entry function of $H_i^b(\boldsymbol{x}, \boldsymbol{y})$.

- Output $\mathcal{B}_i^2 = \{\mathbf{LFE.KGen}(H_{i,j}^b)\}_j$.

3. Outputs a vector $\mathsf{fk}_{\mathbf{Enc}^2 \circ \boldsymbol{F}} = ((\mathcal{C}_i^2)_{i \in [\ell_2+1]}, (\mathcal{B}_i^2)_{i \in [d^1]})$.

- $\mathbf{EDec}^1(\mathsf{fk}_{\mathbf{Enc}^2 \circ \boldsymbol{F}}, \mathsf{ct}^1)$: On a functional key $\mathsf{fk}_{\mathbf{Enc}^2 \circ \boldsymbol{F}} = ((\mathcal{C}_i^2)_{i \in [\ell_2+1]}, (\mathcal{B}_i^2)_{i \in [d^1]})$ and ciphertext $\mathsf{ct}^1 = (\boldsymbol{c}^1, \boldsymbol{b}^1, \boldsymbol{a}) = \mathbf{Enc}^1(\boldsymbol{F}^0(\boldsymbol{m}))$,

1. Compute a vector $\hat{\boldsymbol{c}}_i^2 = (\hat{c}_{i,j}^2)_j$ and a vector $\hat{\boldsymbol{b}}_i^2 = (\hat{\boldsymbol{b}}_{i,j}^2)_j$

$$\hat{c}_{i,j}^2 = \mathbf{LFE.Dec}(\boldsymbol{a}, \mathbf{LFE.KGen}(H_{i,j}^c)) \bmod q_1$$
$$\hat{\boldsymbol{b}}_{i,j}^2 = \mathbf{LFE.Dec}(\boldsymbol{a}, \mathbf{LFE.KGen}(H_{i,j}^b)) \bmod q_1$$

2. Compute

$$\boldsymbol{c}_i^2 \leftarrow \overline{\mathbf{QFE}^1.\mathbf{Dec}}(\mathsf{pp}, \hat{\boldsymbol{c}}_i^2, (\boldsymbol{c}^1, \boldsymbol{b}^1)) \text{ for } i \in [\ell_2 + 1]$$
$$\boldsymbol{b}_i^2 \leftarrow \overline{\mathbf{QFE}^1.\mathbf{Dec}}(\mathsf{pp}, \hat{\boldsymbol{b}}_i^2, (\boldsymbol{c}^1, \boldsymbol{b}^1)) \text{ for } i \in [d^1].$$

3. Output $\mathsf{ct}^2 = (\boldsymbol{c}^2, \boldsymbol{b}^2, \boldsymbol{a})$, where $\boldsymbol{c}^2 = (\boldsymbol{c}_i^2)_{i \in [\ell_2+1]}$ and $\boldsymbol{b}^2 = (\boldsymbol{b}_i^2)_{i \in [d^1]}$.

6.4.3 Decryption Correctness of $\mathbf{EKGen}^1$

By definition of $\hat{\boldsymbol{c}}_i^2$ and $\hat{\boldsymbol{b}}_i^2$, applying the same argument in the decryption correctness of $\mathbf{Dec}^1(\mathsf{fk}_F^1, \mathsf{ct}^1)$, it holds that

$$\hat{\boldsymbol{c}}_{i,j}^2 = \langle H_{i,j}^c, (\boldsymbol{\zeta} \otimes \boldsymbol{\zeta}, \boldsymbol{\zeta}, 1) \rangle \bmod q_1$$
$$\hat{\boldsymbol{b}}_{i,j}^2 = \langle H_{i,j}^b, (\boldsymbol{\zeta} \otimes \boldsymbol{\zeta}, \boldsymbol{\zeta}, 1) \rangle \bmod q_1,$$

as long as the size of each computational noise is less than q. Then, for every i, it holds

$$\hat{\boldsymbol{c}}_i^2 = H_i^c(\boldsymbol{\zeta} \otimes \boldsymbol{\zeta}, \boldsymbol{\zeta}) = \mathbf{LFE}^1.\mathbf{KGen}(\mathsf{fk}'_{\hat{F}_i}, P \cdot \boldsymbol{c}_i^2, \boldsymbol{\delta}_i^{2Q+1})$$
$$\hat{\boldsymbol{b}}_i^2 = H_i^b(\boldsymbol{\zeta} \otimes \boldsymbol{\zeta}, \boldsymbol{\zeta}) = \mathbf{LFE}^1.\mathbf{KGen}(\mathsf{fk}'_{G_{1,i}}, \vec{G}_{2,i}, \vec{G}_{3,i}, \vec{G}_{4,i}, \vec{G}_{5,i}, \vec{G}_{0,i})$$

where

$$\mathsf{fk}'_{\hat{F}_i} = \sum_{1 \le j \le k \le \ell_2+1} F_{j,k}(u_j^1 \cdot u_k^1 \cdot \boldsymbol{\delta}_1^{\ell_2+2} - u_j^1 \cdot \boldsymbol{\delta}_{1+k}^{\ell_2+2} - u_k^1 \cdot \boldsymbol{\delta}_{1+j}^{\ell_2+2})$$
$$\mathsf{fk}'_{G_{1,i}} = \sum_{1 \le j \le k \le \ell_2+1} G_{1,i,j,k}(u_j^1 \cdot u_k^1 \cdot \boldsymbol{\delta}_1^{\ell_2+2} - u_j^1 \cdot \boldsymbol{\delta}_{1+k}^{\ell_2+2} - u_k^1 \cdot \boldsymbol{\delta}_{1+j}^{\ell_2+2}).$$

Once $\hat{\boldsymbol{c}}_i^2$ and $\hat{\boldsymbol{b}}_i^2$ are computed, $\boldsymbol{c}^2$ and $\boldsymbol{b}^2$ are of the form

$$\boldsymbol{c}_i^2 = \sum_{h=1}^{Q} u_{i,h}^2 \cdot \zeta_h + \eta_i + P \cdot \hat{F}_i(\boldsymbol{F}^0(\boldsymbol{m})) \text{ for } i \in [\ell_2 + 1]$$
$$\boldsymbol{b}^2 = \mathbf{LFE}^2.\mathbf{Enc}(s_2^2, \boldsymbol{c}^2 \cdot s_2, \boldsymbol{\zeta}, \boldsymbol{\eta}, 1).$$

with the same computation on $\mathbf{EDec}^0(\mathsf{fk}_{\mathbf{Enc}^1 \circ F}, \mathsf{ct}^0)$. This guarantees that the vector $(\boldsymbol{c}^2, \boldsymbol{b}^2, \boldsymbol{a})$ is a proper encryption form of $\mathbf{Enc}^2(F \circ \boldsymbol{F}^0(\boldsymbol{m}))$. This completes the correctness.

