Fully Encrypted Machine Learning Training Using Function-Hiding Functional Encryption

Linda Scheu-Hachtel and Jasmin Zalonis

University of Mannheim, Mannheim, Germany
{linda.scheu-hachtel, zalonis}@uni-mannheim.de

Abstract. This work presents a practical protocol for fully encrypted machine learning using one-time (OT) function-hiding (FH) multi-input functional encryption (MIFE). Prior works employing functional encryption for privacy-preserving machine learning typically protect intermediate results using differential privacy (DP) techniques, which requires a large privacy budget and a predefined number of model training iterations. Our protocol overcomes these limitations by ensuring that all intermediate results remain encrypted throughout the training process, revealing only the final model, which can be further protected using DP if desired. This allows for more flexibility in the number of training iterations and improves the utility and robustness of the trained model, especially in the high privacy regime.
We demonstrate the practicality of our protocol by training logistic regression models on real-world datasets. To do this, we present the first OT FH-MIFE scheme that ensures correctness over $\mathbb{Z}_p$ and supports a flexible number of decryption keys. This is also the first FH-(MI)FE schemes for affine functions based on lattices. Furthermore, we adapt this construction to efficiently handle affine average functions for horizontally partitioned data.

Keywords: Function-Hiding Functional Encryption · Multi-Input Functional Encryption · Lattices · Privacy-Preserving Machine Learning.

1 Introduction

Functional encryption (FE) is a powerful cryptographic primitive overcoming the all-or-nothing nature of traditional encryption schemes. Specifically, it enables the generation of decryption keys associated with a function f, so that decrypting a ciphertext of x reveals only $f(x)$, without disclosing any additional information about x.

Recently, FE schemes have been employed in privacy-preserving machine learning (PPML) [36, 48] instead of established cryptographic methods such as multi-party computation (MPC) and homomorphic encryption (HE) due to some unique advantages [15, 39, 47]. Specifically multi-input functional encryption (MIFE) schemes are of interest, which allow the evaluation of multivariate functions $f(x_1, \ldots, x_N)$ on multiple ciphertexts. These ciphertexts may come

from multiple clients, which allows the data to be distributed (possibly arbitrary) across multiple data sources.

However, the training of machine learning (ML) models typically requires the evaluation of complex functions. Although there exist FE schemes supporting the circuit class $\mathsf{P/Poly}$ [8, 10, 24], i.e. arbitrary functions, these schemes are far from being used in practice, which is why there has been a related line of works focusing on schemes for specific function families such as inner-product (IP) [2, 7, 34] or quadratic functions [6, 12, 41]. A common approach is to use FE only for the first linear part of the model and then continue the rest of the training on the plain results [20,35,46]. However, by disclosing these intermediate results in plaintext, they inadvertently reveal more information about the private data than necessary. In the worst case, this allows to reconstruct the whole database [14, 26].

One way to protect these intermediate results is to apply differential privacy (DP) [21]. DP is a widely used technique to provide privacy guarantees when analyzing sensitive data. The idea of DP is that given the output of a computation, an analyst cannot decide whether a specific individual's data was included in the considered database or not. This is usually achieved by adding some well chosen noise to the output of the computation, i.e. the intermediate results in the case of PPML.

Zalonis et al. [48] recently proposed a noisy MIFE scheme that can be used to train linearizable ML algorithms such as linear and logistic regression with gradient descent (GD). A noisy MIFE allows the input of multiple ciphertexts to evaluate $f(x_1, \ldots, x_N) + \nu$, where ν is a noise value hidden in the decryption key. They consider a setup that is motivated by practical requirements that typically occur in real-world scenarios, for instance in the medical field. It consists of a set of clients, each holding private data, an analyst who wants to train an ML model on the clients' data, and a trusted authority[1], for example an ethics committee, which might have limited resources. It is natural to assume that the clients might be willing to provide their data for analyses under the restriction that their privacy remains protected, but do not want to actively participate in a complex cryptographic protocol. Using noisy MIFE, this is possible as the analyst can train their model iteratively by only interacting with the authority, who chooses DP noise values for each gradient update to ensure that the overall training process is DP. All the clients have to do is to encrypt their data and hand it to the analyst, who builds a static database using these ciphertexts. This naturally imposes a bound on the number of required ciphertexts per client and allows to focus on one-time (OT) schemes.

Although this mitigates the privacy leakage there are still some major drawbacks, which are also present in other approaches using DP. To ensure DP for the overall protocol, the number of iterations must be fixed in advance, as the privacy budget is spread over all iterations. In other words, convergence may not be reached if this predetermined number is too small, however setting it

[1] This trusted party is common in all solutions discussing (MI)FE for PPML. But even solutions based on MPC and HE require at least one trusted party.

too large can degrade utility as well due to smaller privacy budget per iteration and thus increased noise. Without prior knowledge of the data distribution, it is hard to predict how many iterations are required to reach convergence. This issue is particularly pronounced for small privacy parameters, where the added noise tends to be large, substantially affecting the accuracy of the final model. Indeed, especially if the privacy budget is low, i.e. in the high privacy regime, this approach is not robust, which is undesirable if training is only conducted once in a real world scenario, unless the privacy budget is split even further.

Our Approach. At a high level, we avoid this problem by replacing the DP noise in each iteration with *truly random* noise added to the intermediate results. In each iteration, the analyst requests a decryption key based on the GD update function. Instead of receiving the decryption key associated with the update function, it receives one associated with a perturbed update function, that adds random noise to the true output, where the random values are freshly sampled for each iteration. As the authority has chosen the random noise, for the next iteration they can cancel out the randomness and hand out the decryption key for the next update, whose result will again be concealed by another truly random noise. Since each randomness is only used once, it prevents any potential leakage of the clients' data as it serves as a random pad encryption.

After several iterations, for example when a certain level of convergence is reached, the authority may release the randomness used in the final iteration. In this case, the analyst can recover the true final model. Alternatively, if the goal is to train a DP ML model, the released value consists of the final randomness combined with DP noise. This ensures that the underlying DP noise remains hidden and the analyst only learns the DP model.

This approach requires an MIFE scheme, that

1) is function-hiding (FH), which means that the decryption key does not reveal anything about the underlying function, so that the analyst does not learn the random values and thus the true intermediate results;
2) operates over the complete finite group $\mathbb{Z}_p$ for some integer p to support random pads on the output.

However, so far no such scheme is known. On the one hand, the only known efficient FH-FE schemes existing so far are based on bilinear pairings [13, 43], which, by design, need the output to be in some bounded range to solve the discrete logarithm. Hence, they do not support requirement 2).

On the other hand, the only existing FE schemes supporting evaluations over the complete finite group $\mathbb{Z}_p$, which are based either on the security of random pads [3, 39] or lattices [7], are not FH, therefore they violate requirement 1). There even exists an impossibility result stating that one cannot construct an FH-FE scheme solely based on lattices, which allows an unbounded number of decryption key queries [44]. However, as we will show, it is possible to construct an affine FH-FE scheme for a *bounded* number of decryption keys.

Using the fact that in an affine FH-FE scheme the decryption key generation and encryption algorithm can be swapped, we directly obtain a scheme support-

ing a bounded number of ciphertexts and an unbounded number of decryption keys. This scheme can be further lifted to an FH-MIFE scheme.

Although this scheme is interesting on its own, it is not yet efficient enough for our considered use case of PPML. Nevertheless, it remains of theoretical interest and may find applications in other domains, such as biometric authentication [23], which is why we include it here for completeness.

To achieve a better efficiency, we further tailor the scheme to our use case. Specifically, we note that the GD update function can be interpreted as an averaging operation with an additional constant term. This insight enables us to employ, e.g. secret-sharing techniques to optimize the decryption process and reduce the number of ciphertexts required. A more technical overview on how our scheme is constructed can be found in Section 4.1.

1.1 Our Contribution

Our contribution can be summarized as follows.

- We propose a protocol to train linearizable ML models, e.g. a logistic regression, without any leakage of intermediate results and show its applicability on real-world data. In contrast to previous protocols, the training is not terminated after a predetermined number of iterations, but can continue until convergence is reached. By adding DP noise only to the final model, we achieve better utility and robustness, especially in the high privacy regime.
- We present the first affine FH-MIFE scheme from lattices, which allows for one ciphertext per client and an unbounded number of decryption keys[2]. This is the first FH scheme from lattices and the first to ensure correctness over the whole domain $\mathbb{Z}_p$ for some prime p, which is also interesting from a theoretical point of view. Using standard techniques, this scheme can be lifted to support a bounded number of ciphertexts per client.
- We further adapt this scheme to an OT FH-MIFE scheme for affine average functions, dubbed HIFEL (HIdden Functional Encryption from Lattices). HIFEL supports affine function where all data records are evaluated by the same function and each client only submits one ciphertext.
- Finally we train a logistic regression model on real-world data using the proposed protocol and HIFEL. This is the first PPML training based on FE that remains fully encrypted throughout the training process and does not require the promise of DP to cover the intermediate results. Only the final model is revealed, which can be further protected using DP if desired.

1.2 Related Work

In the following we recall related papers that consider FE for PPML or provide FH-FE constructions.

[2] Or respectively an unbounded number of ciphertexts and one decryption key.

Machine Learning using Functional Encryption. Although a lot of works combining ML and FE simply focus on prediction [14, 20, 30, 38], some also include training. Xu et al. [46] proposed a 5-layer neural network, where the first layer is computed using an inner-product functional encryption (IPFE) scheme, while all subsequent computations are performed in the clear. The performance was improved by Panzade et al. [35], who used an additional FH-IPFE scheme. However, the leakage of intermediate results makes them insecure [26].

Recently, Zalonis et al. [48] proposed a protocol to train an ML model using GD by employing a noisy quadratic MIFE scheme, where the data can be arbitrarily distributed among all clients. They address the leakage by adding DP noise to each function evaluation, making the whole training DP. In a similar manner, Scheu-Hachtel and Zalonis [39] showed how to use an IPFE scheme to train a logistic regression if the data is horizontally distributed. However, both of these approaches require a predetermined number of iterations to divide the privacy budget over all iterations, whereas our approach enables to train over a horizontally split dataset until convergence is reached and allows to spend the whole privacy budget on the final model only. In addition, the framework of [39] considers multiple analysis requests on the same data, which further splits the privacy budget.

Hong et al. [25] also proposed a fully encrypted ML protocol using FE. Their idea is to chain their quadratic FE scheme such that their function evaluation has the same form as their ciphertext. This way, they can create evaluations of higher order and present runtimes for the classification of a 2-layer model. Although their idea could potentially be used to train a ML model as it allows for higher degree functions, they do not support multiple data holder and their runtimes indicate that the training even with one data holder might not be feasible in reasonable time.

Function-Hiding Multi-Input Functional Encryption. Over the years, there have been many constructions considering FH-FE for specific function families, or more precisely IP [5, 17, 28, 42]. Some of this schemes have been directly extended to support multiple inputs [3, 18]. There exists also a generic construction that transforms any weak FH single input scheme into an MIFE scheme [40]. However, all of these schemes are based on pairings, which are not suitable for our proposed protocol, since we require unbounded output over a finite field.

1.3 Organization

The paper is organized as follows. In Section 2, we review the necessary preliminaries on ML and MIFE. In Section 3, we present our protocol for fully encrypted PPML, followed by a description of the underlying cryptographic technique, the OT FH-MIFE scheme dubbed HIFEL, in Section 4. Implementation details and results from training logistic regression models on various datasets are provided in Section 5.

2 Preliminaries

Due to the limited space, we only provide background on GD and MIFE here. For an overview of DP, we refer to [22] and for an overview of lattice-based cryptography to [37].

2.1 Notation

Let λ define the security parameter. Denote by $[n]$ the set $\{1, \ldots, n\}$ and $[n_1, n_2]$ the set $\{n_1, \ldots, n_2\}$ for any $n, n_1, n_2 \in \mathbb{N}$, where $\mathbb{N}$ defines the set of all positive integers and $\mathbb{Z}$ the set of all integers. For any $q \in \mathbb{Z}$, the term $\mathbb{Z}_q$ refers to the residue class ring of size q.

In this work, we always consider row vectors. Vectors and matrices are denoted by bold letters, i.e. $\boldsymbol{x}$ denotes a vector, where $\boldsymbol{x}[j]$ is the j^{th} element of $\boldsymbol{x}$. For vectors $\boldsymbol{x}, \boldsymbol{x}_1, \boldsymbol{x}_2$ of the same length, $\|\boldsymbol{x}\|_p = (\sum_i |\boldsymbol{x}[i]|^p)^{\frac{1}{p}}$ denotes the ℓ_p norm and $\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle = \sum_i \boldsymbol{x}_1[i]\boldsymbol{x}_2[i]$ their scalar product. For a matrix $\mathbf{A}$, $s_1(\mathbf{A})$ denotes its spectral norm. We always denote by $\mathbf{1}_i^j$ the one hot vector of length j with a 1 at position i.

For a distribution $\mathbb{D}$ over some set Δ, $x \leftarrow \mathbb{D}(\Delta)$ denotes that x is sampled according to the distribution $\mathbb{D}$. If $\mathbb{D}$ is clear from the context, we simply write $x \leftarrow \Delta$. $x \xleftarrow{\$} \Delta$ represents the process of uniformly sampling an element $x \in \Delta$. For two distributions $\mathbb{D}_0, \mathbb{D}_1$, we write $\mathbb{D}_0 \equiv \mathbb{D}_1$, if they are identically distributed. The discrete Gaussian distribution over $\mathbb{Z}^n$ is denoted by $\mathcal{D}^n_{\mathbb{Z},\sigma}$, where $\sigma > 0$ is the standard deviation.

