# Non-interactive Fully Encrypted Machine Learning Protocol for Inference

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

[1] Columbia University / New York Genome Center
shong@nygenome.org
[2] Jeonbuk National University
jiseungkim@jbnu.ac.kr
[3] Korea University
changminlee@korea.ac.kr
[4] Duksung Women's University
mhseo@duksung.ac.kr

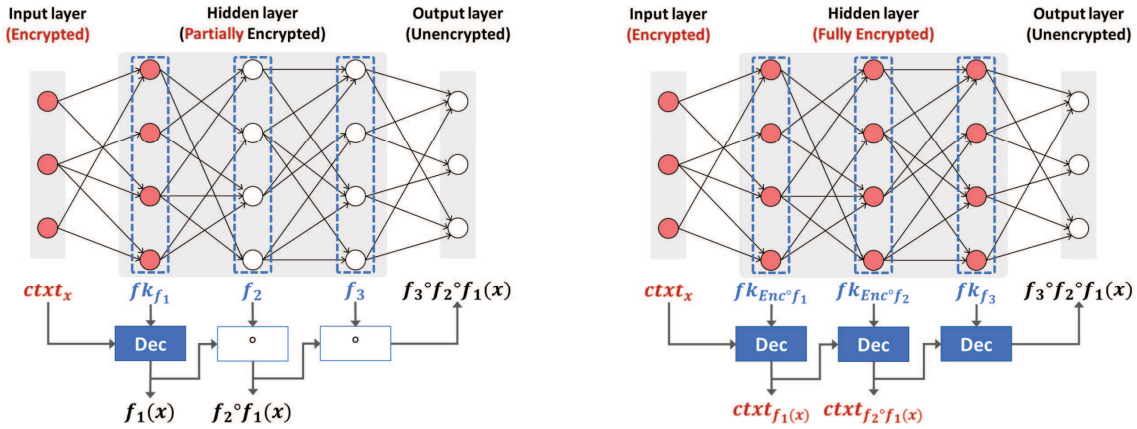
**Abstract.** As privacy concerns have arisen in machine learning, privacy-preserving machine learning (PPML) has received significant attention. Fully homomorphic encryption (FHE) and secure multi-party computation (MPC) are representative building blocks for PPML. However, in PPML protocols based on FHE and MPC, interaction between the client (who provides encrypted input data) and the evaluator (who performs the computation) is essential to obtain the final result in plaintext. Functional encryption (FE) is a promising candidate to remove this constraint, but existing FE-based PPML protocols are restricted to evaluating only simple ML models, such as one-layer neural networks, or they support partially encrypted PPML, which makes them vulnerable to information leakage beyond the inference results.

In this paper, we propose a fully encrypted FE-based PPML protocol, which supports the evaluation of arbitrary functions over encrypted data with no information leakage during computation, for the first time. To achieve this, we introduce a new functional encryption scheme for quadratic polynomials, in which the encryption algorithm is realized as a linear function with respect to a message. This structural property allows for the composition of the encryption algorithm with arbitrary quadratic functions, thereby enabling multiple compositions of quadratic polynomials to compute arbitrary complex functions in an encrypted manner.

Our FE-based PPML protocol is secure in the malicious model, which means that an adversary cannot obtain any information about the input data even though they intentionally deviate from the protocol. We then show how to use our protocol to build a fully encrypted 2-layer neural network model with quadratic activation functions and present experimental results.

## 1 Introduction

Machine Learning (ML) has become a vital technology for companies across various industries, as it enables them to provide services that enhance people's quality of life. In traditional machine learning, the data is generally centralized and available to the machine learning algorithm in its raw form. However, when dealing with sensitive data, it is crucial to safeguard the privacy of individuals represented in the data. For example, in the healthcare industry, machine learning models are used to analyze medical data for diagnosis, treatment, and drug discovery. However, medical data is highly sensitive, containing personal information about patients [33, 38]. Similarly, in finance, machine learning models are utilized for fraud detection, risk assessment, and other applications, which often contain sensitive information about individuals' income and spending habits [9, 36]. Additionally, online advertising, which employs machine learning models to personalize ads for individual users, requires the protection of sensitive information such as browsing habits and interests [8, 26]. As ML increasingly permeates various businesses and organizations, privacy issues concerning the underlying data have become more prominent. Privacy-preserving machine learning (PPML) techniques and approaches have been developed to enable machine learning models to




(a) The existing partially encrypted FE-based PPML

(b) Our fully encrypted FE-based PPML

**Fig. 1.** Privacy-preserving machine learning based on functional encryption

provide a useful service while maintaining the data's privacy. In line with this, research on PPML has begun to draw significant attention [17, 18, 28, 45].

The typical approaches to PPML are based on fully homomorphic encryption (FHE) and secure multi-party computation (MPC). However, FHE-based and MPC-based PPML protocols have their own limitations: MPC-based PPML protocols [28, 31, 35, 40] require computations to be performed in the online phase, necessitating the active involvement of the client who owns the data throughout the entire process of evaluating computations. In FHE-based PPML protocols [13, 15, 20, 24], the client does not have to remain online during computations, i.e., he/she encrypts the data prior to the computation and needs not involve in any intermediate computation. However, after the computation of encrypted data, the evaluator (performing the computations) should interact with the client (data owner) to obtain decrypted results. Additionally, this limitation prevents FHE-based PPML protocols from being deployed in certain applications, such as spam filtering. For example, an e-mail server has to classify an encrypted incoming e-mail as spam or not, but filtering cannot be done without the help of the client (e-mail recipient) because the result of the classification is provided encrypted. After all, spam filtering requires constant user involvement, which is not what we expect from spam filtering.

Functional Encryption (FE) is a promising approach for PPML, as it does not require any interaction during computation. FE allows computation on ciphertexts while revealing only the output of the computation and keeping the inputs private. There have been several studies on FE-based PPML. However, they are limited to either partially encrypted or simple structured ML models because efficient functional encryption schemes so far only support linear or quadratic polynomials. Functional encryption for arbitrary functions can be achieved from various cryptographic primitives such as multilinear map or indistinguishability obfuscation (iO) [4, 12], but they are so inefficient that these are only theoretically feasible. Thus, to date, FE schemes for inner-product and quadratic functionalities have been taken to construct PPML [11, 21, 22, 25, 30, 37, 44] along with an attempt to speed up using hardware accelerator [6]. But the thing is, most works assume 'partially encrypted setting' which means that the first layer is encrypted, but operations in subsequent layers are visible in clear.

Recently, Carpov *et al.* [7] proposed an attack where adversaries can exploit such cleartexts to partially recover the original input data. That is, the intermediate values could yield information leakage about the original encrypted input data, which leaves a gap with real-world scenarios. Thus, it remains an open problem to achieve fully encrypted machine learning protocol via FE that can support complex computations while maintaining strong privacy guarantees.

### 1.1 Contributions

In this paper, we propose a fully encrypted PPML protocol for the first time in the literature that

- does not require any involvement of the client (data owner) in the computation process, and
- allows the evaluator to obtain the final inference result without any interaction (since it is output in plaintext in the output layer), and
- has no intermediate leakage while evaluating arbitrary functions over encrypted data.

We construct our PPML protocol using FE, which enables non-interactive computations on encrypted data and reveals only the final output of the computation. However, to make it "fully encrypted" for evaluating "arbitrary" functions (e.g., 3-degree polynomials or higher), we have considered the following technical ideas:

**1) Imitate the FHE from FE.** Since existing FE could support computation on ciphertext only once, FE-based PPML protocols had to be "partially encrypted". The novel idea of our work is to mimic the FHE in FE framework. To be precise, we consider a composition function of $\mathsf{Enc} \circ f$, where $\mathsf{Enc}$ is an encryption of FE scheme. This composition then allows that an output of evaluation still remains encrypted. For ease of understanding, let $f_i$ be the function corresponding to the $i$-th layer where $i = 1, 2, 3$ as in Fig. 1. In the existing FE-based PPML protocol, the output of the first hidden layer $f_1(x)$ is given in plaintext whereas, in our PPML protocol, the output of the first hidden layer $\mathsf{Enc} \circ f_1(x)$ is presented in ciphertext.

**2) Introduce a new compact[1] FE scheme for function composition.** The composition of the encryption algorithm $\mathsf{Enc}$ and the function $f$ is infeasible in the existing FE schemes. To make the composition work, the structure of the encryption algorithm should be polynomial so that a composition with functions (i.e., $\mathsf{Enc} \circ f$) is a polynomial as well, which is not supported by the existing FE schemes. Therefore, we newly construct a compact FE scheme for quadratic polynomials (LinEnc-QFE in Section 3), such that its encryption algorithm can be represented by a linear function with respect to a message, thereby resulting in ciphertexts that are vectors. Since the encryption algorithm is a linear function, the LinEnc-QFE supports the functionality $\mathsf{Enc} \circ f$ and our protocol works properly as long as the $f$ is a quadratic polynomial. Since all complex functions can be decomposed into a composition of quadratic polynomials, our protocol covers all polynomial functions.

**3) Make the PPML protocol secure in the malicious model.** We then construct a fully encrypted PPML protocol using function compositions based on LinEnc-QFE scheme, called FE-PPML protocol. Although our novel LinEnc-QFE scheme is secure only under bounded ciphertext conditions, it is sufficient to support the FE-PPML protocol.

Note that IND-CPA security of the underlying FE scheme could not guarantee the security of the PPML protocol [7]. To make our PPML protocol secure in the malicious model, we introduce random linear functions $h_i, {h_i}^{-1}$ to randomize the functional keys. In addition, we consider random linear functions $\gamma$ and $\gamma^{-1}$ to randomize the message. Taken together, our PPML protocol is proven to be secure in the malicious model. This means that the adversary cannot obtain any information about the client's input data from the entire transcript of the protocol except for the final result (in the output layer) even if the evaluator may deviate from the protocol arbitrarily and collude with other clients. See more details in Section 4.

Furthermore, we validate the feasibility of our proposed protocol through experimental results. To be precise, we provide an implementation result using inference of 2-layer neural network classifier on the IRIS and Breast Cancer dataset in the UCI Machine Learning Repository [10]. The source code is available in https://github.com/swanhong/fully-encrypted-ml/.

For a detailed comparison of the PPML protocols, please refer to Table 1 and Table 2. In Table 1, the "# interactions (client)" column denotes the number of interactions required from the

[1] There are two definition of the compactness of FE. We here adopt the definition in [3].FE scheme is compact when its encryption time is a polynomial in the security parameter $\lambda$, the number of function queries $Q_{key}$, and the size of input message $m$.

| Research | Type | ML model | Security model | # parties | # interactions (client) |
|---|---|---|---|---|---|
| ABY2.0 [31] | MPC[1] | NN[3] | semi-honest[4] | 2 | ≥ 2 |
| SecureML [28] | | General | semi-honest | 2 | |
| CRYPTGPU [39] | | NN | semi-honest | 3 | |
| CrypTFlow [19] | | NN | semi-honest | 3 | |
| Chameleon [34] | | NN | semi-honest | 3 | |
| SecureNN [40] | | NN | malicious[5] | 3 | |
| ABY[3] [27] | | NN | malicious | 3 | |
| AdamInPrivate [5] | | NN | malicious | 3 | |
| FalconN [41] | | General | malicious | 3 | |
| BLAZE [32] | | General | malicious | 3 | |
| CryptoDL [16] | FHE[2] | NN | semi-honest | 2 | 1 |
| CryptoNets [13] | | NN | semi-honest | 2 | |
| **This work** | FE | NN | malicious | 3 | 0 |

**Table 1.** Comparison with FHE/MPC-based PPML protocol for inference
[1] Secure Multi-Party Computation , [2] Fully Homomorphic Encryption, [3] Neural Network, [4] Adversaries follow the protocol but try to get more information. [5] Adversaries can deviate from the protocol to gain an advantage.

| Research | FE type | ML model | Fully Encrypted |
|---|---|---|---|
| Linger et al. [22] | IPFE[1] | ERT[3] | ✗ |
| Xu et al. [44] | IPFE | 5-layer NN | ✗ |
| Sans et al. [11] | QFE[2] | 2-layer NN | ✗ |
| Ryfell et al. [37] | QFE | 2-layer NN | ✗ |
| **This work** | IPFE / QFE | 2-layer NN | ✓ |

**Table 2.** Implementations of FE-based PPML protocol for inference
[1]Inner Product Functional Encryption [2] Quadratic Functional Encryption [3] Extremely Randomized Trees

client during the inference phase, after transmitting their encrypted data to the evaluator, in order for the evaluator to ultimately obtain the inference result in plaintext. As previously discussed, the proposed PPML protocol operates without any client involvement (i.e., non-interactive) throughout the entire process, including model evaluation and service provision based on the results. This property enhances the practicality of our protocol for deployment in real-world scenarios.

### 1.2 Simple description of protocol

A brief description of our protocol is given here. Our protocol involves three parties: the key distributor, the client, and the evaluator. For simplicity, we assume that an evaluator wants to compute a function $f_3 \circ f_2 \circ f_1$ for an input data $\mathbf{x}$ as in Fig. 1. We denote the underlying FE scheme by $\{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$. Then the proposed protocol is proceeded as follows:

1. The key distributor runs $\mathsf{Setup}$ to obtain $\{\mathsf{pk}, \mathsf{msk}\}$, and samples random pairs $(h_i, {h_i}^{-1})$ for $i = 1, 2$. The key distributor samples random linear functions $(\gamma, \gamma^{-1})$ and sends $(\mathsf{pk}, \gamma)$ to the client.
2. The client encrypts $\gamma(\mathbf{x})$ with a message $\mathbf{x}$ and sends the ciphertext $\mathsf{ct}_{\gamma(\mathbf{x})} \leftarrow \mathsf{Enc}(\mathsf{pk}, \gamma(\mathbf{x}))$ to the evaluator.
3. The evaluator sends functions $f_1, f_2, f_3$ to the key distributor.
4. The key distributor computes functional keys $\mathsf{fk}_1, \mathsf{fk}_2, \mathsf{fk}_3$ and sends them to the evaluator:

- $\mathsf{fk}_1 \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{Enc} \circ F_1)$ where $F_1 = h_1 \circ f_1 \circ \gamma^{-1}$
- $\mathsf{fk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{Enc} \circ F_2)$ where $F_2 = h_2 \circ f_2 \circ h_1^{-1}$
- $\mathsf{fk}_3 \leftarrow \mathsf{KeyGen}(\mathsf{msk}, F_3)$ where $F_3 = f_3 \circ h_2^{-1}$ [5]

5. The evaluator computes the following:
   - $\mathsf{ct}_{h_1 \circ f_1(\mathbf{x})} \leftarrow \mathsf{Dec}(\mathsf{fk}_1, \mathsf{ct}_{\gamma(\mathbf{x})})$
   - $\mathsf{ct}_{h_2 \circ f_2 \circ f_1(\mathbf{x})} \leftarrow \mathsf{Dec}(\mathsf{fk}_2, \mathsf{ct}_{h_1 \circ f_1(\mathbf{x})})$
   - $f_3 \circ f_2 \circ f_1(\mathbf{x}) \leftarrow \mathsf{Dec}(\mathsf{fk}_3, \mathsf{ct}_{h_2 \circ f_2 \circ f_1(\mathbf{x})})$

Note that the evaluator can obtain $f_3 \circ f_2 \circ f_1(\mathbf{x})$ in plaintext at the end of the protocol. Since our LinEnc-QFE is constructed in the symmetric-key setting, we additionally adopt an inner-product (public-key) encryption scheme (pkIPFE in Fig. 2) and combine it with the LinEnc-QFE in our protocol. The client encrypts the input data $\mathbf{x}$ using the pkIPFE scheme in the public-key setting, and then the evaluator converts it into the ciphertext of LinEnc-QFE scheme. See Section 4 for more details.

## 2 Preliminaries

### 2.1 Notation

Throughout this paper, we use bold letters to denote vectors and matrices. Let $\mathbf{O}_n$ be a zero matrix of dimension $n \times n$ and $\mathbf{I}_n$ be an identity matrix of dimension $n \times n$ for any positive integer $n$. Let $\mathbb{Z}$ be the set of all integers and $\mathbb{N}$ the set of all positive integers.

For any $a, b \in \mathbb{Z}$, we simplify $[a, b] \cap \mathbb{Z}$ as $[a, b]$. We also use other simplified interval notations. For any $N \geq 2$, we identify $\mathbb{Z}_N$ as $[-N/2, N/2) \cap \mathbb{Z}$. For any finite set $S$, $s \leftarrow S$ is denoted to sampling $s$ from the uniform distribution over $S$. We denote $\phi$ by the Euler's totient function.

We describe a composition notation for functions. Let $F$ be $(f_1, \ldots, f_k)$, where $f_i : \mathbb{Z}_N^\ell \to \mathbb{Z}_N$ is a quadratic function for every $i$. Then, a composite function $F \circ h$ is denoted by a function of the form $(f_1 \circ h, \ldots, f_k \circ h)$, when the output dimension of the function $h$ is $\ell$. Similarly, $h' \circ f \circ h$ is well-defined for a function $h'$ of input dimension $k$. For every positive integer $i$ and proper input $\mathbf{x}$, $(f_i \circ \cdots \circ f_2 \circ f_1)(\mathbf{x})$ is simply denoted by $\bigcirc_{t=1}^{i} f_t(\mathbf{x})$.

Given $n$-dimensional vector $\mathbf{v} = (v_1, \cdots, v_n)^T$ and group element $g$, we denote $(g^{v_1}, g^{v_2}, \cdots, g^{v_n})^T$ by $g^{\mathbf{v}}$. Moreover, we use a bracket notation $[a]_g$ to denote $g^a$. Similarly, for any vector $\mathbf{v}$ and matrix $\mathbf{A}$, $g^{\mathbf{v}}$ and $g^{\mathbf{A}}$ are denoted by $[\mathbf{v}]_g$ and $[\mathbf{A}]_g$, respectively. Given two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{Z}^n$, we define $[\mathbf{v}]_g^{\mathbf{w}}$ as $g^{\mathbf{v}^T \cdot \mathbf{w}} = g^{\langle \mathbf{v}, \mathbf{w} \rangle}$. In addition, we denote their row concatenation as $(\mathbf{v}, \mathbf{w})$ and their column concatenation as $(\mathbf{v} \| \mathbf{w})$. The Kronecker tensor products of vectors $\mathbf{a} \in \mathbb{Z}_N^n$ and $\mathbf{b} \in \mathbb{Z}_N^m$ or matrices $\mathbf{A} \in \mathbb{Z}_N^{n \times m}$ and $\mathbf{B} \in \mathbb{Z}_N^{r \times s}$ are defined by

$$\mathbf{a} \otimes \mathbf{b} = (a_1\mathbf{b}, a_2\mathbf{b}, \ldots, a_n\mathbf{b}) \in \mathbb{Z}_N^{nm},$$

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{11}\mathbf{B} & \ldots & a_{1m}\mathbf{B} \\ \vdots & & \vdots \\ a_{n1}\mathbf{B} & \ldots & a_{nm}\mathbf{B} \end{pmatrix} \in \mathbb{Z}_N^{nr \times ms}.$$

### 2.2 DCR-based Inner Product Encryption

This section introduces a DCR-based inner product encryption (IPFE) cryptosystem, which employs inner product functionality to design our secure inference protocol. Whereas there are several forms of DCR-based cryptosystems, we serve a scheme, inspired by a functional encryption scheme [1], to guarantee simulation-based security. We let $B_X$ (resp. $B_{\mathcal{F}}$) denote a size bound of a message (resp. function coefficients).Furthermore, we use a function $\mathsf{DCR.pp}(1^\lambda)$ that outputs a pair $(N, p, q, g)$ such that $N = p \cdot q$, $p = 2p' + 1, q = 2q' + 1 \in \mathbb{Z}$ are primes, where $p'$ and $q'$ are

[5] If $f_3$ is a quadratic polynomial, recovering $h_2^{-1}$ from $F_3$ is difficult since it requires solving a multivariate quadratic (MQ) problem, which is computationally intractable in the average case.

also primes, and $g$ is a group generator of $\mathbb{Z}^*_{N^2}$. This parameter is set to be robust against known attacks.

For consistency with the main body, we let $\mathsf{pkIPFE}$ denote this DCR-based scheme. The detailed construction of $\mathsf{pkIPFE}$ is given by Fig. 2.

- $\mathsf{Setup}_{\mathsf{pkIPFE}}(\lambda, \ell, B_X, B_{\mathcal{F}})$:
  1. $(N, p, q, g) \leftarrow \mathsf{DCR.pp}(1^\lambda)$, such that $\ell B_X B_{\mathcal{F}} < p \cdot q/2$.
  2. For $S = 2^{\lambda+\ell-1} \cdot \left(B_X + \ell(\sqrt{\ell} B_{\mathcal{F}})^\ell\right)^{\ell-1} \cdot \ell N^2$, sample a secret vector $\mathbf{s} = (s_1, \ldots, s_\ell) \leftarrow \mathcal{U}([-S, S] \cap \mathbb{Z})$ and then compute $g_i = g^{s_i} \bmod N^2$.
  3. Return $\mathsf{pk} = (N, g, \{g_i\}_{1 \le i \le \ell}, B_X, B_{\mathcal{F}})$ and $\mathsf{msk} = (\{s_i\}_{1 \le i \le \ell})$.
- $\mathsf{KeyGen}_{\mathsf{pkIPFE}}(\mathsf{msk}, \mathbf{y})$:
  1. Return $\mathsf{fk}_{\mathbf{y}} := (\langle \mathbf{y}, \mathsf{msk} \rangle, \mathbf{y})$.
- $\mathsf{Enc}_{\mathsf{pkIPFE}}(\mathsf{pk}, \mathbf{m})$:
  1. Given $\mathbf{m} \in [-B_X, B_X]^\ell$, sample $r \leftarrow \{0, \ldots, \lfloor N/4 \rfloor\}$ and compute $C_0 = g^r \bmod N^2$,
     $C_i = (1 + m_i \cdot N) \cdot g_i^r \bmod N^2$ for $1 \le i \le \ell$.
  2. Return $\mathsf{ct}_{\mathbf{m}} = (C_0, \ldots C_\ell)$.
- $\mathsf{Dec}_{\mathsf{pkIPFE}}(\mathsf{fk}_{\mathbf{y}}, \mathsf{ct}_{\mathbf{m}})$:
  1. Given $\mathsf{ct}_{\mathbf{m}} = (C_0, \ldots, C_\ell)$, output
$$\frac{((\prod_{i=1}^{\ell} C_i^{y_i})/C_0^{\langle \mathbf{y}, \mathsf{msk} \rangle} \bmod N^2) - 1}{N}.$$

**Fig. 2.** DCR-based pkIPFE [1].

**Simulator.** For the proof of Theorem 1 and Theorem 3, a simulator of the DCR-based scheme is required. The simulator consists of the following algorithms:

$$\mathsf{Setup}^{\mathcal{S}}, \mathsf{KeyGen}_0^{\mathcal{S}}, \mathsf{Enc}^{\mathcal{S}}, \mathsf{KeyGen}_1^{\mathcal{S}}, \mathsf{Enc}, \mathsf{Dec}.$$

$\mathsf{KeyGen}_0^{\mathcal{S}}$ is only used before the challenge query. $\mathsf{KeyGen}_1^{\mathcal{S}}$ is used in the post-challenge key queries. The message in challenge phase is denoted by $\mathbf{x}^*$. The detailed construction of simulators is given by Fig. 3. The $\mathsf{Enc}$ and $\mathsf{Dec}$ algorithm exactly coincide with the $\mathsf{Enc}_{\mathsf{pkIPFE}}$ and $\mathsf{Dec}_{\mathsf{pkIPFE}}$. Thus we do not provide an algorithm description.

**Theorem 1 ( [1]).** *The scheme holds semi-adaptive security under the DCR assumption. In particular, it holds* $\mathsf{Adv}_{\mathsf{pkIPFE}} \le \mathsf{Adv}_{\mathsf{DCR}}$.

*Remark 1.* The DCR-based IPFE scheme in [1] satisfies adaptive security. However, in this paper, we consider only the weaker notion of semi-adaptive security.

*Remark 2.* For a matrix $\mathbf{Y}$, we define its functional key by $(\mathbf{Y}, \mathbf{Y} \cdot \mathsf{msk})$. Then one can securely compute a matrix multiplication $\mathbf{Y} \cdot \mathbf{x}$ as well. By an abuse of notation, we will denote it as $\mathsf{pkIPFE.KeyGen}(\mathsf{msk}, \mathbf{Y})$.

## 3 Building Blocks for Fully Encrypted PPML Protocol: FE Scheme with Composition

In this section, we describe the building blocks for the fully encrypted PPML protocol with zero client interaction.

- $\mathsf{Setup}^{\mathcal{S}}(\lambda, \ell, B_X, B_{\mathcal{F}})$: This step is identical to $\mathsf{Setup}$ except that primes $p, q$ are included in the $\mathsf{msk}$. That is, this algorithm returns $\mathsf{pk}^{\mathcal{S}} = (N, g, \{g_i\}_{1 \le i \le \ell}, B_X, B_{\mathcal{F}})$ and $\mathsf{msk}^{\mathcal{S}} = (\{s_i\}_{1 \le i \le \ell}, p, q)$.
- $\mathsf{KeyGen}_0^{\mathcal{S}}(\mathsf{msk}^{\mathcal{S}}, \mathbf{y})$: For $\mathbf{y} \in [-B_{\mathcal{F}}, B_{\mathcal{F}}]^{\ell}$, it returns $\mathsf{fk}_{\mathbf{y}} = (\langle \mathbf{y}, \mathsf{msk}^{\mathcal{S}} \rangle, \mathbf{y})$.
- $\mathsf{Enc}^{\mathcal{S}}(\mathsf{pk}^{\mathcal{S}}, \{(\mathbf{y}_i, z_i)\}_{i=1}^{k})$: For pre-challenge queries $(\mathbf{y}_i, z_i)$ with $z_i = \langle \mathbf{x}^*, \mathbf{y}_i \rangle$, the algorithm computes $\mathsf{ct}^* = (c_0^*, \{c_i^*\}_{i=1}^{\ell}) \in (\mathbb{Z}_{N^2}^*)^{\ell+1}$ computed as follows:
  1. Compute $\overline{\mathbf{x}} \in \mathbb{Z}^{\ell}$ such that $\langle \overline{\mathbf{x}}, \mathbf{y}_i \rangle = z_i$ for all $i \in [k]$.
  2. Sample $a \leftarrow \mathbb{Z}_N$ and $b \leftarrow \mathbb{Z}_{\phi(N)}$ with $\phi(N) = p' \cdot q'$ and computes $c_0^* = (1 + aN) \cdot g^b \bmod N^2$,
     $c_i^* = (1 + \overline{x}_i N) \cdot (c_0^*)^{s_i} \bmod N^2$.
  3. Return $\mathsf{ct}^*$ along with a state $\mathsf{st} = (\overline{\mathbf{x}}, a, \phi(N))$.
- $\mathsf{KeyGen}_1^{\mathcal{S}}(\mathsf{msk}^{\mathcal{S}}, \mathbf{y}, z = \langle \mathbf{y}, \mathbf{x}^* \rangle, \mathsf{st})$ :
  1. Compute $\alpha = (a^{-1} \bmod N) \cdot v\phi(N) \bmod N\phi(N)$, where $v = \phi(N)^{-1} \bmod N$.
  2. Return $\mathsf{fk}'_{\mathbf{y}} = \big(\langle \mathsf{msk}^{\mathcal{S}}, \mathbf{y} \rangle - \alpha \cdot (z - \langle \overline{\mathbf{x}}, \mathbf{y} \rangle), \mathbf{y}\big)$.