Parameter Constraint. To achieve the correctness, a few constraints are needed. As in the previous section, for all i, j, the computational noise is of the form: $\langle H_{i,j}^b, \boldsymbol{e} \rangle$, where $\boldsymbol{e}$ is the noise vector used in $\boldsymbol{a}$. Thus, it is bounded by $P \cdot \sigma_{\mathbf{LFE}} \cdot K_2^b \cdot (Q+2)^2$, where

$$K_2^b = \max_{h,h_1,h_2,i \in [Q+1]} \{\|\mathsf{sk}_{i,h_1,h_2}^{(b,0)}\|_\infty, \|\mathsf{sk}_{i,h}^{(b,1)}\|_\infty, \|\mathsf{sk}_i^{(b,2)}\|_\infty\}.$$

Since each entry of $G_i(x)$ is defined in $\mathbb{Z}_{q_1}$, we can say that $K_2^b \leq q_1$. Hence, it can be expressed as:

$$\sigma_{\mathbf{LFE}} \cdot q_1 \cdot (Q+2)^2 < q/2.$$

Adapting the constraint in Section 6.2.3, for the correctness of $\mathbf{EDec}^1$, the constraint is written as:

$$\frac{\alpha_1 q \cdot P \cdot K \cdot (\ell_1 + 1)^2 \cdot M_1 + P \cdot Q \cdot \alpha_1 q_2}{\alpha_2 q} = \mathsf{negl}(\lambda).$$

7 Security Proof

This section presents the security proof of our composable functional encryption scheme.

Theorem 7.1 *The CFE construction in Section 6 achieves full simulation-based security (Definition 2).*

Proof. We prove the security of our CFE scheme, following the approach of [8]. Since level-2 security follows directly from level-1 security, we focus our proof on the first level. The proof proceeds in the following steps: Suppose that

$$\mathsf{fk}_{F_1}^0 \leftarrow \mathbf{KGen}^0(F_1, i_1)$$
$$\mathsf{fk}_{\mathbf{Enc}^1 \circ F_2} \leftarrow \mathbf{EKGen}^1(F_2, i_2)$$
$$\mathsf{fk}_{F_3}^1 \leftarrow \mathbf{KGen}^1(F_3, i_3)$$
$$\mathsf{fk}_{\mathbf{Enc}^2 \circ F_4} \leftarrow \mathbf{EKGen}^1(F_4, i_4)$$

satisfying $i_1 + i_2 + i_3 + i_4 = Q$, where i_j is a counter of each functional key.

Let ct be a level-0 ciphertext of the form $\mathsf{ct} = (\boldsymbol{c}, \boldsymbol{b}, \boldsymbol{a}) \leftarrow \mathbf{Enc}^0(\boldsymbol{m})$. We again note that a decryption algorithm $\mathbf{Dec}^0$ of which inputs are an encrypted functional key $\mathsf{fk}_{\mathbf{Enc}^1 \circ F}$ and ct moves the ciphertext to a level-1 ciphertext. We then describe a simulator for level 1.

For a more formal description, we should consider $\{F_i\}_i$ and $\left\{\mathbf{Enc}^1 \circ F_j\right\}_j$ such that $i + j \leq \ell_1$. However, for simplicity, we describe only two functions, F_1 and $\mathbf{Enc}^1 \circ F_2$, along with their functional keys, $\mathsf{fk}^0_{F_1}$ and $\mathsf{fk}_{\mathbf{Enc}^1 \circ F_2}$.

Simulator for level 1. Given input the security parameter λ, the length of message $\boldsymbol{m}$, the functions $F_1, \mathbf{Enc}^1 \circ F_2$, the functional keys $\mathsf{fk}^0_{F_1}$ and $\mathsf{fk}_{\mathbf{Enc}^1 \circ F_2}$, and extra information obtained by a decryption process β^0_1, β^0_2, where

$$\beta^0_1 = \sum_{1 \leq j \leq k \leq \ell+1} F_{1,j,k} \cdot c_j \cdot c_k + \mathbf{LFE}^0.\mathbf{Dec}(\boldsymbol{b}, \mathsf{fk}^0_{F_1}) = F_1(P \cdot \boldsymbol{m} + \boldsymbol{e}),$$
$$\beta^0_2 = \sum_{1 \leq j \leq k \leq \ell+1} F_{2,j,k} \cdot c_j \cdot c_k + \mathbf{LFE}^0.\mathbf{Dec}(\boldsymbol{b}, \mathsf{fk}_{\mathbf{Enc}^1 \circ F_2}) = \mathsf{fk}_{\mathbf{Enc}^1 \circ F_2}(P \cdot \boldsymbol{m} + \boldsymbol{e}),$$

where $\boldsymbol{e}$ is a noise vector. We note that β^0_i can be computed independently to $\boldsymbol{c}$ and $\boldsymbol{u}$, which is term involved in a pp The simulator $\mathbf{Sim}(\lambda, |\boldsymbol{m}|, \mathsf{pp}, \{(F_1, \mathsf{fk}^0_{F_1}, \beta^0_1), (\mathbf{Enc}^1 \circ F_2, \mathsf{fk}_{\mathbf{Enc}^1 \circ F_2}, \beta^0_2)\})$ for an encryption conducts the following:

- It randomly picks an element $\boldsymbol{c} \leftarrow R^{\ell+1}_q$.
- Computes β^2_i for $i \in \{1, 2\}$, which are defined later.
- It invokes the $\mathbf{LFE}^0$ simulator with input $\{\beta^2_i\}$. Its output sets as $\boldsymbol{b}$.
- This simulator replaces $(\boldsymbol{c}, \boldsymbol{b})$ in ct, and outputs ct.

To demonstrate indistinguishability, we construct a sequence of hybrid experiments, transitioning from the real to the ideal world via intermediate steps.

The Hybrids. Our hybrid games are described below.