We use the abbreviation PPT to mean probabilistic polynomial time. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}^+$ is said to be *negligible* if for every $c \in \mathbb{N}$, there exists a $\tau \in \mathbb{N}$ such that for all $x \in \mathbb{N}$ with $x > \tau$, it holds that $\mathsf{negl}(x) < 1/x^c$.

2.2 Gradient Descent for Logistic Regression

One common approach to train an ML model, or more precisely its parameters $\boldsymbol{\theta}$, is to use the GD algorithm [1]. Let $\boldsymbol{X} = \{(\boldsymbol{x}_i, y_i)\}_{i \in [N]}$ be a dataset, where each record $(\boldsymbol{x}_i, y_i)$ consists of a constant $\boldsymbol{x}_i[0] := 1$, a certain number of attributes M and the dependent variable $y \in \{0, 1\}$. The goal of GD is to minimize the loss function

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{X}) = \frac{1}{N} \sum_{i \in [N]} l(\boldsymbol{\theta}, (\boldsymbol{x}_i, y_i)) \tag{1}$$

of the model by iteratively updating the model weight using the gradient $\nabla\mathcal{L}$, where l is the loss for a single observation. More precisely, in each iteration t the parameters are updated by computing

$$\boldsymbol{\theta}^{(t+1)} := (1 - \alpha \cdot \Lambda)\boldsymbol{\theta}^{(t)} + \frac{\alpha}{N} \sum_{i \in [N]} \nabla l(\boldsymbol{\theta}^{(t)}, (\boldsymbol{x}_i, y_i)), \tag{2}$$

where α is the learning rate and Λ is the regularization parameter to avoid over-fitting the model. This update step can be captured as a function

$$F^{(t+1)}_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{X}) = \boldsymbol{\theta}^{(t+1)}.$$

In case of a logistic regression model, the gradient of the single loss function is given as $\nabla l(\boldsymbol{\theta}, (\boldsymbol{x}_i, y_i)) = (y_i - \sigma(\boldsymbol{x}_i \boldsymbol{\theta}^\top))\boldsymbol{x}_i$, where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function. Using a Taylor approximation of degree 1, the sigmoid function can be approximated by $\sigma'(x) = 0.5 + 0.25x$, i.e. ∇l is a quadratic function in the data records. Like [39], in order to use an affine functionality to calculate the updated weights, one can precompute the needed monomials and encrypt them, i.e. in this case $((\boldsymbol{x}, y) \otimes (\boldsymbol{x}, y))$, which contains all monomials up to degree 2.

Using symmetry, as each monomial is only needed once, this increases the vector length to be encrypted from $M+1$ to $0.5(5 \cdot M + M^2) + 1$. The same idea can also be used for approximation of higher degree or loss functions of other models, such as linear regression.

2.3 Multi-Input Functional Encryption

Let us recall the standard notion of MIFE.

Definition 1 (MIFE). *Let $\mathcal{F} = \{\mathcal{F}_{N,\pi}\}_{N \in \mathbb{N}}$ be an ensemble where each $\mathcal{F}_{N,\pi}$ is a family of N-ary functions, i.e. $f \in \mathcal{F}_{N,\pi}$ is defined as $f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_N \to \mathcal{Y}$ and π an additional set of parameters. A multi-input functional encryption (MIFE) scheme for $\mathcal{F}$ is a tuple of algorithms $\mathsf{MIFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ defined as follows:*

$\mathsf{Setup}(1^\lambda, \mathcal{F}_{N,\pi}) \to (\mathsf{MSK}, \{\mathsf{EK}_i\}_{i\in[N]}, \mathsf{PP})$**:** *Takes as input a security parameter λ and a description of $\mathcal{F}_{N,\pi} \in \mathcal{F}$. It outputs a master secret key MSK, encryption keys $\{\mathsf{EK}_i\}_{i\in[N]}$ and the public parameters PP, which are implicitly taken by all other algorithms.*

$\mathsf{Enc}(\mathsf{EK}_i, i, x_i) \to \mathsf{CT}_i$ *: Takes as input the encryption key EK_i for slot $i \in [N]$ and a message $x_i \in \mathcal{X}_i$. It outputs a ciphertext CT_i.*

$\mathsf{KeyGen}(\mathsf{MSK}, f) \to \mathsf{DK}_f$**:** *Takes as input the master secret key MSK and a function $f \in \mathcal{F}_{N,\pi}$. It outputs a decryption key DK_f.*

$\mathsf{Dec}(\mathsf{DK}_f, \mathsf{CT}_1, \ldots, \mathsf{CT}_N) \to y$**:** *Takes as input a decryption key DK_f and N ciphertexts. It outputs a value $y \in \mathcal{Y}$.*

A scheme MIFE as defined above is called correct *if for all $N \in \mathbb{N}$, $f \in \mathcal{F}_{N,\pi}$ and all $x_i \in \mathcal{X}_i$, $i \in [N]$, it holds that*

$$\Pr\left(\mathsf{Dec}(\mathsf{DK}_f, \mathsf{CT}_1, \ldots, \mathsf{CT}_N) = f(x_1, \ldots, x_N)\right) \geq 1 - \mathsf{negl}(\lambda),$$

where $(\mathsf{MSK}, \{\mathsf{EK}_i\}_{i\in[N]}, \mathsf{PP}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}_{N,\pi})$, $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{EK}_i, i, x_i) \forall i \in [N]$, $\mathsf{DK}_f \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f)$ and the probability is taken over the coins of Setup, Enc and KeyGen.

If $N = 1$, this is the standard definition of a secret-key *single-input* FE scheme with possible bounds. To obtain a public key (PK) single-input scheme, $\mathsf{EK} = \mathsf{PP}$. Whenever we talk about a PK-FE scheme, we implicitly assume it is single-input.

Security. Let us recall the security definition of FH-MIFE, which we restrict to a single ciphertext per client, i.e. OT security.

$\mathsf{fh\text{-}IND}^{\mathsf{MIFE}}_{\beta}(\lambda, N, \mathcal{A})$

$(\mathsf{MSK}, \{\mathsf{EK}_i\}_{i\in[N]}, \mathsf{PP}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}_{N,\pi})$

$\mathcal{MS} = \{(x_i^0, x_i^1, i)\}_{i\in[N]} \leftarrow \mathcal{A}(1^\lambda, \mathcal{F}_{N,\pi})$

$\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{EK}_i, i, x_i^\beta)\ \forall i \in [N]$

$\alpha \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{KeyGen}(\cdot,\cdot)}(\mathsf{PP}, \{\mathsf{CT}_i\}_{i\in[N]})$

if $\mathsf{adm}(\mathcal{A}) = 0$ **then**

  return α

return $\alpha \xleftarrow{\$} \{0,1\}$

Fig. 1: Security game $\mathsf{fh\text{-}IND}^{\mathsf{MIFE}}_{\beta}(\lambda, N, \mathcal{A})$.

Definition 2 (Security of OT FH-MIFE). *Let* MIFE *be an MIFE scheme for the function family $\mathcal{F}_{N,\pi}$. For $\beta \in \{0,1\}$, we define the experiment* $\mathsf{fh\text{-}IND}^{\mathsf{MIFE}}_{\beta}$ *as in Fig. 1, where* $\mathcal{O}.\mathsf{KeyGen}$ *is an oracle that outputs* $\mathsf{KeyGen}(\mathsf{MSK}, f^\beta)$ *on input* (f^0, f^1) *and* $\mathcal{O}.\mathsf{Enc}$ *an oracle returning* $\mathsf{Enc}(\mathsf{EK}_i, i, x_i^\beta)$ *for input* (x_i^0, x_i^1, i)*. The adversary $\mathcal{A}$ is admissible, denoted by* $\mathsf{adm}(\mathcal{A}) = 0$*, if the following conditions hold. Otherwise, we say that $\mathcal{A}$ is not admissible and write* $\mathsf{adm}(\mathcal{A}) = 1$*.*

– *For all $(x_i^0, x_i^1, i) \in \mathcal{MS}$, for any query (f^0, f^1) to $\mathcal{O}.\mathsf{KeyGen}$, we require that*

$$f^0(x_1^0, \ldots, x_N^0) = f^1(x_1^1, \ldots, x_N^1), \tag{3}$$

i.e. $\mathcal{A}$ cannot distinguish both games through the evaluation of the function.

Define the advantage of an adversary $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathsf{fh\text{-}IND}}_{\mathsf{MIFE},\mathcal{A}}(\lambda, N) = |\Pr[\mathsf{fh\text{-}IND}^{\mathsf{MIFE}}_0(\lambda, N, \mathcal{A}) = 1] - \Pr[\mathsf{fh\text{-}IND}^{\mathsf{MIFE}}_1(\lambda, N, \mathcal{A}) = 1]|.$$

We say that MIFE *is semi-adaptive* ot-fh-IND*-secure if for any number of clients N and any PPT adversary $\mathcal{A}$ it holds that $\mathsf{Adv}^{\mathsf{fh\text{-}IND}}_{\mathsf{MIFE},\mathcal{A}}(\lambda, N) \leq \mathsf{negl}(\lambda)$.*

Function Families. In this work, we focus on the function class $\mathcal{F}^{\mathsf{aff}}_{N,(p,\ell)}$ of affine function evaluations modulo p and two special cases, namely the function class of inner-product $\mathcal{F}^{\mathsf{IP}}_{N,(p,\ell)}$ and average functions $\mathcal{F}^{\mathsf{avg}}_{N,(p,\ell)}$, where ℓ is the length of each input vector. In $\mathcal{F}^{\mathsf{aff}}_{N,(p,\ell)}$, functions can be identified with a vector $\boldsymbol{y}$, which can be split into N fragments of size ℓ and some constant c. More precisely, we obtain $f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \sum_{i\in[N]} \langle \boldsymbol{x}_i, \boldsymbol{y}_i \rangle + c \mod p$. For IP c is always set to zero.

In contrast, for $\mathcal{F}^{\mathsf{avg}}_{N,(p,\ell)}$ it needs to hold that $\boldsymbol{y}_i = \boldsymbol{y}_j$ for all $i, j \in [N]$, i.e. each function f can be seen as a single vector $\boldsymbol{y}$ of length ℓ, where $f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \sum_{i\in[N]} \langle \boldsymbol{x}_i, \boldsymbol{y}\rangle + c \mod p$. We call this the average function class, as the sum can be seen as an average assuming some scaling inside $\boldsymbol{y}$. Note that for $N = 1$, $\mathcal{F}^{\mathsf{aff}}_{1,(p,\ell)} = \mathcal{F}^{\mathsf{avg}}_{1,(p,\ell)}$.

2.4 Inner-Product Functional Encryption Modulo p

In order to construct our FH-MIFE scheme for affine functions with unbounded output range over $\mathbb{Z}_p$, we require an IPFE scheme supporting the same output range. This essentially limits ourselves to the scheme provided by Agrawal et. al [7], to which we refer to as ALS[3].

Construction 1 (IPFE modulo a prime p). *Let m, n be positive integers, $\sigma > 0$ and τ a distribution of dimension $\ell \times m$.*

$\mathsf{Setup}(1^\lambda, \mathcal{F}^{\mathsf{IP}}_{1,(p,\ell)})$**:** *Set parameter $q = p^k$ for some integer k. Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ and $\mathbf{Z} \leftarrow \tau$. Compute $\mathbf{U} = \mathbf{AZ} \in \mathbb{Z}_q^{\ell\times n}$. Output the public and master secret keys $\mathsf{PP} := (\mathbf{A}, \mathbf{U}), \mathsf{MSK} := \mathbf{Z}$.*

$\mathsf{Enc}(\mathsf{PP}, \boldsymbol{x})$**:** *Sample $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{e}_0 \leftarrow \mathcal{D}^m_{\mathbb{Z},\sigma}$, $\boldsymbol{e}_1 \leftarrow \mathcal{D}^\ell_{\mathbb{Z},\sigma}$. Compute $\boldsymbol{c}_0 = \boldsymbol{s}\mathbf{A} + \boldsymbol{e}_0$ and $\boldsymbol{c}_1 = \boldsymbol{s}\mathbf{U} + \boldsymbol{e}_1 + p^{k-1} \cdot \boldsymbol{x}$. Output the ciphertext $\mathsf{CT} := (\boldsymbol{c}_0, \boldsymbol{c}_1)$.*

$\mathsf{KeyGen}(\mathsf{MSK}, \boldsymbol{y})$**:** *Parse $\mathsf{MSK} = \mathbf{Z}$. Set $\mathbf{z}_{\boldsymbol{y}} = \boldsymbol{y}\mathbf{Z}^\top$. Output $\mathsf{DK} := (\boldsymbol{y}, \mathbf{z}_{\boldsymbol{y}})$.*

$\mathsf{Dec}(\mathsf{DK}, \mathsf{CT})$**:** *Parse $\mathsf{DK} = (\boldsymbol{y}, \mathbf{z}_{\boldsymbol{y}})$ and $\mathsf{CT} = (\boldsymbol{c}_0, \boldsymbol{c}_1)$. Compute $\mu' = \boldsymbol{c}_1\boldsymbol{y}^\top - \boldsymbol{c}_0\mathbf{z}_{\boldsymbol{y}}^\top \mod q$. Output $\mu \in \mathbb{Z}_p$ that minimizes $|p^{k-1} \cdot \mu - \mu'|$.*

Construction 1 is adaptively IND-secure assuming the hardness of multi-hint Learning with Errors (LWE), a version of LWE which additionally provides some "hints" about the error of the LWE instance. One can instead rely on plain LWE by leveraging the rerandomization technique as in [45]. This gives us more freedom on how τ can be chosen. We omit the concrete security parameters here, as Appendix A proves the security of a slightly modified version tailored to our use case together with its corresponding parameters.

3 Fully Encrypted Machine Learning

Before we present our concrete protocol, let us briefly recall the use cases described in Section 1, based on [15,47,48]. This also yields some immediate design requirements and features for our protocol.