**Fig. 3.** Simulator of pkIPFE [1].

### 3.1 Technical Overview

Our primary technical insight involves incorporating the encryption algorithm into the function itself and considering a key generation algorithm on the composition of the encryption and the function. This approach enables us to obtain a functional key $\mathsf{fk}_{\mathsf{Enc} \circ f}$ which corresponds to the key generation algorithm applied to $\mathsf{Enc} \circ f$. Using the decryption algorithm on $(\mathsf{fk}_{\mathsf{Enc} \circ f}, \mathsf{ct}_{\mathbf{m}})$, we can produce an evaluated value in the form of an encrypted ciphertext $\mathsf{ct}_{f(\mathbf{m})}$. Since the output is still encrypted, we can proceed with the decryption iteratively. Specifically, we compute the decryption algorithm as

$$\mathsf{Dec}(\mathsf{fk}_{\mathsf{Enc} \circ f_{i+1}}, \mathsf{ct}_{\bigcirc_{t=1}^{i} f_t(\mathbf{m})})$$

for $i = 1, \ldots, \mathcal{E}$, where $\mathcal{E}$ is the number of functions to be computed. This framework is illustrated in Fig. 1.(b).

One major challenge in computing the functional key $\mathsf{fk}_{\mathsf{Enc} \circ f}$ is that existing functional encryption schemes for arbitrary circuits are truly infeasible. To address this limitation, we restrict functionalities to quadratic polynomials, and instantiate the scheme by modifying the Musciagna FE scheme [29] so that the $\mathsf{Enc}$ can be represented by a linear function.

To provide an intuition, we briefly describe the FE scheme in terms of an IPFE. Let $F : \mathbb{Z}^{\ell} \to \mathbb{Z}^m$ be a linear multivariate function and $\mathbf{M}_F \in \mathbb{Z}^{m \times \ell}$ be its matrix representation satisfying $F(\mathbf{x}) = \mathbf{M}_F \cdot \mathbf{x}$. Consider a ciphertext as an encryption of a message $\mathbf{m}$ from a message space $\mathcal{M} = \mathbb{Z}^{\ell}$ using inner product encryption. The functional key is represented by $\mathsf{fk}_F = \mathbf{M}_F \cdot \mathbf{D}_0$, and the ciphertext is of the form $\mathsf{ct}_{\mathbf{m}} = [{\mathbf{D}_0}^{-1} \cdot \mathbf{m}]_g$, where $g$ is a public group element and $\mathbf{D}_0$ is an invertible matrix of size $\ell \times \ell$ computed with the master secret key. Then, the decryption algorithm for this scheme can be computed as

$$\log_g(\mathsf{ct}_{\mathbf{m}}^{\mathsf{fk}_F}) = \log_g([\mathbf{M}_F \cdot \mathbf{D}_0 \cdot \mathbf{D}_0^{-1} \cdot \mathbf{m}]_g) = F(\mathbf{m}),$$

where $\log_g$ is a discrete logarithm over a group with base $g$.

We now introduce a concept to develop a desired functional encryption algorithm. To be precise, we consider a functional key for a composition function $\mathsf{Enc} \circ F$ using another secret invertible matrix $\mathbf{D}_1$: $\mathsf{fk}_{\mathsf{Enc} \circ F} = [\mathbf{D}_1^{-1} \cdot \mathbf{M}_F \cdot \mathbf{D}_0]_g$. It is clear that the decryption algorithm on a pair $(\mathsf{fk}_{\mathsf{Enc} \circ F}, \mathsf{ct}_{\mathbf{m}} = \mathbf{D}_0^{-1} \cdot \mathbf{m})$ outputs a value $\log_g((\mathsf{fk}_{\mathsf{Enc} \circ F})^{\mathsf{ct}_{\mathbf{m}}}) = \mathbf{D}_1^{-1} \cdot \mathbf{M}_F \cdot \mathbf{m} = \mathsf{ct}_{F(\mathbf{m})}$. Hence, the decryption output is a new ciphertext representing the evaluation of $F$ and the original message $\mathbf{m}$ using another secret.

Another challenge is efficiently solving the discrete log problem ($\log_g(\cdot)$) in the decryption algorithm. While an elliptic curve-based FE scheme is efficient, it should have a relatively small message space since the decryption algorithm requires to solve the discrete log problem. This small message space would imply the security issue that the size of every intermediate ciphertext obtained by decryption algorithm of $(\mathsf{fk}_{\mathsf{Enc}\circ F}, \mathsf{ct}_{\mathbf{m}})$ is small. Thus, we need a large message space with an efficient decryption process. To address this, we use a decision composite residuosity (DCR)-based FE scheme that enables efficient solving of the discrete log problem for a relatively large message set. Therefore, we had to consider a group-based FE in which DCR groups could be used, so we modified the DCR-based Musciagna scheme. The detailed discussion of the security of the modified FE scheme and original scheme description is given in Appendix E.

### 3.2 Ciphertext-bounded QFE

We present a private-key *ciphertext-bounded* quadratic functional encryption (QFE) scheme, for short LinEnc-QFE, of which The encryption algorithm is implemented as a linear function of the message. We note that it is sufficient to generate the LinEnc-QFE because any polynomial can be represented as a composite of quadratic polynomials. This scheme can be employed to instantiate a fully encrypted PPML protocol described in Section 4.3.

We first clarify a set of quadratic functionalities $\mathcal{F}$ of our LinEnc-QFE. The functionality $\mathcal{F}$ corresponds to a set of functions of the form $((\mathbf{x}\|1)\otimes(\mathbf{x}\|1))^T \cdot \mathbf{c}_f$ for some $\mathbf{x} \in \mathbb{Z}^\ell$, and $\mathbf{c}_f \in \mathbb{Z}^{(\ell+1)^2}$ such that $\|\mathbf{x}\|_\infty \leq B_X$ and $\|\mathbf{c}_f\|_\infty \leq B_{\mathcal{F}}$. Here, $\mathbf{c}_f$ is a coefficient vector corresponding to a quadratic function $f$.

**Notation.** We introduce notations to describe the LinEnc-QFE. The underlying group size is $N \cdot \phi(N)$ while the message space is $\mathbb{Z}_N$. Throughout this paper, we denote $N \cdot \phi(N)$ by $\Delta$ for simplicity and let $Q$ be the number of ciphertext queries allowed. Since our algorithm is based on the DCR scheme, we employ the $\mathsf{DCR.pp}$ function described in the Section 2.2

As a building block of our scheme, given a wide matrix $\mathbf{D} \in \mathbb{Z}_\Delta^{m\times n}$ $(m < n)$, we define sets of (the right) kernel and inverse as follows.

- $\mathsf{Ker}(\mathbf{D}) = \{\mathbf{D}^\perp \in \mathbb{Z}_\Delta^{n\times(n-m)} : \mathbf{D}\cdot\mathbf{D}^\perp = \mathbf{O} \bmod \Delta\}$
- $\mathsf{Rinv}(\mathbf{D}) = \{\mathbf{D}^{-1} \in \mathbb{Z}_\Delta^{n\times m} : \mathbf{D}\cdot\mathbf{D}^{-1} = \mathbf{I}_m \bmod \Delta\}$

**LinEnc-IPFE.** To construct the LinEnc-QFE, we first describe an inner product encryption scheme (LinEnc-IPFE) as an ingredient The goal of LinEnc-IPFE is to compute the inner product $[\mathbf{y}^T \cdot \mathbf{x}]_g$ for two inputs; a message $\mathbf{x} \in \mathbb{Z}_N^\ell$ and a function $\mathbf{y} \in \mathbb{Z}_N^\ell$ by running the decryption algorithm. The detailed construction of LinEnc-IPFE is given by Fig. 4.

*Correctness.* The correctness of LinEnc-IPFE in Fig. 4 will be given by Appendix C.1.

**LinEnc-QFE.** We now introduce the LinEnc-QFE, built from LinEnc-IPFE. The detailed construction of LinEnc-QFE is then given by Appendix 5.

*Correctness.* The correctness of LinEnc-QFE in Fig. 5 will be given by Appendix C.2.

*Remark 3.* The operations $\mathsf{KeyGen}_{\mathsf{IPFE,QFE}}$ (resp. $\mathsf{Enc}_{\mathsf{IPFE,QFE}}$) were performed on a matrix $\mathbf{M}$ in a column-by-column manner. To illustrate, the function $\mathsf{KeyGen}_{\mathsf{IPFE}}(\mathsf{msk}, \mathsf{pp}, \mathbf{M})$ is defined by the expression $(\mathsf{KeyGen}_{\mathsf{IPFE}}(\mathsf{msk}, \mathsf{pp}, \mathbf{M}[i]))_{i\in[1,\mathsf{col}]}$, where the notation $(\mathbf{M}[i])_{i\in[1,\mathsf{col}]}$ represents the set of all column vectors of length $\mathsf{col}$ comprising the matrix $\mathbf{M}$.

### 3.3 Security proof of LinEnc FE

This section demonstrates that the LinEnc-QFE satisfies the semi-adaptive security when the number of ciphertexts is $Q$-bounded.

- $\mathsf{Setup}_{\mathsf{IPFE}}(\lambda, \ell, Q)$:
  1. Choose $(N, p, q, g) \leftarrow \mathsf{DCR.pp}(\lambda)$ and define $\Delta = N \cdot \phi(N)$.
  2. Sample $\mathbf{a} \leftarrow \mathbb{Z}_{\Delta}^{2}$, $\mathbf{U} \leftarrow \mathbb{Z}_{\Delta}^{\ell \times 2}$, and $\mathbf{D} \leftarrow \mathbb{Z}_{\Delta}^{(\ell+2)\times(\ell+2+Q)}$.
  3. Sample $\mathbf{D}^{\perp} \leftarrow \mathsf{Ker}(\mathbf{D})$ and $\mathbf{D}^{-1} \leftarrow \mathsf{Rinv}(\mathbf{D})$
  4. Return
$$\mathsf{msk} = \{\mathbf{a}, \mathbf{U}, \mathbf{D}, \mathbf{D}^{\perp}, \mathbf{D}^{-1}\} \text{ and } \mathsf{pp} = \{g, N, \Delta\}.$$
- $\mathsf{KeyGen}_{\mathsf{IPFE}}(\mathsf{msk}, \mathsf{pp}, \mathbf{y} \in \mathbb{Z}_{N}^{\ell})$:
  1. Return
$$\mathsf{fk}_{\mathbf{y}} = \left[ \begin{pmatrix} -\mathbf{U}^{T} \cdot \mathbf{y} \\ \mathbf{y} \end{pmatrix}^{T} \cdot \mathbf{D} \right]_{g}.$$
- $\mathsf{Enc}_{\mathsf{IPFE}}(\mathsf{msk}, \mathsf{pp}, \mathbf{x} \in \mathbb{Z}_{N}^{\ell})$:
  1. Sample $r \leftarrow \mathbb{Z}_{\Delta}$ and $\widetilde{\mathbf{r}} \leftarrow \mathbb{Z}_{\Delta}^{Q}$.
  2. Return
$$\mathsf{ct}_{\mathbf{x}} = \mathbf{D}^{\perp} \cdot \widetilde{\mathbf{r}} + \mathbf{D}^{-1} \cdot \begin{pmatrix} r \cdot \mathbf{a} \\ \mathbf{x} + r \cdot \mathbf{U} \cdot \mathbf{a} \end{pmatrix} \in \mathbb{Z}_{\Delta}^{\ell+2+Q}.$$
- $\mathsf{Dec}_{\mathsf{IPFE}}(\mathsf{pp}, \mathsf{fk}_{\mathbf{y}}, \mathsf{ct}_{\mathbf{x}})$:
  1. Return $\mathsf{fk}_{\mathbf{y}}^{\mathsf{ct}_{\mathbf{x}}}$.

**Fig. 4.** LinEnc-IPFE.

**Theorem 2.** *The LinEnc-QFE described in Fig. 5 is semi-adaptively secure under the MDDH and bilateral 2-LIN assumptions for $Q$-bounded ciphertexts. In particular, it holds that*

$$\mathsf{Adv}_{\mathsf{LinEnc\text{-}QFE}} \leq \mathsf{Adv}_{\mathsf{MDDH}} + 2 \cdot \mathsf{Adv}_{\mathsf{2\text{-}Lin}}.$$

To prove this theorem, our strategy is to provide a polynomial time reduction from the quadratic functional encryption scheme, Mus.QFE with $Q$ ciphertexts, which is suggested by Musciagna [29] to the LinEnc-QFE scheme.

On the other hand, [29] proved that the Mus.QFE scheme achieves the semi-adaptive simulation security under MDDH and 2-Lin assumptions. When these conditions are met, the following can be concluded: $\mathsf{Adv}_{\mathsf{LinEnc\text{-}QFE}} \leq \mathsf{Adv}_{\mathsf{Mus.QFE}} \leq \mathsf{Adv}_{\mathsf{MDDH}} + 2 \cdot \mathsf{Adv}_{\mathsf{2\text{-}Lin}}$. We provide both the description and reduction of the Mus.QFE in the Appendix D. The proof of Theorem 2 will be given by Appendix E.

### 3.4 Composite evaluation via LinEnc QFE

We emphasize that for fixed randomness in the encryption of the QFE scheme, $\mathsf{Enc}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE}}, \mathsf{pp}_{\mathsf{QFE}}, \cdot)$ is a linear function for a message. This implies that, rather than requesting the vector quadratic function $F : \mathbb{Z}^{\ell} \to \mathbb{Z}^{m}$ during the key generation process, as in traditional FE, one can query the coefficient vector of the composite function $\mathsf{Enc}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE}}, \mathsf{pp}_{\mathsf{QFE}}, \cdot) \circ F$.

To enable this, we exploit the $\mathsf{Enc}_{\mathsf{QFE}}$ algorithm for a linear or quadratic function input $F$ with a matrix representation $\mathbf{M}_F$. Specifically, $F(\mathbf{x}) = \mathbf{M}_F \cdot ((\mathbf{x}\|1) \otimes (\mathbf{x}\|1))$ (or $F(\mathbf{x}) = \mathbf{M}_F \cdot (\mathbf{x}\|1)$ if $F$ is linear) and outputs a composition of encryption and the function evaluation $\mathbf{M}_{\mathsf{Enc}\circ F}$. We denote this algorithm by cEnc, representing the composition of encryption and a function.

For further composite evaluations, we incorporate two subroutines into the description of the algorithm and will now detail them. For these evaluations, it is necessary to get an encryption of $(F(\mathbf{x})\|1)$. Therefore, we define the expanded matrix representation of $F$, $\overline{\mathbf{M}}_F$, by concatenation of the unit vector $(0, \ldots, 0, 1)$ as the last row of the matrix $\mathbf{M}_F$. It is clear that $\overline{\mathbf{M}}_F \cdot ((\mathbf{x}\|1) \otimes (\mathbf{x}\|1)) = (F(\mathbf{x})\|1)$.

- $\mathsf{Setup}_{\mathsf{QFE}}(\lambda, \ell, Q)$:
  1. Sample $(\mathsf{msk}_{\mathsf{IPFE}}, \mathsf{pp}_{\mathsf{IPFE}} = \{g, N, \Delta\}) \leftarrow \mathsf{Setup}_{\mathsf{IPFE}}(\lambda, 4\ell + 4, Q)$.
  2. Let $c_0$ be a inverse of $\frac{(g^{\phi(N)} \bmod N^2)-1}{N}$ modulo $N$ and $c = c_0 \cdot \phi(N)$ so that it satisfies $g^c = 1 + N \bmod N^2$.
  3. Sample $\mathbf{V} \leftarrow \mathbb{Z}_{\Delta}^{(\ell+1)\times 2}$, $\mathbf{W} \leftarrow \mathbb{Z}_{\Delta}^{(\ell+1)\times 2}$.
  4. Sample $\mathbf{D}_b \leftarrow \mathbb{Z}_{\Delta}^{(\ell+1)\times(\ell+1+Q)}$ for $b \in \{0,1\}$.
  5. Sample $\mathbf{D}_b^{\perp} \leftarrow \mathsf{Ker}(\mathbf{D}_b)$ and $\mathbf{D}_b^{-1} \leftarrow \mathsf{Rinv}(\mathbf{D}_b)$ for $b \in \{0,1\}$.
  6. Return $(\mathsf{pp}, \mathsf{msk})$ where $\mathsf{pp} = \mathsf{pp}_{\mathsf{IPFE}}$ and $\mathsf{msk} = \left\{\mathbf{V}, \mathbf{W}, \{\mathbf{D}_b, \mathbf{D}_b^{\perp}, \mathbf{D}_b^{-1}\}_{b\in\{0,1\}}, c, \mathsf{msk}_{\mathsf{IPFE}}\right\}$.
- $\mathsf{KeyGen}_{\mathsf{QFE}}(\mathsf{msk}, \mathsf{pp}, \mathbf{c}_f \in \mathbb{Z}^{(\ell+1)^2})$:
  1. Sample $\mathsf{fk}_{\mathsf{IPFE}} \leftarrow \mathsf{KeyGen}_{\mathsf{IPFE}}(\mathsf{msk}_{\mathsf{IPFE}}, \mathsf{pp}_{\mathsf{IPFE}}, \mathsf{fk}_0 \cdot \mathbf{c}_f)$ where $\mathsf{fk}_0 = \begin{pmatrix} \mathbf{V}^T \otimes \mathbf{I}_{\ell+1} \\ \mathbf{I}_{\ell+1} \otimes \mathbf{W}^T \end{pmatrix} \in \mathbb{Z}_{\Delta}^{4(\ell+1)\times(\ell+1)^2}$.
  2. Compute $\mathsf{fk}_1 = \left[(\mathbf{D}_0 \otimes \mathbf{D}_1)^T \cdot \mathbf{c}_f\right]_g \in \mathbb{G}^{(\ell+1+Q)^2}$
  3. Return $\mathsf{fk}_f = \{\mathsf{fk}_{\mathsf{IPFE}}, \mathsf{fk}_1\}$.
- $\mathsf{Enc}_{\mathsf{QFE}}(\mathsf{msk}, \mathsf{pp}, \mathbf{x} \in \mathbb{Z}^{\ell})$:
  1. Set $\overline{\mathbf{x}} = (\mathbf{x} \| 1)$.
  2. Sample $\mathbf{r}_b \leftarrow \mathbb{Z}_{\Delta}^2$ for $b \in \{0,1\}$.
  3. Compute $\mathsf{ct}_b$ for $b \in \{0,1\}$ as follows.
$$\mathsf{ct}_0 = \mathbf{D}_0^{\perp} \cdot \widetilde{\mathbf{r}}_0 + \mathbf{D}_0^{-1} \cdot (c \cdot \overline{\mathbf{x}} + \mathbf{V} \cdot \mathbf{r}_0) \in \mathbb{Z}_{\Delta}^{\ell+1+Q},$$
$$\mathsf{ct}_1 = \mathbf{D}_1^{\perp} \cdot \widetilde{\mathbf{r}}_1 + \mathbf{D}_1^{-1} \cdot (\overline{\mathbf{x}} + \mathbf{W} \cdot \mathbf{r}_1) \in \mathbb{Z}_{\Delta}^{\ell+1+Q}.$$
  4. Sample $\mathsf{ct}_{\mathsf{IPFE}} \leftarrow \mathsf{Enc}_{\mathsf{IPFE}}(\mathsf{msk}_{\mathsf{IPFE}}, \mathbf{h})$ for $\mathbf{h} = \left((\mathbf{r}_0 \otimes \overline{\mathbf{x}}) \,\|\, ((c \cdot \overline{\mathbf{x}} + \mathbf{V} \cdot \mathbf{r}_0) \otimes \mathbf{r}_1)\right) \in \mathbb{Z}_{\Delta}^{4(\ell+1)}$.
  5. Return $\mathsf{ct}$ defined as follows: $\mathsf{ct} = \left\{\{\mathsf{ct}_b\}_{b\in\{0,1\}}, \mathsf{ct}_{\mathsf{IPFE}}\right\}$
- $\mathsf{Dec}_{\mathsf{QFE}}(\mathsf{pp}, \mathsf{fk}_f, \mathsf{ct})$:
  1. Compute $\mathbf{e}_1$ and $\mathbf{e}_2$ as follows.
$$\mathbf{e}_1 = \mathsf{fk}_1^{\mathsf{ct}_0 \otimes \mathsf{ct}_1},$$
$$\mathbf{e}_2 = \left[\mathsf{Dec}_{\mathsf{IPFE}}(\mathsf{pp}_{\mathsf{IPFE}}, \mathsf{fk}_{\mathsf{IPFE}}, \mathsf{ct}_{\mathsf{IPFE}})\right]_g.$$
  2. Return $\log_{(1+N)}(\mathbf{e}_1/\mathbf{e}_2)$.

**Fig. 5.** LinEnc-QFE.

We then implement a functional key of $\mathsf{Enc}_{\mathsf{QFE}} \circ F$ and its decryption process by using the expanded matrix representation. The detailed algorithms are given by Fig. 7. In the description, we denote $\overline{\mathbf{M}}_F[i]$ as the $i$-th column vector of $\overline{\mathbf{M}}_F$.

In order to compute composite evaluations, it is necessary to impose size restrictions on the ciphertexts of the LinEnc-QFE since the correctness of LinEnc-QFE only works when an evaluated value is less than $N/2$. For this purpose, given $L \in \mathbb{Z}$, we set $Y = \lfloor \Delta^{1/L} \rceil$. In other words, it holds that $Y^{L-1} < \Delta < Y^L$. To restrict the size of ciphertexts, we then define two functions:

$$\begin{aligned} \mathsf{Decomp}_L : \mathbb{Z}_{\Delta} &\mapsto \mathbb{Z}^L \\ v &\to (v_0, v_1, \ldots, v_{L-1}) \\ \mathsf{Power}_L : \mathbb{Z}^L &\mapsto \mathbb{Z} \\ (w_0, w_1, \ldots, w_{L-1}) &\to \sum_{i=0}^{L-1} w_i \cdot Y^i, \end{aligned}$$

where the $\{v_i\}$ holds that $\sum_{i=0}^{L-1} v_i \cdot Y^i = v$. By the definition of both functions, it is clear that

$$\mathsf{Power}_L(\mathsf{Decomp}_L(v)) = v.$$

In the case where the input is a matrix, we apply the $\mathsf{Decomp}_Y$ function to each entry. The power map is then properly defined as an inverse map of the decomposition function on matrices. We now show the details of the algorithm in Fig. 6.

- $\mathsf{cEnc}(\mathsf{msk}_{\mathsf{QFE}}, \mathsf{pp}_{\mathsf{QFE}}, \mathbf{M}_F, L)$
  1. Sample $\mathbf{r}_b \leftarrow \mathbb{Z}_\Delta^2$ for $b \in \{0,1\}$.
  2. Let $m$ and $\mathsf{col}$ be the number of rows and columns of $\mathbf{M}_F$, respectively, and define $\overline{\mathbf{M}}_F = \begin{pmatrix} \mathbf{M}_F \\ \mathbf{e}_{\mathsf{col}} \end{pmatrix}$, where $\mathbf{e}_{\mathsf{col}}$ is a unit vector $(0,\ldots,0,1)$ of length $\mathsf{col}$.
  3. For $1 \le i < \mathsf{col}$, sample $\widetilde{\mathbf{r}}_{b,i} \leftarrow \mathbb{Z}_\Delta^Q$ for $b \in \{0,1\}$ and compute the following:
$$\mathsf{ct}_{0,i} = \mathbf{D}_0^{\perp} \cdot \widetilde{\mathbf{r}}_{0,i} + \mathbf{D}_0^{-1} \cdot c \cdot \overline{\mathbf{M}}_F[i] \in \mathbb{Z}_\Delta^{m+1+Q}$$
$$\mathsf{ct}_{1,i} = \mathbf{D}_1^{\perp} \cdot \widetilde{\mathbf{r}}_{1,i} + \mathbf{D}_1^{-1} \cdot \overline{\mathbf{M}}_F[i] \in \mathbb{Z}_\Delta^{m+1+Q}$$
$$\mathsf{ct}_{\mathsf{IPFE},i} \leftarrow \mathsf{Enc}_{\mathsf{IPFE}}(\mathsf{msk}_{\mathsf{IPFE}}, \mathsf{pp}_{\mathsf{IPFE}}, \mathbf{h}_i),$$
     where $\mathbf{h}_i = \big((\mathbf{r}_0 \otimes \overline{\mathbf{M}}_F[i]) \| (c \cdot \overline{\mathbf{M}}_F[i] \otimes \mathbf{r}_1)\big) \in \mathbb{Z}_\Delta^{4(m+1)}$.
  4. Sample $\widetilde{\mathbf{r}}_{b,\mathsf{col}} \leftarrow \mathbb{Z}_\Delta^Q$ for $b \in \{0,1\}$ and compute the following:
$$\mathsf{ct}_{0,\mathsf{col}} = \mathbf{D}_0^{\perp} \cdot \widetilde{\mathbf{r}}_{0,\mathsf{col}} + \mathbf{D}_0^{-1} \cdot (c \cdot \overline{\mathbf{M}}_F[\mathsf{col}] + \mathbf{V} \cdot \mathbf{r}_0) \bmod \Delta$$
$$\mathsf{ct}_{1,\mathsf{col}} = \mathbf{D}_1^{\perp} \cdot \widetilde{\mathbf{r}}_{1,\mathsf{col}} + \mathbf{D}_1^{-1} \cdot (\overline{\mathbf{M}}_F[\mathsf{col}] + \mathbf{W} \cdot \mathbf{r}_1) \bmod \Delta$$
$$\mathsf{ct}_{\mathsf{IPFE},\mathsf{col}} \leftarrow \mathsf{Enc}_{\mathsf{IPFE}}(\mathsf{msk}_{\mathsf{IPFE}}, \mathbf{h}_{\mathsf{col}}),$$
     where $\mathbf{h}_{\mathsf{col}} = \big(\mathbf{r}_0 \otimes \overline{\mathbf{M}}_F[\mathsf{col}] \| (c \cdot \overline{\mathbf{M}}_F[\mathsf{col}] + \mathbf{V} \cdot \mathbf{r}_0) \otimes \mathbf{r}_1\big) \in \mathbb{Z}_\Delta^{4(m+1)}$.
  5. Set a matrix $\mathbf{M}_{\mathsf{Enc}\circ F}$ of which $i$-th column vector is the concatenation of three ciphertexts, i.e. for $1 \le i \le \mathsf{col}$, define
$$\mathbf{M}_{\mathsf{Enc}\circ F}[i] \leftarrow (\mathsf{ct}_{0,i} \| \mathsf{ct}_{1,i} \| \mathsf{ct}_{\mathsf{IPFE},i}) \in \mathbb{Z}_\Delta^{6m+3Q+8}.$$
  6. Decompose $\mathbf{M}_{\mathsf{Enc}\circ F}$ over columns to get $\mathbf{M}_{\mathsf{Enc}\circ F}^{\mathsf{decomp}} \in \mathbb{Z}_\Delta^{L(6m+3Q+8)\times \mathsf{col}}$
  7. Return $\mathbf{M}_{\mathsf{Enc}\circ F}^{\mathsf{decomp}}$.