- **Hybrid 0**: This is the real world.
- **Hybrid 1**: In this hybrid, the only thing that is different is that $\boldsymbol{b}$ in the $\mathbf{Enc}^0(\boldsymbol{m})$ is computed using the $\mathbf{LFE}^0$ simulator as

$$\boldsymbol{b}' = \mathbf{LFE}^0.\mathbf{Sim}(\{(F_1, \mathsf{fk}^0_{F_1}, \beta^0_1), (\mathbf{Enc}^1 \circ F_2, \mathsf{fk}_{\mathbf{Enc}^1 \circ F_2}, \beta^0_2)\}).$$

- **Hybrid 2**: In this hybrid, $\boldsymbol{b}'$ is changed into $\boldsymbol{b}''$, where $\boldsymbol{b}''$ is generated as follows:

$$\boldsymbol{b}'' = \mathbf{LFE}^0.\mathbf{Sim}(\{(F_1, \mathsf{fk}^0_{F_1}, \beta^2_1), (\mathbf{Enc}^1 \circ F_2, \mathsf{fk}_{\mathbf{Enc}^1 \circ F_2}, \beta^2_2)\})$$

  where $\beta^2_i = \beta^0_i - \boldsymbol{e}_i$ and $\boldsymbol{e}_i$ is the computational noise: Let $\boldsymbol{e}$ be an initial noise vector used in $\boldsymbol{c}$, it is then defined by:

$$\boldsymbol{e}_1 = F_1(P \cdot \boldsymbol{m} + \boldsymbol{e}) - P^2 \cdot F_1(\boldsymbol{m})$$
$$\boldsymbol{e}_2 = F_2(P \cdot \boldsymbol{m} + \boldsymbol{e}) - P^2 \cdot F_2(\boldsymbol{m}).$$

- **Hybrid 3**: In this hybrid, sample $\boldsymbol{c}$ at random. This is the simulation world.

Lemma 7.2 ***Hybrid 0*** *is indistinguishable from* ***Hybrid 1*** *assuming that* $\mathbf{LFE}^0$ *is selectively secure.*

Proof. We first describe how to instantiate a simulator of $\mathbf{LFE}^0$ even though the scheme only achieves selective security. This is because the notion of full-simulation security [8] does not allow the attacker to generate functional keys after seeing the challenge query. As a consequence, the selective security of $\mathbf{LFE}^0$ is sufficient to achieve the indistinguishability of two hybrids.

More precisely, due to the definition of a full-simulation security, the adversary submits function queries, and obtains functional keys before submitting the challenge message $\boldsymbol{x}$.

Let $\{\boldsymbol{v}_i\}_{i=1}^{Q}$ be a set of queried functions. Given evaluated values $\{\varepsilon_i = \langle \boldsymbol{v}_i, \boldsymbol{x}\rangle\}_{i\in[Q]}$, the simulator can find a vector $\boldsymbol{x}'$ such that it holds that $\langle \boldsymbol{v}_i, \boldsymbol{x}'\rangle = \varepsilon_i$. Finding $\boldsymbol{x}'$ terminates in polynomial time since it only requires basic linear algebra computations. The simulator of $\mathbf{LFE}^0$'s then returns $\boldsymbol{b}' = \mathbf{LFE}^0.\mathbf{Enc}(\boldsymbol{x}')$ as its output. Then, the selective security of $\mathbf{LFE}^0$ says that no one can efficiently distinguish $\mathbf{LFE}^0.\mathbf{Enc}(\boldsymbol{x})$ and $\mathbf{LFE}^0.\mathbf{Enc}(\boldsymbol{x}')$. Thus, it can be regarded as a simulator $\mathbf{LFE}^0$.

We now show the indistinguishability using the simulator of $\mathbf{LFE}^0$. Recall that a decryption algorithm that takes input $\mathsf{fk}^0_{F_1}$ and ct computes the following:

$$\sum_{1\le j\le k\le \ell+1} F_{j,k}\cdot c_j\cdot c_k + \mathbf{LFE}^0.\mathbf{Dec}(\boldsymbol{b}, \mathsf{fk}^0_{F_1}).$$

Since the simulator has access to information $\sum_{1\le j\le k\le \ell+1} F_{j,k}\cdot c_j\cdot c_k$, $\mathbf{LFE}$ simulator can be invoked with β^0_1. Similarly, $\mathbf{QFE}^0.Dec(\mathsf{fk}_{\mathbf{Enc}^1\circ F_2}, \mathsf{ct})$ is computed as follows:

$$\sum_{1\le j\le k\le \ell+1} F_{j,k}\cdot c_j\cdot c_k + \mathbf{LFE}^0.\mathbf{Dec}(\mathsf{fk}_{\mathbf{Enc}^1\circ F_2}),$$

which implies that $\mathbf{LFE}^0$ simulator is invoked with β^0_2. This is because letting $\mathsf{ct}' = (\boldsymbol{c}, \boldsymbol{b}', \boldsymbol{a})$, this modification always guarantees

$$\begin{aligned}
\mathbf{Dec}^0(\mathsf{fk}^0_{F_1}, \mathsf{ct}) &= \mathbf{Dec}^0(\mathsf{fk}^0_{F_1}, \mathsf{ct}')\\
\mathbf{EDec}^0(\mathsf{fk}_{\mathbf{Enc}^1\circ F_2}, \mathsf{ct}) &= \mathbf{EDec}^0(\mathsf{fk}_{\mathbf{Enc}^1\circ F_2}, \mathsf{ct}')\\
\mathbf{Dec}^1(\mathsf{fk}^1_{F_3}, \mathbf{EDec}^0(\mathsf{fk}_{\mathbf{Enc}^1\circ F_2}, \mathsf{ct})) &= \mathbf{Dec}^1(\mathsf{fk}^1_{F_3}, \mathbf{EDec}^0(\mathsf{fk}_{\mathbf{Enc}^1\circ F_2}, \mathsf{ct}'))\\
\mathbf{EDec}^1(\mathsf{fk}_{\mathbf{Enc}^2\circ F_4}, \mathbf{EDec}^0(\mathsf{fk}_{\mathbf{Enc}^1\circ F_2}, \mathsf{ct})) &= \mathbf{EDec}^1(\mathsf{fk}_{\mathbf{Enc}^2\circ F_4}, \mathbf{EDec}^0(\mathsf{fk}_{\mathbf{Enc}^1\circ F_2}, \mathsf{ct}'))
\end{aligned}$$

for any F_1, F_2, F_3 and F_4 of which domain and co-domain are well-defined.

Hence, **Hybrid 0** is indistinguishable from **Hybrid 1** assuming that $\mathbf{LFE}^0$ is secure. □

Lemma 7.3 ***Hybrid 1*** *and* ***Hybrid 2*** *are statistically indistinguishable*

Proof. This lemma can be directly obtained from the choice of $\boldsymbol{\eta}$. Indeed, due to the choice of $\boldsymbol{\eta}$, β^0_i and β^2_i are statistically indistinguishable. □

Lemma 7.4 ***Hybrid 2*** *and* ***Hybrid 3*** *are indistinguishable under the hardness of RLWE.*

Proof. Under the RLWE assumption, a pair $(\boldsymbol{u}, \boldsymbol{c})$ satisfying $\boldsymbol{c} = \boldsymbol{u} \cdot s + \boldsymbol{e} + P \cdot \boldsymbol{m}' \in R^{\ell+1}$, where $\boldsymbol{u}$ is a randomly sampled public parameter, $s \leftarrow \mathcal{D}_{R,\alpha_0 q}$, and $\boldsymbol{e} \leftarrow \mathcal{D}^{\ell+1}_{R,\alpha_0 q}$, is indistinguishable from a random vector.