Although we restrict ourselves to linearizable ML models, the protocol itself is generic, provided that an FH-MIFE scheme exists which supports the respective function family of the update function.

[3] Technically, there also exist other versions, but all of them have ALS at their core.

3.1 Overview

We assume multiple clients holding sensitive data, collectively creating a horizontally partitioned database. They are willing to provide their data for research, as long as their privacy remains intact and they are not involved in any ongoing or heavy computations. In this case, clients only have to submit their data once, naturally imposing a bound on the number of required ciphertexts. On the other hand, an untrusted analyst wants to train some ML model on this data, e.g. a logistic regression model. Additionally, we assume the presence of a trusted authority that oversees the training process and manages the FE scheme, i.e. sets up the scheme and answers the decryption key queries posed by the analyst. Note that a trusted party is always needed when the clients themselves do not want to be involved in the training process.

The protocol proposed in [48] allows to train linearizable ML models over an iteration-based GD algorithm (Section 2.2) by using a noisy MIFE scheme to update the model weights. As the MIFE scheme only outputs one value per decryption key, the analyst requests one decryption key per entry of the model weight vector $\boldsymbol{\theta}$ in each iteration. This yields a total of $M + 1$ decryption key requests per iteration, where M is the number of attributes in the dataset. By evaluating the decryption key per entry, the analyst is able to compute $\boldsymbol{\theta}^{(t+1)}$ in iteration t.

To mitigate information leakage from those intermediate values, [48] use a noisy MIFE scheme, where the decryption key outputs perturbed functions. That is, instead of computing $\boldsymbol{\theta}^{(t)}[j]$ the j^{th} decryption key evaluates to $\boldsymbol{\theta}^{(t)}[j] + \boldsymbol{\nu}^{(t)}[j]$, where $\boldsymbol{\nu}^{(t)}$ is drawn from a distribution that ensures DP and is not leaked by the decryption key. However, to achieve overall DP, their approach requires the number of training iterations to be known in advance, as the privacy budget must be allocated across all iterations. Inherently, they encounter a trade-off between number of iterations, directly connected with runtime, utility and privacy. Training a small number of iterations would mean less noise added to the decryption, however could also limit utility. More importantly, it is hard to determine how many iterations need to be trained, without having insights about the underlying data itself.

We propose two main changes to the protocol to circumvent these problems.

- First, instead of protecting the intermediate values with small DP noise, we protect the decryption results through random values over a finite field, ensuring no leakage from these results. Only the final model is protected by DP noise, allowing to spend all of the privacy budget in a single iteration, which improves utility and robustness in the high privacy regime.
- Second, instead of training a predefined number of iterations, our protocol assumes a minimal and maximal iteration number as well as a convergence bound. The training process terminates once convergence is achieved after at least a minimum number of iterations, or when the maximum number of iterations has been reached. The latter one can be arbitrarily high without influencing the security or correctness of our protocol. It simply considers the trade-off between utility and training time.

In the following we want to explain our changes and the challenges that come with them in greater detail.

Protection of Intermediate Values. In contrast to previous protocols, where the intermediate values are either leaked directly or are protected only by DP noise, we do not want any leakage from those decryption results. The idea is to encrypt these intermediate results using random noise instead of DP noise. More precisely, instead of evaluating to $f(X)$ or $f(X) + \nu$ for some small value ν, the decryption algorithm outputs $f(X) + \phi \mod p$, where ϕ is random over $\mathbb{Z}_p$ and hidden inside the decryption key. As our use case only considers one ciphertext per client, this suffices as a random pad and therefore ensures that no information is leaked.

This poses two immediate requirements on the MIFE scheme used. First, the decryption algorithm needs to work over the whole group $\mathbb{Z}_p$ to enable random pad encryptions. Second, to train the parameters of the ML model over multiple iterations using GD, we need to be able to cancel out this randomness during the next iteration step to obtain convergence in the end. More specific, if we consider Eq. (2) as the requested function, the j^{th} decryption key for the $(t-1)^{th}$ round will not output $\boldsymbol{\theta}^{(t)}[j]$ but some

$$\boldsymbol{r}^{(t)}[j] = \boldsymbol{\theta}^{(t)}[j] + \boldsymbol{\phi}^{(t)}[j] \mod p. \tag{4}$$

In the next iteration, i.e. iteration t, the analyst requests an GD update function with weight $\boldsymbol{r}^{(t)}$. As the authority knows the randomness $\boldsymbol{\phi}^{(t)}$, they can instead create $M + 1$ decryption keys for the following function:

$$F^{(t+1)}_{\boldsymbol{r}^{(t)} - \boldsymbol{\phi}^{(t)}}(\boldsymbol{X}) = (1 - \alpha\Lambda)(\boldsymbol{r}^{(t)} - \boldsymbol{\phi}^{(t)}) + \frac{\alpha}{N} \sum_{i \in [N]} \nabla l((\boldsymbol{r}^{(t)} - \boldsymbol{\phi}^{(t)}), (\boldsymbol{x}_i, y_i)), \tag{5}$$

By Eq. (4), $F^{(t+1)}_{\boldsymbol{r}^{(t)} - \boldsymbol{\phi}^{(t)}}(\boldsymbol{X}) = F^{(t+1)}_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{X})$, i.e. this is exactly the update step of the GD algorithm as in Eq. (2). To protect these new intermediate results, a fresh random value is embedded into the decryption keys again.

Cancelling out the randomness inside the function values requires an FH-MIFE scheme, because otherwise the decryption key would leak the coefficients of the encoded function, which would enable the analyst to reconstruct $\boldsymbol{\theta}^{(t)}$.

In Section 4.2 we present a tailored FH-MIFE construction that meets these requirements. Note that previously no scheme existed that is both FH and works over $\mathbb{Z}_p$ as all concrete FH-MIFE schemes operate over bilinear groups, which by design need to bound the output.

Convergence. Instead of training the model for a predefined number of iterations, a more intuitive approach is to train until a convergence threshold is reached. To this end, we introduce a convergence check after a minimum number of iterations, which does not leak information to the analyst. More specific,

to check the convergence, the analyst computes the vector of differences between the intermediate results of two iterations $t+1$ and t:

$$\boldsymbol{\Delta} = \boldsymbol{r}^{(t+1)} - \boldsymbol{r}^{(t)} \in \mathbb{Z}_p^{M+1}.$$

Since the intermediate values are protected by random noise, this also ensures no leakage from $\boldsymbol{\Delta}$. The authority however can compute the overall difference between $\boldsymbol{\theta}^{(t+1)}$ and $\boldsymbol{\theta}^{(t)}$ by computing $\Delta_\theta = \|\boldsymbol{\Delta} - \boldsymbol{\phi}^{(t+1)} + \boldsymbol{\phi}^{(t)} \mod p\|_1$, where $\boldsymbol{\phi}^{(t+\beta)}$, $\beta \in \{0,1\}$, is the randomness used in the respective iteration. By comparing Δ_θ to the given convergence threshold, they can either terminate the training earlier if convergence is reached, or continue with the next iteration.

3.2 Protocol

In the following, we present our concrete protocol including all modifications as described above. The protocol is divided into two phases. First, a data gathering phase, where the analyst obtains all the ciphertexts from the clients. Second, the actual training phase, which solely involves the authority and the analyst.

Data Gathering. Analogue to [48], we first start with a data gathering phase. This requires an authority, which provides a connection platform between the analyst and the data holders. The authority first generates the keys MSK and $\{\mathsf{EK}_i\}_{i\in[N]}$ and distributes the encryption keys EK_i to each data holder i. To ensure that their privacy requirements are met, each data holder provides their privacy budget (ϵ_i, δ_i) to the authority. After receiving the encryption keys, the data holders encrypt their data and send the ciphertext CT_i to the analyst.

Training. The next step is the training of the ML model, which is depicted in Fig. 2. In each iteration the analyst requests a vector of decryption keys for Eq. (5). The authority generates the decryption keys, including fresh random noise over $\mathbb{Z}_p$ as a constant value per key. Since each round cancels out the random noise of the previous round, this is coherent with the GD algorithm presented in Section 2.2.

As the exact number of iterations does not need to be predetermined, we include convergence checks after a minimal number of iterations and stop the training after convergence is reached (or after a predefined maximal number of iterations). These convergence checks take place in each iteration after the minimal number of iterations $\mathtt{it}_{\mathtt{min}}$ is reached (see the second loop in Fig. 2) and work as follows. The analyst computes the difference $\boldsymbol{\Delta}$ between the perturbed model weights of two iterations t and $t+1$ and sends it to the authority. The authority can then compute the absolute distance between the model weights and verify whether it is below the convergence threshold. If this is the case, the training algorithm can stop early, if not, the analyst requests the next decryption key for the next iteration. After convergence is reached or the maximal number of iterations is trained, the analyst can request the release of the model weights, say in iteration T. In this case, the authority samples noise $\boldsymbol{\nu}$ from a distribution

$\mathbb{D}$, which provides DP with respect to the minimal received privacy budget (ϵ, δ) and the loss function, if desired. Then they hand out the final randomness minus $\boldsymbol{\nu}$ to the analyst, which allows the analyst to compute $\boldsymbol{\theta}^{(T)} + \boldsymbol{\nu}$ by extracting this value.

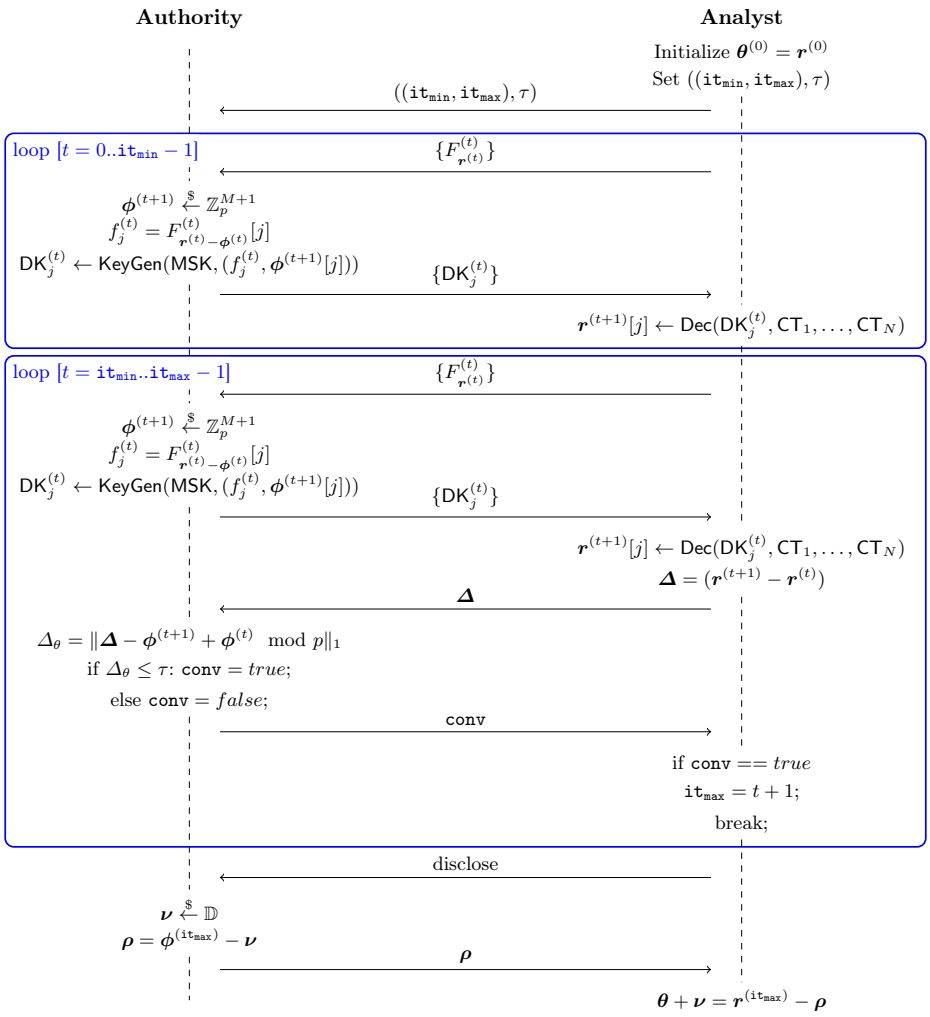


Fig. 2: Fully encrypted training phase using FH-MIFE, where $\boldsymbol{\phi}^{(0)} := \mathbf{0}$.

3.3 Threat Model and Privacy Analysis

The privacy goal of our modified protocol is to leak no intermediate results and provide DP guarantee for the participating clients with respect to their privacy

budgets. We assume honest-but-curious data holder, that are inactive after the setup, which aligns with the real world where data holder do not want to be actively involved. The analyst can behave maliciously in the sense, that they can request functions that are not limited to the training update function. However, doing so would not yield a correct training evaluation, since the randomness would not cancel out correctly. The authority needs to be trusted in the sense that they do not collude with the analyst and follow the protocol correctly.

Using an FH-MIFE scheme ensures that the ciphertexts and decryption keys do not leak any information about the underlying encoded data, beyond the output of the decryption algorithm, which can be protected by DP noise. Since only one ciphertext per client exists, no mix-and-match attacks[4] are possible. Moreover, by working over $\mathbb{Z}_p$ and including random pads in the decryption keys, our scheme ensures that the training process stays fully encrypted, thereby eliminating leakage from intermediate values. Only the last decryption is disclosed by the authority, guaranteeing that only the final model itself, possibly protected by DP, is known to the analyst.

4 Function-Hiding Multi-Input Functional Encryption from Lattices

As we have established in the previous sections, our protocol requires an FH-MIFE scheme that works over $\mathbb{Z}_p$, which did not yet exist.