**Fig. 6.** Composition algorithm of $\mathsf{Enc}_{\mathsf{QFE}}$ and a matrix $\mathbf{M}_F$, a matrix representation of a function $F$. Recall that $\overline{\mathbf{M}}_F[i]$ is $i$-th column of the matrix $\overline{\mathbf{M}}_F$.

*Correctness* (of Fig. 6 and Fig. 7). Due to the decryption correctness of LinEnc-QFE, $\mathsf{cDec}_{\mathsf{QFE}}$ returns a vector of the form $\mathsf{Power}_L(\mathbf{t})$ for $\mathbf{t} = \left(\mathbf{M}_{\mathsf{Enc}\circ F}^{\mathsf{decomp}} \cdot \mathbf{y}\right)$, where $\mathbf{y}$ is $(\mathbf{x}\|1) \otimes (\mathbf{x}\|1)$. Let $B_X$ be a bound of $\mathbf{x}$. Since each entry of $\mathbf{M}_{\mathsf{Enc}\circ F}^{\mathsf{decomp}}$ is less than $Y = \lfloor \Delta^{1/L} \rceil$, each entry of the product $\mathbf{M}_{\mathsf{Enc}\circ F}^{\mathsf{decomp}} \cdot ((\mathbf{x}\|1) \otimes (\mathbf{x}\|1))$ is less than $Y \cdot B_X \cdot (\ell+1)^2$. Thus, if we choose large enough $L$ to satisfy $Y \cdot B_X \cdot (\ell+1)^2 < N/2$, then $\mathsf{cDec}_{\mathsf{QFE}}$ in Fig. 7 will output an exact decryption value $\mathbf{t}_{\mathsf{idx}}$. Similarly, $\mathsf{cDec}_{\mathsf{pkIPFE}}$ yields an exact decryption value $\mathbf{t}_{\mathsf{idx}}$ as long as $Y \cdot B_X \cdot (\ell+1) < N/2$.

Hence, from the linear property, $\mathsf{Decomp}_L(\mathbf{t})$ is represented by a vector $(\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_{\mathsf{IPFE}})$ of the form

$$\mathsf{ct}_0 = \mathbf{D}_0^{\perp} \cdot \widetilde{\mathbf{r}}_0 + \mathbf{D}_0^{-1} \cdot (c \cdot (F(\mathbf{x})\|1) + \mathbf{V} \cdot \mathbf{r}_0) \bmod \Delta$$
$$\mathsf{ct}_1 = \mathbf{D}_1^{\perp} \cdot \widetilde{\mathbf{r}}_1 + \mathbf{D}_1^{-1} \cdot ((F(\mathbf{x})\|1) + \mathbf{W} \cdot \mathbf{r}_1) \bmod \Delta$$
$$\mathsf{ct}_{\mathsf{IPFE}} = \mathsf{Enc}_{\mathsf{IPFE}}(\mathsf{msk}_{\mathsf{IPFE}}, \mathbf{h})$$

with $\mathbf{h} = \big(\mathbf{r}_0 \otimes F(\mathbf{x}) \| (c \cdot F(\mathbf{x}) + \mathbf{V} \cdot \mathbf{r}_0) \otimes \mathbf{r}_1\big)$ for some $\widetilde{\mathbf{r}}_0$ and $\widetilde{\mathbf{r}}_1$. Thus, the output of both $\mathsf{cDec}$ can be regarded as a ciphertext of message $F(\mathbf{x})$.

*Remark 4.* The randomness $(\mathbf{r}_0, \mathbf{r}_1)$ in $\mathsf{cDec}_{\mathsf{QFE}}(\mathsf{pp}_{\mathsf{QFE}}, \mathsf{fk}_{\mathsf{Enc}\circ F}, \mathsf{ct}, L)$ is shared with that of $\mathsf{fk}_{\mathsf{Enc}\circ F}$. Therefore, it cannot be guaranteed that the ciphertext is secure. Accordingly, when instantiating FE-PPML using the current LinEnc-QFE, an additional factor is required to ensure sufficient randomness. This additional factor will be discussed in greater detail in Section 4.

## 4 Fully Encrypted PPML protocol

In this section, we propose a fully encrypted PPML protocol using functional encryption for quadratic polynomials. We consider a machine learning model of $\mathcal{E}$ layers as a composition of

- $\mathsf{cKeyGen}_{\mathsf{pkIPFE}}(\mathsf{msk}_{\mathsf{pkIPFE}}, \mathsf{pk}_{\mathsf{pkIPFE}}, \mathsf{msk}_{\mathsf{QFE}}, \mathsf{pp}_{\mathsf{QFE}}, f : \mathbb{Z}^{\ell} \to \mathbb{Z}^{m}, L)$
  1. For given a linear function $f$, Let $\mathbf{M}_f \in \mathbb{Z}^{m \times \ell}$ be a matrix representation of $f$ such that $f(\mathbf{x}) = \mathbf{M}_f \cdot \mathbf{x}$.
  2. Compute $\mathbf{M}^{\mathsf{decomp}}_{\mathsf{Enc} \circ f} \leftarrow \mathsf{cEnc}(\mathsf{msk}_{\mathsf{QFE}}, \mathsf{pp}_{\mathsf{QFE}}, \mathbf{M}_f, L)$
  3. For every $1 \le \mathsf{idx} \le L(6m + 3Q + 8)$, compute
  $$\mathsf{fk}_{\mathsf{Enc} \circ F}[\mathsf{idx}] \leftarrow \mathsf{KeyGen}_{\mathsf{pkIPFE}}(\mathsf{msk}_{\mathsf{pkIPFE}}, \mathsf{pk}_{\mathsf{pkIPFE}}, (\mathbf{M}^{\mathsf{decomp}}_{\mathsf{Enc} \circ f})^T[\mathsf{idx}]),$$
  where $\mathbf{M}^T_{\mathsf{Enc} \circ f}[\mathsf{idx}]$ is the $\mathsf{idx}$-th column vector of $\mathbf{M}^T_{\mathsf{Enc} \circ f}$,
  4. Return a set of functional keys $\mathsf{fk}_{\mathsf{Enc} \circ f} = \{\mathsf{fk}_{\mathsf{Enc} \circ f}[\mathsf{idx}]\}^{L(6m+3Q+8)}_{\mathsf{idx}=1}$.
- $\mathsf{cDec}_{\mathsf{pkIPFE}}(\mathsf{pk}_{\mathsf{pkIPFE}}, \mathsf{fk}_{\mathsf{Enc} \circ f}, \mathsf{ct}, L)$
  1. Parse $\mathsf{fk}_{\mathsf{Enc} \circ f} = \{\mathsf{fk}_{\mathsf{Enc} \circ f}[\mathsf{idx}]\}_{\mathsf{idx}}$. Compute $\mathbf{t} \in \mathbb{Z}^{L(6m+3Q+8)}$ where $\mathsf{idx}$-th entry of $\mathbf{t}$ is
  $$\mathbf{t}_{\mathsf{idx}} \leftarrow \mathsf{Dec}_{\mathsf{pkIPFE}}(\mathsf{pk}_{\mathsf{pkIPFE}}, \mathsf{fk}_{\mathsf{Enc} \circ f}[\mathsf{idx}], \mathsf{ct}).$$
  2. Return a vector $\mathsf{Power}_L(\mathbf{t})$.
- $\mathsf{cKeyGen}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE}}, \mathsf{pp}_{\mathsf{QFE}}, \mathsf{msk}_{\mathsf{QFE}'}, \mathsf{pp}_{\mathsf{QFE}'}, F : \mathbb{Z}^{\ell} \to \mathbb{Z}^{m}, L)$
  1. For given quadratic function $F$, let $\mathbf{M}_F \in \mathbb{Z}^{m \times (\ell+1)^2}$ be a matrix representation of $F$ s.t. $F(\mathbf{x}) = \mathbf{M}_F \cdot (\overline{\mathbf{x}} \otimes \overline{\mathbf{x}})$, where $\overline{\mathbf{x}} = (\mathbf{x} \| 1)$.
  2. Compute $\mathbf{M}^{\mathsf{decomp}}_{\mathsf{Enc} \circ F} \leftarrow \mathsf{cEnc}(\mathsf{msk}_{\mathsf{QFE}'}, \mathsf{pp}_{\mathsf{QFE}'}, \mathbf{M}_F, L)$
  3. For every $1 \le \mathsf{idx} \le L(6m + 3Q + 8)$, compute
  $$\mathsf{fk}_{\mathsf{Enc} \circ F}[\mathsf{idx}] \leftarrow \mathsf{KeyGen}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE}}, \mathsf{pp}_{\mathsf{QFE}}, (\mathbf{M}^{\mathsf{decomp}}_{\mathsf{Enc} \circ F})^T[\mathsf{idx}]),$$
  where $\mathbf{M}^T_{\mathsf{Enc} \circ F}[\mathsf{idx}]$ is the $\mathsf{idx}$-th column vector of $\mathbf{M}^T_{\mathsf{Enc} \circ F}$.
  4. Return a set of functional keys $\mathsf{fk}_{\mathsf{Enc} \circ F} = \{\mathsf{fk}_{\mathsf{Enc} \circ F}[\mathsf{idx}]\}^{L(6m+3Q+8)}_{\mathsf{idx}=1}$.
- $\mathsf{cDec}_{\mathsf{QFE}}(\mathsf{pp}_{\mathsf{QFE}}, \mathsf{fk}_{\mathsf{Enc} \circ F}, \mathsf{ct}, L)$
  1. Parse $\mathsf{fk}_{\mathsf{Enc} \circ F} = \{\mathsf{fk}_{\mathsf{Enc} \circ F[\mathsf{idx}]}\}_{\mathsf{idx}}$. Compute $\mathbf{t} \in \mathbb{Z}^{L(6m+3Q+8)}$ where $\mathsf{idx}$-th entry of $\mathbf{t}$ is $\mathbf{t}_{\mathsf{idx}} \leftarrow \mathsf{Dec}_{\mathsf{QFE}}(\mathsf{pp}_{\mathsf{QFE}}, \mathsf{fk}_{\mathsf{Enc} \circ F}[\mathsf{idx}], \mathsf{ct})$.
  2. Return a vector $\mathsf{Power}_L(\mathbf{t})$.

**Fig. 7.** Algorithms for composite evaluation.

quadratic functions $\bigcirc_{i=1}^{\mathcal{E}} f_i = f_{\mathcal{E}} \circ \cdots \circ f_1$, where the function $f_i$ represents the computation of the $i$-th layer of the model. That is, the inference result $F(\mathbf{x})$ can be represented by $\bigcirc_{i=1}^{\mathcal{E}} f_i(\mathbf{x}) = (f_{\mathcal{E}} \circ \cdots \circ f_1)(\mathbf{x})$. We will demonstrate the application of our protocol through privacy-preserving quadratic neural networks, as detailed in Section 5.3. Our approach effectively addresses the pervasive issue of intermediate leakage, which has been identified as a significant challenge in existing FE-based PPML solutions.

The proposed protocol $\mathcal{P}$ in Section 4.3 is constructed using a (public-key) inner product encryption scheme, $\mathsf{pkIPFE}$ in Fig. 2, and a FE scheme for quadratic polynomials, LinEnc-QFE in Fig. 5. The final result of the protocol is the composition of quadratic polynomials $\bigcirc_{i=1}^{\mathcal{E}} f_i(\mathbf{x})$, where each $f_i : \mathbb{Z}^{\ell_i} \to \mathbb{Z}^{\ell_{i+1}}$ is a vector quadratic polynomial. Throughout this paper, we set $\ell_i = \ell_{i+1}$ for all indices for simplicity. To ensure the correctness and the security of $\mathcal{P}$, specific requirements and conditions must be met by LinEnc-QFE. Subsequent sections will provide a detailed analysis of these aspects, with a view to evaluating the reliability and security of the protocol.

### 4.1 Fully Encrypted PPML Framework and Security Definition

We consider a fully encrypted privacy-preserving machine learning protocol using functional encryption (FE-PPML) as a secure inference on encrypted data. Our FE-PPML framework is intended for scenarios in which a service provider process clients' data to perform machine learning

operations, thereby deriving meaningful outcomes and delivering valuable services to the clients based on these results (as outlined in Section 1). Accordingly, the FE-PPML framework involves the following three types of entities: a key distributor (KD), an evaluator (E), and a set of clients ($C_j$ for $j \in [0, J]$). The key distributor is a trusted third-party authority to generate a public key ($\mathsf{pk}$) and a master secret key ($\mathsf{msk}$) and providing functional keys ($\mathsf{fk}$) based on the evaluatior's query.

Specifically, we assume that an evaluator possesses machine learning models in advance. The client encrypts the data using $\mathsf{pk}$ and sends the ciphertext ($\mathsf{ct}$) to the evaluator. The evaluator performs inference on the encrypted data by obtaining a functional key $\mathsf{fk}_F$ associated with a pre-trained model $F$ from the key distributor and the ciphertext $\mathsf{ct}_{\mathbf{x}}$ associated with the input data $\mathbf{x}$, and then outputs the computation result $F(\mathbf{x})$ in plaintext.

**General framework for FE-PPML protocol** We construct a FE-PPML protocol for multiple clients by adopting functional encryption (FE) scheme for general circuits as in Fig. 8.

**Participants**: Clients ($\{C_j\}_{j\in[0,J]}$), key distributor ($KD$), and evaluator ($E$)
**Protocol**:

**Protocol Setup by** $KD$

1. $KD$ samples $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\lambda, \zeta)$ for the security parameter $\lambda$ and pre-determined parameter $\zeta$ and sends $\mathsf{pk}$ to every client.

**Encryption by** $C$

1. Let $\mathbf{x}_j$ be the input data of $C_j$ for $j \in [0, J]$. Every $C_j$ encrypts a message to generate $\mathsf{ct}_{\mathbf{x}_j} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{x}_j)$ and sends the ciphertext to $E$.

**Functional Key Generation between** $KD$ **and** $E$

1. Let $\{F_l\}_{l\in[0,\zeta]}$ be a set of (pre-trained) machine learning models of the form $F_l = f_{l,\mathcal{E}} \circ \cdots \circ f_{l,1}$ for any function $f_{l,i}$. $E$ sends the family of models $\{F_l\}_{l\in[0,\zeta]}$ to $KD$.
2. $KD$ computes a set of functional keys $\{\mathsf{fk}_{\mathsf{Enc}\circ f_{l,i}}\}_{l\in[0,\zeta],i\in[1,\mathcal{E}-1]}$ and $\{\mathsf{fk}_{f_{l,\mathcal{E}}}\}_{l\in[0,\zeta]}$ defined as below. Then, $KD$ sends them to $E$.

$$\mathsf{fk}_{\mathsf{Enc}\circ f_{l,i}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{Enc} \circ f_{l,i})$$
$$\mathsf{fk}_{f_{l,\mathcal{E}}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_{l,\mathcal{E}})$$

**Model Evaluation by** $E$

1. For each $j \in [0, J]$, $E$ computes $\mathsf{Enc}(f_{l,1}(\mathbf{x}_j)) \leftarrow \mathsf{Dec}(\mathsf{pk}, \mathsf{fk}_{\mathsf{Enc}\circ f_{l,1}}, \mathsf{ct}_{\mathbf{x}_j})$ for every $l \in [0, \zeta]$. Sequentially, $E$ computes $\mathsf{Enc}(\bigcirc_{i=1}^{\mathcal{E}-1} f_{l,i}(\mathbf{x}_j))$ for each $j \in [0, J]$.
2. Using $\mathsf{fk}_{f_{l,\mathcal{E}}}$ for every $l \in [0, \zeta]$, $E$ obtains $F_l(\mathbf{x}_j)$ for each $j \in [0, J]$.

$$F_l(\mathbf{x}_j) \leftarrow \mathsf{Dec}(\mathsf{pk}, \mathsf{fk}_{f_{l,\mathcal{E}}}, \mathsf{ct}_{\bigcirc_{i=1}^{\mathcal{E}-1} f_{l,i}(\mathbf{x}_j)})$$

**Fig. 8.** General framework for FE-PPML.

The problem is that previously known FE schemes for general circuits are infeasible, and the same is true for LinEnc FE. Therefore, we aim to demonstrate how to construct a secure FE-PPML protocol using a LinEnc FE scheme for quadratic functions (LinEnc-QFE), which is truly feasible for real-world implementations. In brief, we first convert a pre-trained model $F_l$ for $l \in [0, \zeta]$ into compositions of quadratic functions $F_l = \bigcirc_{i=1}^{\mathcal{E}} f_{l,i}$, and then apply LinEnc-QFE iteratively. In Section 3, we construct a (ciphertext-bounded) LinEnc-QFE scheme. In LinEnc-QFE, an encryption algorithm (denoted by $\mathsf{Enc}$) itself can be represented as a linear function. This allows us to generate a functional key for the composition of encryption algorithm and arbitrary quadratic function

$f$ (i.e., $\mathsf{Enc} \circ f$). Note that existing FE schemes cannot support this property, and therefore the composition of $\mathsf{Enc}$ and a function $f$ is not possible. A full description of FE-PPML protocol is given in Fig. 9.

**Security of FE-PPML** We revisit the security definition for PPML protocol in the malicious model by adapting the security by Lindell [23]. In the FE-PPML, the following parties are involved in the protocol.

- the key distributor, denoted by $KD$.
- the honest client $C_0$ and other (malicious) clients $\{C_j\}_{j\in[1,J]}$ that have a message $\mathbf{x}_j$ for $j \in [0, J]$.
- the adversary $\mathcal{A}$.
- the evaluator $E$ which has several machine learning models $\{F_l\}$ where $\{F_l\}_{l\in[0,\zeta]}$ is a set of machine learning models of the form

$$F_l = \bigcirc_{i=1}^{\mathcal{E}} f_{l,i} = f_{l,\mathcal{E}} \circ \ldots \circ f_{l,1}$$

  for some functions $f_{l,i}$.

We note that there exist a bunch of intermediate values, which can be used to get information associated with $\mathbf{x}_0$, in the PPML protocol such as $\mathsf{Enc}((f_{l,i} \circ f_{l,i-1} \circ \ldots \circ f_{l,1})(\mathbf{x}_j))$ for any indices $l, i, j$. We say that FE-PPML is secure when $\mathcal{A}$ cannot learn any information of $\mathbf{x}_0$ even if $\mathcal{A}$ interacts with clients $\{C_j\}_{j\in[1,J]}$ and the evaluator $E$. More formally, it can be defined as follows:

**Definition 1.** *We say that the FE-PPML protocol $\mathcal{P}$ securely computes $\{F_l(\mathbf{x}_0)\}$ for $l \in [0, \zeta]$ in the presence of static malicious adversaries if for every probabilistic polynomial-time adversary $\mathcal{A}$ in the real-world, there exists a probabilistic polynomial-time algorithm $\mathcal{S}$ in the ideal-world such that for every input data $\mathbf{x}_j$ and machine learning model $F_l$, we have that the following two distribution ensembles (over the security parameter $\lambda$) are computationally indistinguishable:*

$$\{\mathsf{REAL}_{\mathcal{P},\mathcal{A}}(\{\mathbf{x}_j\}_{j\in[0,J]}, \{F_l\}_{l\in[0,\zeta]}, \lambda, \zeta)\}$$
$$\overset{c}{\approx} \{\mathsf{IDEAL}_{\mathcal{S}}(\{\mathbf{x}_j\}_{j\in[0,J]}, \{F_l\}_{l\in[0,\zeta]}, \lambda, \zeta)\},$$

*where $\{\mathsf{REAL}_{\mathcal{P},\mathcal{A}}(\{\mathbf{x}_j\}_{j\in[0,J]}, \{F_l\}_{l\in[0,\zeta]}, \lambda, \zeta)\}$ denotes the view of the corrupted clients and the adversary $\mathcal{A}$ from the real execution of $\mathcal{P}$ on inputs $(\mathbf{x}_j, F_l, \lambda, \zeta)$ for $j \in [0, J]$ and $l \in [0, \zeta]$, and $\{\mathsf{IDEAL}_{\mathcal{S}}(\{\mathbf{x}_j\}_{j\in[0,J]}, \{F_l\}_{l\in[0,\zeta]}, \lambda, \zeta)\}$ denotes the (simulated) view of corrupted clients and the simulator $\mathcal{S}$ from the ideal execution of $\mathcal{I}$ on inputs $(\mathbf{x}_j, F_l, \lambda, \zeta)$ for $j \in [0, J]$ and $l \in [0, \zeta]$. Both views are defined as the outputs of the following experiments:*

– *In the* $\mathsf{REAL}_{\mathcal{P},\mathcal{A}}(\{\mathbf{x}_j\}_{j\in[0,J]}, \{F_l\}_{l\in[0,\zeta]}, \lambda, \zeta)$*:*
  1. $\mathsf{pk} \leftarrow KD^{\mathsf{Setup}(1^\lambda,\zeta)}$
  2. $f_{l,i} \leftarrow E$ *for* $l \in [0, \zeta]$, $i \in [1, \mathcal{E}]$
  3. $\mathsf{ct}_{\mathbf{x}_0} \leftarrow C_0^{\mathsf{Enc}(\mathsf{pk},\mathbf{x}_0)}$
  4. $(\mathsf{ct}_{\mathbf{x}_j}, \mathbf{x}_j) \leftarrow C_j^{\mathsf{Enc}(\mathsf{pk},\mathbf{x}_j)}$ *for* $j \in [1, J]$
  5. $\mathsf{fk}_{\mathsf{Enc}\circ f_{l,i}} \leftarrow KD^{\mathsf{KeyGen}(\mathsf{msk},\mathsf{Enc}\circ f_{l,i})}$ *for* $l \in [0, \zeta]$, $i \in [1, \mathcal{E}]$
  6. $\mathsf{Enc}(\bigcirc_{t=1}^{i} f_{l,t}(\mathbf{x}_j)) \leftarrow E^{\mathsf{Dec}(\mathsf{pk},\mathsf{fk}_{\mathsf{Enc}\circ f_{l,t}},\mathsf{ct}_{\bigcirc_{t=1}^{i-1} f_{l,t}(\mathbf{x}_j)})}$
  7. $F_l(\mathbf{x}_j) \leftarrow E^{\mathsf{Dec}(\mathsf{pk},\mathsf{fk}_{f_{l,\mathcal{E}}},\mathsf{Enc}(\bigcirc_{i=1}^{\mathcal{E}-1} f_{l,i}(\mathbf{x}_j)))}$

– *In the* $\mathsf{IDEAL}_{\mathcal{S}}(\{\mathbf{x}_j\}_{j\in[0,J]}, \{F_l\}_{l\in[0,\zeta]}, \lambda, \zeta)$*:*
  1. $\mathsf{pk}^{\mathcal{S}} \leftarrow KD^{\mathsf{Setup}^{\mathcal{S}}(1^\lambda,\zeta)}$
  2. $f_{l,i} \leftarrow E$ *for* $l \in [0, \zeta]$, $i \in [1, \mathcal{E}]$
  3. $\mathsf{ct}_{\mathbf{x}_0} \leftarrow C_0^{\mathsf{Enc}^{\mathcal{S}}(\mathsf{pk}^{\mathcal{S}},F_l(\mathbf{x}_0))}$
  4. $(\mathsf{ct}_{\mathbf{x}_j}, \mathbf{x}_j) \leftarrow C_j^{\mathsf{Enc}(\mathsf{pk}^{\mathcal{S}},\mathbf{x}_j)}$ *for* $j \in [1, J]$
  5. $\mathsf{fk}_{\mathsf{Enc}\circ f_{l,i}} \leftarrow KD^{\mathsf{KeyGen}^{\mathcal{S}}(\mathsf{msk},\mathsf{Enc}\circ f_{l,i})}$ *for* $l \in [0, \zeta]$, $i \in [1, \mathcal{E}]$

6. $\mathsf{Enc}(\bigcirc_{t=1}^{i} f_{l,t}(\mathbf{x}_j)) \leftarrow E^{\mathsf{Dec}^{\mathcal{S}}(\mathsf{pk}^{\mathcal{S}}, \mathsf{fk}_{\mathsf{Enc} \circ f_{l,t}}, \mathsf{ct}_{\bigcirc_{t=1}^{i-1} f_{l,t}(\mathbf{x}_j)})}$
7. $F_l(\mathbf{x}_j) \leftarrow E^{\mathsf{Dec}^{\mathcal{S}}(\mathsf{pk}^{\mathcal{S}}, \mathsf{fk}_{f_{l,\mathcal{E}}}, \mathsf{Enc}(\bigcirc_{i=1}^{\mathcal{E}-1} f_{l,i}(\mathbf{x}_j)))}$

In Section 4.4, we will prove that the protocol in Fig. 9 achieves the security. Here, we consider the evaluator $E$ as the malicious party.

### 4.2 Extra Ingredient: integrating linear functions for security

Our main idea is to express higher-order polynomial operations required in ML as compositions of quadratic polynomials as $F = \bigcirc_{i=1}^{\mathcal{E}} f_i$, and to perform each quadratic polynomial using LinEnc-QFE. To ensure security, instead of directly generating functional keys of $f_i$, we compute functional keys for $\mathsf{Enc}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE}}, \mathsf{pp}_{\mathsf{QFE}}, \cdot) \circ f_i$. Consequently, the decryption algorithm $\mathsf{Dec}_{\mathsf{QFE}}$ using such a functional key outputs the encrypted ciphertext of intermediate evaluations. However, this approach renders the encryption deterministic, thereby compromising security. To reintroduce randomness for security, we compose linear functions during key generation. Specifically, for a function $f_i : \mathbb{Z}_N^{\ell} \to \mathbb{Z}_N^{\ell}$, we define a function $G_i : \mathbb{Z}_N^{\ell+k} \to \mathbb{Z}_N^{\ell+k}$ for $i \in [1, \mathcal{E}-1]$ by $G_i = h_i \circ f_i \circ h'_{i-1}$. Here, $h_i$ and $h'_i$ are linear functions that serve to calculate the composition order and substantiate the security of our protocol.