Given an adversary $\mathcal{B}$ that distinguishes between **Hybrid 2** and **Hybrid 3**, we construct an adversary $\mathcal{A}$ that breaks the RLWE assumption by distinguishing $\boldsymbol{c}$ from a random vector. The adversary $\mathcal{A}$ receives $\boldsymbol{u}$ and simulates the view of $\mathcal{B}$ as follows:

1. Invoke QFE^t for $t \in \{0, 1, 2\}$ and LFE to get $\mathsf{QFE}^t.\mathsf{pp}$ and $\mathsf{LFE}.\mathsf{pp}$.
2. Sample $u^t_{i,h} \leftarrow R_q$ for $i \in [\ell_1], h \in [Q+1]$, and $t \in [2]$
3. Return $\mathsf{pp} = (\{\mathsf{QFE}^i.\mathsf{pp}\}_i, \mathsf{LFE}.\mathsf{pp}, \{u^t_{i,h}\}, Q, \{\alpha_i\}, \mathsf{LFE}^0.\mathsf{pp}, \boldsymbol{u})$ to $\mathcal{B}$.
4. When $\mathcal{B}$ requests keys $F_1, \mathsf{Enc}^1 \circ F_2$. construct them as in **Hybrid 0**.
5. When $\mathcal{B}$ outputs the challenge $\boldsymbol{m}$, $\mathcal{A}$ outputs the same.
6. $\mathcal{A}$ receives $\boldsymbol{c}$ where $\boldsymbol{c} = \boldsymbol{u} \cdot s + \boldsymbol{e} + P \cdot \boldsymbol{m}'$ or random.
7. $\mathcal{A}$ computes β_i^2 as in **Hybrid 2**. It invokes $\mathsf{LFE}^0.\mathsf{Sim}(\cdot)$ receives LFE^0 ciphertext $\boldsymbol{b}$. It returns to $(\boldsymbol{c}, \boldsymbol{b})$ to $\mathcal{B}$.
8. Finally, when $\mathcal{B}$ outputs a guess bit b, A outputs the same bit.

Clearly, if $b = 0$, then $\mathcal{B}$ sees the distribution of **Hybrid 2**, whereas if $b = 1$, it sees the distribution of **Hybrid 2**. Hence, we complete the proof.

Under the lemmas, we can complete the proof. □

Acknowledgment. We acknowledge the use of generative AI (Grammarly, ChatGPT) for grammar/spell checking and basic editing.

References

1. M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. Simple functional encryption schemes for inner products. In *IACR International Workshop on Public Key Cryptography*, pages 733–751. Springer, 2015.
2. M. Abdalla, A. De Caro, and K. Mochetti. Lattice-based hierarchical inner product encryption. In *Progress in Cryptology–LATINCRYPT 2012: 2nd International Conference on Cryptology and Information Security in Latin America, Santiago, Chile, October 7-10, 2012. Proceedings 2*, pages 121–138. Springer, 2012.
3. S. Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In *Annual international cryptology conference*, pages 3–35. Springer, 2017.
4. S. Agrawal, F. Kitagawa, A. Modi, R. Nishimaki, S. Yamada, and T. Yamakawa. Bounded functional encryption for turing machines: Adaptive security from general assumptions. In *Theory of Cryptography Conference*, pages 618–647. Springer, 2022.

5. S. Agrawal, B. Libert, M. Maitra, and R. Titiu. Adaptive simulation security for inner product functional encryption. In *Public Key Cryptography (PKC 2020)*, 2020.
6. S. Agrawal, B. Libert, and D. Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *Annual International Cryptology Conference*, pages 333–362. Springer, 2016.
7. S. Agrawal, M. Maitra, N. S. Vempati, and S. Yamada. Functional encryption for turing machines with dynamic bounded collusion from lwe. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part IV 41*, pages 239–269. Springer, 2021.
8. S. Agrawal and A. Rosen. Functional encryption for bounded collusions, revisited. In *Theory of Cryptography Conference*, pages 173–205. Springer, 2017.
9. P. Ananth, D. Boneh, S. Garg, A. Sahai, and M. Zhandry. Differing-inputs obfuscation and applications. *Cryptology ePrint Archive*, 2013.
10. P. Ananth and A. Jain. Indistinguishability obfuscation from compact functional encryption. In *Annual Cryptology Conference*, pages 308–326. Springer, 2015.
11. P. Ananth, A. Jain, and A. Sahai. Indistinguishability obfuscation from functional encryption for simple functions. *Cryptology ePrint Archive*, 2015.
12. P. Ananth and V. Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In *Theory of Cryptography Conference*, pages 174–198. Springer, 2019.
13. C. E. Z. Baltico, D. Catalano, D. Fiore, and R. Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In *Annual International Cryptology Conference*, pages 67–98. Springer, 2017.
14. A. Belel, R. Dutta, and S. Mukhopadhyay. Hierarchical identity-based inner product functional encryption for unbounded hierarchical depth. In *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, pages 274–288. Springer, 2023.
15. K. Bhushan, A. Korb, and A. Sahai. Dynamic bounded-collusion streaming functional encryption from minimal assumptions. In *Annual International Cryptology Conference*, pages 137–169. Springer, 2025.
16. N. Bitansky and V. Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *Journal of the ACM (JACM)*, 65(6):1–37, 2018.
17. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference*, pages 253–273. Springer, 2011.
18. Z. Brakerski, N. Chandran, V. Goyal, A. Jain, A. Sahai, and G. Segev. Hierarchical functional encryption. In C. H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 8:1–8:27. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
19. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.
20. S. Carpov, C. Fontaine, D. Ligier, and R. Sirdey. Illuminating the dark or how to recover what should not be seen in fe-based classifiers. *Proceedings on Privacy Enhancing Technologies*, 2020(2):5–23, 2020.
21. Y. Chen, V. Vaikuntanathan, B. Waters, H. Wee, and D. Wichs. Traitor-tracing from lwe made simple and attribute-based. In *Theory of Cryptography: 16th International Conference, TCC 2018, Panaji, India, November 11–14, 2018, Proceedings, Part II 16*, pages 341–369. Springer, 2018.