If we want to support arbitrary splits of the data among the clients, our FH-MIFE would at least have to support quadratic functions. However, all quadratic MIFE schemes known to date operate over bilinear groups and so far, it remains an open question how they can be transformed to FH. More crucially, by design, bilinear groups cannot operate over the whole group $\mathbb{Z}_p$, since the output needs to be bounded to ensure decryption is possible in polynomial time.

However, if we consider the natural use case that each client is in possession of their entire record, i.e. the data is horizontally distributed, it suffices to have an FH-MIFE scheme that can evaluate affine functions. In fact, as each function coefficient is the same for each client, we can focus on building a scheme for average functions. In the following we present HIFEL, a semi-adaptive FH-MIFE scheme for average functions, supporting one encryption per client.

We will first provide a high level overview of the main ideas used to construct HIFEL, before presenting the scheme itself.

4.1 Technical Overview

Let us start by providing a technique to turn any IPFE scheme into an FH-IPFE scheme, which is bounded in either the ciphertexts or decryption keys. That is, using an IPFE scheme operating over all of $\mathbb{Z}_p$, e.g. the scheme ALS from [7]

[4] Mix-and-match attacks describe attacks, where an attacker exploits different combination of ciphertexts and decryption keys.

(Section 2.4), yields an OT FH-IPFE scheme covering all of $\mathbb{Z}_p$. In a second step, we show how this construction can be leveraged to build HIFEL, an OT FH-MIFE scheme that supports average functions, operating all over $\mathbb{Z}_p$.

Let $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ be an arbitrary IPFE scheme over $\mathbb{Z}_p$. We show how this scheme can be transformed to an OT FH scheme by leveraging a masking technique.

Instead of using the key vector $\boldsymbol{y}$ directly in the key generation algorithm, we mask it with some randomness $\boldsymbol{\zeta}$ and generate a key for $\boldsymbol{y} + \boldsymbol{\zeta} \mod p$. Additionally we restrict us to one key requests. This can be easily extended to Q queries using standard techniques.

To ensure correctness and get rid of the value $\langle \boldsymbol{x}, \boldsymbol{\zeta} \rangle$ for a message $\boldsymbol{x}$, we extend both the ciphertext and the key vector by one coordinate. In particular, we compute

$$\mathsf{CT} \leftarrow \mathsf{iEnc}((\boldsymbol{x}, -\langle \boldsymbol{x}, \boldsymbol{\zeta} \rangle \mod p)), \tag{6}$$

$$\mathsf{DK} \leftarrow \mathsf{iKeyGen}((\boldsymbol{y} + \boldsymbol{\zeta} \mod p, 1)). \tag{7}$$

If iFE is correct, then decryption yields $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. To ensure security, we utilize the security of iFE and the fact, that if we restrict the number of key queries to one, we can use $\boldsymbol{\zeta}$ once as random pad.

To construct an OT scheme that supports an unbounded number of key queries, we can exploit the symmetry of affine FH-FE schemes operating in the secret key setting. From now on, we will always assume a bound in the ciphertexts instead of the decryption keys, i.e. iKeyGen is called for encryption and iEnc for the generation of the decryption keys.

In the next step, we want to transform the scheme to an FE scheme supporting multiple inputs for the function family of average functions. The most straightforward approach, analogous to [18], would be to execute the scheme N times and embed a zero-share into the decryption keys, ensuring that correct decryption is only possible when all ciphertexts are evaluated together with their corresponding decryption keys. This construction would directly yield an FH-MIFE scheme for arbitrary inner-products. For completeness and as it might be of its own interest, we include this variant in Appendix B, together with a full security proof.

Unfortunately, this approach requires setting up the scheme N times and running the key generation algorithm N times, which in the case of ALS introduces a substantial overhead. Therefore, we now consider solution strategies that directly exploit the fact that we do not need to compute an arbitrary IP, but only an average function.

A natural way to achieve this using the OT FH scheme described above, would be to allow N ciphertexts in the following sense: For each client $i \in [N]$ build the ciphertext

$$\mathsf{CT}_i \leftarrow \mathsf{iKeyGen}((\boldsymbol{x} + \boldsymbol{\zeta}_i \mod p, 1))$$

using a different $\boldsymbol{\zeta}_i$ for all $i \in [N]$. As a decryption key generate

$$\mathsf{DK} \leftarrow \mathsf{iEnc}((\boldsymbol{y}, -N^{-1}(\sum_{i\in[N]} \langle \boldsymbol{\zeta}_i, \boldsymbol{y}\rangle - c)) \mod p), \tag{8}$$

where c is a constant coefficient of the function. The decryption algorithm now evaluates DK on each CT_i and sums up over all intermediate results. However, building this variant using ALS, leads to big parameters to ensure security, yielding a scheme that is again not efficient enough for PPML. One reason is that the modified message vectors $(\boldsymbol{x}+\boldsymbol{\zeta}_i \mod p, 1)$, which are given to $\mathsf{iKeyGen}$, are likely not linearly independent, e.g. because it is likely that the number of clients is larger than the vector length of $\boldsymbol{x}$. This can be circumvented by encoding a one-hot vector with length N into the ciphertexts. More specifically, one can change the ciphertexts to

$$\mathsf{CT}_i \leftarrow \mathsf{iKeyGen}((\boldsymbol{x} + \boldsymbol{\zeta}_i \mod p, \mathbf{1}_i^N)) \tag{9}$$

and use

$$(\boldsymbol{y}, -\langle \boldsymbol{\zeta}_1, \boldsymbol{y}\rangle + \omega_1, \ldots, -\langle \boldsymbol{\zeta}_N, \boldsymbol{y}\rangle + \omega_N) \in \mathbb{Z}_p^{|\boldsymbol{y}|+N}$$

as the key vector, where the ω_i's are randomly chosen such that $\sum_{i\in[N]} \omega_i = c$. This would automatically ensure linear independence, but also increase the vector length by N, which in turn increases all parameters of the scheme. Although this is already an improvement of the parameters, it still lacks the efficiency we are looking for.

Instead, we use a secret sharing technique to generate a master ciphertext mCT by leveraging that the sum of ALS decryption keys remains an ALS decryption key for an appropriate choice of parameters. Recall, that the $\mathsf{iKeyGen}$ algorithm of ALS outputs a tuple $(\boldsymbol{x}, \mathbf{z}_{\boldsymbol{y}})$. To build a ciphertext, each client computes

$$\begin{aligned}(\widetilde{\boldsymbol{x}}_i, \mathbf{c}_i) &\leftarrow \mathsf{iKeyGen}(\widetilde{\boldsymbol{x}} = (\boldsymbol{x}_i + \boldsymbol{\zeta}_i \mod p, 1)),\\ \mathsf{CT}_i &= (\widetilde{\boldsymbol{x}}_i, \mathbf{c}_i + \boldsymbol{\rho}_i \mod q),\end{aligned}$$

where $\boldsymbol{\rho}_i$ is part of the encryption key and $\sum_{i\in[N]} \boldsymbol{\rho}_i = 0$. By computing $\mathsf{mCT} = \sum_{i\in[N]} \mathsf{CT}_i$, we obtain one valid ALS decryption key. The decryption key for HIFEL is build as described in Eq. (8) and evaluated on mCT. This circumvents the problem of linear dependent requests, while still ensuring a small vector length.

Through this linear independence, it suffices to sample the coefficients of the master secret key, i.e. $\mathbf{Z}$, from $\{-1, 1\}$ instead of a complex and potentially large distribution to maintain a high enough min-entropy during the security proof. This significantly improves our parameters as larger coefficients of $\mathbf{Z}$ increase the overall parameters quite fast. More details on the parameter choices, as well as the corresponding security proof for the modified ALS variant, dubbed 1-ALS, can be found in Appendix A.

4.2 OT FH-MIFE for Average Functions

In the following we present HIFEL, an OT FH-MIFE scheme for average functions, which can be used to train a linearizable model according to our presented protocol.

Construction 2 (HIFEL). *Let m, n be positive integers and $q = p^k$ for some integer k and a prime p such that $N < p$.*

$\mathsf{Setup}(1^\lambda, \mathcal{F}^{\mathsf{avg}}_{N,(p,\ell)}, 1)$**:** *Sample* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, \mathbf{Z} \xleftarrow{\$} \{-1,1\}^{m\times(\ell+1)}$, $\boldsymbol{\zeta}_i \xleftarrow{\$} \mathbb{Z}_p^\ell$ *and* $\boldsymbol{\rho}_i \in \mathbb{Z}_q^m$ *for all* $i \in [N]$ *such that* $\sum_{i\in[N]} \boldsymbol{\rho}_i = \mathbf{0} \mod q$. *Set*

$$\mathbf{U} = \mathbf{AZ} \in \mathbb{Z}_q^{n\times(\ell+1)}.$$

Return $\mathsf{PP} := \mathbf{A}$, $\{\mathsf{EK}_i := (\mathbf{Z}, \boldsymbol{\zeta}_i, \boldsymbol{\rho}_i)\}_{i\in[N]}$ *and* $\mathsf{MSK} := (\{\boldsymbol{\zeta}_i\}_{i\in[N]}, \mathbf{U})$.

$\mathsf{Enc}(\mathsf{EK}_i, i, \boldsymbol{x} \in \mathbb{Z}_p^\ell)$**:** *Parse* $\mathsf{EK}_i = (\mathbf{Z}, \boldsymbol{\zeta}_i, \boldsymbol{\rho}_i)$, *set*

$$\widetilde{\boldsymbol{x}}_i = (\boldsymbol{x}_i + \boldsymbol{\zeta}_i, 1) \in \mathbb{Z}_p^{\ell+1}, \tag{10}$$

compute

$$\mathbf{c}_i = \widetilde{\boldsymbol{x}}_i \mathbf{Z}^\top + \boldsymbol{\rho}_i \in \mathbb{Z}_q^m$$

and return $\mathsf{CT}_i = (\widetilde{\boldsymbol{x}}_i, \mathbf{c}_i)$.

$\mathsf{KeyGen}(\mathsf{MSK}, (\boldsymbol{y}, c))$**:** *Sample* $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{e}_0 \leftarrow \mathcal{D}^m_{\mathbb{Z},\sigma}$, $\boldsymbol{e}_1 \leftarrow \mathcal{D}^{\ell+1}_{\mathbb{Z},\sigma}$. *Set*

$$\widetilde{\boldsymbol{y}} = (\boldsymbol{y}, -N^{-1}(\sum_{j\in[N]} \langle \boldsymbol{\zeta}_j, \boldsymbol{y}\rangle - c))\mathbb{Z}_q^{\ell+1}, \tag{11}$$

where N^{-1} *is the inverse of* N *in* $\mathbb{Z}_p$. *Calculate*

$$\mathbf{k}_0 = \boldsymbol{s}\mathbf{A} + \boldsymbol{e}_0 \in \mathbb{Z}_q^m, \ \mathbf{k}_1 = \boldsymbol{s}\mathbf{U} + \boldsymbol{e}_1 + p^{k-1}\widetilde{\boldsymbol{y}} \in \mathbb{Z}_q^{\ell+1}.$$

Return $\mathsf{DK} = (\mathbf{k}_0, \mathbf{k}_1)$.

$\mathsf{Dec}(\mathsf{DK}, \mathsf{CT}_1, \ldots, \mathsf{CT}_N)$**:** *Compute the master ciphertext as*

$$\mathsf{mCT} = \sum_{i\in[N]} \mathsf{CT}_i = (\sum_{i\in[N]} \widetilde{\boldsymbol{x}}_i, \sum_{i\in[N]} \mathbf{c}_i).$$

Parse $\mathsf{DK} = (\mathbf{k}_0, \mathbf{k}_1)$ *and* $\mathsf{mCT} = (\boldsymbol{x}, \mathbf{c})$. *Compute* $\mu' = \langle \boldsymbol{x}, \mathbf{k}_1\rangle - \langle \mathbf{c}, \mathbf{k}_0\rangle \mod q$. *Return* $\min_\mu |p^{k-1}\mu - \mu'|$.

As the construction requires the computation of the inverse of N, i.e. the number of clients, in $\mathbb{Z}_p$, we require that p does not divide N. To simplify the parameter restrictions, we assume that $N < p$, i.e. the inverse of N in $\mathbb{Z}_p$ exists.

Parameters. The security of the scheme relies directly on the security of 1-ALS with improved parameters (see Appendix A), therefore we can choose the following parameters: $n = \mathsf{poly}(\lambda)$, $m = \max\{2n\log q, 2\ell + kn + \frac{2\lambda+2\log\ell}{\log p}\}$, $\sigma = 2\alpha q\tau$, $\tau = \sqrt{\ell+1} + \sqrt{m} + \sqrt{\lambda}$, $q \geq 4p^2\alpha q\tau N(\ell+1)(1+m)$ for $4 < nk$ and $\ell \geq 4$.

More details on the restrictions of the parameters and their role in the security proof can be found in Appendix A. Note that these are the same parameters as those of 1-ALS if we set $\mathsf{B} = Np$ in the modulus q, where B is a bound imposed on the size of the 1-ALS function coefficients to ensure correctness.