Given positive integers $\ell$, $N$, $k$ and $B_{\mathcal{H}}$, we sample a matrix $\mathbf{H} \in \mathbb{Z}^{(\ell+k)\times\ell}$ and its left-inverse $\mathbf{H}' \in \mathbb{Z}^{\ell\times(\ell+k)}$, and then define linear functions $h(\mathbf{x}) = \mathbf{H} \cdot \mathbf{x}$ and $h'(\mathbf{x}) = \mathbf{H}' \cdot \mathbf{x}$. The explicit algorithm is described in Alg. 1. Furthermore, in order to guarantee the security of the protocol, we consider a matrix $\mathbf{\Gamma} \in \{0,1\}^{2\ell\times\ell}$ and its left-inverse $\mathbf{\Gamma}^{-1} \in \mathbb{Z}_N^{\ell\times 2\ell}$ to randomize the messages in the protocol. We denote that $\gamma$ and $\gamma^{-1}$ are linear functions that correspond to $\mathbf{\Gamma}$ and $\mathbf{\Gamma}^{-1}$, respectively.

**Algorithm 1** Algorithm for generating linear functions $h$ and $h'$ for randomness

**function** GENLIN($\ell, N, k, B_{\mathcal{H}}$)
  Sample a matrix $\mathbf{U} \leftarrow \{-1, 0, 1\}^{(\ell+k)\times k}$ such that $\mathbf{U}[i]^T \cdot \mathbf{U}[j] = 0$ for all $i \neq j$
  Compute a kernel matrix $\mathbf{V} \in \mathbb{Z}^{\ell\times(\ell+k)}$ such that $\mathbf{V} \cdot \mathbf{U} = \mathbf{O}$
  $\mathbf{H} \leftarrow [-B_{\mathcal{H}}, B_{\mathcal{H}}]^{(\ell+k)\times\ell}$ until $\mathbf{V} \cdot \mathbf{H}$ is left-invertible over $\mathbb{Z}_N$
  Define $\mathbf{T} \in \mathbb{Z}_N^{\ell\times\ell}$ as the left inverse of $\mathbf{V} \cdot \mathbf{H}$ over $\mathbb{Z}_N$
  Define $h(\mathbf{x}) = \mathbf{H} \cdot \mathbf{x}, h'(\mathbf{x}) = \mathbf{H}' \cdot \mathbf{x}$ for $\mathbf{H}' = \mathbf{T} \cdot \mathbf{V} \bmod N$
  **return** $h, h'$
**end function**

For the sake of security, the parameter $k$ requires a condition to ensure the sufficient number of possible $\mathbf{H}$'s for sampling.

**Lemma 1.** *Given $(h, h') \leftarrow$ GENLIN$(\ell, N, k, B_{\mathcal{H}})$, there exist at least $(2 \cdot B_{\mathcal{H}})^{\ell k}$-matrices $\overline{\mathbf{H}} \in [-B_{\mathcal{H}}, B_{\mathcal{H}}]^{(2\ell+k)\times\ell}$ such that $\mathbf{H}' \cdot \overline{\mathbf{H}} = \mathbf{I}_\ell$.*

*Proof.* Given the definitions of matrices $\mathbf{U}$ and $\mathbf{V}$, it holds that

$$\mathbf{H}' \cdot (\mathbf{H} + \mathbf{U} \cdot \mathbf{B}) = \mathbf{I}_\ell \bmod \mathbb{Z}_N$$

for any matrix $\mathbf{B} \in \mathbb{Z}^{k\times\ell}$. This requires us to enumerate the matrices $\mathbf{B}$ such that $(\mathbf{H} + \mathbf{U} \cdot \mathbf{B}) \in [-B_{\mathcal{H}}, B_{\mathcal{H}}]^{(\ell+k)\times\ell}$.

Define $\mathbf{B}[i]$ and $\mathbf{H}[i]$ as the $i$-th column vector of $\mathbf{B}$ and $\mathbf{H}$, respectively. The task then is to determine the cardinality of the set:

$$\mathcal{S}_i := \{\mathbf{B}[i] \mid \mathbf{H}[i] + \mathbf{U} \cdot \mathbf{B}[i] \in [-B_{\mathcal{H}}, B_{\mathcal{H}}]\}.$$

The total number of suitable matrices $\mathbf{B}$ is given by the product $\prod_{i=1}^{\ell} |\mathcal{S}_i|$.

According to the Alg. 1, the orthogonality of $\mathbf{U}$'s column vectors implies that the entries of $\mathbf{B}[i]$ do not interfere with each other's magnitudes. This allows for the assessment of each entry's potential size in $\mathbf{B}[i]$ and the computation of the set $\mathcal{S}_i$'s size.

Only the possible coefficients for the first entry need to be counted for each $\mathbf{B}[i]$ due to symmetry. Therefore, the problem simplifies to counting the number of $c_1$ such that each entry size of $\mathbf{H}[i] + \mathbf{U}[1] \cdot c_1$ remains below $B_{\mathcal{H}}$. Initially, counting occurrences where $\mathbf{U}[1] \cdot c_1$ fits within the set $[-B_{\mathcal{H}}, B_{\mathcal{H}}]^{\ell+k}$ indicates $c_1 \in [-B_{\mathcal{H}}, B_{\mathcal{H}}]$, yielding exactly $2 \cdot B_{\mathcal{H}} + 1$ possibilities. The focus is on shifted instances from these cases. Decomposing the $\mathbf{H}[i]$ into $c_1' \cdot \mathbf{H}[i] + \mathbf{H}[i]^{\perp}$, where $\mathbf{H}[i]^{\perp}$ is an orthogonal vector to $\mathbf{H}[i]$, it is required to $c_1 + c_1' \in [-B_{\mathcal{H}}, B_{\mathcal{H}}]$ so that $\mathbf{H}[i] + c_1 \cdot \mathbf{U}[1] \in [-B_{\mathcal{H}}, B_{\mathcal{H}}]^{\ell+k}$. It guarantees at least $2 \cdot B_{\mathcal{H}}$ cases for $c_1$. As a result, for all $\mathbf{U}[i]$, $(2 \cdot B_{\mathcal{H}})^k$ cases are possible and $|\mathcal{S}_i| \geq (2 \cdot B_{\mathcal{H}})^k$.

Combining all indices, we have $\prod_{i=1}^{\ell} |\mathcal{S}_i| \geq (2 \cdot B_{\mathcal{H}})^{\ell k}$, which completes the proof. □

From Lemma 1, we set the parameters $k$ and $B_{\mathcal{H}}$ to satisfy the following condition.

$$(2 \cdot B_{\mathcal{H}})^{\ell k} \geq 2^{\lambda} \tag{1}$$

This ensures a sufficiently large number of possible $\mathbf{H}$ configurations for security.

### 4.3 Protocol description

This section provides a fully encrypted PPML protocol using composite evaluations via LinEnc-QFE. The participants of our protocol consist of three types of entities: clients $\{C_j\}_{j \in [0,J]}$, a key distributor $KD$, and an evaluator $E$. As discussed, instead of generating functional keys for $f_i$ directly, we compose linear functions to add randomness. In the protocol, the key distributor generates a pair of linear function $(\gamma, \gamma^{-1})$ as described in Section 4.2 and $\mathcal{E}$ distinct pairs of linear functions $\{(h_i, h_i')\}$ using Alg. 1 for $i \in [0, \mathcal{E}-1]$, then define $G_i$ by $h_i \circ f_i \circ h_{i-1}'$ for each $i \in [1, \mathcal{E}-1]$ and $G_{\mathcal{E}}$ by $f_{\mathcal{E}} \circ h_{\mathcal{E}-1}'$. Then, we observe that for any $i \in [1, \mathcal{E}-1]$ and a vector $\mathbf{x}$,

$$\begin{aligned}
G_{i+1}(h_i(\mathbf{x})) &= (h_{i+1} \circ f_{i+1} \circ h_i')(h_i(\mathbf{x})) \\
&= h_{i+1}(f_{i+1}(\mathbf{x})) \bmod N, \\
\left(\bigcirc_{t=1}^{i} G_t\right)(h_0(\mathbf{x})) &= h_i(\bigcirc_{t=1}^{i} f_t(\mathbf{x})) \bmod N, \\
\left(\bigcirc_{t=1}^{\mathcal{E}} G_t\right)(h_0(\mathbf{x})) &= \bigcirc_{t=1}^{\mathcal{E}} f_t(\mathbf{x}) \bmod N.
\end{aligned}$$

The whole progress of our protocol is described in Fig. 9. In the protocol, we denote by $\mathsf{E}_i$ the encryption algorithm corresponding to the $i$-th invocation of the LinEnc-QFE scheme. Moreover, there are two additional subscripts, denoted by $l$ and $j$. The index $l$ indicates several machine learning model $F_l$ that is compositions of quadratic polynomials $f_{l,i}$, and the other index $j$ indicates a client $C_j$.

*Correctness* (of the protocol in Fig. 9). By definition of $\mathsf{fk}_{\mathsf{E}_1 \circ G_{l,0,j}}$, it holds that

$$\begin{aligned}
M_{l,0,j} &= \mathsf{cDec}_{\mathsf{pkIPFE}}(\mathsf{pk}_{\mathsf{pkIPFE}}, \mathsf{fk}_{\mathsf{E}_1 \circ G_{l,0,j}}, \mathsf{ct}_{l,j}, L) \\
&= (\mathsf{E}_1 \circ h_{l,0,j} \circ \gamma_{l,j}^{-1})(\gamma_{l,j}(\mathbf{x}_j)) = \mathsf{E}_1(h_{l,0,j}(\mathbf{x}_j)).
\end{aligned}$$

Furthermore, we claim that $M_{l,i,j} = \mathsf{E}_{i+1}(h_{l,i,j} \bigcirc_{t=1}^{i} f_{l,t}(\mathbf{x}_j))$ for each $i \in [1, \mathcal{E}-1]$. Using the fact that $h_{l,i,j}' \circ h_{l,i,j}$ is an identity function for each $i$, it can be checked inductively as follows:

$$\begin{aligned}
M_{l,i,j} &= \mathsf{cDec}_{\mathsf{QFE}}(\mathsf{pp}_{\mathsf{QFE},i}, \mathsf{fk}_{\mathsf{E}_{i+1} \circ G_{l,i,j}}, M_{l,i-1,j}, L) \\
&= (\mathsf{E}_{i+1} \circ h_{l,i,j} \circ f_{l,i} \circ h_{l,i-1,j}') \left((h_{l,i-1,j} \bigcirc_{t=1}^{i-1} f_{l,t}(\mathbf{x}_j)\right) \\
&= (\mathsf{E}_{i+1} \circ h_{l,i,j})(\bigcirc_{t=1}^{i} f_{l,t}(\mathbf{x}_j)).
\end{aligned}$$

Therefore, it satisfies

$$M_{l,\mathcal{E}-1,j} = (\mathsf{E}_{\mathcal{E}} \circ h_{l,\mathcal{E}-1,j})(\bigcirc_{t=1}^{\mathcal{E}-1} f_{l,t}(\mathbf{x}_j)).$$

Then the final result is:

$$\begin{aligned} M_{l,\mathcal{E},j} &= \mathsf{Dec}_{\mathsf{QFE}}(\mathsf{pp}_{\mathsf{QFE},\mathcal{E}}, \mathsf{fk}_{G_{l,\mathcal{E},j}}, M_{l,\mathcal{E}-1,j}) \\ &= (f_{l,\mathcal{E}} \circ h'_{l,\mathcal{E}-1,j}) \left( (h_{l,\mathcal{E}-1,j} \bigcirc_{t=1}^{\mathcal{E}-1} f_{l,t}(\mathbf{x}_j) \right) \\ &= \bigcirc_{i=1}^{\mathcal{E}} f_{l,i}(\mathbf{x}_j) = F_l(\mathbf{x}_j). \end{aligned}$$

**Time complexity.** Let $\{f_{l,i}\}$ be a family of quadratic functions and $\{h_{l,i,j}\}$ be a family of invertible linear functions in Section 4.2. It is clear that the underlying algorithms including $\mathsf{cDec}, \mathsf{cKeygen}, \mathsf{Enc}$ terminates in polynomial time in input parameters. Then Fig. 9 terminates in polynomial time in input parameters as well.

### 4.4 Security proof of the protocol

In this section, we show that the protocol $\mathcal{P}$ described in Fig. 9 does not reveal information about an unknown message under a malicious model. We recall the malicious security of the protocol based on Definition 1. Let $KD$, $E$, and $\{C_j\}_{j\in[0,J]}$ denote the (trusted) key distributor, evaluator, and clients, respectively, who are participants in the protocol. We here note that $C_0$ is only one honest client, and the evaluator $E$ has $\zeta + 1$ machine learning models. Each model consists of a series of quadratic polynomials. Let $F_0 = \bigcirc_{i=1}^{\mathcal{E}} f_{0,i}$ be a machine learning model requested by $C_0$. The other models, denoted as $\{F_l\}_{l\in[1,\zeta]} := \{\bigcirc_{i=1}^{\mathcal{E}} f_{l,i}\}_{l\in[1,\zeta]}$, are not requested by $C_0$.

The objective is to demonstrate that an adversary $\mathcal{A}$ cannot distinguish between a protocol $\mathcal{P}$ and an ideal protocol from its view. Then, it ensures the privacy of any intermediate values of the honest client $C_0$ during machine learning model computations because $\mathcal{A}$ cannot distinguish where the intermediate value comes from. This indistinguishability holds even if $\mathcal{A}$ interacts with other clients $\{C_j\}_{j\in[1,J]}$ and an evaluator $E$. To elucidate the adversary's view, we describe both real-world and ideal-world protocols.

**Real-World.** $\mathcal{A}$ interacting with the corrupted clients and an evaluator can be described as follows:

- Functional Keys: $E$ queries several models $\{F_l\}_{l\in[0,\zeta]}$ corresponding to all clients to $KD$ and transmits $\{\mathsf{F}_{l,j}\}_{l\in[0,\zeta],j\in[0,J]}$ with $\mathsf{F}_{l,j} = \{\mathsf{fk}_{\mathsf{E}_{i+1}\circ G_{l,i,j}}\}_{i\in[0,\mathcal{E}-1]} \cup \mathsf{fk}_{G_{l,\mathcal{E},j}}$ to $\mathcal{A}$, where $G_{l,i,j} = h_{l,i,j} \circ f_{l,i} \circ h'_{l,i-1,j}$ as in the Fig. 9.
- Target ciphertext: The honest client $C_0$ selects a message $\mathbf{x}_0$ and makes a ciphertext $\mathsf{ct}_0 := \mathsf{ct}_{0,0} = \mathsf{Enc}_{\mathsf{pkIPFE}}(\mathsf{pk}_{\mathsf{pkIPFE}}, \gamma_{0,0}(\mathbf{x}_0))$, given a linear function $\gamma_{0,0}$. Then, $C_0$ sends it to $E$.
- Additional Ciphertexts: For $l \in [0,\zeta]$ and $j \in [1,J]$, each client $C_j$ also generates a ciphertext $\mathsf{ct}_{l,j} := \mathsf{Enc}_{\mathsf{pkIPFE}}(\mathsf{pk}_{\mathsf{pkIPFE}}, \gamma_{l,j}(\mathbf{x}_j))$ for some message $\mathbf{x}_j$ and a linear function $\gamma_{l,j}$, and sends $\mathsf{ct}_{l,j}$ to $E$ and $(\mathsf{ct}_{l,j}, \mathbf{x}_j)$ to $\mathcal{A}$.
- Evaluation: For every $i \in [1, \mathcal{E}-1], l \in [0,\zeta]$ and $j \in [0,J]$, $E$ computes intermediate values $(\mathsf{E}_i(h_{l,i-1,j} \bigcirc_{t=1}^{i-1} f_{l,t}(\mathbf{x}_j)))$ from ciphertexts and functional keys, and final results of the form $F_l(\mathbf{x}_j)$. $E$ sends them to $\mathcal{A}$.

Based on the description of $\mathcal{P}$, we define $\mathsf{REAL}_{\mathcal{P}}(\mathbf{x}_0, \{\mathbf{x}_j\}_{j\in[1,J]}, \{F_l\}_{l\in[0,\zeta]}, \lambda, \zeta)$ to be the view of $\mathcal{A}$ in the real world.

$$\begin{aligned} \mathsf{REAL}_{\mathcal{P},\mathcal{A}} = \Big\{ & \{\mathbf{x}_j\}_{j\in[1,J]}, \{F_l\}_{l\in[0,\zeta]}, \{F_l(\mathbf{x}_j)\}_{l\in[0,\zeta],j\in[0,J]}, \\ & \{\mathsf{F}_{l,j}\}_{l\in[0,\zeta],j\in[0,J]}, \{\mathsf{ct}_{l,j}\}_{l\in[0,\zeta],j\in[0,J]}, \\ & \{\mathsf{E}_i(h_{i-1,j} \bigcirc_{t=1}^{i-1} f_{l,t}(\mathbf{x}_j))\}_{i\in[1,\mathcal{E}-1],l\in[0,\zeta],j\in[0,J]} \Big\} \end{aligned}$$

**Ideal-World.** In an ideal-world, there exists a simulator $\mathcal{S}$ for a client $C_0$, which mimics the $\mathsf{KeyGen}$ and $\mathsf{Enc}_{\mathsf{pkIPFE}}$. An ideal-world interaction then coincides with the real-world except for terms related to the honest client $C_0$; the target ciphertext and functional key:

- Functional Keys for client $C_0$: $E$ queries function $\{F_l\}_{l\in[0,\zeta]}$ corresponding to all clients to $KD$. Then $KD$ transmits functional keys $\{\mathsf{F}^{\mathcal{S}}_{l,0}\}_{l\in[0,\zeta]}$ with $\mathsf{F}^{\mathcal{S}}_{l,0} = \{\mathsf{fk}_{\mathsf{E}_{i+1}\circ G^{\mathcal{S}}_{l,i,0}}\}_{i\in[0,\mathcal{E}-1]} \cup \mathsf{fk}_{G^{\mathcal{S}}_{l,\mathcal{E},j}}$ to $\mathcal{A}$.

– Target ciphertext: The honest client $C_0$ sends a ciphertext

$$\mathsf{ct}_0^* := \mathsf{Enc}^{\mathcal{S}}_{\mathsf{pkIPFE}}(\mathsf{pk}^{\mathcal{S}}_{\mathsf{pkIPFE}}, \{h_{0,0,0} \circ {\gamma_{0,0}}^{-1}, h_{0,0,0}(\mathbf{x}_0)\})$$

to $E$.

– Additional ciphertexts: These ciphertexts are generated by $\mathsf{ct}_{l,j} := \mathsf{Enc}_{\mathsf{pkIPFE}}(\mathsf{pk}^{\mathcal{S}}_{\mathsf{pkIPFE}}, \gamma_{l,j}(\mathbf{x}_j))$.

Analogues to the real-world, we define $\mathsf{IDEAL}_{\mathcal{S}}(\mathbf{x}_0, \{\mathbf{x}_j\}_{j\in[1,J]}, \{F_l\}_{l\in[0,\zeta]}, \lambda, \zeta)$ to be the view of $\mathcal{A}$ in the ideal-world.

$$\begin{aligned}\mathsf{IDEAL}_{\mathcal{S}} = \Big\{ & \{\mathbf{x}_j\}_{j\in[1,J]}, \{F_l\}_{l\in[0,\zeta]}, \{F_l(\mathbf{x}_j)\}_{l\in[0,\zeta],j\in[0,J]}, \\ & \{\mathsf{F}^{\mathcal{S}}_{l,j}\}_{l\in[0,\zeta],j\in[0,J]}, \mathsf{ct}_0^*, \{\mathsf{ct}_{l,j}\}_{l\in[0,\zeta],j\in[0,J]}, \\ & \{\mathsf{E}_i(h_{i-1,j} \bigcirc_{t=1}^{i-1} f_{0,t}(\mathbf{x}_j))\}_{i\in[1,\mathcal{E}-1],l\in[0,\zeta],j\in[0,J]} \Big\}\end{aligned}$$

We then aim to show that the following two distributions are computationally indistinguishable:

$$\begin{aligned}&\mathsf{REAL}_{\mathcal{P},\mathcal{A}}(\mathbf{x}_0, \{\mathbf{x}_j\}_{j\in[1,J]}, \{F_l\}_{l\in[0,\zeta]}, \lambda, \zeta) \overset{c}{\approx} \\ &\mathsf{IDEAL}_{\mathcal{S}}(\{F_l(\mathbf{x}_0)\}, \{\mathbf{x}_j\}_{j\in[1,J]}, \{F_l\}_{l\in[0,\zeta]}, \lambda, \zeta). \end{aligned} \tag{2}$$

As a high-level idea for proof, a collection of linear functions $\{h_{l,i,j}\}$ (represented by $\{\mathbf{H}_{l,i,j}\}$ for each $l, i$ and $j$) plays a significant role in ensuring security. This matrix allows composite operations only when each $l, j$ is coincided. Hence, the matrix $\{\mathbf{H}_{l,i,0}\}$ are independent to other matrices. In other words, the diversity of $\{h_{l,i,0}\}$ compensates for the lack of randomness of an encryption function corresponding to $\mathsf{E}_i$. Given that the evaluator lacks knowledge of $h_{l,i,0}$, $\mathsf{E}_i(h_{l,i-1,j}(\bigcirc_{t=1}^{i-1} f_{l,t}(\mathbf{x}_j)))$ for any $i \in [1, \mathcal{E}-1]$ seems to be an encryption of random value in the adversary's view. In addition, other clients' ciphertext is not helpful. In the following, we show that there are at least $2^\lambda$-ensembles $(h_{l,i-1,0,k}, \mathbf{x}_{0,k})$ $\mathsf{E}_i(h_{l,i-1,0}(\bigcirc_{t=1}^{i-1} f_{l,t}(\mathbf{x}_0))) = \mathsf{E}_i(h_{l,i-1,0,k}(\mathbf{x}_{0,k}))$. Consequently, the adversary is unable to distinguish $f_{l,i-1}(\mathbf{x}_0)$ and $\mathbf{x}_{0,k}$, even if $\mathcal{A}$ is already familiar with the function $F_l$.

**Theorem 3.** *Let $\lambda$ be the security parameter and $(k, B_{\mathcal{H}})$ be integers such that $(2 \cdot B_{\mathcal{H}})^{\ell k} \geq 2^\lambda$. The protocol in Fig. 9 securely computes $F_0 = \bigcirc_{t=1}^{\mathcal{E}} f_{0,t}$ over $\mathbb{Z}^\ell$, when pkIPFE is adaptively simulation secure and LinEnc-QFE is $(2\ell+1)$-ciphertext bounded semi-adaptively secure. More precisely, there exists a simulator $\mathcal{S}$ such that any adversary $\mathcal{A}$ cannot distinguish the Eq.* (2) *except for the following advantages:*

$$\mathsf{Adv}_{\mathsf{pkIPFE}} + \mathcal{E} \cdot \mathsf{Adv}_{\mathsf{LinEnc\text{-}QFE}} + \frac{\mathcal{E}}{2^\lambda}$$

*where $\mathsf{Adv}_{\mathsf{pkIPFE}}, \mathsf{Adv}_{\mathsf{LinEnc\text{-}QFE}}$ are the advantages of pkIPFE and LinEnc-QFE, respectively.*

*Proof of Theorem 3.* The proof of Theorem 3 will be given by Appendix F. □

## 5 Benchmarks and Applications to ML Classification

In this section, we present the results of our protocol's performance and demonstrate its application in secure inference for classification. All benchmarks were performed on Linux with Intel(R) Xeon(R) Silver 4208 CPU @ 2.10GHz with 30GB RAMS.

### 5.1 Benchmarks

First, we provide the benchmarks of our protocol with toy parameters. Our implementation utilizes a public key IPFE [1] based on the Decisional composite residuosity (DCR) assumption. To achieve 128-bit security, we preselect $p$ and $q$ by 3072-bit primes for both DCR-based IPFE and our protocol. These values are hard-coded into our implementation.

Under the above settings, we execute our protocol Fig. 9 for two-layer model across various dimensions $\ell$, where $\ell$ ranges from 1 to 6. As an input, we randomly set a set of vectors $\{\mathbf{c}_i\}_{i\in[1,\ell]} \subset \mathbb{Z}^\ell$ and $\mathbf{d} \in \mathbb{Z}^\ell$. Then, we consider a composition of two quadratic functions as an evaluation model: a multinomial polynomial $f_1 : \mathbb{Z}^\ell \to \mathbb{Z}^\ell$ and $f_2 : \mathbb{Z}^\ell \to \mathbb{Z}$, defined as the following formula.

$$f_1(\mathbf{x}) = \left((\langle \mathbf{c}_i, \mathbf{x}\rangle)^2\right)_{1\le i\le \ell}, f_2(\mathbf{x}) = (\langle \mathbf{d}, \mathbf{x}\rangle)^2$$

Consequently, the protocol outputs the evaluation of two quartic functions $f_2 \circ f_1(\mathbf{x})$ for an input $\mathbf{x}$. It is important to note that the time cost of the protocol is independent of the size of an input vector and the coefficients of the functions. Therefore, these parameters are randomly chosen from small integers.

The result is in the Table 3. The time cost is summed up by the Setup, KeyGen, Enc, and Dec categories. In details, using the notation of protocol in Fig. 9, we elaborate as following.

- "Setup" in the Key Distributor involves "Protocol Setup" step.
- "KeyGen" in the Key Distributor implies "Functional Key Generation" step.
- "Enc" in the Client implies "Encryption" step.
- "Dec" in the Evaluator implies "Model Evaluation" step.

Note that every Setup and KeyGen algorithms in Fig. 9 are computed by a key distributor, and both the client and evaluator are only required to compute encryption and decryption, respectively.

We note that the KeyGen and Dec steps in our experimental results significantly dominate the total time costs. This is because both steps involve power operations on a group, specifically computing the power of an integer sampled in a 12288-bit space. Additionally, since we use the DCR group in our implementation, the discrete logarithm with base $N+1$ can be performed using simple arithmetic operations, which does not require much time. In addition, other parameters in our protocol are set to $Q_i = 2\ell_i + 1$ and $k = 1$. The details of our implementation can be found in the github repository[6].

| Dim ($\ell$) | Key Distributor | | Evaluator | Client |
|---|---|---|---|---|
| | Setup | KeyGen | Dec | Enc |
| 1 | 0.69s | 0.55h | 0.58h | 0.70s |
| 2 | 0.70s | 0.60h | 0.64h | 0.75s |
| 3 | 0.87s | 1.08h | 1.13h | 0.77s |
| 4 | 1.17s | 1.79h | 1.84h | 0.97s |
| 5 | 1.46s | 2.75h | 2.78h | 1.02s |
| 6 | 1.74s | 4.04h | 4.02h | 1.11s |

**Table 3.** Benchmarks for protocol under 128-bit security level. The protocol runs for two-layer model with input and intermediate dimension $\ell$ and outputs a scalar value. Note that we precompute 3072-bit prime in advance so the time to select 3072-bit prime is not included during the Setup.

### 5.2 Complexity Analysis

In this section, we provide a detailed complexity analysis of our proposed method for evaluating multi-layer neural networks. Our analysis focuses on the computational complexities of dominant operations. In our case, assuming that we only deal with small dimensions in neural networks and a group for a 128-bit security parameter, the dominant operation is the exponentiation over $\mathbb{Z}^*_{N^2}$ with $N = p \cdot q$, the product of two 3072-bit primes $p, q$. We let $\ell_i$ denote the message dimension of the $i$-th layer in neural networks and $L$ be a decomposition parameter.