22. J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in cryptology–ASIACRYPT 2017: 23rd international conference on the theory and applications of cryptology and information security, Hong kong, China, December 3-7, 2017, proceedings, part i 23*, pages 409–437. Springer, 2017.
23. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.
24. E. Dufour-Sans, R. Gay, and D. Pointcheval. Reading in the dark: Classifying encrypted digits with functional encryption. *Cryptology ePrint Archive*, 2018.
25. R. Garg, R. Goyal, and G. Lu. Dynamic collusion functional encryption and multi-authority attribute-based encryption. In *IACR International Conference on Public-Key Cryptography*, pages 69–104. Springer, 2024.
26. R. Garg, R. Goyal, G. Lu, and B. Waters. Dynamic collusion bounded functional encryption from identity-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 736–763. Springer, 2022.
27. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016.
28. R. Gay. A new paradigm for public-key functional encryption for degree-2 polynomials. In *Public Key Cryptography (1)*, pages 95–120, 2020.
29. C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, STOC '09, pages 169–178, New York, NY, USA, May 2009. Association for Computing Machinery.
30. R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*, pages 201–210. PMLR, 2016.
31. S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 555–564, 2013.
32. S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan. Robustness of the learning with errors assumption. 2010.
33. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In *Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 162–179. Springer, 2012.
34. J. Guan, A. Korb, and A. Sahai. Streaming functional encryption. In *Annual International Cryptology Conference*, pages 433–463. Springer, 2023.
35. A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 60–73, 2021.
36. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in Cryptology–EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings 27*, pages 146–162. Springer, 2008.
37. A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product

encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 62–91. Springer, 2010.

38. Q. Li, Z. Huang, W.-j. Lu, C. Hong, H. Qu, H. He, and W. Zhang. Homopai: A secure collaborative machine learning platform based on homomorphic encryption. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1713–1717. IEEE, 2020.
39. D. Ligier, S. Carpov, C. Fontaine, and R. Sirdey. Privacy preserving data classification using inner-product functional encryption. In *ICISSP*, pages 423–430, 2017.
40. H. Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 28–57. Springer, 2016.
41. H. Lin and S. Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local prgs. In *Annual International Cryptology Conference*, pages 630–660. Springer, 2017.
42. A. Lombardi and V. Vaikuntanathan. Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In *Theory of Cryptography Conference*, pages 119–137. Springer, 2017.
43. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 1–23. Springer, 2010.
44. J. M. B. Mera, A. Karmakar, T. Marc, and A. Soleimanian. Efficient lattice-based inner-product functional encryption. In *IACR International Conference on Public-Key Cryptography*, pages 163–193. Springer, 2022.
45. T. Okamoto and K. Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 99(1):92–117, 2016.
46. A. O'Neill. Definitional issues in functional encryption. *IACR Cryptol. ePrint Arch.*, 2010:556, 2010.
47. P. Panzade and D. Takabi. Fenet: Privacy-preserving neural network training with functional encryption. In *Proceedings of the 9th ACM International Workshop on Security and Privacy Analytics*, pages 33–43, 2023.
48. B. D. Rouhani, M. S. Riazi, and F. Koushanfar. Deepsecure: Scalable provably-secure deep learning. In *Proceedings of the 55th annual design automation conference*, pages 1–6, 2018.
49. T. Ryffel, E. Dufour-Sans, R. Gay, F. Bach, and D. Pointcheval. Partially encrypted machine learning using functional encryption. In *NeurIPS 2019-Thirty-third Conference on Neural Information Processing Systems*, 2019.
50. E. Tairi and A. Ünal. Lower bounds for lattice-based compact functional encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 249–279. Springer, 2024.
51. S. Wagh, D. Gupta, and N. Chandran. Securenn: 3-party secure computation for neural network training. *Proc. Priv. Enhancing Technol.*, 2019(3):26–49, 2019.
52. R. Xu, J. B. Joshi, and C. Li. Cryptonn: Training neural networks over encrypted data. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1199–1209. IEEE, 2019.

A Missing Preliminaries

A.1 Lattice Backgrounds

A discrete subgroup Λ of $\mathbb{R}^n$ is called a lattice. If every vector in Λ is generated by integer linear combinations of $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ and $\boldsymbol{b}_i$'s are linearly independent to each other, then we call $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ a basis of Λ.

Discrete Gaussian. For $\sigma > 0$ and $\boldsymbol{c} \in \mathbb{R}^n$, we define the Gaussian function $\rho_{\boldsymbol{c},\sigma} : \mathbb{R}^n \to (0, 1]$:

$$\rho_{\boldsymbol{c},\sigma}(\boldsymbol{x}) = \exp(-\pi\|\boldsymbol{x} - \boldsymbol{c}\|^2/\sigma^2).$$

Similar to the Gaussian function, we define the discrete Gaussian distribution on a lattice Λ, denoted by $\mathcal{D}_{\Lambda,\boldsymbol{c},\sigma} : \Lambda \to (0, 1]$:

$$\mathcal{D}_{\Lambda,\boldsymbol{c},\sigma}(\boldsymbol{x}) = \frac{\rho_{\boldsymbol{c},\sigma}(\boldsymbol{x})}{\sum_{\boldsymbol{x}\in\Lambda} \rho_{\boldsymbol{c},\sigma}(\boldsymbol{x})}.$$

We usually take $\boldsymbol{c} = \boldsymbol{0}$ when omitted. Moreover, if Λ is obvious, then we also omit it.

Lemma 1 (Flooding Lemma [32]). *Let $n \in \mathbb{N}$. For any $\sigma \geq \omega(\sqrt{\log n})$, and any $\boldsymbol{c} \in \mathbb{Z}^n$, the statistical distance between $\mathcal{D}_{\Lambda,\boldsymbol{c},\sigma}$ and $\mathcal{D}_{\Lambda,\sigma}$ is bounded by $\frac{\|\boldsymbol{c}\|}{\sigma}$.*

Hardness Assumptions. We briefly introduce hardness assumptions of our construction.

Definition 3 (Ring-LWE [43]). *Let $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$ be a ring with a power of two integer n. Let $R_q = R/qR$ for any $q \geq 2$ and χ be a probability distribution over R_q. For $s \in R_q$, let $A_{s,\chi}$ be the probability distribution on $R_q \times R_q$ defined as follows: 1) sample $a \leftarrow R_q$ and $e \leftarrow \chi$ and 2) return a pair $(a, a \cdot s + e)$.*

The decision Ring-LWE problem is to distinguish a RLWE distribution $A_{s,\chi}$ and the uniform random distribution on $R_q \times R_q$.