Correctness. Let DK be a decryption key for the function $(\boldsymbol{y}, c)$ and $\mathsf{CT}_i = (\widetilde{\boldsymbol{x}}_i, \mathbf{c}_i)$ an encryption of $\boldsymbol{x}_i$ for all $i \in [N]$. By the design of $\mathbf{c}_i$, it holds that $\sum_{i\in[N]} \mathbf{c}_i = (\sum_{i\in[N]} \widetilde{\boldsymbol{x}}_i)\mathbf{Z}^\top$ in $\mathbb{Z}_q$. Hence, $\mathsf{mCT} = (\sum_{i\in[N]} \widetilde{\boldsymbol{x}}_i, (\sum_{i\in[N]} \widetilde{\boldsymbol{x}}_i)\mathbf{Z}^\top)$ and $(\langle \sum_{i\in[N]} \widetilde{\boldsymbol{x}}_i, \mathbf{k}_1\rangle - \langle \sum_{i\in[N]} \mathbf{c}_i, \mathbf{k}_0\rangle) = p^{k-1}(\langle \sum_{i\in[N]} \boldsymbol{x}_i, \boldsymbol{y}\rangle + c) + \boldsymbol{e}_{\mathsf{Dec}}$ in $\mathbb{Z}_q$, where $\boldsymbol{e}_{\mathsf{Dec}} = \sum_{i\in[N]} \widetilde{\boldsymbol{x}}_i \boldsymbol{e}_1^\top - \boldsymbol{e}_0(\sum_{i\in[N]} \widetilde{\boldsymbol{x}}_i)\mathbf{Z}^\top$. As $|\sum_{i\in[N]} \widetilde{\boldsymbol{x}}_i \boldsymbol{e}_1^\top| \leq \ell N\sigma p$ and $|\boldsymbol{e}_0(\sum_{i\in[N]} \widetilde{\boldsymbol{x}}_i)\mathbf{Z}^\top| \leq \ell N\sigma pm$, it holds that $|\boldsymbol{e}_{\mathsf{Dec}}| \leq \frac{1}{2}p^{k-1}$, yielding correctness.

Security. The structure of the presented Construction 2 is similar to Construction 3, i.e. 1-ALS, which is secure assuming the hardness of LWE and appropriate parameters. This enables us to prove the security of 2 by using Construction 3 as a black box. The crucial part is that the encryption and key generation algorithms are basically swapped.

Theorem 1. *Assuming that Construction 3 is adaptively* IND*-secure for one decryption key, then Construction 2 is* ot-fh-IND*-secure according to Definition 2.*

In order to prove Theorem 1, we define a sequence of hybrid games G_0 to G_5 and show that the advantage of an adversary distinguishing between two consecutive games is negligible. Denote by $\{(\boldsymbol{x}_i^0, \boldsymbol{x}_i^1)\}_{i\in[N]}$ the messages chosen by $\mathcal{A}$ per client during the challenge phase and $\widetilde{\boldsymbol{x}}_i^\beta = (\boldsymbol{x}_i^\beta + \boldsymbol{\zeta}_i, 1)$.

The sequence of games is as follows.

G_0**:** This game corresponds to the original game for $\beta = 0$.

G_1**:** This game is the same as G_0, except that $\mathbf{c}_1 = (\sum_{i\in[N]} \widetilde{\boldsymbol{x}}_i^0)\mathbf{Z}^\top + \boldsymbol{\rho}_1$ and $\mathbf{c}_i = \boldsymbol{\rho}_i$ for all $i \in [2, N]$.

G_2**:** This game is the same as G_1, except that during KeyGen, $\widetilde{\boldsymbol{y}} \in \mathbb{Z}_p^{\ell+1}$ is set as

$$\widetilde{\boldsymbol{y}}[i] = \begin{cases} 0, & i \in [\ell], \\ N^{-1}(\sum_{j\in[N]} \langle \boldsymbol{x}_j^0, \boldsymbol{y}^0\rangle + c^0), & i = \ell + 1. \end{cases}$$

G_3**:** This game is the same as G_2, except that for all $i \in [N]$, $\widetilde{\boldsymbol{x}}_i^1$ is used instead of $\widetilde{\boldsymbol{x}}_i^0$ in $\mathbf{c}_1$.

G_4**:** This game is the same as G_3, except that during KeyGen, $\widetilde{\boldsymbol{y}} \in \mathbb{Z}_p^{\ell+1}$ is set as

$$\widetilde{\boldsymbol{y}}[i] = \begin{cases} \boldsymbol{y}^1[i], & i \in [\ell], \\ -N^{-1}(\sum_{j\in[N]} \langle \boldsymbol{\zeta}_j, \boldsymbol{y}^1\rangle - c^1), & i = \ell + 1. \end{cases}$$

G_5**:** This game corresponds to the original game for $\beta = 1$.

Proof. Note that the sequence of games depicts the advantage of an adversary in distinguishing between $\text{ot-fh-}\mathsf{IND}_0^{\mathsf{MIFE}}(\lambda, N, \mathcal{A})$ and $\text{ot-fh-}\mathsf{IND}_1^{\mathsf{MIFE}}(\lambda, N, \mathcal{A})$. The advantage of an adversary in distinguishing between G_0 and G_4 can be bounded using Lemma 1 to 4, which is negligible assuming the security of Construction 3 and thus $\mathsf{LWE}_{q,n,\alpha}$. Lastly, the advantage to distinguish between G_4 and G_5 is the same as going in reverse from G_4 to G_0, which is negligible by the reasoning above. Thus, the theorem is concluded.

Lemma 1. *For any PPT adversary $\mathcal{A}$, it holds that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{G}_0 \leftrightarrow \mathsf{G}_1}(\lambda) = 0$.*

Proof. The difference between both games is the generation of the $\mathbf{c}_i$ values. An adversary has to distinguish between

$$(\widetilde{\boldsymbol{x}}_1 \mathbf{Z}^\top + \boldsymbol{\rho}_1 \mod q, \ldots, \widetilde{\boldsymbol{x}}_N \mathbf{Z}^\top + \boldsymbol{\rho}_N \mod q) \tag{12}$$

and

$$((\sum_{i \in [N]} \widetilde{\boldsymbol{x}}_i)\mathbf{Z}^\top + \boldsymbol{\rho}_1 \mod q, \boldsymbol{\rho}_2, \ldots, \boldsymbol{\rho}_N). \tag{13}$$

However, both (12) and (13) are uniformly distributed over $\mathbb{Z}_q^{mN}$ as the $\boldsymbol{\rho}_i$'s are sampled uniformly at random over $\mathbb{Z}_q^m$.

Lemma 2. *For any PPT adversary $\mathcal{A}$ distinguishing G_1 and G_2, there exists a PPT adversary $\mathcal{B}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{G}_1 \leftrightarrow \mathsf{G}_2}(\lambda) \leq \mathsf{Adv}_{\mathcal{B}}^{1\text{-}\mathsf{ALS}}(\lambda)$, where ALS has function coefficient bound $\mathsf{B} = Np$.*

Proof. Let $\mathcal{B}$ be an adversary against 1-ALS for the family $\mathcal{F}_{1,(p,\ell)}^{\mathsf{IP}}$ with function coefficient bound $\mathsf{B} = Np$. $\mathcal{B}$ simulates $\mathcal{A}$'s view as follows.

Upon receiving the public parameters $(\mathbf{A}, \mathbf{U})$ from their own oracle and $\{(\boldsymbol{x}^0, \boldsymbol{x}^1)\}$ from $\mathcal{A}$, $\mathcal{B}$ samples $\boldsymbol{\rho}_i \xleftarrow{\$} \mathbb{Z}_q^m$ such that $\sum_{i \in [N]} \boldsymbol{\rho}_i = \mathbf{0}$, $\boldsymbol{\zeta}_i \xleftarrow{\$} \mathbb{Z}_p^\ell$ for all $i \in [N]$.

To generate the ciphertext for client 1, $\mathcal{B}$ computes as $\widetilde{\boldsymbol{x}}_i^0 = (\boldsymbol{x}_i^0 + \boldsymbol{\zeta}_i, 1) \mod p$ and $\sum_{i \in [N]} \widetilde{\boldsymbol{x}}_i^0$. They query their own KeyGen oracle for $\sum_{i \in [N]} \widetilde{\boldsymbol{x}}_i^0$ to obtain $(\sum_{i \in [N]} \widetilde{\boldsymbol{x}}_i^0, (\sum_{i \in [N]} \widetilde{\boldsymbol{x}}_i^0)\mathbf{Z}^\top)$. They set $\mathsf{CT}_1 = (\widetilde{\boldsymbol{x}}_1^0, (\sum_{i \in [N]} \widetilde{\boldsymbol{x}}_i^0)\mathbf{Z}^\top + \boldsymbol{\rho}_1)$ and $\mathsf{CT}_i = (\widetilde{\boldsymbol{x}}_i^0, \boldsymbol{\rho}_i)$ for all $i \geq 2$. The set of ciphertexts $\{\mathsf{CT}_i\}_{i \in [N]}$ is given to $\mathcal{A}$.

Whenever $\mathcal{A}$ poses a decryption key query $((\boldsymbol{y}^0, c^0), (\boldsymbol{y}^1, c^1))$, $\mathcal{B}$ sets $\widetilde{\boldsymbol{y}}^0$ according to (11) and $\widetilde{\boldsymbol{y}}^1 \in \mathbb{Z}_p^{\ell+1}$ as

$$\widetilde{\boldsymbol{y}}^1[i] = \begin{cases} 0, & i \in [\ell], \\ N^{-1}(\sum_{j \in [N]} \langle \boldsymbol{x}_j^0, \boldsymbol{y}^0 \rangle - c^0), & i = \ell + 1. \end{cases}$$

They query their own encryption oracle for $(\widetilde{\boldsymbol{y}}^0, \widetilde{\boldsymbol{y}}^1)$ and submit the result to $\mathcal{A}$.

In the end, they output the same guess as $\mathcal{A}$. As $\mathcal{B}$ perfectly simulates $\mathcal{A}$'s view and is admissible due to $\mathcal{A}$'s admissibility and the fact that they can perfectly simulate all $\mathsf{CT}_i, i \neq 1$, the theorem is concluded.

Lemma 3. *For any PPT adversary $\mathcal{A}$, it holds that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{G}_2 \leftrightarrow \mathsf{G}_3}(\lambda) = 0$.*

Proof. This lemma can be proven using a straightforward statistical argument. By the previous game, it holds that the first ℓ slots of $\widetilde{\boldsymbol{x}}_i$ are multiplied by zero during decryption and thus do not contribute to the final result. As $\boldsymbol{\zeta}_i \xleftarrow{\$} \mathbb{Z}_p^\ell$ is independent and uniform over $\mathbb{Z}_p^\ell$ for all $i \in [N]$, it holds that $\widetilde{\boldsymbol{x}}_i$ is statistically indistinguishable from a vector where the first ℓ slots are uniform over $\mathbb{Z}_p^\ell$. More precisely, for all $i \in [N]$, we have that $\boldsymbol{x}_i^0 + \boldsymbol{\zeta}_i \equiv \boldsymbol{\zeta}_i \equiv \boldsymbol{x}_i^1 + \boldsymbol{\zeta}_i$ over $\mathbb{Z}_p^\ell$, i.e. the adversary has no advantage in distinguishing between the two games.

Table 1: Number of trained iterations T, with $\mathtt{it}_{\mathtt{max}} = 50$, and runtime for a logistic regression training on different datasets consisting of N records and M attributes, together with used LWE parameters.

Dataset	N	M	T	n	m	q	σ	**Time (min)**
LBW	189	8	34	1728	414720	10000019^5	118130195237.6527	48.77
PCS	380	7	27	1760	422400	10000019^5	70796420456.999	36.78
UIS	575	7	34	1760	422400	10000019^5	46787199606.3646	47.07

Lemma 4. *For any PPT adversary $\mathcal{A}$ distinguishing G_3 and G_4, there exists a PPT adversary $\mathcal{B}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{G}_3 \leftrightarrow \mathsf{G}_4}(\lambda) \leq \mathsf{Adv}_{\mathcal{B}}^{1\text{-}\mathsf{ALS}}(\lambda)$.*

Proof. Note that this is basically the reverse of Lemma 1 and thus the proof follows analogue.

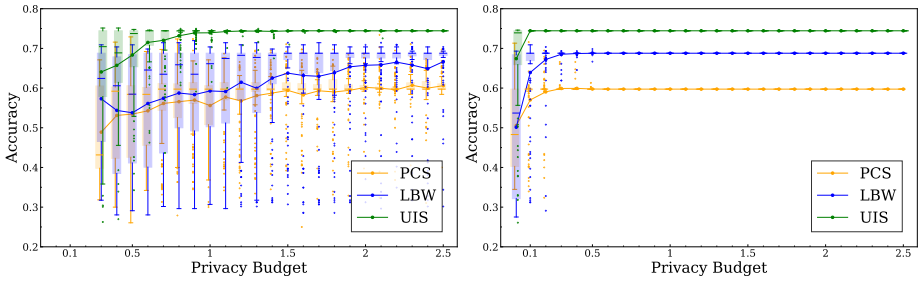
5 Implementation and Results

To show the feasibility of our approach we evaluate it regarding efficiency and utility on three different standard datasets for PPML: Low birth weight (LBW) study [31], prostate cancer study (PCS) [32] and umaru impact study (UIS) [33]. The training was conducted on a virtual machine with access to 425 GB of DDR-5 memory and 22 CPU threads. The security parameter used for each dataset can be found in Table 1, yielding a security level of at least 80 bits as estimated by [9].

We train the logistic regression model using our proposed protocol (Sec. 3) setting the minimal training iterations to 25 and the maximal to 50. The training terminates after convergence of $\Delta_\theta < 10^{-3}$ is reached and the final result is perturbed using the Gaussian mechanism [11], i.e. noise drawn from a Gaussian distribution using the sensitivity $\frac{\sqrt{M+1}}{N\Lambda}$, which is the sensitivity of the loss function for logistic regression with respect to the domain of $\boldsymbol{x}$ [16]. This is enough to achieve DP for a given privacy budget (ϵ, δ). Table 1 shows the number of trained iterations T until convergence and the resulting runtime of the training phase per dataset. For all datasets the setup took about 15 minutes, encrypting all existing records including generating the master ciphertext mCT, i.e. summing up all ciphertexts, which took between 2-6 min, depending on the number of records N in the dataset. As mCT is the same throughout all decryptions, this step needs to be conducted only once.

The utility of our approach is shown in 3b, in dependency of the privacy budget.