Adapting algorithms and parameters from Fig. 9, the complexity for each step in our algorithm is as below. Note that we set $k = O(1)$, $L = O(\log \ell_{i+1})$, and $Q_i = O(\ell_i)$ for the QFE of $i$-th layer.

[6] https://github.com/swanhong/fully-encrypted-ml

| | Enc | Keygen | First Layer | Intermediate Layer | Last Layer |
|---|---|---|---|---|---|
| [37] | $O(\ell_1)(E_1+E_2)$ | $E_2$ | $O(\ell_1^2)(E_1+P)$ | $*$ | $*$ |
| ours | $O(\ell_1)E$ | $\left(\sum_{i=1}^{\mathcal{E}-1} O(\ell_i^2 \cdot \ell_{i+1})\right) E$ | $O(\ell_1 \log \ell_2)E$ | $O(\ell_i^2 \ell_{i+1} \log \ell_{i+1})E$ | $O(\ell_{\mathcal{E}-1}^2 \ell_{\mathcal{E}})E$ |

**Table 4.** Comparison of the complexity analysis for evaluating multi-layer neural networks. [37] utilizes a pairing map $G_1 \times G_2 \to G_T$ over the groups $G_1, G_2$, and $G_T$, whereas our method employs a group $G = \mathbb{Z}_{N^2}$. We denote the exponentiation complexities over $G_1, G_2$, and $G$ by $E_1, E_2$, and $E$, respectively, and the pairing complexity by $P$. In our environment, each operation costs on average $E_1 = 16$ms, $E_2 = 16$ms, and $P = 22$ms using the Charm library [2], and $E = 321$ms in our implementation. An asterisk$(*)$ indicates that all computations in the layer are performed in plaintext.

- Encryption (Enc): The encryption step computes $\mathsf{pkIPFE}$ encryption for a vector of dimension $2\ell_1$, which costs $O(\ell_1)$ exponentiations.
- Functional Key Generation (Keygen): In the functional key generation step, the key distributor generates one $\mathsf{pkIPFE}$ functional key of dimension $\ell_1 + k$ and a $\mathsf{LinEnc\text{-}QFE}$ key of dimension $\ell_i + k$ for $\ell_{i+1}$ times for each $i \in [1, \mathcal{E}-1]$. For the last layer, a $\mathsf{LinEnc\text{-}QFE}$ key is generated for dimension $\ell_{\mathcal{E}}$. Hence, the total number of exponentiations is $\sum_{i=1}^{\mathcal{E}-1} O(\ell_i^2 \cdot \ell_{i+1}) + O(\ell_{\mathcal{E}})$, where the last $O(\ell_{\mathcal{E}})$ term can be omitted.
- Evaluation (First Layer): For the first layer, the evaluator computes $\mathsf{pkIPFE}$ decryption, which requires $L \cdot O(\ell_1) = O(\ell_1 \log \ell_2)$ exponentiations.
- Evaluation (Intermediate Layers): For each intermediate layer $i$ (where $i \in [2, \mathcal{E}-1]$), the evaluator performs $\mathsf{LinEnc\text{-}QFE}$ decryption, which evaluates the quadratic function of input dimension $\ell_i$ and output dimension $\ell_{i+1}$. This requires $L \cdot O(\ell_{i+1}) = O(\ell_{i+1} \log \ell_{i+1})$ number of $\mathsf{LinEnc\text{-}QFE}$ decryption operations of input dimension $\ell_i$, totaling $O(\ell_i^2 \ell_{i+1} \log \ell_{i+1})$ exponentiations.
- Evaluation (Last Layer): For the last layer, a single $\mathsf{LinEnc\text{-}QFE}$ decryption from dimension $\ell_{\mathcal{E}-1}$ to $\ell_{\mathcal{E}}$ costs $O(\ell_{\mathcal{E}-1}^2 \ell_{\mathcal{E}})$ exponentiations.

Table 4 summarizes the complexity analysis and shows the comparison with [37].

### 5.3 Application to Quadratic Neural Networks with UCI Datasets

In this section, we briefly introduce Quadratic Neural Networks (QNN) and their application in secure inference for image classification.

**Quadratic Neural Networks.** Quadratic Neural Networks (QNN) utilize a neural network model with a *quadratic* activation function, which is mainly used for data classification. The input data consists of pairs $(\mathbf{a}_k, b_k)$, where $\mathbf{a}_k \in \mathbb{R}^\ell$ represents the input features, and $b_k \in [d]$ is a discrete class label, where $d$ is the number of classes. The primary goal of a neural network for classification is to predict the probability that an input belongs to a specific class.

We consider a two-layer neural network, so that we assume the presence of $u$ units in the hidden layer and $d$ distinct classes. The model involves two steps of computation:

1. In the hidden layer, each unit computes $f_h(\mathbf{x}) = \left((\langle \mathbf{W}_{h,i}, \mathbf{x} \rangle + \beta_{h,i})^2\right)_{1 \le i \le u}$, where $\mathbf{W}_h = (\mathbf{W}_{h,1}, \ldots, \mathbf{W}_{h,u}) \in \mathbb{R}^{u \times \ell}$ is a weight matrix and $\beta_h \in \mathbb{R}^u$ is a bias.
2. In the output layer, each unit calculates $f_o(\mathbf{x}) = \left((\langle \mathbf{W}_{o,j}, \mathbf{x} \rangle + \beta_{o,j})^2\right)_{1 \le j \le d}$, where $\mathbf{W}_o = (\mathbf{W}_{o,1}, \ldots, \mathbf{W}_{o,d}) \in \mathbb{R}^{d \times u}$ is a weight matrix and $\beta_o \in \mathbb{R}^d$ is a bias.

After training phase, we find the models $f_h = (f_{h,1}, f_{h,2}, \ldots, f_{h,u})$ and $f_o = (f_{o,1}, f_{o,2}, \ldots, f_{o,d})$ for some univariate functions $\{f_{h,i}\}_{i=1}^u$ and $\{f_{o,j}\}_{h=1}^d$, and then the inference phase can get a prediction vector $\mathbf{b}$ of a new input data $\mathbf{a}$ by computing following multinomial quadratic formulas:

$$\mathbf{a}^* = (f_{h,1}(\mathbf{a}), f_{h,2}(\mathbf{a}), \ldots, f_{h,u}(\mathbf{a}))\,,$$
$$\mathbf{b} = (f_{o,1}(\mathbf{a}^*), f_{o,2}(\mathbf{a}^*), \ldots, f_{o,d}(\mathbf{a}^*))\,.$$

**Application to UCI Datasets.** In this section, we describe the implementation of the inference step of QNN with real datasets using our protocol. We employed the Iris and Breast Cancer datasets from the UCI Machine Learning Repository [10]. For both datasets, we implemented a 2-layer neural network model with quadratic activation functions and utilized cross-entropy as the loss function. The model's architecture includes an input layer of dimension $\ell = 4$ for Iris (or $\ell = 9$ for Breast Cancer), a hidden layer with 4 nodes ($u = 4$), and an output layer with $d = 3$ for Iris (or $d = 2$ for Breast Cancer). Other parameters for the protocol are chosen the same as our previous benchmarks.

We converted the real numbers in both dataset and model parameters into integers by scaling each value by $S = 2^{30}$.To enable evaluating $f_h$ and $f_o$ using our protocol, we scale model parameters and define corresponding matrices

$$M_h = \left(S \cdot \mathbf{W}_h, S^2 \cdot \beta_h\right), M_o = \left(S \cdot \mathbf{W}_o, S^5 \cdot \beta_o\right).$$

Then, after performing a linear operation, the output is scaled as follows (rounding is performed after scaling but ignored in the description below).

1. To evaluate the hidden layer $f_h$, compute the scaled output as follows.
   $(M_h \otimes M_h) \cdot ((S \cdot \mathbf{a} \| 1) \otimes (S \cdot \mathbf{a} \| 1))$
   $= \left((S^2 \cdot \mathbf{W}_{h,i} \cdot \mathbf{a} + S^2 \cdot \beta_{h,i})^2\right)_{1 \le i \le u} = S^4 \cdot f_h(\mathbf{a})$.
2. To evaluate the output layer $f_o$, compute the scaled output as follows.
   $(M_o \otimes M_o) \cdot \left((S^4 \cdot f_h(\mathbf{a}) \| 1) \otimes (S^4 \cdot f_h(\mathbf{a}) \| 1)\right)$
   $= \left((S^5 \cdot \mathbf{W}_{o,j} \cdot f_h(\mathbf{a}) + S^5 \cdot \beta_{o,j})^2\right)_{1 \le j \le d}$
   $= S^{10} \cdot f_o(f_h(\mathbf{a}))$.

We rescale the result by dividing it by $S^{10} = 2^{300}$ to obtain the desired output $f_o(f_h(\mathbf{a}))$ as real numbers. This scaling factor allowed us to achieve inference results in our protocol that were nearly identical to the results obtained in plain computation, with an error of less than $10^{-7}$.

The benchmark results, including the time taken for setup, key generation, decryption, and encryption, are presented in Table 5.

| | Key Distributor | Evaluator | Client | |
|---|---|---|---|---|
| | Setup | KeyGen | Dec | Enc |
| **Iris** | 1.47s | 1.05h | 1.11h | 0.93s |
| **Breast** | 4.92s | 3.61h | 3.32h | 1.13s |

**Table 5.** Benchmarks for secure inference per one input for the 128-bit security level. Note that 1) we precompute 3072-bit prime in advance so the time to select 3072-bit prime is not included during the Setup. 2) Iris data and Breast data use parameters $(\ell, u, d) = (4, 4, 3)$ and $(9, 4, 2)$, respectively.

## 6 Conclusion

In this paper, we introduce the notion of LinEnc functional encryption and, based on this, construct the first "fully encrypted" PPML protocol that supports "arbitrary" functions. Unlike existing FE-based PPML protocols, our protocol ensures no intermediate leakage during computation and achieves security in the malicious model. Furthermore, unlike existing MPC-based or FHE-based PPML protocols, it requires no client (i.e., data owner) involvement during the inference phase, except for providing input data to the evaluator.

While our algorithm excels in terms of security and user convenience, it does have a significant drawback in terms of operation time. Generating a ciphertext format necessitates considering a large underlying space. Specifically, we have adopted the decision composite residuosity (DCR) based scheme to accommodate this extensive space. However, computations for the DCR scheme

involve exponentiations and multiplications, resulting in significant computational costs. Notably, exponentiation operations are approximately $10^6\times$ more time-consuming than multiplication operations, which is (normally) a basic computation in other protocols. Nevertheless, we believe that our protocol serves as a new approach to secure computations.

**Acknowledgment.** J.Kim was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. RS-2024-00399491, Development of Privacy-Preserving Multiparty Computation Techniques for Secure Multiparty Data Integration). M. Seo was supported by Samsung Electronics Co., Ltd(IO221213-04119-01).

## References

1. S. Agrawal, B. Libert, M. Maitra, and R. Titiu. Adaptive simulation security for inner product functional encryption. In *IACR International Conference on Public-Key Cryptography*, pages 34–64. Springer, 2020.
2. J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3:111–128, 2013.
3. P. Ananth and A. Jain. Indistinguishability obfuscation from compact functional encryption. In *Annual Cryptology Conference*, pages 308–326. Springer, 2015.
4. P. Ananth and A. Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In *Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part I*, pages 152–181. Springer, 2017.
5. N. Attrapadung, K. Hamada, D. Ikarashi, R. Kikuchi, T. Matsuda, I. Mishina, H. Morita, and J. C. Schuldt. Adam in private: Secure and fast training of deep neural networks with adaptive moment estimation. *Proceedings on Privacy Enhancing Technologies*, 4:746–767, 2022.
6. M. Bahadori, K. Järvinen, T. Marc, and M. Stopar. Speed reading in the dark: Accelerating functional encryption for quadratic functions with reprogrammable hardware. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 1–27, 2021.
7. 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.
8. J.-A. Choi and K. Lim. Identifying machine learning techniques for classification of target advertising. *ICT Express*, 6(3):175–180, 2020.
9. J. De Spiegeleer, D. B. Madan, S. Reyners, and W. Schoutens. Machine learning for quantitative finance: fast derivative pricing, hedging and fitting. *Quantitative Finance*, 18(10):1635–1643, 2018.
10. D. Dua and C. Graff. UCI machine learning repository, 2017.
11. E. Dufour-Sans, R. Gay, and D. Pointcheval. Reading in the dark: Classifying encrypted digits with functional encryption. *Cryptology ePrint Archive*, 2018.
12. 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.
13. 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.
14. R. Goyal, V. Koppula, and B. Waters. Semi-adaptive security and bundling functionalities made generic and easy. In *Theory of Cryptography: 14th International Conference, TCC 2016-B, Beijing, China, October 31-November 3, 2016, Proceedings, Part II*, pages 361–388. Springer, 2016.
15. T. Graepel, K. Lauter, and M. Naehrig. Ml confidential: Machine learning on encrypted data. In *Information Security and Cryptology–ICISC 2012: 15th International Conference, Seoul, Korea, November 28-30, 2012, Revised Selected Papers 15*, pages 1–21. Springer, 2013.
16. E. Hesamifard, H. Takabi, and M. Ghasemi. Cryptodl: Deep neural networks over encrypted data. *arXiv preprint arXiv:1711.05189*, 2017.
17. E. Hesamifard, H. Takabi, M. Ghasemi, and R. N. Wright. Privacy-preserving machine learning as a service. *Proc. Priv. Enhancing Technol.*, 2018(3):123–142, 2018.
18. N. Koti, M. Pancholi, A. Patra, and A. Suresh. Swift: Super-fast and robust privacy-preserving machine learning. In *USENIX Security Symposium*, pages 2651–2668, 2021.

19. N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma. Cryptflow: Secure tensorflow inference. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 336–353. IEEE, 2020.
20. 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.
21. D. Ligier, S. Carpov, C. Fontaine, and R. Sirdey. Information leakage analysis of inner-product functional encryption based data classification. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 303–3035. IEEE, 2017.
22. D. Ligier, S. Carpov, C. Fontaine, and R. Sirdey. Privacy preserving data classification using inner-product functional encryption. In *ICISSP*, pages 423–430, 2017.
23. Y. Lindell. How to simulate it–a tutorial on the simulation proof technique. *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, pages 277–346, 2017.
24. C. Liu, Z. L. Jiang, X. Zhao, Q. Chen, J. Fang, D. He, J. Zhang, and X. Wang. Efficient and privacy-preserving logistic regression scheme based on leveled fully homomorphic encryption. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2022.
25. T. Marc, M. Stopar, J. Hartman, M. Bizjak, and J. Modic. Privacy-enhanced machine learning with functional encryption. In *European Symposium on Research in Computer Security*, pages 3–21. Springer, 2019.
26. A. Miklosik and N. Evans. Impact of big data and machine learning on digital transformation in marketing: A literature review. *IEEE Access*, 8:101284–101292, 2020.
27. P. Mohassel and P. Rindal. Aby3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 35–52, 2018.
28. P. Mohassel and Y. Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*, pages 19–38. IEEE, 2017.
29. G. Musciagna. Functional encryption for higher degree polynomials assuming multilinear maps. Master's thesis, ETH Zürich, Switzerland, 2021.
30. P. Panzade and D. Takabi. Towards faster functional encryption for privacy-preserving machine learning. In *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 21–30. IEEE, 2021.
31. A. Patra, T. Schneider, A. Suresh, and H. Yalame. Aby2. 0: Improved mixed-protocol secure two-party computation. In *USENIX Security Symposium*, pages 2165–2182, 2021.
32. A. Patra and A. Suresh. Blaze: Blazing fast privacy-preserving machine learning. In *Proceedings 2020 Network and Distributed System Security Symposium*. Internet Society, 2020.
33. A. Qayyum, J. Qadir, M. Bilal, and A. Al-Fuqaha. Secure and robust machine learning for healthcare: A survey. *IEEE Reviews in Biomedical Engineering*, 14:156–180, 2020.
34. M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar. Chameleon: A hybrid secure computation framework for machine learning applications. In *Proceedings of the 2018 on Asia conference on computer and communications security*, pages 707–721, 2018.
35. 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.
36. F. Rundo, F. Trenta, A. L. Di Stallo, and S. Battiato. Machine learning for quantitative finance applications: A survey. *Applied Sciences*, 9(24):5574, 2019.
37. 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.
38. A. Saxena and S. Chandra. *Artificial intelligence and machine learning in healthcare*. Springer, 2021.
39. S. Tan, B. Knott, Y. Tian, and D. J. Wu. Cryptgpu: Fast privacy-preserving machine learning on the gpu. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1021–1038. IEEE, 2021.
40. 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.
41. S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin. Falcon: Honest-majority maliciously secure framework for private deep learning. *Proceedings on Privacy Enhancing Technologies*, 2021.
42. H. Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In *Theory of Cryptography: 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, pages 206–233. Springer, 2017.

43. H. Wee. Functional encryption for quadratic functions from k-lin, revisited. In *Theory of Cryptography Conference*, pages 210–228. Springer, 2020.
44. 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.
45. J. Zhao, R. Mortier, J. Crowcroft, and L. Wang. Privacy-preserving machine learning based data analytics on edge devices. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 341–346, 2018.

# A Functional Encryption and Its Security

In this section, we describe the definitions of private-key and public-key functional encryption schemes and their security models.

**Definition 2 (Private-key functional encryption).** *Let $\mathcal{F}$ be a function space, and $\mathcal{M}$ a message space. Then, a private key functional encryption scheme for $\mathcal{F}$ with $\mathcal{M}$ is composed of four probabilistic polynomial-time (PPT) algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

- $\mathsf{Setup}(\lambda, \mathcal{F})$*: For the security parameter $\lambda$, it outputs the master secret key* $\mathsf{msk}$ *and the public parameter* $\mathsf{pp}$.
- $\mathsf{KeyGen}(\mathsf{msk}, f \in \mathcal{F}, \mathsf{pp})$ : *For* $\mathsf{msk}$ *and a function $f$ from $\mathcal{F}$, it outputs a functional key* $\mathsf{fk}_f$.
- $\mathsf{Enc}(\mathsf{msk}, m \in \mathcal{M}, \mathsf{pp})$ : *For* $\mathsf{msk}$ *and a message $m$ from $\mathcal{M}$, it outputs a ciphertext* $\mathsf{ct}_m$.
- $\mathsf{Dec}(\mathsf{ct}_m, \mathsf{fk}_f, \mathsf{pp})$ : *For* $\mathsf{ct}_m$ *and* $\mathsf{fk}_f$*, it outputs a value $\alpha$.*

*A private-key functional encryption scheme* $\mathsf{skFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to be correct if for every security parameter $\lambda$,* $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(\lambda, \mathcal{F})$*, for all $m \in \mathcal{M}$ and $f \in \mathcal{F}$,* $\mathsf{fk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f, \mathsf{pp})$*, and* $\mathsf{ct}_m \leftarrow \mathsf{Enc}(\mathsf{msk}, m, \mathsf{pp})$*,* $\mathsf{Dec}(\mathsf{ct}_m, \mathsf{fk}_f, \mathsf{pp}) = f(m)$ *with all but a negligible probability.*

**Definition 3 (Semi-adaptive Security of private-key FE [14, 29, 43]).** *A private-key functional encryption scheme* $\mathsf{skFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *for $\mathcal{F}$ with $\mathcal{M}$ is semi-adaptively secure if for every PPT adversary $\mathcal{A}$, there is a negligible function* $\mathsf{neg}(\cdot)$ *such that*

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = |\Pr[\mathsf{G}(\mathcal{A}, 0) = 1] - \Pr[\mathsf{G}(\mathcal{A}, 1) = 1]| \leq \mathsf{neg}(\lambda) \tag{3}$$

*for $\lambda \in \mathbb{N}$, where $\mathsf{G}(\mathcal{A}, b)$ for $b \in \{0, 1\}$ is a semi-adaptive security game between $\mathcal{A}$ and a challenger defined as follows.*

1. *(Setup Phase) The challenger samples* $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(\lambda, \mathcal{F})$ *and gives* $\mathsf{pp}$ *to $\mathcal{A}$.*
2. *(Challenge Phase) $\mathcal{A}$ submits $\big((m_{0,1}, \ldots, m_{0,T}), (m_{1,1}, \ldots, m_{1,T})\big)$, where $m_{0,i}$, $m_{1,i} \in \mathcal{M}$ for all $i \in [1, T]$, and the challenger returns* $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{msk}, m_{b,i})$ *for all $i \in [1, T]$ and a randomly chosen $b \in \{0, 1\}$.*
3. *(Query Phase) $\mathcal{A}$ queries the challenger with $f \in \mathcal{F}$ such that $f(m_{0,i}) = f(m_{1,i})$ for all $i \in [1, T]$. For each $f$, the challenger returns* $\mathsf{fk}_f$.
4. *(Output Phase) $\mathcal{A}$ outputs a bit $b'$ as an output of the game.*

**Definition 4.** *We say that a private key* $\mathsf{FE}$ *is $Q$-ciphertext bounded when the scheme is secure for adversaries requesting $Q$-ciphertexts.*

**Definition 5 (Public-key functional encryption).** *Let $\mathcal{F}$ be a function space, and $\mathcal{M}$ a message space. Then, a private key functional encryption scheme,* $\mathsf{pkFE}$*, for $\mathcal{F}$ with $\mathcal{M}$ is composed of four PPT algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

- $\mathsf{Setup}(\lambda, \mathcal{F})$*: For the security parameter $\lambda$, it outputs the master public/secret key pair* $(\mathsf{pk}, \mathsf{msk})$.
- $\mathsf{KeyGen}(\mathsf{msk}, f \in \mathcal{F})$ : *For* $\mathsf{msk}$ *and a function $f$ from $\mathcal{F}$, it outputs a functional key* $\mathsf{fk}_f$.
- $\mathsf{Enc}(\mathsf{pk}, m \in \mathcal{M})$ : *For* $\mathsf{pk}$ *and a message $m$ from $\mathcal{M}$, it outputs a ciphertext* $\mathsf{ct}_m$.
- $\mathsf{Dec}(\mathsf{pk}, \mathsf{ct}_m, \mathsf{fk}_f)$ : *For* $\mathsf{ct}_m$ *and* $\mathsf{fk}_f$*, it outputs a value $\alpha$.*

*A public-key functional encryption scheme* $\mathsf{pkFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to be correct if for every security parameter* $\lambda$, $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\lambda, \mathcal{F})$, *for all* $m \in \mathcal{M}$ *and* $f \in \mathcal{F}$, $\mathsf{fk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$, *and* $\mathsf{ct}_m \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$, $\mathsf{Dec}(\mathsf{pk}, \mathsf{ct}_m, \mathsf{fk}_f) = f(m)$ *with all but a negligible probability.*

**Definition 6 (Adaptive simulation-based security of public-key FE [1]).** *A public-key functional encryption scheme* $\mathsf{pkFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *for* $\mathcal{F}$ *with* $\mathcal{M}$ *is adaptively simulation secure if for every PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, *there exists a PPT simulator* $\mathcal{S} = (\mathsf{Setup}^{\mathcal{S}}, \mathsf{KeyGen}_0^{\mathcal{S}}, \mathsf{Enc}^{\mathcal{S}}, \mathsf{KeyGen}_1^{\mathcal{S}})$ *such that the* Real *and the* Ideal *experiments, defined as follows, are computationally indistinguishable.*

- *In the* Real *experiment:*
  1. $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\lambda, \mathcal{F})$
  2. $(m^*, st) \leftarrow \mathcal{A}_1^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{pk})$
  3. $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m^*)$
  4. $\alpha \leftarrow \mathcal{A}_2^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{pk}, \mathsf{ct}, st)$
  5. *Output* $(m^*, \alpha)$
- *In the* Ideal *experiment:*
  1. $(\mathsf{pk}^{\mathcal{S}}, \mathsf{msk}^{\mathcal{S}}) \leftarrow \mathsf{Setup}^{\mathcal{S}}(\lambda, \mathcal{F})$
  2. $(m^*, st) \leftarrow \mathcal{A}_1^{\mathsf{KeyGen}_0^{\mathcal{S}}(\mathsf{msk}^{\mathcal{S}}, \cdot)}(\mathsf{pk}^{\mathcal{S}})$
  3. *Let* $\mathcal{V} = \left\{(f_i, f_i(m^*), fk_{f_i})\right\}_{i=1}^{k}$ *where* $\{f_i \in \mathcal{F}\}_{i=1}^{k}$ *are the functions for which the adversary requests their corresponding keys* $\{fk_{f_i}\}_{i=1}^{k}$
  4. $\mathsf{ct}^* \leftarrow \mathsf{Enc}^{\mathcal{S}}(\mathsf{pk}^{\mathcal{S}}, \mathsf{msk}^{\mathcal{S}}, \mathcal{V}, 1^{|m^*|})$
  5. $\alpha \leftarrow \mathcal{A}_2^{\mathsf{KeyGen}_1^{\mathcal{S}}(\mathsf{msk}^{\mathcal{S}}, \cdot)}(\mathsf{pk}^{\mathcal{S}}, \mathsf{ct}^*, st)$
  6. *Output* $(m^*, \alpha)$

**Definition 7 (Compact FE, Adaptation from [3]).** *We say that a* $Q$*-ciphertext bounded private key FE scheme* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *for* $\mathcal{F}$ *with* $\mathcal{M}$ *is compact if for all* $\lambda \in \mathbb{N}$, *the running time of the encryption algorithm* Enc *is a polynomial with respect to parameters* $\lambda, Q$ *and* $m$, *where* $m \in \mathcal{M}$.

*Remark 5.* Even though the function space $\mathcal{F}$ is used as the input of the Setup algorithm in the FE definition, we employ more specific notations instead of $\mathcal{F}$.

## B Hardness Assumptions

This section provides the hardness assumption used in this paper. We consider groups of order $N \cdot \phi(N)$ where $p, q$ are large primes such that factoring $N$ is hard. In the following section, we assume that the following assumptions hold.

**Definition 8 (Bilinear map).** *Let* $\mathbb{G}_a, \mathbb{G}_b$, *and* $\mathbb{G}_T$ *be groups. We say that* $e : \mathbb{G}_a \times \mathbb{G}_b \to \mathbb{G}_T$ *is a bilinear map if* $e$ *satisfies the following:*

1. $\mathbb{G}_a, \mathbb{G}_b, \mathbb{G}_T$ *are groups of the same order that satisfy the discrete logarithm problem is hard on each group.*
2. *For any* $g_a \in \mathbb{G}_a, g_b \in \mathbb{G}_b$ *and* $x_a, x_b \in \mathbb{Z}$, *it holds that*
$$e(g_a^{x_a}, g_b^{x_b}) = e(g_a, g_b)^{x_a x_b}.$$
3. *If* $g_a, g_b$ *are generators of* $\mathbb{G}_a, \mathbb{G}_b$ *respectively, then* $e(g_a, g_b)$ *is a generator of* $\mathbb{G}_T$.