Lemma 2 (Hardness of Ring-LWE [43]). *Let $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$ be a ring with a power of two integer n and $q \geq 2$ be an integer. Let $\alpha q \geq \omega(\sqrt{\log n})$ be a real number. Then, there exists a polynomial time randomized reduction from $2^{\omega(\sqrt{\log n})}(1/\alpha)$ approximate Ideal-SVP to $RLWE_{R,q,\chi}$, where χ is the Gaussian distribution with the standard deviation αq.*

A.2 Lattice-based Linear Functional Encryption

This section introduces the construction of a linear functional encryption scheme from lattices, **LFE**, and its properties. For a detailed discussion on the construction, we refer to the LFE literature [1, 5, 6, 44].

Definition 4 (LFE). *Let $\mathcal{K}$ be a key space and $\mathcal{P}$ a message space. A (public-key) linear functional encryption scheme for $\mathcal{K}$ with message space $\mathcal{P}$ is a tuple of four probabilistic polynomial-time (PPT) algorithms:*

- ***Setup***$(\lambda, \mathcal{P}, \mathcal{K})$*: On input security parameter λ, outputs public parameters* pp *and master secret key* msk*.*
- ***KGen***$(\mathsf{msk}, \boldsymbol{x} \in \mathcal{K})$*: On input master secret key* msk *and key $\boldsymbol{x}$, outputs a functional key* $\mathsf{fk}_{\boldsymbol{x}}$*.*
- ***Enc***$(\mathsf{pp}, \boldsymbol{y} \in \mathcal{P})$*: On input* pp *and message $\boldsymbol{y}$, outputs a ciphertext* ct *encrypting $\boldsymbol{y}$.*
- ***Dec***$(\mathsf{pp}, \mathsf{ct}, \mathsf{fk}_{\boldsymbol{x}})$*: On input* pp*,* ct*, and* $\mathsf{fk}_{\boldsymbol{x}}$*, outputs a value μ.*

A scheme $\mathbf{LFE} = (\mathbf{Setup}, \mathbf{KGen}, \mathbf{Enc}, \mathbf{Dec})$ *is correct if for all* $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathbf{Setup}(\lambda, \mathcal{P}, \mathcal{K})$*, any $\boldsymbol{y} \in \mathcal{P}$, and any $\boldsymbol{x} \in \mathcal{K}$, for* $\mathsf{fk}_{\boldsymbol{x}} \leftarrow \mathbf{KGen}(\mathsf{msk}, \boldsymbol{x})$ *and* $\mathsf{ct} \leftarrow \mathbf{Enc}(\mathsf{pp}, \boldsymbol{y})$*, we have that* $\mathbf{Dec}(\mathsf{pp}, \mathsf{ct}, \mathsf{fk}_{\boldsymbol{x}}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle$ *with all but negligible probability.*

For simplicity, we describe the construction from [1], but one can employ alternative LFE constructions such as [5, 6, 44] to build quadratic functional encryption schemes (QFE). We note that these constructions can be also applicable to build QFE or CFE. If then, the only differences lie in the choice of parameters and the length of ciphertexts. In other words, the essential constructions are the same as the construction built from [1]. Moreover, our construction in the main body is a modular construction.

Let P, ℓ and K be integers. A message space of $\mathbf{LFE}$ is $\mathcal{P} = \{0, 1, \ldots, P-1\}^{\ell} \subset R^{\ell}$, and secret keys are associated with vectors in $\mathcal{K} = \{0, 1, \ldots, K-1\}^{\ell} \subset R^{\ell}$.

$\mathbf{LFE.Setup}(\lambda, \ell, P, K)$: Set integers $q \geq 2$ a power of P and real $\alpha \in (0, 1)$ and a polynomial ring $R = \mathbb{Z}[x]/\langle x^n + 1 \rangle$ ensuring correctness and λ-bit security. These q and n are computed from ℓ, P, and K. Sample $a \leftarrow R_q$, vectors $\boldsymbol{s} \leftarrow \mathcal{D}^{\ell}_{R,\alpha q}$ and $\boldsymbol{e} \leftarrow \mathcal{D}^{\ell}_{R,\alpha q}$. Return the public parameter $\mathsf{pp} = ((a, \boldsymbol{b} = a \cdot \boldsymbol{s} + \boldsymbol{e} \bmod q), \mathcal{P}, \mathcal{K})$ and the master secret key $\mathsf{msk} = \boldsymbol{s}$.

$\mathbf{LFE.KGen}(\mathsf{msk}, \boldsymbol{x} \in \mathcal{K})$: On inputs $\boldsymbol{x}$ and msk, return a functional key $\mathsf{fk} = (\boldsymbol{x}, \langle \boldsymbol{x}, \boldsymbol{s} \rangle)$.

$\mathbf{LFE.Enc}(\mathsf{pp}, \boldsymbol{y} \in \mathcal{P})$: Sample $r, e_0 \leftarrow \mathcal{D}_{R,\alpha q}$ and $\boldsymbol{e}_1 \leftarrow \mathcal{D}^{\ell}_{R,\alpha q}$. Next, return

$$c_0 = a \cdot r + e_0$$
$$\boldsymbol{c}_1 = \boldsymbol{b} \cdot r + \boldsymbol{e}_1 + P \cdot \boldsymbol{y} \in R^{\ell}_q.$$

Last, return $\mathsf{ct} = (c_0, \boldsymbol{c}_1) \in R^{1+\ell}_q$.

$\mathbf{LFE.Dec}(\mathsf{pp}, \mathsf{ct}, \mathsf{fk})$: On inputs pp, ct, and fk, parse $\mathsf{ct} = (c_0, \boldsymbol{c}_1)$ and $\mathsf{fk} = (\boldsymbol{fk}_0, fk_1)$. Then, compute $\mu' = \langle \boldsymbol{fk}_0, \boldsymbol{c}_1 \rangle - c_0 \cdot fk_1$, and return the value $\lfloor \frac{1}{P} \cdot \mu' \rceil$.