Although our approach is based on [48], who have a comparable runtime to us (48-150 min), they allow for an arbitrarily split data set, whereas we require a horizontally split data set. Therefore, we compare ourselves to [39], who have the same framework but also assume horizontally split data. Even though they surpass our runtime as their training takes under a minute for each dataset, Fig. 3a shows that their approach has a very high variance for small



(a) DP gradient perturbation [39].

(b) Our approach.

Fig. 3: Accuracy of logistic regression model trained on different datasets with varying privacy budget ϵ starting with $\epsilon = 0.01$ and fixed $\delta = \frac{1}{N}$. Blank spaces mean that the model did not converge.

ϵ. As training is only conducted once, it is unsure beforehand whether a good model is obtained or not. If the training is repeated to obtain a more stable model, the privacy budget needs to be split even further, which impacts the accuracy again. In contrast, as shown in Fig. 3b, our approach using HIFEL only has a high variance for $\epsilon = 0.01$ and can be seen as robust for $\epsilon \geq 0.1$. In other words, with our approach it is possible to obtain high privacy guarantees while still achieving good and robust model utility.

6 Conclusion

In this work, we present a fully encrypted PPML protocol to train linearizable ML models using GD. To show the applicability of our protocol, we present HIFEL, an OT FH-MIFE scheme for average functions, which can be extended to a scheme for affine functions. This is the first lattice based FH-MIFE scheme for average functions and the first with an unbounded output range over $\mathbb{Z}_p$.

Our protocol can also be used to train more complex model, provided that one designs more expressive FH-MIFE schemes. In order to enhance the efficiency, one could use further optimizations for the matrix multiplications or design a scheme based on ring LWE instead of plain LWE. This requires to develop new insights on rings to lift ALS for $\mathbb{Z}_p$ to a ring version, especially if we want to maintain semi-adaptive security. One major problem is that the decisional version of ring LWE requires a prime q, whereas we need $q = p^k$ for some $k > 1$. We leave these points open for future research.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Abadi, M., Chu, A., Goodfellow, I.J., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 308–318. ACM Press (Oct 2016). https://doi.org/10.1145/2976749.2978318
2. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Berlin, Heidelberg (Mar / Apr 2015). https://doi.org/10.1007/978-3-662-46447-2_33
3. Abdalla, M., Catalano, D., Fiore, D., Gay, R., Ursu, B.: Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 597–627. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1_20
4. Abdalla, M., Catalano, D., Gay, R., Ursu, B.: Inner-product functional encryption with fine-grained access control. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 467–497. Springer, Cham (Dec 2020). https://doi.org/10.1007/978-3-030-64840-4_16
5. Abdalla, M., Gay, R., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 601–626. Springer, Cham (Apr / May 2017). https://doi.org/10.1007/978-3-319-56620-7_21
6. Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption from pairings. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 208–238. Springer, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84259-8_8
7. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 333–362. Springer, Berlin, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53015-3_12
8. Agrawal, S., Rosen, A.: Functional encryption for bounded collusions, revisited. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 173–205. Springer, Cham (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_7
9. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9**(3), 169–203 (Oct 2015). https://doi.org/10.1515/jmc-2015-0016
10. Ananth, P., Vaikuntanathan, V.: Optimal bounded-collusion secure functional encryption. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part I. LNCS, vol. 11891, pp. 174–198. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-36030-6_8
11. Balle, B., Wang, Y.X.: Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 394–403. PMLR (10–15 Jul 2018), https://proceedings.mlr.press/v80/balle18a.html
12. Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 67–98. Springer, Cham (Aug 2017). https://doi.org/10.1007/978-3-319-63688-7_3

13. Bishop, A., Jain, A., Kowalczyk, L.: Function-hiding inner product encryption. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 470–491. Springer, Berlin, Heidelberg (Nov / Dec 2015). https://doi.org/10.1007/978-3-662-48797-6_20
14. Carpov, S., Fontaine, C., Ligier, D., Sirdey, R.: 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)
15. Chang, Y., Zhang, K., Gong, J., Qian, H.: Privacy-preserving federated learning via functional encryption, revisited. IEEE Transactions on Information Forensics and Security **18**, 1855–1869 (2023). https://doi.org/10.1109/TIFS.2023.3255171
16. Chaudhuri, K., Monteleoni, C., Sarwate, A.D.: Differentially private empirical risk minimization. Journal of Machine Learning Research **12**(3) (2011)
17. Datta, P., Dutta, R., Mukhopadhyay, S.: Functional encryption for inner product with full function privacy. In: Cheng, C.M., Chung, K.M., Persiano, G., Yang, B.Y. (eds.) PKC 2016, Part I. LNCS, vol. 9614, pp. 164–195. Springer, Berlin, Heidelberg (Mar 2016). https://doi.org/10.1007/978-3-662-49384-7_7
18. Datta, P., Okamoto, T., Tomida, J.: Full-hiding (unbounded) multi-input inner product functional encryption from the k-Linear assumption. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 245–277. Springer, Cham (Mar 2018). https://doi.org/10.1007/978-3-319-76581-5_9
19. Ducas, L., Micciancio, D.: Improved short lattice signatures in the standard model. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 335–352. Springer, Berlin, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2_19
20. Dufour-Sans, E., Gay, R., Pointcheval, D.: Reading in the dark: Classifying encrypted digits with functional encryption. Cryptology ePrint Archive (2018)
21. Dwork, C.: Differential privacy. In: International colloquium on automata, languages, and programming. pp. 1–12. Springer (2006)
22. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Foundations and trends® in theoretical computer science **9**(3-4), 211–487 (2014)
23. Ernst, J., Mitrokotsa, A.: A framework for UC secure privacy preserving biometric authentication using efficient functional encryption. In: Tibouchi, M., Wang, X. (eds.) ACNS 2023, Part II. LNCS, vol. 13906, pp. 167–196. Springer, Cham (Jun 2023). https://doi.org/10.1007/978-3-031-33491-7_7
24. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Berlin, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_11
25. Hong, S., Kim, J., Lee, C., Seo, M.: Non-interactive fully encrypted machine learning protocol for inference. Cryptology ePrint Archive, Paper 2024/1859 (2024), https://eprint.iacr.org/2024/1859
26. Ioniţă, A., Ioniţă, A.: Functional encryption in secure neural network training: Data leakage and practical mitigations (2025), https://arxiv.org/abs/2509.21497
27. Katsumata, S., Yamada, S.: Partitioning via non-linear polynomial functions: More compact IBEs from ideal lattices and bilinear maps. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 682–712. Springer, Berlin, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53890-6_23
28. Kim, S., Lewi, K., Mandal, A., Montgomery, H., Roy, A., Wu, D.J.: Function-hiding inner product encryption is practical. In: Catalano, D., De Prisco, R. (eds.) SCN 18. LNCS, vol. 11035, pp. 544–562. Springer, Cham (Sep 2018). https://doi.org/10.1007/978-3-319-98113-0_29

29. Lai, Q., Liu, F.H., Wang, Z.: New lattice two-stage sampling technique and its applications to functional encryption - stronger security and smaller ciphertexts. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 498–527. Springer, Cham (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_18
30. Ligier, D., Carpov, S., Fontaine, C., Sirdey, R.: Privacy preserving data classification using inner-product functional encryption. In: International Conference on Information Systems Security and Privacy. vol. 2, pp. 423–430. SciTePress (2017)
31. LogisticDx: Diagnostic tests for models with a binomial response. lbw: Low birth weight study data (6 2024), https://rdrr.io/rforge/LogisticDx/man/lbw.html
32. LogisticDx: Diagnostic tests for models with a binomial response. pcs: Prostate cancer study data (6 2024), https://rdrr.io/rforge/LogisticDx/man/pcs.html
33. LogisticDx: Diagnostic tests for models with a binomial response. uis: Umaru impatct study data (6 2024), https://rdrr.io/rforge/LogisticDx/man/uis.html
34. Mera, J.M.B., Karmakar, A., Marc, T., Soleimanian, A.: Efficient lattice-based inner-product functional encryption. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022, Part II. LNCS, vol. 13178, pp. 163–193. Springer, Cham (Mar 2022). https://doi.org/10.1007/978-3-030-97131-1_6
35. Panzade, P., Takabi, D.: Fenet: Privacy-preserving neural network training with functional encryption. In: Proceedings of the 9th ACM International Workshop on Security and Privacy Analytics. pp. 33–43 (2023)
36. Qian, X., Li, H., Hao, M., Yuan, S., Zhang, X., Guo, S.: Cryptofe: Practical and privacy-preserving federated learning via functional encryption. In: GLOBECOM 2022-2022 IEEE Global Communications Conference. pp. 2999–3004. IEEE (2022)
37. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 84–93. ACM Press (May 2005). https://doi.org/10.1145/1060590.1060603
38. Ryffel, T., Dufour-Sans, E., Gay, R., Bach, F., Pointcheval, D.: Partially encrypted machine learning using functional encryption. arXiv preprint arXiv:1905.10214 (2019)
39. Scheu-Hachtel, L., Zalonis, J.: Enhancing noisy functional encryption for privacy-preserving machine learning. In: 2025 Annual Computer Security Applications Conference (ACSAC). IEEE (2025)
40. Tomida, J.: Tightly secure inner product functional encryption: Multi-input and function-hiding constructions. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 459–488. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-34618-8_16
41. Tomida, J.: Unbounded quadratic functional encryption and more from pairings. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part III. LNCS, vol. 14006, pp. 543–572. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30620-4_18
42. Tomida, J., Abe, M., Okamoto, T.: Efficient functional encryption for inner-product values with full-hiding security. In: Bishop, M., Nascimento, A.C.A. (eds.) ISC 2016. LNCS, vol. 9866, pp. 408–425. Springer, Cham (Sep 2016). https://doi.org/10.1007/978-3-319-45871-7_24
43. Tomida, J., Takashima, K.: Unbounded inner product functional encryption from bilinear maps. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 609–639. Springer, Cham (Dec 2018). https://doi.org/10.1007/978-3-030-03329-3_21

44. Ünal, A.: Impossibility results for lattice-based functional encryption schemes. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 169–199. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45721-1_7
45. Wang, Z., Fan, X., Liu, F.H.: FE for inner products and its application to decentralized ABE. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 97–127. Springer, Cham (Apr 2019). https://doi.org/10.1007/978-3-030-17259-6_4
46. Xu, R., Joshi, J.B., Li, C.: Cryptonn: Training neural networks over encrypted data. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). pp. 1199–1209. IEEE (2019)
47. Zalonis, J., Armknecht, F., Scheu-Hachtel, L.: Differentially private functional encryption. Proceedings on Privacy Enhancing Technologies (2024)
48. Zalonis, J., Scheu-Hachtel, L., Armknecht, F.: A new quadratic noisy functional encryption scheme and its application for privacy preserving machine learning. In: Fischlin, M., Moonsamy, V. (eds.) ACNS 2025, Part III. LNCS, vol. 15827, pp. 220–250. Springer, Cham (Jun 2025). https://doi.org/10.1007/978-3-031-95767-3_9

A 1-ALS

In this section we will provide some insight on 1-ALS, which supports one decryption key request and is used in our OT FH-MIFE scheme. This variant will also be used in the more general construction of FH-MIFE for IP, which is why we prove a more general version than needed for Construction 2. In contrast to the standard ALS over $\mathbb{Z}_p$, we will rely on the rerandomization technique [27] during the security proof, which gives a broader variety on how $\mathbf{Z}$ is chosen. In fact, we can choose $\mathbf{Z} \xleftarrow{\$} \{-1, 1\}^{m \times \ell}$, which suffices to obtain a high enough min-entropy.

To support the generation of mCT in our final construction, we need to ensure that mCT is a valid decryption key for 1-ALS. In other words, we need to ensure that $\sum_{i \in [N]} \widetilde{\boldsymbol{x}}_i$ is a valid input for the ALS decryption key algorithm. As each coefficient $\widetilde{\boldsymbol{x}}_i$ is smaller than p, each coefficient of $\sum_{i \in [N]} \widetilde{\boldsymbol{x}}_i$ is smaller than Np and a valid function vector request for the underlying 1-ALS scheme if it allows decryption keys for function vectors whose coefficients are not greater than Np. More generally, in our ALS variant, we bound the coefficients of the key vector with a bound B, i.e. for our Construction 2 it needs to hold that $Np \leq \mathsf{B}$. To facilitate the min-entropy requirement, we further assume $\mathsf{B} < p^2$. This does not influence our use case as $p \gg N$ to obtain meaningful results.

Finally, we will also require that the function key vector contains at least one non-zero element. Note that our plaintexts and function vector are all of size $\ell + 1$ to be directly coherent with our OT FH-MIFE construction for the parameter selection.

Before we present 1-ALS, let us first state a lemma regarding the rerandomization technique.

Lemma 5 (Noise Rerandomization [4,27]). *Let $\mathbf{V} \in \mathbb{Z}^{t \times m}$ and $\tau > s_1(\mathbf{V})$. Then there exists a procedure $\mathsf{NoiseGen}(\mathbf{V}, \tau)$ such that the distributions $\boldsymbol{e}\mathbf{V} + \boldsymbol{e}'$ with $\boldsymbol{e} \leftarrow \mathcal{D}^t_{\mathbb{Z},\sigma}, \boldsymbol{e}' \leftarrow \mathsf{NoiseGen}(\mathbf{V}, \tau)$ and $\boldsymbol{e} \leftarrow \mathcal{D}^m_{\mathbb{Z},2\sigma\tau}$ are statistically close.*

Construction 3 (1-ALS). *Let m, n be positive integers, $\mathsf{st} := \emptyset$, $\sigma > 0$ and B a bound on the coefficients of the key vector.*

$\mathsf{Setup}(1^\lambda, \mathcal{F}^{\mathsf{IP}}_{1,(p,\ell+1)})$**:** *Set parameter $q = p^k$ for some integer k. Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ and $\mathbf{Z} \xleftarrow{\$} \{-1,1\}^{m\times(\ell+1)}$. Compute $\mathbf{U} = \mathbf{AZ} \in \mathbb{Z}_q^{(\ell+1)\times n}$. Output the public and master secret keys $\mathsf{PP} := (\mathbf{A}, \mathbf{U}), \mathsf{MSK} := \mathbf{Z}$.*

$\mathsf{Enc}(\mathsf{PP}, \boldsymbol{x})$**:** *Parse $\mathsf{PP} = (\mathbf{A}, \mathbf{U})$. Sample $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{e}_0 \leftarrow \mathcal{D}^m_{\mathbb{Z},\sigma}$, $\boldsymbol{e}_1 \leftarrow \mathcal{D}^{\ell+1}_{\mathbb{Z},\sigma}$. Compute $\mathbf{c}_0 = \boldsymbol{s}\mathbf{A} + \boldsymbol{e}_0$ and $\mathbf{c}_1 = \boldsymbol{s}\mathbf{U} + \boldsymbol{e}_1 + p^{k-1} \cdot \boldsymbol{x}$.*
Output the ciphertext $\mathsf{CT} := (\mathbf{c}_0, \mathbf{c}_1)$.