**Definition 9 (DCR assumption).** *Let* $p, q$ *be prime numbers and* $N = pq$. *The decision composite residuosity (DCR) assumption is that the following distributions are computationally indistinguishable:*
$$Dist_1 : \{z = z_0^N \bmod N^2 \mid z_0 \leftarrow \mathbb{Z}_N^*\}$$
$$Dist_2 : \{z \leftarrow \mathbb{Z}_{N^2}^*\},$$
*where* $\mathbb{Z}_N^*$ *is a multiplicative group of* $\mathbb{Z}_N$.

**Definition 10 (DDH assumption).** *For $N = pq$ with primes $p$ and $q$, let $\mathbb{G}$ be a group of order $N \cdot \phi(N)$ and $g$ a generator of $\mathbb{G}$.*

*The decisional Diffie-Hellman (DDH) assumption over $\mathbb{G}$ is that the following distributions are computationally indistinguishable:*

$$Dist_1 : \{(g, g^x, g^y, g^{xy}) \mid x, y \leftarrow \mathbb{Z}_{N\cdot\phi(N)}\}$$
$$Dist_2 : \{(g, g^x, g^y, g^z) \mid x, y, z \leftarrow \mathbb{Z}_{N\cdot\phi(N)}\}.$$

**Definition 11 ($\chi$-MDDH assumption).** *For $N = pq$ with primes $p$ and $q$, let $\mathbb{G}$ be a group of order $N \cdot \phi(N)$ and $g$ a generator of $\mathbb{G}$.*

*Let $\chi$ be a distribution which returns a vector over $\mathbb{G}^2$. Then, $\chi$-MDDH assumption holds on $\mathbb{G}$ with a generator $g$ if any PPT adversary $\mathcal{A}$ cannot distinguish the following distributions.*

$$Dist_1 : \{[\mathbf{a}]_g, [\mathbf{a} \cdot w]_g : \mathbf{A} \leftarrow \chi, w \leftarrow \mathbb{Z}_{N\cdot\phi(N)}\}$$
$$Dist_2 : \{[\mathbf{a}]_g, [\mathbf{u}]_g : \mathbf{a} \leftarrow \chi, \mathbf{u} \leftarrow \mathbb{G}^2\}.$$

There is a simple reduction from DDH to $\chi$-MDDH, so it holds that DDH$\leq \chi$-MDDH. When a bilinear map $e : \mathbb{G}_a \times \mathbb{G}_b \to \mathbb{Z}^*_{N^2}$ is given, the DDH problem over two groups would be considered simultaneously. We describe the problem, the so-called Bilateral 2-LIN assumption, which is used in security proof for bilinear map-based schemes [43].

**Definition 12 (Bilateral 2-LIN assumption).** *Let $\mathbb{G}_a, \mathbb{G}_b$ be groups of order $N \cdot \phi(N)$ and $e : \mathbb{G}_a \times \mathbb{G}_b \to \mathbb{Z}^*_{N^2}$ a bilinear group. We say bilateral 2-LIN assumption holds on groups $\mathbb{G}_a$ and $\mathbb{G}_b$ if*

$$\{[x]_{g_a}, [y]_{g_a}, [xy]_{g_a}, [x]_{g_b}, [y]_{g_b}, [xy]_{g_b}\} \approx \{[x]_{g_a}, [y]_{g_a}, [z]_{g_a}, [x]_{g_b}, [y]_{g_b}, [z]_{g_b}\},$$

*where $x, y, z \leftarrow \mathbb{Z}_{N\cdot\phi(N)}$, $g_a \in \mathbb{G}_a$ and $g_b \in \mathbb{G}_b$.*

# C Correctness of LinEnc FE Schemes

## C.1 Correctness of LinEnc-IPFE

*Correctness* (of LinEnc-IPFE in Fig. 4). To this end, it suffices to show that $\mathsf{Dec}_{\mathsf{IPFE}}(\mathsf{pp}, \mathsf{fk}_{\mathbf{y}}, \mathsf{ct}_{\mathbf{x}}) = [\mathbf{y}^T \cdot \mathbf{x}]_g$. It can be easily proven by examining the decryption procedure. Specifically, we can observe that:

$$\begin{aligned}
\mathsf{fk}_{\mathbf{y}}^{\mathsf{ct}_{\mathbf{x}}} &= \left[ \begin{pmatrix} -\mathbf{U}^T \cdot \mathbf{y} \\ \mathbf{y} \end{pmatrix}^T \cdot \mathbf{D} \cdot \mathsf{ct}_{\mathbf{x}} \right]_g \\
&= \left[ \begin{pmatrix} -\mathbf{U}^T \cdot \mathbf{y} \\ \mathbf{y} \end{pmatrix}^T \cdot \mathbf{D} \cdot \mathbf{D}^{\perp} \cdot \widetilde{\mathbf{r}} + \begin{pmatrix} -\mathbf{U}^T \cdot \mathbf{y} \\ \mathbf{y} \end{pmatrix}^T \mathbf{D} \cdot \mathbf{D}^{-1} \cdot \begin{pmatrix} r \cdot \mathbf{a} \\ \mathbf{x} + r \cdot \mathbf{U} \cdot \mathbf{a} \end{pmatrix} \right]_g \\
&= \left[ \begin{pmatrix} -\mathbf{U}^T \cdot \mathbf{y} \\ \mathbf{y} \end{pmatrix}^T \cdot \begin{pmatrix} r \cdot \mathbf{a} \\ \mathbf{x} + r \cdot \mathbf{U} \cdot \mathbf{a} \end{pmatrix} \right]_g \\
&= \left[ (-\mathbf{U}^T \cdot \mathbf{y})^T \cdot r \cdot \mathbf{a} + \mathbf{y}^T \cdot \mathbf{x} + r \cdot \mathbf{y}^T \cdot \mathbf{U} \cdot \mathbf{a} \right]_g = \left[ \mathbf{y}^T \cdot \mathbf{x} \right]_g
\end{aligned}$$

This completes the correctness.

### C.2 Correctness of LinEnc-QFE

*Correctness* (of LinEnc-QFE in Fig. 5). We first observe the term $\mathbf{e}_1 = [(\mathbf{D}_0 \otimes \mathbf{D}_1)^T \cdot \mathbf{c}_f]_g^{(\mathsf{ct}_0 \otimes \mathsf{ct}_1)}$:

$$\begin{aligned}
\mathbf{e}_1 &= [(\mathbf{D}_0 \otimes \mathbf{D}_1)^T \cdot \mathbf{c}_f]_g^{(\mathsf{ct}_0 \otimes \mathsf{ct}_1)} = [\langle (\mathbf{D}_0 \otimes \mathbf{D}_1)^T \cdot \mathbf{c}_f, (\mathsf{ct}_0 \otimes \mathsf{ct}_1) \rangle]_g \\
&= [\mathbf{c}_f^T \cdot ((\mathbf{D}_0 \cdot \mathsf{ct}_0) \otimes (\mathbf{D}_1 \cdot \mathsf{ct}_1))]_g = \left[\mathbf{c}_f^T \cdot \left((c \cdot \overline{\mathbf{x}} + \mathbf{V} \cdot \mathbf{r}_0) \otimes (\overline{\mathbf{x}} + \mathbf{W} \cdot \mathbf{r}_1)\right)\right]_g \\
&= \left[\mathbf{c}_f^T \cdot \left(c \cdot \overline{\mathbf{x}} \otimes \overline{\mathbf{x}} + c \cdot \overline{\mathbf{x}} \otimes \mathbf{W} \cdot \mathbf{r}_1 + \mathbf{V} \cdot \mathbf{r}_0 \otimes \overline{\mathbf{x}} + \mathbf{V} \cdot \mathbf{r}_0 \otimes \mathbf{W} \cdot \mathbf{r}_1\right)\right]_g \\
&= \left[\mathbf{c}_f^T \cdot \left(c \cdot \overline{\mathbf{x}} \otimes \overline{\mathbf{x}} + (\mathbf{V} \cdot \mathbf{r}_0) \otimes \overline{\mathbf{x}} + (c \cdot \overline{\mathbf{x}} + \mathbf{V} \cdot \mathbf{r}_0) \otimes (\mathbf{W} \cdot \mathbf{r}_1)\right)\right]_g \\
&= \left[\mathbf{c}_f^T \cdot \left(c \cdot \overline{\mathbf{x}} \otimes \overline{\mathbf{x}} + (\mathbf{V} \otimes \mathbf{I}_{\ell+1}) \cdot (\mathbf{r}_0 \otimes \overline{\mathbf{x}}) + (\mathbf{I}_{\ell+1} \otimes \mathbf{W}) \cdot ((c \cdot \overline{\mathbf{x}} + \mathbf{V} \cdot \mathbf{r}_0) \otimes \mathbf{r}_1)\right)\right]_g \\
&= \left[ c \cdot \mathbf{c}_f^T \cdot (\overline{\mathbf{x}} \otimes \overline{\mathbf{x}}) + \underbrace{\begin{pmatrix} \mathbf{r}_0 \otimes \overline{\mathbf{x}} \\ (c \cdot \overline{\mathbf{x}} + \mathbf{V} \cdot \mathbf{r}_0) \otimes \mathbf{r}_1 \end{pmatrix}^T}_{\mathbf{h}^T} \underbrace{\begin{pmatrix} \mathbf{V}^T \otimes \mathbf{I}_{\ell+1} \\ \mathbf{I}_{\ell+1} \otimes \mathbf{W}^T \end{pmatrix} \cdot \mathbf{c}_f}_{\mathsf{fk}_0 \cdot \mathbf{c}_f} \right]_g .
\end{aligned}$$

On the other hand, by the correctness of $\mathsf{Dec}_{\mathsf{IPFE}}$, we obtain that

$$\mathbf{e}_2 = [\mathsf{Dec}_{\mathsf{IPFE}}(\mathsf{pp}_{\mathsf{IPFE}}, \mathsf{fk}_{\mathsf{IPFE}}, \mathsf{ct}_{\mathsf{IPFE}})]_g = [\mathbf{h}^T \cdot \mathsf{fk}_0 \cdot \mathbf{c}_f]_g.$$

The ratio $\mathbf{e}_1/\mathbf{e}_2$ is then identical to

$$[c \cdot \mathbf{c}_f^T \cdot (\overline{\mathbf{x}} \otimes \overline{\mathbf{x}})]_g = (1+N)^{\mathbf{c}_f^T \cdot (\overline{\mathbf{x}} \otimes \overline{\mathbf{x}})} = (1+N)^{f(\mathbf{x})}.$$

Hence, the $\log_{(1+N)}(\mathbf{e}_1/\mathbf{e}_2)$ outputs $f(\mathbf{x})$ mod $N$. It directly implies that $\mathsf{Dec}_{\mathsf{QFE}}(\mathsf{pp}, \mathsf{fk}_f, \mathsf{ct}) = f(\mathbf{x})$ as long as $|f(\mathbf{x})| < N/2$.

## D Functional Encryption for Linear/Quadratic Polynomials due to Musciagna

We overview FE schemes for linear/quadratic polynomials, proposed by Musciagna [29]. They generalize the technique for building the FE for quadratic polynomials assuming $k$-LIN due to Wee [43].

Both constructions in this section achieve the semi-adaptive simulation based security (SA-SIM), where it is more stronger security, under several assumptions in Supplementary Material B. The semi-adaptive game introduced by [42], where the adversary is restricted to perform functional key queries to the $\mathsf{KeyGen}$ oracle only after it sees the challenge ciphertext.

**Notation.** Suppose that 2-linear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is given. For appropriate sizes of matrices $[\mathbf{M}_i]_i \in \mathbb{G}_i$, $e([\mathbf{M}_1]_1, [\mathbf{M}_2]_2)$ is defined by $[[\mathbf{M}_1]_1 \cdot [\mathbf{M}_2]_2]_T$ by exploiting $e$.

Let $\mathsf{GGen}(1^\lambda)$ be a PPT algorithm which takes as input the security parameter $\lambda$, and returns $(\mathbb{G}, g, p)$. Here, $\mathbb{G}$ is a group of composite order $p \gg 2^\lambda$ and $g$ is its generator, which has $\lambda$-bit hardness of the discrete log problem. We also use a bracket notation $[x]$ to denote $g^x$. We note that while the order $p$ corresponds to a composite number $\Delta = N \cdot \phi(N)$, an order of hybrid protocol in the main body, we use the notation $p$ to follow the same description of the scheme [29].

### D.1 Functional Encryption for Linear Polynomials

In this section, we provide a brief overview of the simplified IPFE proposed by Musciagna in [29], without security proofs. For more detailed information, we refer readers to the original paper [29].

We begin by describing Mus.IPFE, a FE scheme for inner products (IPFE) proposed by Musciagna [29], in Fig. 10. The security of Mus.IPFE is based on the hardness of the $\chi$-MDDH assumption (Definition 11).

*Correctness* (of Fig. 10). From the decryption procedure, we observe that

$$\begin{aligned}[\mathsf{ct}^T \cdot \mathsf{sk}_{\mathbf{y}}]_T &= \left[ \begin{pmatrix} r \cdot \mathbf{a} \\ \mathbf{x} + r \cdot \mathbf{U} \cdot \mathbf{a} \end{pmatrix}^T \cdot \begin{pmatrix} -\mathbf{U}^T \cdot \mathbf{y} \\ \mathbf{y} \end{pmatrix} \right]_T \\ &= [(r \cdot \mathbf{a})^T (-\mathbf{U} \cdot \mathbf{y}) + \mathbf{x}^T \mathbf{y} + (r \cdot \mathbf{U} \cdot \mathbf{a})^T \cdot \mathbf{y}]_T \\ &= [\mathbf{x}^T \cdot \mathbf{y}]_T \end{aligned}$$

We omit the security proof of Mus.IPFE. For more details, we refer the original paper [29, Section 4].

**Theorem 4 (Security of Mus.IPFE [29]).** *This scheme provides the semi-adaptive simulation based security if the $\mathcal{D}_1$-MDDH assumption holds on $\mathbb{G}_1$.*

### D.2 FE for Quadratic Polynomials

We will now describe FE scheme for quadratic polynomials (QFE) in Fig. 11. The goal of QFE is to compute a vector of the form $(\mathbf{x} \otimes \mathbf{y})^T \cdot \mathbf{f}$ for some $\mathbf{x} \in \mathbb{Z}_p^n, \mathbf{y} \in \mathbb{Z}_p^n$, and $\mathbf{f} \in \mathbb{Z}_p^{n^2}$.

*Correctness* (of Fig. 11). We first observe the term $[(\mathsf{ct}_{\mathbf{x}} \otimes \mathsf{ct}_{\mathbf{y}}) \cdot \mathbf{f}]_T$:

$$\begin{aligned} &[(\mathsf{ct}_{\mathbf{x}} \otimes \mathsf{ct}_{\mathbf{y}}) \cdot \mathbf{f}]_T \\ &= [(\mathbf{x} \otimes \mathbf{y} + \mathbf{x} \otimes (\mathbf{V} \cdot \mathbf{s}) + (\mathbf{U} \cdot \mathbf{r}) \otimes \mathbf{y} + (\mathbf{U} \cdot \mathbf{r}) \otimes (\mathbf{V} \cdot \mathbf{s}))^T \cdot \mathbf{f}]_T \\ &= [(\mathbf{x} \otimes \mathbf{y} + (\mathbf{U} \cdot \mathbf{r}) \otimes \mathbf{y} + \mathsf{ct}_{\mathbf{x}} \otimes (\mathbf{V} \cdot \mathbf{s}))^T \cdot \mathbf{f}]_T \\ &= [(\mathbf{x} \otimes \mathbf{y} + (\mathbf{U} \otimes \mathbf{I}_n)(\mathbf{r} \otimes \mathbf{y}) + (\mathbf{I}_n \otimes \mathbf{V})(\mathsf{ct}_{\mathbf{x}} \otimes \mathbf{s}))^T \cdot \mathbf{f}]_T \\ &= [(\mathbf{x} \otimes \mathbf{y})^T \cdot \mathbf{f}]_T + \left[ \underbrace{\begin{pmatrix} \mathbf{r} \otimes \mathbf{y} \\ \mathsf{ct}_{\mathbf{x}} \otimes \mathbf{s} \end{pmatrix}^T}_{\mathbf{h}^T} \cdot \underbrace{\begin{pmatrix} \mathbf{U}^T \otimes \mathbf{I}_n \\ \mathbf{I}_n \otimes \mathbf{V}^T \end{pmatrix} \cdot \mathbf{f}}_{\mathbf{M} \cdot \mathbf{f}} \right]_T . \end{aligned}$$

On the other hand, by the correctness of $\mathsf{Mus.Dec}_{\mathsf{IPFE}}$, we obtain that

$$\mathsf{Mus.Dec}_{\mathsf{IPFE}}(\mathsf{mpk}_{\mathsf{IPFE}}, \mathsf{ct}_{\mathsf{IPFE}}, \mathsf{sk}_{\mathsf{IPFE}}) = [\mathbf{h}^T \cdot \mathbf{M} \cdot \mathbf{f}]_T.$$

Hence, the correctness must hold since we obtain $[(\mathbf{x} \otimes \mathbf{y})^T \cdot \mathbf{f}]_T$.

As the above, we also omit the security proof of Mus.QFE. For more details, we refer an original paper [29, Section 5].

**Theorem 5 (Security of Mus.QFE).** *Mus.QFE is semi-adaptive simulation based secure if the bilateral 2-LIN assumptions holds in $(\mathbb{G}_1, \mathbb{G}_2)$ and the Mus.IPFE is semi-adaptive simulation based secure.*

## E Deferred Security Proof

The purpose of this section is to present the security proof of Theorem 2. We will mainly focus on the security proof of Mus.QFE, as the technique is identical to the proof for Mus.IPFE throughout this section.

**Strategy.** To prove the theorem, we convert from the Mus.QFE scheme [29] into our LinEnc-QFE scheme in Section 3 under semi-adaptive security game. That is, we will show that

$$\mathsf{Adv}_{\mathsf{LinEnc\text{-}QFE}} \leq \mathsf{Adv}_{\mathsf{Mus.QFE}}$$

which finishes proof. This is because that [29] proved that the scheme achieves the semi-adaptive security under the cryptographic hard assumptions.

We make use of an assumption that there exists a secure bilinear map oracle $\mathcal{M}$ capable of computing

$$\mathcal{M}(g_1^x, g_2^y) = g_T^{xy}$$

for some groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ with generators $g_1, g_2, g_T \in \mathbb{Z}_{N^2}$. The only required thing for proof is that the whole group $\mathbb{G}_i$ is an order $p$ group. Thus we skip the group description. It is important to note that efficiency, in terms of storage cost, is not a requirement for the bilinear maps under consideration. Rather, the main concern is security, which makes lookup tables a viable option. The bilinear map oracle will be used for decryption during the security game.

We then show that the LinEnc-QFE scheme is semi-adaptive secure when only $Q$-ciphertext queries are given.

**Lemma 2.** LinEnc-QFE *achieves $Q$-ciphertext bounded semi-adaptive simulation based security under the assumption that the bilateral 2-LIN assumptions holds in $(\mathbb{G}_1, \mathbb{G}_2)$ and the above IPFE is semi-adaptive simulation based secure.*

*Proof of Lemma 2.* Our goal is to demonstrate that if an adversary $\mathcal{A}_{\mathsf{LinEnc\text{-}QFE}}$ can break the semi-adaptive simulation-based security of our scheme, then Mus.QFE can also be broken. To accomplish this, we construct an adversary of Mus.QFE using $\mathcal{A}_{\mathsf{QFE}}$ that breaks the LinEnc-QFE.

Now, we introduce an algorithm, called matrix-vector splitting algorithm (Alg. 2) to bridge between ciphertexts of Mus.QFE and LinEnc-QFE. The algorithm is a core technique of the reduction for the proof.

For ease of presentations, we omit the underlying group, $\mathbb{G}$, and the dimension of given vectors, $L$[7], in the input of each algorithm.

**Algorithm 2** Matrix-vector splitting

**Input:** Vectors $\{\mathbf{v}_i\}_{i\in[1,Q]} \subset \mathbb{G}^L$

**Output:** $[\mathbf{D}] \in \mathbb{G}^{L\times L+Q}$ and $\{\mathbf{d}_i\}_{i\in[1,Q]} \subset \mathbb{Z}_{N\cdot\phi(N)}^{L+Q}$ such that $[\mathbf{D}]^{\mathbf{d}_i} = \mathbf{v}_i$

1: Set $\mathbf{A} \leftarrow \mathcal{D}_{\mathbb{Z}^{L+Q\times L+Q}}$ until $\mathbf{A}$ is invertible.

2: Sample $\widetilde{\mathbf{B}} \leftarrow \mathbb{Z}_{N\cdot\phi(N)}^{L\times L}$ and set

$$[\mathbf{D}] = (\mathbf{v}_1\| \cdots \|\mathbf{v}_Q\|[\widetilde{\mathbf{B}}])^{\mathbf{A}^{-1}} \in \mathbb{G}^{L\times(L+Q)}$$

3: Set $\mathbf{d}_i = \mathbf{A}\cdot\mathbf{e}_i$ for $i \in [1, Q]$.

4: **return** $[\mathbf{D}]$ and $\{\mathbf{d}_i\}$.

Alg. 2 is used to convert ciphertexts of Mus.QFE (resp. Mus.IPFE) into LinEnc-QFE (resp. LinEnc-IPFE). The detailed algorithms for transforming Mus.IPFE and Mus.QFE into LinEnc-IPFE and LinEnc-QFE are given by Fig. 12 and Fig. 13, respectively.

We would like to note that the set of output vectors $\{\mathbf{d}_{\mathbf{x}_i}, \mathbf{d}_{\mathbf{y}_i}, \mathbf{d}_{\mathsf{IPFE},i}\}_i$ obtained from Fig. 13 has the same distribution of $\mathsf{Enc}_{\mathsf{QFE}}$ in Section 3 when $\mathbf{x}_i = \mathbf{y}_i$. Additionally, we note that the terms $\widetilde{\mathsf{KeyGen}}_{\mathsf{QFE}}(\mathbf{f}, \{[\mathbf{D}_0]_1, [\mathbf{D}_1]_2, [\mathbf{D}_{\mathsf{IPFE}}]_2\}, \mathsf{KeyGen}_{\mathsf{QFE}}(\mathbf{f}))$ can serve as functional keys for LinEnc-QFE because it has the same distribution of $\mathsf{KeyGen}_{\mathsf{QFE}}$ as well. As a result, it is able to obtain a set of ciphertexts and a family of functions that serve as legitimate inputs.

Let $\mathsf{Mus.QFE}^{\mathcal{S}}$ (resp. $\mathsf{Mus.IPFE}^{\mathcal{S}}$) be a simulator of Mus.QFE (resp. Mus.IPFE). Since these schemes are semi adaptive simulation based secure scheme, such a simulator exists. In the same vein, Alg. 2 converts the $\mathsf{Mus.QFE}^{\mathcal{S}}$ into a simulator of LinEnc-QFE. We denote it as $\mathsf{LinEnc\text{-}QFE}^{\mathcal{S}}$.

The probability of distinguishing between LinEnc-QFE and $\mathsf{LinEnc\text{-}QFE}^{\mathcal{S}}$ should be non-negligible if not Mus.QFE and its simulator should be distinguished with non-negligible probability. This completes the proof.

□

[7] This notation is independent to the $L$ in the main body.

### E.1 Detailed Proof in Theorem 3

In this section, we provide a detailed proof of $\mathcal{P}_{2,i} \sim \mathcal{P}_{2,i+1}$. For this purpose, we prove the following lemma.

**Lemma 3.** *For any $1 \leq i < \mathcal{E}$, $\mathcal{P}_{2,i}$ and $\mathcal{P}_{2,i+1}$ are computationally indistinguishable under the assumption that* LinEnc-QFE *is $(2\ell+1)$-bounded semi-adaptively secure. The advantage of distinguishing between $\mathcal{D}_{2,i}$ and $\mathcal{D}_{2,i+1}$ is smaller than $\mathsf{Adv}_{\mathsf{LinEnc\text{-}QFE}}$. Furthermore, no one computationally obtains any information about $\bigcirc_{i=1}^{\mathcal{E}} f_{0,i}(\mathbf{x}_0)$ from $\mathsf{E}_{i+1}(h_{0,i,0}(\bigcirc_{t=1}^{i} f_{0,t}(\mathbf{x}_0)))$.*

*Proof of Lemma 3.* Suppose that there exists a PPT adversary $\mathcal{B}_i$ that can distinguish between $\mathcal{P}_{2,i}$ and $\mathcal{P}_{2,i+1}$. Then, we construct an adversary $\mathcal{A}_{\mathsf{LinEnc\text{-}QFE}}$ to break the security game of LinEnc-QFE. Assume that $\mathcal{C}$ is a challenger of the security game of LinEnc-QFE.