Correctness of the LFE. By the definition of $\mathbf{LFE.Dec}$, we first compute $\mu' = \langle \boldsymbol{fk}_0, \boldsymbol{c}_1 \rangle - c_0 \cdot fk_1$. Since $c_0 \cdot \mathsf{fk}_1 = (a \cdot r + e_0) \cdot \langle \boldsymbol{x}, \boldsymbol{s} \rangle$ and $\langle \mathsf{fk}_0, \boldsymbol{c}_1 \rangle = \langle \boldsymbol{x}, \boldsymbol{b} \cdot r +$

$\boldsymbol{e}_1 + P \cdot \boldsymbol{y}\rangle$, we have

$$\begin{aligned}
\mu' &= \langle \mathsf{fk}_0, \boldsymbol{c}_1 \rangle - c_0 \cdot \mathsf{fk}_1 \\
&= \langle \boldsymbol{x}, \boldsymbol{b} \cdot r + \boldsymbol{e}_1 + P \cdot \boldsymbol{y} \rangle - (a \cdot r + e_0) \cdot \langle \boldsymbol{x}, \boldsymbol{s} \rangle \\
&= \langle \boldsymbol{x}, (a \cdot \boldsymbol{s} + \boldsymbol{e}) \cdot r + \boldsymbol{e}_1 + P \cdot \boldsymbol{y} \rangle - (a \cdot r + e_0) \cdot \langle \boldsymbol{x}, \boldsymbol{s} \rangle \\
&= \langle \boldsymbol{x}, \boldsymbol{e} \cdot r \rangle + \langle \boldsymbol{x}, \boldsymbol{e}_1 \rangle + P \cdot \langle \boldsymbol{x}, \boldsymbol{y} \rangle - e_0 \cdot \langle \boldsymbol{x}, \boldsymbol{s} \rangle \\
&= P \cdot \langle \boldsymbol{x}, \boldsymbol{y} \rangle + e^*.
\end{aligned}$$

If $\|e^*\|_\infty < P/2$, the μ is equal to $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$.

Security of the LFE. Since **LFE** is a ring variant of LWE based inner product encryption in [1], it satisfies selective IND-CPA security as in the original paper. It suffices to prove the full-simulation security of our CFE.

Property of the LFE. We state the following additional properties:

- **Malleability**: The ciphertext components of **LFE** are malleable so that if $\mathsf{ct} = \mathbf{LFE.Enc}(\mathsf{pp}, \boldsymbol{x})$ then $\mathsf{ct} + P \cdot \boldsymbol{y} = \mathbf{LFE.Enc}(\mathsf{pp}, \boldsymbol{x} + \boldsymbol{y})$ as long as $\boldsymbol{x} + \boldsymbol{y}$ belong to the message space.
- **Additive homomorphism of ciphertext and functional key**: The ciphertext and functional key enjoy additive homomorphism in the following sense. If $\mathsf{ct}_i = \mathbf{LFE.Enc}(\mathsf{pp}, y_i)$ then $\sum_i x_i \cdot \mathsf{ct}_i = \mathbf{LFE.Enc}(\mathsf{pp}, \sum_i x_i \cdot y_i)$ as long as $\sum_i x_i \cdot y_i$ belongs to the message space. Similarly, given $\mathsf{fk}_i = \mathbf{LFE.KGen}(\mathsf{msk}, x_i)$, $\sum_i y_i \cdot \mathsf{fk}_i = \mathbf{LFE.KGen}(\mathsf{msk}, \sum_i y_i \cdot x_i)$ as long as $\sum_i x_i \cdot y_i$ belongs to the function space.
- **Modulus Switching**: The ciphertext of the form $(c_0, \boldsymbol{c}_1) = \mathbf{LFE.Enc}(\mathsf{pp}, P \cdot x)$ with randomness $r, e_0, \boldsymbol{e}_1$ can be converted into a ciphertext $\mathbf{LFE.Enc}(\mathsf{pp}, x)$ with less modulus q/P via computing $\lfloor \frac{\mathsf{ct}}{P} \rceil$ as long as $\|r\|_1 + \frac{\|e_0, \boldsymbol{e}_1\|_\infty}{P} < P$, where $\|e_0, \boldsymbol{e}_1\|_\infty$ is the maximal absolute coefficient values between e_0 and $\boldsymbol{e}_1$.

B Indistinguishability-Based Security Definition

Definition 5 (Indistinguishability-based security of CFE). *Let* $\boldsymbol{CFE} = (\boldsymbol{Setup}, \boldsymbol{Enc}^0, \boldsymbol{EKGen}^t, \boldsymbol{EDec}^t, \boldsymbol{KGen}^t, \boldsymbol{Dec}^t)$ *be a Q-bounded T-level composable functional encryption scheme for a function family $\mathcal{F}$. For every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we define security against chosen-plaintext attacks (IND-CPA security, for short) via the security game depicted as follows:*

$\mathsf{Exp}^{IND\text{-}CPA\text{-}b}_{\mathbf{CFE},\lambda}(\mathcal{A})$:

(1) $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathbf{Setup}(\lambda, \mathcal{M}, \mathcal{F}, Q, T)$

(2) $\mathbf{F}_1 \leftarrow \mathcal{A}_1(1^\lambda, \mathsf{pp}, \mathcal{F})$ *where* $\mathbf{F}_1 = \{F_i\}$ *for* $i = 1, \ldots, Q_1$

(3) $(m_0^*, m_1^*, st) \leftarrow \mathcal{A}_1^{\mathbf{EKGen}^t(\mathsf{msk},\cdot),\mathbf{KGen}^t(\mathsf{msk},\cdot)}(\mathsf{pp}, \mathbf{F}_1)$ *where* $|m_0^*| = |m_1^*|$

(4) $\mathsf{ct}^* \leftarrow \mathbf{Enc}^0(\mathsf{pp}, m_b^*)$

(5) $\mathbf{F}_2 \leftarrow \mathcal{A}_2(1^\lambda, \mathsf{pp}, \mathcal{F})$ *where* $\mathbf{F}_2 = \{F_i\}$ *for* $i = Q_1 + 1, \ldots, Q$

(6) $b' \leftarrow \mathcal{A}_2^{\mathbf{EKGen}^t(\mathsf{msk},\cdot),\mathbf{KGen}^t(\mathsf{msk},\cdot)}(\mathsf{pp}, \mathsf{ct}^*, st, \mathbf{F}_2)$

(7) Output b'

Define an adversary $\mathcal{A}$ *as one which makes at most* Q *quries and* $F_i(m_0^*) = F_i(m_1^*)$ *for every* $i \in [Q]$*. The advantage of* $\mathcal{A}$ *in the above game is defined as*

$$\mathsf{Adv}_{\mathbf{CFE}}(\mathcal{A}) := \left|\Pr[\mathsf{Exp}^{IND\text{-}CPA\text{-}0}_{\mathbf{CFE},\lambda}(\mathcal{A}) = 1] - \Pr[\mathsf{Exp}^{IND\text{-}CPA\text{-}1}_{\mathbf{CFE},\lambda}(\mathcal{A}) = 1]\right|.$$

We say that **CFE** *is secure against chosen-plaintext attacks (IND-CPA secure, for short) if any polynomial time adversary* $\mathcal{A}$ *has at most negligible advantage in* λ *in the above game.*

C Parameter Selection

This section provides parameter selection for the security and the correctness of our CFE construction. Note that Q, K, M are inputs of **Setup**. The parameters are chosen to satisfy

- To guarantee λ-bit security for the RLWE assumption, we set n, q, α_0 following Lemma 2.
- Correctness/Noise flooding lemma for four decryption algorithms

We summarize parameter constraints for four decryption algorithms.