$\mathsf{KeyGen}(\mathsf{MSK}, \boldsymbol{y})$**:** *Parse $\mathsf{MSK} = \mathbf{Z}$ and set $\mathbf{z}_{\boldsymbol{y}} = \boldsymbol{y}\mathbf{Z}^\top$.*
Output $\mathsf{DK} := (\boldsymbol{y}, \mathbf{z}_{\boldsymbol{y}})$.

$\mathsf{Dec}(\mathsf{DK}, \mathsf{CT})$**:** *takes as input the decryption key DK and ciphertext CT, Compute $\mu' = \mathbf{c}_1\widetilde{\boldsymbol{y}}^\top - \mathbf{c}_0\mathbf{z}_{\boldsymbol{y}}^\top \mod q$. Output $\mu \in \mathbb{Z}_p$ that minimizes $|p^{k-1} \cdot \mu - \mu'|$.*

Parameters Setting. To guarantee both correctness and security, the parameters must satisfy the following constraints:

1. The final magnitude of decryption error must be less than $\frac{1}{2}p^{k-1}$ for the correctness, i.e. $q \geq 2p\sigma\mathsf{B}(\ell+1)(1+m)$
2. To ensure the hardness of $\mathsf{LWE}_{q,n,\alpha}$, where q is the modulus, n is the size of the secret vector and α the standard deviation of the error distribution, we require $\alpha q \geq \Omega(\sqrt{n})$ [37].
3. We require $\tau > s_1(\mathbf{Z})$ for Lemma 5. By [19], the spectral norm $s_1(\mathbf{Z})$ is bounded by $\sqrt{\ell+1} + \sqrt{m} + \sqrt{\lambda}$.
4. To ensure a high enough min-entropy, $m \geq \max\{2n\log q, 4\ell + kn + \frac{2\lambda+2\log\ell}{\log p}\}$ and $4 < nk$ and $\ell \geq 4$

Our parameters could be chosen as: $n = \mathsf{poly}(\lambda)$, $m = \max\{2n\log q, 2\ell + kn + \frac{2\lambda+2\log\ell}{\log p}\}$, $\tau = \sqrt{\ell+1} + \sqrt{m} + \sqrt{\lambda}$, $\sigma = 2\alpha q\tau$, $q \geq 4p\alpha q\tau\mathsf{B}(\ell+1)(1+m)$ for $4 < nk$ and $\ell \geq 4$.

Correctness. For $\boldsymbol{y}$, we have that $\mu' = p^{k-1}(\langle \boldsymbol{x}, \boldsymbol{y}\rangle \mod p) + \left(\boldsymbol{e}_1\boldsymbol{y}^\top - \boldsymbol{e}_0\mathbf{z}_{\boldsymbol{y}}^\top\right) \mod q$. This yields the correct result if $|\boldsymbol{e}_1\boldsymbol{y}^\top - \boldsymbol{e}_0\mathbf{z}_{\boldsymbol{y}}^\top| \leq \frac{1}{2}p^{k-1}$, which is given by our choice of parameters.

Security. Construction 3 is adaptively IND-secure for one decryption key query under the $\mathsf{LWE}_{q,n,\alpha}$ assumption. The difference between this security notion and Definition 2 is that the adversary can call on $\mathcal{O}.\mathsf{KeyGen}$ at any time during the game, but only once. As the encryption algorithm only requires knowledge of PP, it suffices to show indistinguishability for one challenge message to conclude the same for multiple messages.

Theorem 2. *Assuming the hardness of the $\mathsf{LWE}_{q,n,\alpha}$ assumption, Construction 3 is adaptively IND-secure for one decryption key query for the parameters above.*

We prove the theorem by a sequence of games.

G_1**:** This is the original security game.

G_2**:** This game is the same as G_1, except that the challenge ciphertext is generated as follows. Sample $\boldsymbol{e} \leftarrow \mathcal{D}^m_{\mathbb{Z},\alpha q}$ and set $\mathbf{c} = \boldsymbol{s}\mathbf{A} + \boldsymbol{e}$. Set $\mathbf{V} = (\mathbf{I}_m || \mathbf{Z}) \in \mathbb{Z}^{m\times(m+\ell+1)}$, where $\mathbf{I}_m$ is the identity matrix of size m. Calculate $\mathbf{c}' = \mathbf{c}\mathbf{V} + \mathsf{NoiseGen}(\mathbf{V}, \tau)$ and split it into two parts $\mathbf{c}_0 \in \mathbb{Z}^m_q$ and $\mathbf{c}_1 \in \mathbb{Z}^{\ell+1}_q$. Return $\mathsf{CT} = (\mathbf{c}_0, \mathbf{c}'_1 = \mathbf{c}_1 + p^{k-1}\boldsymbol{x}^\beta)$.

G_3**:** This game is the same as G_2, except that during the generation of the challenge ciphertext, $\mathbf{c}$ is sampled uniformly at random from $\mathbb{Z}^m_q$.

The following lemmas bound the advantage of a PPT adversary $\mathcal{A}$ in distinguishing between two consecutive games from G_1 to G_3. In G_3, we argue that the probability of an adversary winning this game is close to $\frac{1}{2}$. As the advantage is negligible in each step for the given parameters and assuming the hardness of $\mathsf{LWE}_{q,n,\alpha}$, the overall advantage of $\mathcal{A}$ is also negligible and the theorem follows.

Lemma 6. *For all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{G}_1\leftrightarrow\mathsf{G}_2}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.*

Proof. The claim holds due to the indistinguishability provided by Lemma 5 and the choice of parameters. Sample $\boldsymbol{e} \leftarrow \mathcal{D}^m_{\mathbb{Z},\alpha q}$. Compute $\mathbf{c} = \boldsymbol{s}\mathbf{A} + \boldsymbol{e}$ and set $\mathbf{V} = (\mathbf{I}_m || \mathbf{Z}) \in \mathbb{Z}^{m\times(m+\ell+1)}$, where $\mathbf{I}_m$ is the identity matrix of size m. By [19], it holds that $s_1(\mathbf{Z}) < \sqrt{\ell+1} + \sqrt{m} + \sqrt{\lambda}$ with probability at least $1 - \frac{1}{\exp(2\pi\lambda)}$, as $\mathbf{Z}$ is a subgaussian matrix with parameter $\sqrt{2\pi}$. Thus, $s_1(\mathbf{V}) < \sqrt{(\sqrt{\ell+1} + \sqrt{m} + \sqrt{\lambda})^2 + 1}$, meaning that $\tau > s_1(\mathbf{V})$ with high probability.

By computing $\mathbf{c}' = \mathbf{c}\mathbf{V} + \mathsf{NoiseGen}(\mathbf{V}, \tau)$ and splitting it into two parts $\mathbf{c}_0 \in \mathbb{Z}^m_q$ and $\mathbf{c}_1 \in \mathbb{Z}^{\ell+1}_q$, the distribution of $(\mathbf{c}_0, \mathbf{c}_1)$ is statistically close to the distribution of $(\boldsymbol{s}\mathbf{A}+\boldsymbol{e}_0, \boldsymbol{s}\boldsymbol{u}+\boldsymbol{e}_1)$ by Lemma 5, where $\boldsymbol{e}_0 \leftarrow \mathcal{D}^m_{\mathbb{Z},\sigma}$ and $\boldsymbol{e}_1 \leftarrow \mathcal{D}^{\ell+1}_{\mathbb{Z},\sigma}$ for $\sigma = 2\alpha q\tau$. Therefore, to simulate the challenge ciphertext correctly, all that remains is to add $p^{k-1}\boldsymbol{x}^0$ to $\mathbf{c}_1$.

Lemma 7. *For all PPT adversaries $\mathcal{A}$, there exists an adversary $\mathcal{B}$ against $\mathsf{LWE}_{q,n,\alpha}$ such that $\mathsf{Adv}^{\mathsf{G}_2\leftrightarrow\mathsf{G}_3}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{LWE}_{q,n,\alpha}}_{\mathcal{B}}$.*

Proof. Changing $\mathbf{c} = \boldsymbol{s}\mathbf{A} + \boldsymbol{e}$ for $\boldsymbol{e} \leftarrow \mathcal{D}^m_{\mathbb{Z},\alpha q}$ to a uniformly random $\mathbf{c} \leftarrow \mathbb{Z}^m_q$ is exactly the $\mathsf{LWE}_{q,n,\alpha}$ assumption and thus the claim follows.

Lemma 8. *For all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{G}_3}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.*

Proof. Denote by G^*_3 the selective version of G_3, where the adversary has to pose the key queries before the challenge ciphertext is generated. Using a complexity leveraging (CL) argument, it suffices to prove that $\mathsf{Adv}^{\mathsf{G}^*_3}_{\mathcal{A}^*}(\lambda) \leq \mathsf{B}^{-\ell} \cdot \mathsf{negl}(\lambda)$. In the following, we will use that $\mathsf{B} < p^2$ by assumption.

Let $\boldsymbol{y}$ be the key query posed by $\mathcal{A}^*$ and define the matrix $\mathbf{Y}_{\mathsf{top}} = \widetilde{\boldsymbol{y}} \in \mathbb{Z}^{\ell+1}_p$. As $\mathbf{Y}_{\mathsf{top}}$ contains at least one non-zero element, $\mathbf{Y}_{\mathsf{top}}$ has rank 1. Hence, we can find an orthogonal matrix $\mathbf{Y}_{\mathsf{bot}} \in \mathbb{Z}^{(\ell+1)\times\ell}_p$ such that

– $\mathbf{Y}_{\mathsf{top}}\mathbf{Y}_{\mathsf{bot}} = \mathbf{0}$,
– $\mathbf{Y}_{\mathsf{bot}}$ has rank ℓ over $\mathbb{Z}_q$,
– $\boldsymbol{x}\mathbf{Y}_{\mathsf{bot}} \neq \mathbf{0}$ for challenge difference vector $\boldsymbol{x} = \boldsymbol{x}^0 - \boldsymbol{x}^1 \mod p$.

Define the full rank matrix $\mathbf{Y} = (\mathbf{Y}_{\mathsf{top}}||\mathbf{Y}_{\mathsf{bot}}) \in \mathbb{Z}_p^{(\ell+1)\times(\ell+1)}$, which is invertible over $\mathbb{Z}_q$.

Consider now $\mathbf{c}_1\mathbf{Y}$. As $\mathbf{Y}$ can be seen as a bijective mapping, if $\mathbf{c}_1\mathbf{Y}$ reveals almost nothing about β, then also $\mathbf{c}_1$ reveals almost nothing about β.

By the admissibility of $\mathcal{A}^*$, for all key queries $\boldsymbol{y}$, it holds that $\boldsymbol{x}\mathbf{Y}_{\mathsf{top}} = \mathbf{0} \mod p$, i.e. the first part is independent of β. We will show that $\mathbf{Z}\mathbf{Y}_{\mathsf{bot}}$ has a high enough min-entropy give $(\mathbf{A}, \mathbf{A}\mathbf{Z}, \mathbf{Y}_{\mathsf{top}}, \mathbf{Z}\mathbf{Y}_{\mathsf{bot}})$ to argue that $(\boldsymbol{u}_j, \boldsymbol{u}_j\mathbf{Z}\mathbf{Y}_{\mathsf{bot}})$ is statistically close to uniform for seeds $\boldsymbol{u}_j \xleftarrow{\$} \mathbb{Z}_q^m$, $j \in [\ell]$ using a variant of the Leftover Hash Lemma (LHL) as in [7]. In fact, we show that

$$H_\infty(\mathbf{Z}\mathbf{Y}_{\mathsf{bot}}|\mathbf{A}, \mathbf{A}\mathbf{Z}, \mathbf{Y}_{\mathsf{top}}, \mathbf{Y}_{\mathsf{top}}) \geq 2\log(\ell) + 4\ell\log(p) + n\log q + 2\lambda. \tag{14}$$

Note that $2\log(\ell)$ is required as we apply the LHL ℓ times, $4\ell\log(p)$ is needed for our CL argument and the rest of the term is a given through the LHL.