1. $\mathcal{C}$ samples $(\mathsf{msk}_{\mathsf{QFE},i+1}, \mathsf{pp}_{\mathsf{QFE},i+1}) \leftarrow \mathsf{Setup}_{\mathsf{QFE}}(\lambda, \ell, 2\ell+1)$ and sends $\mathsf{pp}_{\mathsf{QFE},i+1}$ to $\mathcal{A}_{\mathsf{LinEnc-QFE}}$.
2. $\mathcal{A}_{\mathsf{LinEnc-QFE}}$ proceeds up to steps as follows:
   (a) Sample
   $$\{\mathsf{msk}^{\mathcal{S}}_{\mathsf{pkIPFE}}, \mathsf{pk}^{\mathcal{S}}_{\mathsf{pkIPFE}}\} \leftarrow \mathsf{Setup}^{\mathcal{S}}_{\mathsf{pkIPFE}}(\lambda, 2\ell+1, B_X, B_F),$$
   $$\{\mathsf{msk}_{\mathsf{QFE},t}, \mathsf{pp}_{\mathsf{QFE},t}\} \leftarrow \mathsf{Setup}_{\mathsf{QFE}}(\lambda, \ell, 2\ell+1)$$
   for each $t \in [1, \mathcal{E}] \setminus \{i+1\}$.
   (b) Set $\mathsf{pk}$ as in $\mathcal{P}$ and send it to $\mathcal{B}_i$.
   (c) Sample a pair $(\mathbf{H}_{0,t,0}, \mathbf{H}'_{0,t,0})$ for $t \in [0, \mathcal{E}] \setminus \{i\}$.
   (d) Sample a matrix $\overline{\mathbf{H}}_{0,t,0}$ satisfying $\mathbf{H}'_{0,t,0} \cdot \overline{\mathbf{H}}_{0,t,0} = \mathbf{H}'_{0,t,0} \cdot \mathbf{H}_{0,t,0} \bmod \Delta$ for $0 \leq t < i$.
   (e) Sample a pair $(\mathbf{\Gamma}_{l,0}, \mathbf{\Gamma}^{-1}_{l,0})$ and a simulated ciphertext $\mathsf{Enc}^{\mathcal{S}}_1$ as follows.
   $$\mathsf{Enc}^{\mathcal{S}}_1 \leftarrow \mathsf{Enc}^{\mathcal{S}}_{\mathsf{pkIPFE}}(\mathsf{msk}^{\mathcal{S}}_{\mathsf{pkIPFE}}, \mathsf{pk}^{\mathcal{S}}_{\mathsf{pkIPFE}}, \{h_{0,0,0} \circ \gamma^{-1}_{l,0}, h_{0,0,0}(\mathbf{x})\})$$
3. $\mathcal{B}_i$ queries a set of models $\{F_l\}$ to the $\mathcal{A}_{\mathsf{LinEnc-QFE}}$.
4. $\mathcal{A}_{\mathsf{LinEnc-QFE}}$ computes a functional key of $\widetilde{G}_{0,0,0} = G_{0,0,0} = h_{0,0,0} \circ \gamma^{-1}_{l,0}$ as follows.
   $$\mathsf{fk}_{\mathsf{E}_1 \circ G^{\mathcal{S}}_{0,0,0}} \leftarrow \mathsf{cKeyGen}^{\mathcal{S}}_{\mathsf{pkIPFE},0}(\mathsf{msk}^{\mathcal{S}}_{\mathsf{pkIPFE}}, \mathsf{pk}^{\mathcal{S}}_{\mathsf{pkIPFE}}, \mathsf{msk}_{\mathsf{QFE},1}, \mathsf{pp}_{\mathsf{QFE},1}, G_{0,0,0}, L)$$
5. For every $t < i$, $\mathcal{A}_{\mathsf{LinEnc-QFE}}$ computes functions keys $\mathsf{fk}_{\mathsf{E}_{t+1} \circ G^{\mathcal{S}}_{0,t,0}}$ as follows:
   $$\mathsf{fk}_{\mathsf{E}_{t+1} \circ G^{\mathcal{S}}_{0,t,0}} \leftarrow \mathsf{cKeyGen}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE},t}, \mathsf{pp}_{\mathsf{QFE},t}, \mathsf{msk}_{\mathsf{QFE},t+1}, \mathsf{pp}_{\mathsf{QFE},t+1}, G^{\mathcal{S}}_{0,t,0}, L),$$
   where $G^{\mathcal{S}}_{0,t,0} = \bar{h}_{0,t,0} \circ f_{l,t} \circ h'_{0,t-1,0}$ .
6. For every $t > i, i+1$, $\mathcal{A}_{\mathsf{LinEnc-QFE}}$ computes functions keys $\mathsf{fk}_{\mathsf{E}_{t+1} \circ G_{0,t,0}}$ as follows:
   $$\mathsf{fk}_{\mathsf{E}_{t+1} \circ G_{0,t,0}} \leftarrow \mathsf{cKeyGen}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE},t}, \mathsf{pp}_{\mathsf{QFE},t}, \mathsf{msk}_{\mathsf{QFE},t+1}, \mathsf{pp}_{\mathsf{QFE},t+1}, G_{0,t,0}, L)$$
   where $G_{0,t,0} = h_{0,t,0} \circ f_{l,t} \circ h'_{0,t-1,0}$.
7. To compute $\mathsf{fk}_{\mathsf{E}_{i+1} \circ G^{\mathcal{S}}_{0,i,0}}, \mathsf{fk}_{\mathsf{E}_{i+2} \circ G_{0,i+1,0}}$, $\mathcal{A}_{\mathsf{LinEnc-QFE}}$ sends $(\mathbf{H}_{0,i,0}, \bar{\mathbf{H}}_{0,i,0})$ to $\mathcal{C}$.
8. $\mathcal{C}$ randomly chooses $h_{0,i,0} \in \{\mathbf{H}_{0,i,0}, \overline{\mathbf{H}}_{0,i,0}\}$ and returns $\mathsf{ct} \leftarrow \mathsf{E}_{i+1}(h_{0,i,0})$.
9. $\mathcal{A}_{\mathsf{LinEnc-QFE}}$ computes
   $$\mathsf{fk}_{\mathsf{E}_{i+1} \circ G^{\mathcal{S}}_{0,i,0}} \leftarrow \mathsf{KeyGen}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE},i}, \mathsf{pp}_{\mathsf{QFE},i}, \mathsf{ct} \cdot \mathbf{M}_{f_{0,i}} \cdot (\mathbf{H}'_{0,i-1,0} \otimes \mathbf{H}'_{0,i-1,0}), L).$$
10. $\mathcal{A}_{\mathsf{LinEnc-QFE}}$ requests a function query for $G_{0,i+1,0} = h_{0,i+1,0} \circ f_{0,i} \circ h'_{0,i,0}$.
11. $\mathcal{C}$ computes a functional key $\mathsf{fk}_{\mathsf{E}_{i+2} \circ G_{0,i+1,0}}$ as follows
    $$\mathsf{fk}_{\mathsf{E}_{i+2} \circ G_{0,i+1,0}} \leftarrow \mathsf{cKeyGen}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE},i+1}, \mathsf{pp}_{\mathsf{QFE},i+1}, \mathsf{msk}_{\mathsf{QFE},i+2}, \mathsf{pp}_{\mathsf{QFE},i+2}, G_{0,i+1,0}, L)$$
    and sends it to $\mathcal{A}_{\mathsf{LinEnc-QFE}}$. Last, $\mathcal{A}_{\mathsf{LinEnc-QFE}}$ transmits it to $\mathcal{B}_i$.
12. $\mathcal{B}_i$ returns $\beta \in \{i, i+1\}$ to $\mathcal{A}_{\mathsf{LinEnc-QFE}}$.

13. If $\beta = i+1$, $\mathcal{A}_{\mathsf{Enc}_{2,i+1}}$ outputs $\overline{\mathbf{H}}_{0,i,0}$. Otherwise, returns $\mathbf{H}_{0,i,0}$.

By construction, $\mathsf{fk}_{\mathsf{E}_{i+1} \circ G^{\mathcal{S}}_{0,i,0}}$ is identical to an output of

$$\mathsf{cKeyGen}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE},i}, \mathsf{pp}_{\mathsf{QFE},i}, \mathsf{msk}_{\mathsf{QFE},i+1}, \mathsf{pp}_{\mathsf{QFE},i+1}, h_{0,i,0} \circ f_{0,i} \circ h_{0,i-1,0}, L).$$

Thus, this game directly implies that

$$\mathsf{Adv}_{i,i+1} \leq \mathsf{Adv}_{\mathsf{LinEnc\text{-}QFE}} \text{ for any } i,$$

where $\mathsf{Adv}_{i,i+1}$ stands for denoting the advantage to distinguish two protocols $\mathcal{P}_{2,i}$ and $\mathcal{P}_{2,i+1}$.

Analogs to the case of $\overline{\mathbf{H}}_{0,0,0}$, we may assume that the adversary $\mathcal{A}$ knows $\mathbf{H}'_{0,i,0}$ and $\overline{\mathbf{H}}_{0,i,0} \cdot \bigcirc_{t=1}^{i} f_{l,t}(\mathbf{x})$ exactly. However the possible number of $\overline{\mathbf{H}}_{0,i,0}$ is larger than $2^{\lambda}$. Hence, no one computationally recovers the $\bigcirc_{t=1}^{i} f_{l,t}(\mathbf{x}_0)$ for each $i$. □

## F Security Proof of Theorem 3

*Proof of Theorem 3.* In the security of the protocol $\mathcal{P}$ in Fig. 9, the adversary is provided with a distribution

$$\mathcal{D} = \mathsf{REAL}_{\mathcal{P},\mathcal{A}}(\mathbf{x}_0, \{\mathbf{x}_j\}_{j=1}^{J}, \{F_l\}_{l=0}^{\zeta}, \lambda).$$

We then consider $(\mathcal{E}+1)$-modified protocols denoting $\mathcal{P}_1$ and $\mathcal{P}_{2,i}$ for $i \in [1, \mathcal{E}]$ such that

$$\mathcal{P}_{2,\varepsilon} = \mathsf{IDEAL}_{\mathcal{S}}(\mathbf{x}_0, \{\mathbf{x}_j\}_{j=1}^{J}, \{F_l\}_{l=0}^{\zeta}, \lambda).$$

These modified protocols coincide with $\mathcal{P}$ for clients $\{C_j\}_{j=1}^{J}$ except for $C_0$. We therefore only describe the protocols for the client $C_0$. For ease of exposition, we denote that $\mathcal{P}_1$ (resp. $\mathcal{P}_{2,i}$) yields a distribution $\mathcal{D}_1$ (resp. $\mathcal{D}_{2,i}$).

The main objective is to demonstrate that the $(\mathcal{E}+1)$-protocols adhere to the following relation

$$\mathcal{P} \overset{\mathsf{Adv}_{\mathsf{pkIPFE}}}{\approx} \mathcal{P}_1 \overset{\mathsf{Adv}_{\mathsf{LinEnc\text{-}QFE}}}{\approx} \mathcal{P}_{2,1} \overset{\mathsf{Adv}_{\mathsf{LinEnc\text{-}QFE}}}{\approx} \cdots \overset{\mathsf{Adv}_{\mathsf{LinEnc\text{-}QFE}}}{\approx} \mathcal{P}_{2,\mathcal{E}},$$

where the notation $\overset{\mathsf{Adv}}{\approx}$ indicates that the distributions cannot be distinguished with the advantage $\mathsf{Adv}$. It immediately says that the advantage for distinguishing between $\mathcal{D}$ and $\mathcal{D}_{2,\mathcal{E}}$ is less than

$$\mathsf{Adv}_{\mathsf{pkIPFE}} + \mathcal{E} \cdot \mathsf{Adv}_{\mathsf{LinEnc\text{-}QFE}},$$

where $\mathsf{Adv}_{\mathsf{pkIPFE}}, \mathsf{Adv}_{\mathsf{LinEnc\text{-}QFE}}$ are the advantages corresponding to pkIPFE and LinEnc-QFE, respectively. As the last step, we show that the $\mathcal{D}_{2,\mathcal{E}}$ does not leak information about $\mathbf{x}_0$. By combining the two statements, we conclude the proof. In the following, we will prove that each statement is correct.

**Protocol $\mathcal{P}_1$.** First of all, we introduce a protocol $\mathcal{P}_1$. Intuitively, the protocol is identical to the original protocol $\mathcal{P}$ except for $C_0$, and the only difference between $\mathcal{P}$ and $\mathcal{P}_1$ is how to implement pkIPFE algorithm. We thus mainly introduce how protocol $\mathcal{P}_1$ is executed for an honest client $C_0$. We let

$$\mathsf{pkIPFE}^{\mathcal{S}} = \{\mathsf{Setup}^{\mathcal{S}}_{\mathsf{pkIPFE}}, \mathsf{KeyGen}^{\mathcal{S}}_{\mathsf{pkIPFE},0}, \mathsf{KeyGen}^{\mathcal{S}}_{\mathsf{pkIPFE},1}, \mathsf{Enc}^{\mathcal{S}}_{\mathsf{pkIPFE}}, \mathsf{Dec}_{\mathsf{pkIPFE}}\}$$

denote the simulator.

The protocol $\mathcal{P}_1$ is exactly the same as $\mathcal{P}$ except for the algorithm related to $\mathsf{pkIPFE}$. The detailed description of $\mathcal{P}_1$ is given in Fig. 14.

Let $\mathcal{P}_0$ be the origin protocol $\mathcal{P}$, and $\mathcal{G}_i$ be a game in which the challenger interacts with the adversary of $\mathcal{P}_i$ with $i \in \{0,1\}$. At the start of the game, the challenger randomly chooses $b \leftarrow \{0,1\}$ and interacts with the adversary in $\mathcal{G}_b$. At the end of the game, $\mathcal{A}$ returns $b' \in \{0,1\}$. We define $\mathsf{Adv}_{01}(\mathcal{A})$ as $|\Pr[b'=b] - 1/2|$. We directly obtain Lemma 4 from the results in [1].

**Lemma 4.** *$\mathcal{P}$ and $\mathcal{P}_1$ are computationally indistinguishable under the assumption that* $\mathsf{pkIPFE}$ *is adaptively simulation secure. More precisely, it holds that* $\mathsf{Adv}_{01}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{pkIPFE}}$.

*Proof of Lemma 4.* The only difference between $\mathcal{P}_0$ and $\mathcal{P}_1$ is the usage of $\mathsf{pkIPFE}$. $\mathcal{P}_0$ exploits a real $\mathsf{pkIPFE}$ and $\mathcal{P}_1$ uses a simulator of $\mathsf{pkIPFE}$. Thus, under the assumption that $\mathsf{pkIPFE}$ achieves simulation-based security proved by [1], this modification cannot affect $\mathcal{A}$'s view, which yields that $\mathsf{Adv}_{01}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{pkIPFE}}$. □

**Protocol $\mathcal{P}_{2,1}$.** Let $\mathcal{P}_{2,1}$ be a protocol identical to $\mathcal{P}_1$ except for generating a functional key $\mathsf{fk}_{G_{0,0,0}}$.To this end, we define the function $\mathsf{cKeyGen}^{\mathcal{S}}_{\mathsf{pkIPFE},0}$, which mirrors the $\mathsf{cKeyGen}$ function except in its execution of $\mathsf{KeyGen}$. Instead of the standard $\mathsf{KeyGen}$ component, it executes $\mathsf{KeyGen}^{\mathcal{S}}_0$. The new functional key $\mathsf{fk}_{\mathsf{E}_1 \circ G^{\mathcal{S}}_{0,0,0}}$ in $\mathcal{P}_{2,1}$ is then generated by

$$\mathsf{fk}_{\mathsf{E}_1 \circ G^{\mathcal{S}}_{0,0,0}} \leftarrow \mathsf{cKeyGen}^{\mathcal{S}}_{\mathsf{pkIPFE},0}(\mathsf{msk}^{\mathcal{S}}_{\mathsf{pkIPFE}}, \mathsf{pk}^{\mathcal{S}}_{\mathsf{pkIPFE}}, \mathsf{msk}_{\mathsf{QFE},1}, \mathsf{pp}_{\mathsf{QFE},1}, G^{\mathcal{S}}_{0,0,0}, L)$$

where $G^{\mathcal{S}}_{0,0,0} = \overline{h}_{0,0,0} \circ \gamma^{-1}_{0,0}$ is a linear function that has a representation matrix $\overline{\mathbf{H}}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0}$ such that

$$\mathbf{H}'_{0,0,0} \cdot \overline{\mathbf{H}}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0} = \mathbf{H}'_{0,0,0} \cdot \mathbf{H}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0} \bmod \Delta. \tag{4}$$

The matrix $\overline{\mathbf{H}}_{0,0,0}$ satisfying Eq. (4) always exists from the Lemma 1. Then, the following lemma holds.

**Lemma 5.** *$\mathcal{P}_1$ and $\mathcal{P}_{2,1}$ are computationally indistinguishable under the assumption that* $\mathsf{E}_1$ *is* $(2\ell+1)$ *ciphertext-bounded semi-adaptively secure. The advantage of distinguishing between $\mathcal{D}_1$ and $\mathcal{D}_{2,1}$ is smaller than* $\mathsf{Adv}_{\mathsf{LinEnc\text{-}QFE}}$.

*Furthermore, no one computationally obtains any information about $\mathbf{x}_0$ from $\mathsf{E}_1(h_{0,0,0}(\mathbf{x}_0))$.*

*Proof of Lemma 5.* We first note that $\mathsf{E}_1$ is also a linear function. That is, there exists a matrix $\mathbf{C}_{2,1}$ such that $\mathsf{E}_1(\mathbf{x}) = \mathbf{C}_{2,1} \cdot \overline{\mathbf{x}}$ with $\overline{\mathbf{x}} = (\mathbf{x} \| 1)$. It also holds that $\mathsf{E}_1(\mathbf{H}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0}) = \mathbf{C}_{2,1} \cdot \overline{\mathbf{H}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0}}$ with $\overline{\mathbf{H}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0}} = \begin{pmatrix} \mathbf{H}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0} \\ \mathbf{1}_{\mathsf{col}} \end{pmatrix}$, where $\mathsf{col}$ is the number columns of $\mathbf{H}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0}$ and $\mathbf{1}_{\mathsf{col}}$ is a vector $(1, \ldots, 1, 1)$ of length $\mathsf{col}$.

The functional key $\mathsf{fk}_{\mathsf{E}_1 \circ G_{0,0,0}}$ in $\mathcal{P}_1$ can be represented as $(\mathbf{A} \cdot \mathsf{msk}^{\mathcal{S}}_{\mathsf{pkIPFE}}, \mathbf{A})$, where $\mathbf{A} = \mathbf{C}_{2,1} \cdot \overline{\mathbf{H}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0}}$ and $\mathsf{msk}^{\mathcal{S}}_{\mathsf{pkIPFE}}$ is the secret key of $\mathsf{pkIPFE}^{\mathcal{S}}$. By definition, we also observe that

$$(\mathbf{A} \cdot \overline{\mathsf{msk}^{\mathcal{S}}_{\mathsf{pkIPFE}}}, \mathbf{A}) = (\mathsf{E}_1(\mathbf{H}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0} \cdot \mathsf{msk}^{\mathcal{S}}_{\mathsf{pkIPFE}}), \mathsf{E}_1(\mathbf{H}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0})),$$

where $\overline{\mathsf{msk}^{\mathcal{S}}_{\mathsf{pkIPFE}}} = (\mathsf{msk}^{\mathcal{S}}_{\mathsf{pkIPFE}} \| 1)$ We note that this $\mathsf{fk}_{\mathsf{E}_1 \circ G_{0,0,0}}$ exactly corresponds to $(2\ell+1)$-ciphertexts of $\mathsf{E}_1$ algorithm. For the sake of simplicity, we let $\mathsf{fk}_{\mathsf{E}_1 \circ G_{0,0,0}}[\mathsf{idx}]$ denote the $\mathsf{idx}$-th column vector of $(\mathbf{A} \cdot \overline{\mathsf{msk}^{\mathcal{S}}_{\mathsf{pkIPFE}}}, \mathbf{A})$ in this proof. Then, each $\mathsf{fk}_{\mathsf{E}_1 \circ G_{0,0,0}}[\mathsf{idx}]$ is a ciphertext of the form

$$\mathsf{fk}_{\mathsf{E}_1 \circ G_{0,0,0}}[\mathsf{idx}] = \begin{cases} \mathsf{E}_1(\mathbf{H}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0} \cdot \mathsf{msk}^{\mathcal{S}}_{\mathsf{pkIPFE}}) & \text{if } \mathsf{idx} = 1 \\ \mathsf{E}_1(\mathbf{H}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0}[\mathsf{idx}]) & \text{if } 2 \leq \mathsf{idx} \leq 2\ell+1. \end{cases}$$

On the other hand, $\mathsf{fk}_{\mathsf{E}_1 \circ G^{\mathcal{S}}_{0,0,0}}$ in $\mathcal{P}_{2,1}$ is of the form

$$(\mathsf{E}_1(\overline{\mathbf{H}}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0} \cdot \mathsf{msk}^{\mathcal{S}}_{\mathsf{pkIPFE}}), \mathsf{E}_1(\overline{\mathbf{H}}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0})),$$

which corresponds to $(2\ell+1)$-ciphertexts of the form $(\mathsf{E}_1(\overline{\mathbf{H}}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0} \cdot \mathsf{msk}^{\mathcal{S}}_{\mathsf{pkIPFE}}), \mathsf{E}_1(\overline{\mathbf{H}}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0}))$. Similarly, $\mathsf{fk}_{\mathsf{E}_1 \circ G^{\mathcal{S}}_{0,0,0}}[\mathsf{idx}]$ is denoted by the $\mathsf{idx}$-th column vector. From the constraint in Eq. (4) and definition of Fig. 14, it implies that for $\mathsf{idx} \in [1, 2\ell]$, we have

$$\begin{aligned} \mathsf{cDec}_{\mathsf{QFE}}(\mathsf{pp}_{\mathsf{QFE},2}, \mathsf{fk}_{\mathsf{E}_2 \circ G_{0,2,0}}, \mathsf{ct}_{0,1,0}[\mathsf{idx}]) &= \mathsf{cDec}_{\mathsf{QFE}}(\mathsf{pp}_{\mathsf{QFE},2}, \mathsf{fk}_{\mathsf{E}_2 \circ G_{0,2,0}}, \mathsf{E}_1(\mathbf{H}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0}[\mathsf{idx}])) \\ &= \mathsf{cDec}_{\mathsf{QFE}}(\mathsf{pp}_{\mathsf{QFE},2}, \mathsf{fk}_{\mathsf{E}_2 \circ G_{0,2,0}}, \mathsf{E}_1(\overline{\mathbf{H}}_{0,0,0} \cdot \mathbf{\Gamma}^{-1}_{0,0}[\mathsf{idx}])) \\ &= \mathsf{cDec}_{\mathsf{QFE}}(\mathsf{pp}_{\mathsf{QFE},2}, \mathsf{fk}_{\mathsf{E}_2 \circ G_{0,2,0}}, \mathsf{fk}_{\mathsf{E}_1 \circ G^{\mathcal{S}}_{0,0,0}}[\mathsf{idx}]) \end{aligned} \tag{5}$$

where $\mathsf{fk}_{\mathsf{E}_2 \circ G_{0,2,0}}$ is defined as Fig. 14, and $\mathsf{ct}_{0,1,0}[\mathsf{idx}] = \mathsf{fk}_{\mathsf{E}_1 \circ G_{0,0,0}}[\mathsf{idx}]$. We briefly remark that the second equality holds due to Eq. (4).

In summary, we observe that $\mathsf{fk}_{\mathsf{E}_1 \circ G_{0,0,0}}$ and $\mathsf{fk}_{\mathsf{E}_1 \circ G^{\mathcal{S}}_{0,0,0}}$ are both considered as $(2\ell{+}1)$ ciphertexts of $\mathsf{E}_1$. Moreover, they provide the same evaluated values given functional keys $\mathsf{fk}_{\mathsf{E}_2 \circ G_{0,2,0}}$ due to Eq. (5).

Any adversary $\mathcal{A}$ in distinguishing between $\mathcal{P}_1$ and $\mathcal{P}_{2,1}$ can obtain $(2\ell{+}1)$ ciphertexts $\mathsf{fk}_{\mathsf{E}_1 \circ G_{0,0,0}}$ or $\mathsf{fk}_{\mathsf{E}_1 \circ G^{\mathcal{S}}_{0,0,0}}$ depending on a protocol. However, $\mathcal{A}$ cannot distinguish between $\mathcal{P}_1$ and $\mathcal{P}_{2,1}$ using this information since $\mathsf{E}_1$ is $(2\ell + 1)$ ciphertext-bounded semi-adaptively secure.

We further argue that the information of $\mathbf{x}_0$ computationally is hidden. We point out that an adversary can obtain $\mathsf{E}_1(\overline{h}_{0,0,0}(\mathbf{x}_0))$ under the protocol $\mathcal{P}_{2,1}$.

We may assume that the adversary learns exactly $\overline{\mathbf{H}}_{0,0,0} \cdot \mathbf{x}_0$ from the $\mathsf{E}_1(\overline{h}_{0,0,0}(\mathbf{x}_0))$. Since $\overline{\mathbf{H}}_{0,0,0}$ is a tall matrix, determining $\overline{\mathbf{H}}_{0,0,0}$ immediately recovers $\mathbf{x}_0$. That is the number of possible candidates of $\mathbf{x}_0$ is obtained from that of $\overline{\mathbf{H}}_{0,0,0}$. As discussed in Lemma 1, the possible number of $\overline{\mathbf{H}}_{0,0,0}$ is larger than $2^\lambda$ by the setup. Hence, the message $\mathbf{x}_0$ can be recovered with probability at most $\frac{1}{2^\lambda}$. □

**Protocol $\mathcal{P}_{2,i}$.** For $i \geq 1$, $\mathcal{P}_{2,i+1}$ is the same protocol as $\mathcal{P}_{2,i}$ except for the $i$-th functional key of LinEnc-QFE. As $\mathcal{P}_{2,1}$ is defined in the above, $\mathcal{P}_{2,i}$ with $i \geq 2$ could be well defined. To this end, we define some notations. For each $i \in [1, \mathcal{E})$, let $\overline{\mathbf{H}}_{0,i,0}$ be a matrix such that

$$\mathbf{H}'_{0,i,0} \cdot \overline{\mathbf{H}}_{0,i,0} = \mathbf{H}'_{0,i,0} \cdot \mathbf{H}_{0,i,0} \bmod \Delta \tag{6}$$

and $\overline{h}_{0,i,0}$ is a linear function that corresponds to $\overline{\mathbf{H}}_{0,i,0}$ for each $1 \leq i < \mathcal{E}$. Given functions $G^{\mathcal{S}}_{0,i,0} = \overline{h}_{0,i,0} \circ f_{0,i} \circ h'_{0,i-1,0}$, the modified functional keys $\mathsf{fk}_{\mathsf{E}_{i+1} \circ G^{\mathcal{S}}_{0,i,0}}$ of $\mathcal{P}_{2,i+1}$ are then generated by

$$\mathsf{fk}_{\mathsf{E}_{i+1} \circ G^{\mathcal{S}}_{0,i,0}} \leftarrow \mathsf{cKeyGen}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE},i}, \mathsf{pp}_{\mathsf{QFE},i}, \mathsf{msk}_{\mathsf{QFE},i+1}, \mathsf{pp}_{\mathsf{QFE},i+1}, G^{\mathcal{S}}_{0,i,0}, L).$$

Similarly to Lemma 5, we remark that $\mathsf{fk}_{\mathsf{E}_{i+1} \circ G_{0,i,0}}$ is $(2\ell + 1)$ ciphertexts of the form

$$\mathsf{fk}_{\mathsf{E}_{i+1} \circ G_{0,i,0}}[\mathsf{idx}] = \begin{cases} \mathsf{E}_{i+1}(\mathbf{H}_{0,i,0} \cdot \mathbf{M}_{f_{0,i}} \cdot (\mathbf{H}'_{0,i-1,0} \otimes \mathbf{H}'_{0,i-1,0}) \cdot \overline{\mathsf{sk}_{\mathsf{QFE},i}}) & \text{if } \mathsf{idx} = 1 \\ \mathsf{E}_{i+1}(\mathbf{H}_{0,i,0} \cdot \mathbf{M}_{f_{0,i}} \cdot (\mathbf{H}'_{0,i-1,0} \otimes \mathbf{H}'_{0,i-1,0})[\mathsf{idx}]) & \text{o.w} \end{cases}$$

where $\overline{\mathsf{sk}_{\mathsf{QFE},i}} = (\mathsf{sk}_{\mathsf{QFE},i} \| 1)$ with $\mathsf{sk}_{\mathsf{QFE},i}$, a certain secret vector and $\mathbf{M}_{f_{0,i}}$ is a matrix representation of $f_{0,i}$. Then, with the same argument of Eq. (5) and the constraint in Eq. (6), it holds that

$$\mathsf{cDec}(\mathsf{pp}_{\mathsf{QFE},i+1}, \mathsf{fk}_{\mathsf{E}_{i+2} \circ G_{0,i+1,0}}, \mathsf{ct}_{0,i,0}[\mathsf{idx}], L) = \mathsf{cDec}_{\mathsf{QFE}}(\mathsf{pp}_{\mathsf{QFE},i+1}, \mathsf{fk}_{\mathsf{E}_{i+2} \circ G_{0,i+1,0}}, \overline{\mathsf{ct}}_{0,i,0}[\mathsf{idx}], L).$$

where $\mathsf{ct}_{0,i,0}[\mathsf{idx}] = \mathsf{fk}_{\mathsf{E}_{i+1} \circ G_{0,i,0}}[\mathsf{idx}]$ and $\overline{\mathsf{ct}}_{0,i,0}[\mathsf{idx}] = \mathsf{fk}_{\mathsf{E}_{i+1} \circ G^{\mathcal{S}}_{0,i,0}}[\mathsf{idx}]$ respectively.