$\mathbf{Dec}^0$. Correctness and Noise flooding lemma of $\mathbf{Dec}^0$ are exactly same to that of QFE. Hence, it requires that

$$\alpha_1 \cdot q \leq P/2 \text{ and } \frac{\alpha_0 q \cdot (\ell+1)^2 \cdot K \cdot P \cdot M}{\alpha_1 q} = \mathsf{negl}(\lambda)$$

$\mathbf{EDec}^0$. From the decryption correctness, a constraint is required:

$$\frac{\alpha_0 q \cdot P \cdot K \cdot (\ell+1)^2 \cdot M + P \cdot Q \cdot \alpha_0 q_1}{\alpha_1 q} = \mathsf{negl}(\lambda)$$

$\mathbf{Dec}^1$. From the decryption correctness, two constraints are required:

$$\sigma_{\mathsf{LFE}} \cdot K \cdot (Q+2)^2 < q/2 \text{ and } \frac{\alpha_1 q \cdot (\ell_1+1)^2 \cdot K \cdot P \cdot M_1}{\alpha_2 q} = \mathsf{negl}(\lambda),$$

where M_1 is the largest message size after the $\mathbf{EDec}^0$.

$\mathbf{EDec}^1$. From the decryption correctness, two constrains are required:

$$\sigma_{\mathsf{LFE}} \cdot q_1 \cdot (Q+2)^2 < q/2 \text{ and } \frac{\alpha_1 q \cdot P \cdot K \cdot (\ell_1+1)^2 \cdot M_1 + P \cdot Q \cdot \alpha_1 q_2}{\alpha_2 q} = \mathsf{negl}(\lambda)$$

D Expressing Higher-Degree maps via Quadratic Maps

By proving the following theorem, we claim that our composable FE is able to evaluate high degree polynomials.

Theorem 1. *Any multivariate polynomial $f(x_1, \ldots, x_n)$ of total degree $d \geq 2$ over a field $\mathbb{F}$ can be expressed as a finite composition of quadratic polynomial maps.*

Proof. The proof can proceed by induction on the number of variables n and then on the degree d.
Base case for n: If $n = 1$, $f(x_1)$ is a univariate polynomial of degree d.

- **Base case for d:** If $d \leq 2$, $f(x_1)$ is itself a quadratic polynomial.
- **Inductive step for d:** Assume any univariate polynomial $g(x_1)$ of degree $d' < d$ can be expressed as a composition of quadratic maps. A polynomial $f(x_1)$ of degree $d > 2$ can be written as $f(x_1) = P(x_1) \cdot x_1 + c_0$, where $P(x_1) = \sum_{i=1}^{d} c_i x_1^{i-1}$ has degree $d-1$ and c_0 is a constant. By the inductive hypothesis, $P(x_1)$ is a composition of quadratic maps, say $\mathcal{C}_P$. The constant c_0 is represented by a quadratic map $\mathcal{C}_{c_0}$. The operations $y \cdot z$ and $y + z$ are elementary quadratic maps (f_{mult} and f_{add}). Thus, $f(x_1)$ is formed by composing $\mathcal{C}_P$, $\mathcal{C}_{c_0}$, f_{mult} (to compute $P(x_1) \cdot x_1$), and f_{add} (to add c_0). Hence, $f(x_1)$ is a composition of quadratic maps.

Thus, any univariate polynomial can be expressed as a composition of quadratic maps.
Inductive step for n: Assume that any polynomial in $k < n$ variables (of any degree) can be expressed as a composition of quadratic maps. We now prove for n variables. This itself uses induction on the degree d of $f(x_1, \ldots, x_n)$.

- **Base case for d:** If $d \leq 2$, $f(x_1, \ldots, x_n)$ is itself a quadratic polynomial.
- **Inductive step for d:** Assume any polynomial in n variables of degree $d' < d$ can be expressed as a composition of quadratic maps, and any polynomial in $n-1$ variables (of any degree) can also be so expressed (from the induction on n).
  Let $f(x_1, \ldots, x_n)$ be a polynomial of degree $d > 2$ and $n \geq 1$. Pick a variable x_n such that f is not constant with respect to x_n. We can write f as: $f(x_1, \ldots, x_n) = \sum_{k=0}^{m} a_k(x_1, \ldots, x_{n-1}) x_n^k$, where $m = \deg_{x_n}(f) \geq 0$.

If $m = 0$, then $f(\vec{x}) = a_0(x_1, \ldots, x_{n-1})$. This is a polynomial in $n-1$ variables of degree d. By the inductive hypothesis (on n), a_0 can be expressed as a composition of quadratic maps.
If $m \geq 1$, we can write $f(\vec{x}) = P(\vec{x}) \cdot x_n + a_0(x_1, \ldots, x_{n-1})$, where $P(\vec{x}) = \sum_{k=1}^{m} a_k(x_1, \ldots, x_{n-1}) x_n^{k-1}$. The polynomial $P(\vec{x})$ has n variables. If the highest degree terms of f involve x_n, then the total degree of $P(\vec{x})$ is $d-1$. By the inductive hypothesis (on d), $P(\vec{x})$ can be expressed as a composition of quadratic maps, denoted $\mathcal{C}_P$. The polynomial $a_0(x_1, \ldots, x_{n-1})$ has $n-1$ variables and its degree is at most d. By the inductive hypothesis (on n), a_0 can be expressed as a composition of quadratic maps, denoted $\mathcal{C}_{a_0}$.
Let y_P be the result of $\mathcal{C}_P$ and y_{a_0} be the result of $\mathcal{C}_{a_0}$. We then construct f using elementary quadratic maps:

1. $M(y_P, x_n) = y_P \cdot x_n$. (Let the output be y_M).
2. $A(y_M, y_{a_0}) = y_M + y_{a_0}$. (This output is f).

The overall computation of f is thus a composition of $\mathcal{C}_P$, $\mathcal{C}_{a_0}$, and the maps M and A. Since $\mathcal{C}_P$ and $\mathcal{C}_{a_0}$ are compositions of quadratic maps, their sequential application followed by M and A (which are also quadratic maps) means that f itself is expressed as a composition of quadratic maps.

This completes the induction. Therefore, any multivariate polynomial of total degree d can be expressed as a finite composition of quadratic polynomial maps.