As the coefficients of $\mathbf{Z}$ are sampled independent, we can analyze each row of $\mathbf{Z}\mathbf{Y}_{\mathsf{bot}}$ separately. For each $\boldsymbol{z}_j$, conditioned on $\mathbf{z}_j\mathbf{Y}_{\mathsf{top}}$, the distribution of $\mathbf{z}_j$ is uniform over an affine space of size 2^ℓ. As $\boldsymbol{z}_j\mathbf{Y}_{\mathsf{bot}}$ is a linear mapping of $\boldsymbol{z}_j$ and $\mathbf{Y}_{\mathsf{bot}}$ is basis of orthogonal lattice to $\mathbf{Y}_{\mathsf{top}}$, the map from the orthogonal lattice to $\mathbb{Z}_q^\ell$ is injective for each fixed $\mathbf{z}_j\mathbf{Y}_{\mathsf{top}}$. This means that $\mathbf{z}_j\mathbf{Y}_{\mathsf{bot}}$ can take at most 2^ℓ values and each value has the same probability, i.e. the min-entropy per columns is $\geq \ell$. As all the columns of $\mathbf{Z}\mathbf{Y}_{\mathsf{bot}}$ are independent, the overall min-entropy is at least $m\ell$ given $(\mathbf{Y}_{\mathsf{top}}, \mathbf{Z}\mathbf{Y}_{\mathsf{top}})$.

Conditioning further on $(\mathbf{A}, \mathbf{A}\mathbf{Z})$ is the same as conditioning on $(\mathbf{A}, \mathbf{A}\mathbf{Z}\mathbf{Y}_{\mathsf{bot}})$ as $\mathbf{Y}$ is a bijection and we already know $\mathbf{A}$ and $\mathbf{Z}\mathbf{Y}_{\mathsf{top}}$. Thus, we have an additional entropy loss of at most $n\log q$ bits, yielding

$$H_\infty(\mathbf{Z}\mathbf{Y}_{\mathsf{bot}}|\mathbf{A}, \mathbf{A}\mathbf{Z}, \mathbf{Y}_{\mathsf{top}}, \mathbf{Y}_{\mathsf{top}}) \geq m\ell - n\log q. \tag{15}$$

If $m \geq 2n\log q$, $4 < nk$ and $\ell \geq 4$, it holds that Eq. (15) implies Eq. (14).

Further, to be statistically close to uniform by the LHL through the CL argument, we require $m \geq 4\ell + kn + \frac{2\lambda+2\log\ell}{\log p}$. Thus, we set $m = \max\{2n\log q, 4\ell + kn + \frac{2\lambda+2\log\ell}{\log p}\}$ and conclude the proof.

B Function-Hiding Multi-Input Functional Encryption for Affine Functions

Let us present the more generic version of Construction 2 supporting the broader family $\mathcal{F}^{\mathsf{aff}}_{N,(p,\ell)}$. The construction follows the idea of Datta et al. [18], where FH-MIFE is achieved by running N independent instances of the same FH-IPFE scheme and force to combine the independent results by incorporating a secret sharing of zero. To use this idea, we do not consider 1-ALS directly in

the security proof, but view our Construction 2 as a single-input FH scheme, i.e. $N = 1$, which means it supports the function family $\mathcal{F}^{\mathsf{avg}}_{1,(p,\ell)}$. To allow for smaller keys, the matrix $\mathbf{A}$ is shared among all clients, which is why we do not directly instantiate N independent copies of Construction 2.

Construction 4. *Let m, n be positive integers and $q = p^k$ for some integer k and a prime p.*

$\mathsf{Setup}(1^\lambda, \mathcal{F}^{\mathsf{avg}}_{N,(p,\ell)})$**:** *Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, \mathbf{Z}_i \xleftarrow{\$} \{-1,1\}^{m\times(\ell+1)}$ and $\boldsymbol{\zeta}_i \xleftarrow{\$} \mathbb{Z}_p^\ell$ for all $i \in [N]$. Set $\mathbf{U}_i = \mathbf{A}\mathbf{Z}_i \in \mathbb{Z}_q^{n\times(\ell+1)}$.*
Return $\mathsf{PP} := \mathbf{A}$, $\mathsf{EK}_i := (\mathbf{Z}_i, \boldsymbol{\zeta}_i)$ and $\mathsf{MSK} := (\{\boldsymbol{\zeta}_i, \mathbf{U}_i\}_{i\in[N]})$.

$\mathsf{Enc}(\mathsf{EK}_i, i, \boldsymbol{x}_i \in \mathbb{Z}_p^\ell)$**:** *Parse $\mathsf{EK}_i = (\mathbf{Z}_i, \boldsymbol{\zeta}_i)$, set $\widetilde{\boldsymbol{x}}_i = (\boldsymbol{x}_i + \boldsymbol{\zeta}_i, 1)$, compute $\mathbf{c}_i = \widetilde{\boldsymbol{x}}_i \mathbf{Z}_i^\top$ and return $\mathsf{CT}_i = (\widetilde{\boldsymbol{x}}_i, \mathbf{c}_i)$.*

$\mathsf{KeyGen}(\mathsf{MSK}, (\boldsymbol{y}, c))$**:** *For all $i \in [N]$, sample $\boldsymbol{s}_i \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{e}_{0,i} \leftarrow \mathcal{D}^m_{\mathbb{Z},\sigma}$, $\boldsymbol{e}_{1,i} \leftarrow \mathcal{D}^{\ell+1}_{\mathbb{Z},\sigma}$ and $\omega_i \xleftarrow{\$} \mathbb{Z}_p$ such that $\sum_{i\in[N]} \omega_i = c$. Set*

$$\widetilde{\boldsymbol{y}}_i = (\boldsymbol{y}_i[j], -\langle \boldsymbol{\zeta}_i, \boldsymbol{y}_i \rangle + \omega_i) \tag{16}$$

and compute

$$\mathbf{k}_{0,i} = \boldsymbol{s}_i \mathbf{A} + \boldsymbol{e}_{0,i} \in \mathbb{Z}_q^m, \quad \mathbf{k}_{1,i} = \boldsymbol{s}_i \mathbf{U}_i + \boldsymbol{e}_{1,i} + p^{k-1}\widetilde{\boldsymbol{y}}_i \in \mathbb{Z}_q^{\ell+1}.$$

Return $\mathsf{DK} = (\{\mathbf{k}_{0,i}, \mathbf{k}_{1,i}\}_{i\in[N]})$.

$\mathsf{Dec}(\mathsf{DK}, \mathsf{CT}_1, .., \mathsf{CT}_N)$**:** *Parse $\mathsf{DK} = (\{\mathbf{k}_{0,i}, \mathbf{k}_{1,i}\}_{i\in[N]})$ and conduct the following steps for all $i \in [N]$.*
- *Parse $\mathsf{CT}_i = (\widetilde{\boldsymbol{x}}_i, \mathbf{c}_i)$.*
- *Calculate $\mu'_i = \langle \widetilde{\boldsymbol{x}}_i, \mathbf{k}_{1,i} \rangle - \langle \mathbf{c}_i, \mathbf{k}_{0,i} \rangle \mod q$.*
- *Compute $r_i = \min_\mu |p^{k-1}\mu - \mu'_i|$.*

Return $\sum_{i\in[N]} r_i \mod p$.

The construction can be lifted to Q messages per client using the same idea as in (9), where the one-hot vector indicates the number of the ciphertext generated. To determine this number, the encryption algorithm needs to be stateful. This can be circumvented by using cover-free sets [29] at the cost of a blow up in the vector length.

Correctness. The correctness follows immediately as $r_i = \langle \boldsymbol{x}_i, \boldsymbol{y}_i \rangle + \omega_i$ for all $i \in [N]$ by the same argument as the correctness for Construction 2. Summing up all r_i yields the desired result.

Security. Overall, the security follows from the security of Construction 2.

Theorem 3. *Assuming that Construction 2 is* ot-fh-IND*-secure for the function family $\mathcal{F}^{\mathsf{avg}}_{1,(p,\ell)}$, the MIFE scheme from Construction 4 is also* ot-fh-IND*-secure.*

Proof. Let $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ be Construction 2 for the function family $\mathcal{F}^{\mathsf{avg}}_{1,(p,\ell)}$. Consider the following sequence of games.

G_0**:** This is the original security game $\mathsf{Expt}_0^{\mathsf{MIFE}}(\mathcal{A}, 1^\lambda)$.

$\mathsf{G}_{1,\kappa}$**:** For $\kappa \in [N]$, this game is the same as $\mathsf{G}_{1,\kappa-1}$ with $\mathsf{G}_{1,0} = \mathsf{G}_0$, except for the following changes. First, client $\kappa \in N$ encrypts $\boldsymbol{x}_\kappa^1$ instead of $\boldsymbol{x}_\kappa^0$. Second, during KeyGen the decryption key $\mathbf{k}_{1,\kappa}$ is generated for $\boldsymbol{y}_\kappa^1$ instead of $\boldsymbol{y}_\kappa^0$ and, $\omega'_\kappa = \omega_\kappa + \langle \mathbf{x}_\kappa^0, \boldsymbol{y}_\kappa^0 \rangle - \langle \mathbf{x}_\kappa^1, \boldsymbol{y}_\kappa^1 \rangle$ is added in $\widetilde{\boldsymbol{y}}_\kappa$ instead of ω_κ.

G_2**:** This is the original security game $\mathsf{Expt}_1^{\mathsf{MIFE}}(\mathcal{A}, 1^\lambda)$.

The advantage of any adversary between $\mathsf{G}_{1,\kappa}$ and $\mathsf{G}_{1,\kappa-1}$ for all $\kappa \in [N]$ can be bounded by relying on the security of iFE. Details are given in Lemma 9.

The only difference between G_2 and $\mathsf{G}_{1,N}$ is that ω'_i is added instead of ω_i. However, as the ω_i values are sampled uniformly at random, the distribution of the ω'_i values is also uniform at random. By the admissibility of the adversary,

$$\sum_{i\in[N]} \omega'_i = \sum_{i\in[N]} (\omega_i + \langle \mathbf{x}_i^0, \boldsymbol{y}_i^0 \rangle - \langle \mathbf{x}_i^1, \boldsymbol{y}_i^1 \rangle) = c^1.$$

This is exactly how the random values in the keys are sampled and thereby, both are indistinguishable.

Lemma 9. *Let* $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ *be Construction 2 for* $\mathcal{F}^{\mathsf{avg}}_{1,(p,\ell)}$. *For any PPT adversary* $\mathcal{A}$ *distinguishing* $\mathsf{G}_{\kappa-1}$ *and* G_κ *for all* $\kappa \in [N]$, *there exists a PPT adversary* $\mathcal{B}$ *such that* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{G}_{\kappa-1}\leftrightarrow\mathsf{G}_\kappa}(\lambda) \leq \mathsf{Adv}_{\mathsf{iFE},\mathcal{B}}^{\mathsf{ot\text{-}fh\text{-}IND}}(\lambda)$.

Proof. Let $\mathcal{B}$ be an adversary against the $\mathsf{ot\text{-}fh\text{-}IND}$-security of iFE, then $\mathcal{B}$ can simulate $\mathcal{A}$'s view as follows. Upon receiving $\mathcal{MS}$, $\mathcal{B}$ extracts $(\boldsymbol{x}_\kappa^0, \boldsymbol{x}_\kappa^1, \kappa)$ and submits $(\boldsymbol{x}_\kappa^0, \boldsymbol{x}_\kappa^1)$ as their message set to their oracle to receive $\mathsf{iPP}_\kappa = \mathbf{A}$ and CT_κ.

Let $\mathsf{iSetup}(\cdot,\cdot,\cdot;\mathbf{A})$ denote the setup algorithm of iFE that takes as input the matrix $\mathbf{A}$, but calculates everything else as before. For all $i \in [N]\setminus\{\kappa\}$, $\mathcal{B}$ generates $(\mathsf{iPP}_i, \mathsf{iEK}_i, \mathsf{iMSK}_i) \leftarrow \mathsf{iSetup}(1^\lambda, \mathcal{F}^{\mathsf{avg}}_{1,(p,\ell)}; \mathbf{A})$ and computes the ciphertexts for $i \neq \kappa$ as

$$\mathsf{CT}_i = \begin{cases} \mathsf{iEnc}(\mathsf{iEK}_i, \boldsymbol{x}_i^1) & i < \kappa, \\ \mathsf{iEnc}(\mathsf{iEK}_i, \boldsymbol{x}_i^0) & i > \kappa. \end{cases}$$

$\mathcal{B}$ hands $(\{\mathsf{iPP}\}_{i\in[N]}, \{\mathsf{CT}_i\}_{i\in[N]})$ to $\mathcal{A}$.

Whenever $\mathcal{A}$ poses a decryption key query $((\boldsymbol{y}^0, c^0), (\boldsymbol{y}^1, c^1))$, $\mathcal{B}$ parses $\boldsymbol{y}^b = (\boldsymbol{y}_1^b, \ldots, \boldsymbol{y}_N^b)$, $b \in \{0,1\}$, and samples $\omega_i \xleftarrow{\$} \mathbb{Z}_p$ such that $\sum_{i\in[N]} \omega_i = c^0$. For all $i \leq \kappa$, they set $\omega'_i = \omega_i + \langle \boldsymbol{x}_i^0, \boldsymbol{y}_i^0 \rangle - \langle \boldsymbol{x}_i^1, \boldsymbol{y}_i^1 \rangle$ and compute

$$\mathsf{DK}_i = \begin{cases} \mathsf{iKeyGen}(\mathsf{iMSK}_i, (\boldsymbol{y}_i^1, \omega'_i)) & i < \kappa, \\ \mathsf{iKeyGen}(\mathsf{iMSK}_i, (\boldsymbol{y}_i^0, \omega)) & i > \kappa. \end{cases}$$

In addition, they query their own oracle for $(\boldsymbol{y}^0 = (\boldsymbol{y}_\kappa^0, \omega_\kappa), \boldsymbol{y}^1 = (\boldsymbol{y}_\kappa^1, \omega'_\kappa))$ to receive DK_κ. Then, $\mathcal{B}$ returns $\mathsf{DK} = \{\mathsf{DK}_i\}_{i\in[N]}$ to $\mathcal{A}$.

Finally, $\mathcal{B}$ outputs the same bit as $\mathcal{A}$. Note that $\mathcal{B}$ perfectly simulates $\mathcal{A}'s$ view and is admissible by the admissibility of $\mathcal{A}$.