According to the hardness proof of the LinEnc-QFE, an adversary $\mathcal{A}$ cannot distinguish between $\mathcal{P}_{2,i}$ and $\mathcal{P}_{2,i+1}$. For more details, we leave this proof in the Lemma 3. For a $\mathcal{P}_{2,i+1}$, a matrix for a randomness is converted into $\overline{\mathbf{H}}_{0,i,0}$. In the same vein as the above, thus $\mathcal{A}$ cannot obtain information related to $\bigcirc_{t=1}^{i} f_{l,t}(\mathbf{x}_0)$ from an intermediate value $\mathsf{E}_{i+1}(h_i \bigcirc_{t=1}^{i} f_{l,t}(\mathbf{x}_0))$ for a protocol $\mathcal{P}_{2,i+1}$.

In summary, we demonstrate that any adversary $\mathcal{A}$ only distinguishes between $(\mathcal{D}, \mathcal{D}_{2,\mathcal{E}})$ with at most advantage $\mathsf{Adv} \leq \mathsf{Adv}_{\mathsf{pkIPFE}} + \mathcal{E} \cdot \mathsf{Adv}_{\mathsf{LinEnc\text{-}QFE}}$. Although, $\mathcal{D}_{2,\mathcal{E}})$ for $0 \leq i < \mathcal{E}$ gives every intermediate values $\mathsf{E}_{i+1}(h_{0,i,0}(\bigcirc_{t=1}^{i} f_{l,t}(\mathbf{x}_0)))$. the message $\bigcirc_{t=1}^{i} f_{l,t}(\mathbf{x}_0)$ for some $i$ can be recovered with probability at most $\frac{\mathcal{E}}{2^\lambda}$. Putting it together, the advantage of adversary $\mathcal{A}$ in revealing any information about the message $\mathbf{x}_0$ is at most $\mathsf{Adv}_{\mathsf{pkIPFE}} + \mathcal{E} \cdot \mathsf{Adv}_{\mathsf{LinEnc\text{-}QFE}} + \frac{\mathcal{E}}{2^\lambda}$.

□

**Participants**: Clients ($\{C_j\}_{j\in[0,J]}$), key distributor ($KD$), and evaluator ($E$)
**Protocol**:

**Protocol Setup by $KD$**

1. $KD$ samples keys of $\mathsf{pkIPFE}$:

$$\{\mathsf{msk}_{\mathsf{pkIPFE}}, \mathsf{pk}_{\mathsf{pkIPFE}}\} \leftarrow \mathsf{Setup}_{\mathsf{pkIPFE}}(\lambda, 2\ell, B_X, B_F)$$

2. $KD$ samples keys of LinEnc-QFE $\mathcal{E}$ times for $i \in [1, \mathcal{E}]$:

$$\{\mathsf{msk}_{\mathsf{QFE},i}, \mathsf{pp}_{\mathsf{QFE},i}\} \leftarrow \mathsf{Setup}_{\mathsf{QFE}}(\lambda, \ell, 2\ell+1)$$

3. $KD$ sets the smallest integer $k$ as Eq. (1) and samples pairs of linear functions $(h_{l,i,j}, h'_{l,i,j})$ as Alg. 1 for each $l \in [0, \zeta], i \in [0, \mathcal{E}-1]$, and $j \in [0, J]$.

$$(h_{l,i,j}, h'_{l,i,j}) \leftarrow \textsc{genLin}(\ell, N, k, B_{\mathcal{H}})$$

4. For every $l, j$, $KD$ samples $\mathbf{\Gamma}_{l,j} \leftarrow \{0,1\}^{2\ell\times\ell}$ until there exists its left-inverse $\mathbf{\Gamma}_{l,j}^{-1}$. Define $\gamma_{l,j}(\mathbf{x}) = \mathbf{\Gamma}_{l,j} \cdot \mathbf{x}$ and $\gamma_{l,j}^{-1}(\mathbf{x}) = \mathbf{\Gamma}_{l,j}^{-1} \cdot \mathbf{x}$ for any $\mathbf{x}$.
5. Set $\{\mathsf{msk}_j\}_{j\in[0,J]}$ and $\mathsf{pk}$ as follows:

$$\mathsf{msk}_j = \{\mathsf{msk}_{\mathsf{pkIPFE}}, \{\mathsf{msk}_{\mathsf{QFE},i}\}_{i\in[1,\mathcal{E}]}, \{h_{l,i,j}, h'_{l,i,j}\}_{l\in[0,\zeta],i\in[0,\mathcal{E}-1]}\}$$
$$\mathsf{pk} = \{\mathsf{pk}_{\mathsf{pkIPFE}}, \{\mathsf{pp}_{\mathsf{QFE},i}\}_{i\in[1,\mathcal{E}]}\}.$$

**Encryption between $C_j$ and $KD$**

1. $KD$ sends a tuple $(\mathsf{pk}_{\mathsf{pkIPFE}}, \{h_{l,0,j} \circ \gamma_{l,j}^{-1}\}_{l\in[0,\zeta]}, \gamma_{l,j})$ to $C_j$.
2. $C_j$ encrypts $\gamma_{l,j}(\mathbf{x}_j)$ to generate $\mathsf{ct}_{l,j}$:

$$\mathsf{ct}_{l,j} \leftarrow \mathsf{Enc}_{\mathsf{pkIPFE}}(\mathsf{pk}_{\mathsf{pkIPFE}}, \gamma_{l,j}(\mathbf{x}_j))$$

3. $C_j$ sends $\mathsf{ct}_{l,j}$ to the $E$.

**Functional key generation between $KD$ and $E$**

1. $E$ sends a model $F_l = \bigcirc_{i=1}^{\mathcal{E}} f_{l,i}$ for a certain $l$ to $KD$.
2. With an encryption map $\mathsf{E}_i : \mathbf{x} \mapsto \mathsf{Enc}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE},i}, \mathsf{pp}_{\mathsf{QFE},i}, \mathbf{x})$, $KD$ sends a set of functional keys $\mathsf{F}_{l,j} := \{\mathsf{fk}_{\mathsf{E}_{i+1}\circ G_{l,i,j}}\}_{i\in[0,\mathcal{E}-1],j\in[0,J]} \cup \{\mathsf{fk}_{G_{l,\mathcal{E},j}}\}$ to E. For every $i \in [1, \mathcal{E}-1]$, $j \in [0, J]$, and a decomposition parameter $L$, define functional keys as follows:
   - $\mathsf{fk}_{\mathsf{E}_1\circ G_{l,0,j}} \leftarrow \mathsf{cKeyGen}_{\mathsf{pkIPFE}}(\mathsf{msk}_{\mathsf{pkIPFE}}, \mathsf{pk}_{\mathsf{pkIPFE}}, \mathsf{msk}_{\mathsf{QFE},1}, \mathsf{pp}_{\mathsf{QFE},1}, \underbrace{h_{l,0,j} \circ \gamma_{l,j}^{-1}}_{:=G_{l,0,j}}, L)$
   - $\mathsf{fk}_{\mathsf{E}_{i+1}\circ G_{l,i,j}} \leftarrow \mathsf{cKeyGen}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE},i}, \mathsf{pp}_{\mathsf{QFE},i}, \mathsf{msk}_{\mathsf{QFE},i+1}, \mathsf{pp}_{\mathsf{QFE},i+1}, \underbrace{h_{l,i,j} \circ f_{l,i} \circ h'_{l,i-1,j}}_{:=G_{l,i,j}}, L)$
   - $\mathsf{fk}_{G_{l,\mathcal{E},j}} \leftarrow \mathsf{KeyGen}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE},\mathcal{E}}, \mathsf{pp}_{\mathsf{QFE},\mathcal{E}}, \underbrace{f_{l,\mathcal{E}} \circ h'_{l,\mathcal{E}-1,j}}_{:=G_{l,\mathcal{E},j}}, L)$

**Model Evaluation by $E$**

1. $E$ inductively computes the following:
   - $M_{l,0,j} \leftarrow \mathsf{cDec}_{\mathsf{pkIPFE}}(\mathsf{pk}_{\mathsf{pkIPFE}}, \mathsf{fk}_{\mathsf{E}_1\circ G_{l,0,j}}, \mathsf{ct}_{l,j}, L)$
   - $M_{l,i,j} \leftarrow \mathsf{cDec}_{\mathsf{QFE}}(\mathsf{pp}_{\mathsf{QFE},i}, \mathsf{fk}_{\mathsf{E}_i\circ G_{l,i,j}}, M_{l,i-1,j}, L)$ for $i \in [1, \mathcal{E}-1]$
   - $M_{l,\mathcal{E},j} \leftarrow \mathsf{Dec}_{\mathsf{QFE}}(\mathsf{pp}_{\mathsf{QFE},i}, \mathsf{fk}_{G_{l,\mathcal{E},j}}, M_{l,\mathcal{E}-1,j})$
2. $E$ obtains $M_{l,\mathcal{E},j}$.

**Fig. 9.** Fully encrypted PPML protocol $\mathcal{P}$ using function compositions $\bigcirc_{i=1}^{\mathcal{E}} f_{l,i}(\mathbf{x})$.

- $\mathsf{Mus.Setup}_{\mathsf{IPFE}}(1^\lambda, n, e)$):
  1. Sample $\mathbf{a} \leftarrow \mathbb{Z}_p^2$, $\mathbf{U} \leftarrow \mathbb{Z}_p^{n\times 2}$.
  2. Set $\mathsf{mpk} = \{[\mathbf{a}]_1, [\mathbf{U}\cdot\mathbf{a}]_1\}$ and $\mathsf{msk} = \{\mathbf{a}, \mathbf{U}\}$ and return $\mathsf{mpk}, \mathsf{msk}$.
- $\mathsf{Mus.KeyGen}_{\mathsf{IPFE}}(\mathsf{msk}, \mathsf{mpk}, [\mathbf{y}]_2 \in \mathbb{G}_2^n)$:
  1. Return $\mathsf{sk}_{\mathbf{y}} = \left[\begin{pmatrix} -\mathbf{U}^T\cdot\mathbf{y} \\ \mathbf{y}\end{pmatrix}\right]_2 \in \mathbb{G}_2^{n+2}$.
- $\mathsf{Mus.Enc}_{\mathsf{IPFE}}(\mathsf{mpk}, [\mathbf{x}]_1 \in \mathbb{G}_1^n)$:
  1. Sample $r \leftarrow \mathbb{Z}_p$.
  2. Return $\mathsf{ct} = \left[\begin{pmatrix} r\cdot\mathbf{a} \\ \mathbf{x} + r\cdot\mathbf{U}\cdot\mathbf{a}\end{pmatrix}\right]_1 \in \mathbb{G}_1^{n+2}$.
- $\mathsf{Mus.Dec}_{\mathsf{IPFE}}(\mathsf{sk}_f, \mathsf{ct}, \mathsf{mpk})$:
  1. Return $\left[\mathsf{ct}^T\cdot\mathsf{sk}_{\mathbf{y}}\right]_T$.

**Fig. 10.** Mus.IPFE.

- $\mathsf{Mus.Setup}_{\mathsf{QFE}}(1^\lambda, e, n)$:
  1. Sample $\mathbf{U} \leftarrow \mathbb{Z}_p^{n\times 2}$, $\mathbf{V} \leftarrow \mathbb{Z}_p^{n\times 2}$ and compute
  $$\mathbf{M} = \begin{pmatrix} \mathbf{U}^T \otimes \mathbf{I}_n \\ \mathbf{I}_n \otimes \mathbf{V}^T \end{pmatrix}.$$
  2. Sample
  $$\{\mathsf{msk}_{\mathsf{IPFE}}, \mathsf{mpk}_{\mathsf{IPFE}}\} \leftarrow \mathsf{Mus.Setup}_{\mathsf{IPFE}}(1^\lambda, 4n).$$
  3. Set $\mathsf{mpk}, \mathsf{msk}$ as follows:
  $$\mathsf{mpk} = \{[\mathbf{U}]_a, [\mathbf{V}]_b, \mathsf{mpk}_{\mathsf{IPFE}}\}$$
  $$\mathsf{msk} = \{\mathbf{U}, \mathbf{V}, \mathbf{M}, \mathsf{msk}_{\mathsf{IPFE}}\}$$
  4. Return $\{\mathsf{mpk}, \mathsf{msk}\}$.
- $\mathsf{Mus.KeyGen}_{\mathsf{QFE}}(\mathsf{msk}, \mathsf{mpk}, \mathbf{f} \in \mathbb{Z}_p^{n^2})$:
  1. Sample $\mathsf{sk}_{\mathsf{IPFE}} \leftarrow \mathsf{KeyGen}_{\mathsf{IPFE}}(\mathsf{msk}_{\mathsf{IPFE}}, [\mathbf{M}\cdot\mathbf{f}]_2)$.
  2. Return $\mathsf{sk}_{\mathbf{f}} = \{\mathsf{sk}_{\mathsf{IPFE}}, \mathbf{f}\}$.
- $\mathsf{Mus.Enc}_{\mathsf{QFE}}(\mathsf{mpk}, \mathbf{x} \in \mathbb{Z}_p^n, \mathbf{y} \in \mathbb{Z}_p^n)$:
  1. Sample $\mathbf{r} \leftarrow \mathbb{Z}_p^2$, $\mathbf{s} \leftarrow \mathbb{Z}_p^2$ and compute the following:
  $$[\mathsf{ct}_{\mathbf{x}}]_1 = [\mathbf{x} + \mathbf{U}\cdot\mathbf{r}]_1,$$
  $$[\mathsf{ct}_{\mathbf{y}}]_2 = [\mathbf{y} + \mathbf{V}\cdot\mathbf{s}]_2,$$
  $$[\mathbf{h}]_1 = \left[\begin{pmatrix} \mathbf{r}\otimes\mathbf{y} \\ \mathsf{ct}_{\mathbf{x}}\otimes\mathbf{s}\end{pmatrix}\right]_1,$$
  $$\mathsf{ct}_{\mathsf{IPFE}} \leftarrow \mathsf{Mus.Enc}_{\mathsf{IPFE}}(\mathsf{msk}_{\mathsf{IPFE}}, [\mathbf{h}]_1)$$
  2. Return $\mathsf{ct} = \{[\mathsf{ct}_{\mathbf{x}}]_1, [\mathsf{ct}_{\mathbf{y}}]_2, \mathsf{ct}_{\mathsf{IPFE}}\}$.
- $\mathsf{Mus.Dec}_{\mathsf{QFE}}(\mathsf{sk}_{\mathbf{f}}, \mathsf{ct}, \mathsf{mpk})$:
  1. Compute
  $$\mathbf{t} = \mathsf{Mus.Dec}_{\mathsf{IPFE}}(\mathsf{mpk}_{\mathsf{IPFE}}, \mathsf{ct}_{\mathsf{IPFE}}, \mathsf{sk}_{\mathsf{IPFE}}).$$
  2. Return $\log_{g_T}([(\mathsf{ct}_{\mathbf{x}} \otimes \mathsf{ct}_{\mathbf{y}})\cdot\mathbf{f}]_T/\mathbf{t})$.

**Fig. 11.** Mus.QFE.

$\widetilde{\mathsf{Enc}}_{\mathsf{IPFE}}(\mathsf{Mus{:}ctxt}_{\mathsf{IPFE}}(\mathbf{x}_i))$ :
1. Sample $[\mathbf{D}]_2, \mathbf{d}_i \leftarrow Alg.\ 2(\mathsf{Mus{:}ctxt}_{\mathsf{IPFE}}(\mathbf{x}_i))$.
2. Return $\mathbf{d}_i$ for every $i$.

$\widetilde{\mathsf{KeyGen}}_{\mathsf{IPFE}}(\mathbf{f}, [\mathbf{D}]_2, \mathsf{Mus.KeyGen}_{\mathsf{IPFE}}(\mathbf{f}))$ :
1. Denote $\mathsf{Mus.KeyGen}_{\mathsf{IPFE}}(\mathbf{f})$ by $[\mathsf{sk}_f]_1$.
2. Return $e([\mathsf{sk}_f^T]_1, [\mathbf{D}]_2) = [\mathsf{sk}_f^T \cdot \mathbf{D}]_T$.

**Fig. 12.** Construction of $\widetilde{\mathsf{Enc}}_{\mathsf{IPFE}}$ and $\widetilde{\mathsf{KeyGen}}_{\mathsf{IPFE}}$ given $\{\mathsf{Mus{:}ctxt}_{\mathsf{IPFE}}(\mathbf{x}_i)\}$ and $\mathbf{f}$.

$\widetilde{\mathsf{Enc}}_{\mathsf{QFE}}(\{\mathsf{Mus{:}ctxt}_{\mathsf{QFE}}(\mathbf{x}_i, \mathbf{y}_i)\}_{i \le Q})$ :

1. Sample $[\mathbf{D}_0]_1, \{\mathbf{d}_{\mathbf{x}_i}\}_{i \le Q} \leftarrow \mathit{Alg.}\ 2(\{\mathsf{ct}_{\mathbf{x}_i}\}_{i \le Q})$.
2. Sample $[\mathbf{D}_1]_2, \{\mathbf{d}_{\mathbf{y}_i}\}_{i \le Q} \leftarrow \mathit{Alg.}\ 2(\{\mathsf{ct}_{\mathbf{y}_i}\}_{i \le Q})$.
3. Sample
$$[\mathbf{D}_{\mathsf{IPFE}}]_2, \{\mathbf{d}_{\mathsf{IPFE},i}\}_{i \le Q} \leftarrow \widetilde{\mathsf{Enc}}_{\mathsf{IPFE}}(\{\mathsf{ct}_{\mathsf{IPFE},i}\}_{i \le Q}).$$
4. Return $(\{\mathbf{d}_{\mathbf{x}_i}, \mathbf{d}_{\mathbf{y}_i}, \mathbf{d}_{\mathsf{IPFE},i}\}_{i \le Q})$.

$\widetilde{\mathsf{KeyGen}}_{\mathsf{QFE}}(\mathbf{f}, \{[\mathbf{D}_0]_1, [\mathbf{D}_1]_2, [\mathbf{D}_{\mathsf{IPFE}}]_2\}, \mathsf{KeyGen}_{\mathsf{QFE}}(\mathbf{f}))$ :

1. Parse $\mathsf{KeyGen}_{\mathsf{QFE}}(\mathbf{f})$ by $(\mathsf{sk}_{\mathsf{IPFE},\mathbf{f}}, [\mathbf{f}]_c)$.
2. Compute $\mathsf{fk}_1 := [\mathbf{f}^T \cdot (\mathbf{D}_0 \otimes \mathbf{D}_1)]_T$ via bilinear map $e$ and $\mathbf{f}^T$, $[\mathbf{D}_0]_1$, and $[\mathbf{D}_1]_2$.
3. Sample
$$\mathsf{fk}_{\mathsf{IPFE}} \leftarrow \widetilde{\mathsf{KeyGen}}_{\mathsf{IPFE}}(\mathsf{sk}_{\mathsf{IPFE},f}, [\mathbf{D}_{\mathsf{IPFE}}]_2).$$
4. Return $\mathsf{fk}_1$ and $\mathsf{fk}_{\mathsf{IPFE}}$.

**Fig. 13.** Construction of $\widetilde{\mathsf{Enc}}_{\mathsf{QFE}}$ and $\widetilde{\mathsf{KeyGen}}_{\mathsf{QFE}}$ given $\{\mathsf{Mus{:}ctxt}_{\mathsf{QFE}}(\mathbf{x}_i, \mathbf{y}_i) = \{\mathsf{ct}_{\mathbf{x}_i}, \mathsf{ct}_{\mathbf{y}_i}, \mathsf{ct}_{\mathsf{IPFE},i}\}\}_{i \le Q}$ and $\mathbf{f}$.

**Participants**: Client ($C_0$), key distributor ($KD$), and evaluator ($E$)
**Protocol**:

**Protocol Setup by $KD$**

1. $KD$ samples keys of $\mathsf{pkIPFE}$:

$$\{\mathsf{msk}^{\mathcal{S}}_{\mathsf{pkIPFE}}, \mathsf{pk}^{\mathcal{S}}_{\mathsf{pkIPFE}}\} \leftarrow \mathsf{Setup}^{\mathcal{S}}_{\mathsf{pkIPFE}}(\lambda, 2\ell, B_X, B_F)$$

2. $KD$ samples keys of LinEnc-QFE $\mathcal{E}$ times for $i \in [1, \mathcal{E}]$:

$$\{\mathsf{msk}_{\mathsf{QFE},i}, \mathsf{pp}_{\mathsf{QFE},i}\} \leftarrow \mathsf{Setup}_{\mathsf{QFE}}(\lambda, \ell, 2\ell+1)$$

3. $KD$ sets the smallest integer $k$ as Eq. (1) and samples pairs of linear functions $(h_{l,i,0}, h'_{l,i,0})$ as Alg. 1 for each $l \in [0, \zeta]$ and $i \in [0, \mathcal{E}-1]$.

$$(h_{l,i,0}, h'_{l,i,0}) \leftarrow \textsc{genLin}(\ell, N, k, B_{\mathcal{H}})$$

4. $KD$ samples $\mathbf{\Gamma}_{0,0} \leftarrow \{0,1\}^{2\ell \times \ell}$ until there exists its left-inverse $\mathbf{\Gamma}^{-1}_{0,0}$. Define $\gamma_{0,0}(\mathbf{x}) = \mathbf{\Gamma}_{0,0} \cdot \mathbf{x}$ and $\gamma^{-1}_{0,0}(\mathbf{x}) = \mathbf{\Gamma}^{-1}_{0,0} \cdot \mathbf{x}$.
5. Set $\mathsf{msk}_0$ and $\mathsf{pk}$ as follows:

$$\mathsf{msk}_0 = \{\mathsf{msk}^{\mathcal{S}}_{\mathsf{pkIPFE}}, \{\mathsf{msk}_{\mathsf{QFE},i}\}_{i\in[1,\mathcal{E}]}, \{h_{l,i,0}, h'_{l,i,0}\}_{l\in[0,\zeta],i\in[0,\mathcal{E}-1]}\}$$
$$\mathsf{pk} = \{\mathsf{pk}^{\mathcal{S}}_{\mathsf{pkIPFE}}, \{\mathsf{pp}_{\mathsf{QFE},i}\}_{i\in[1,\mathcal{E}]}\}.$$

**Encryption between $C_0$ and $KD$**

1. $KD$ sends a tuple $(\mathsf{pk}_{\mathsf{pkIPFE}}, h_{0,0,0} \circ \gamma^{-1}_{0,0}, \gamma_{0,0})$ to $C_0$.
2. $C_0$ encrypts $\gamma_{0,0}(\mathbf{x}_0)$ to generate $\mathsf{ct}^*_0$:

$$\mathsf{ct}^*_0 \leftarrow \mathsf{Enc}^{\mathcal{S}}_{\mathsf{pkIPFE}}(\mathsf{pk}^{\mathcal{S}}_{\mathsf{pkIPFE}}, h_{0,0,0}(\mathbf{x}_0))$$

3. $C_0$ sends $\mathsf{ct}^*_0$ to the $E$.

**Functional key generation between $KD$ and $E$**

1. $E$ sends a model $F_0 = \bigcirc_{i=1}^{\mathcal{E}} f_{0,i}$ to $KD$.
2. With an encryption map $\mathsf{E}_i : \mathbf{x} \mapsto \mathsf{Enc}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE},i}, \mathsf{pp}_{\mathsf{QFE},i}, \mathbf{x})$, $KD$ sends a set of functional keys $\mathsf{F}_{0,0} := \{\mathsf{fk}_{\mathsf{E}_{i+1} \circ G_{0,i,0}}\}_{i\in[0,\mathcal{E}]} \cup \mathsf{fk}_{G_{0,\mathcal{E},0}}$ to $E$. For every $i \in [1, \mathcal{E}-1]$, and a decomposition parameter $L$, define functional keys as follows:
   - $\mathsf{fk}_{\mathsf{E}_1 \circ G_{0,0,0}} \leftarrow \mathsf{cKeyGen}_{\mathsf{pkIPFE}}(\mathsf{msk}_{\mathsf{pkIPFE}}, \mathsf{pk}_{\mathsf{pkIPFE}}, \mathsf{msk}_{\mathsf{QFE},1}, \mathsf{pp}_{\mathsf{QFE},1}, \underbrace{h_{0,0,0} \circ \gamma^{-1}_{0,0}}_{:=G_{0,0,0}}, L)$
   - $\mathsf{fk}_{\mathsf{E}_{i+1} \circ G_{0,i,0}} \leftarrow \mathsf{cKeyGen}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE},i}, \mathsf{pp}_{\mathsf{QFE},i}, \mathsf{msk}_{\mathsf{QFE},i+1}, \mathsf{pp}_{\mathsf{QFE},i+1}, \underbrace{h_{0,i,0} \circ f_{0,i} \circ h'_{0,i-1,0}}_{:=G_{0,i,0}}, L)$
   - $\mathsf{fk}_{G_{0,\mathcal{E},0}} \leftarrow \mathsf{KeyGen}_{\mathsf{QFE}}(\mathsf{msk}_{\mathsf{QFE},\mathcal{E}}, \mathsf{pp}_{\mathsf{QFE},\mathcal{E}}, \underbrace{f_{0,\mathcal{E}} \circ h'_{0,\mathcal{E}-1,0}}_{:=G_{0,\mathcal{E},0}}, L)$

**Model Evaluation by $E$**

1. $E$ inductively computes the following:
   - $M_{0,0,0} \leftarrow \mathsf{cDec}_{\mathsf{pkIPFE}}(\mathsf{pk}_{\mathsf{pkIPFE}}, \mathsf{fk}_{\mathsf{E}_1 \circ G_{0,0,0}}, \mathsf{ct}^*_0, L)$
   - $M_{0,i,0} \leftarrow \mathsf{cDec}_{\mathsf{QFE}}(\mathsf{pp}_{\mathsf{QFE},i}, \mathsf{fk}_{\mathsf{E}_i \circ G_{0,i,0}}, M_{0,i-1,0})$ for $i \in [1, \mathcal{E}-1]$
   - $M_{0,\mathcal{E},0} \leftarrow \mathsf{Dec}_{\mathsf{QFE}}(\mathsf{pp}_{\mathsf{QFE},i}, \mathsf{fk}_{G_{0,\mathcal{E},0}}, M_{0,\mathcal{E}-1,0})$
2. $E$ obtains $M_{0,\mathcal{E},0}$.

**Fig. 14.** Protocol $\mathcal{P}_1$ for a client $C_0$.