FedML-HE: An Efficient Homomorphic-Encryption-Based Privacy-Preserving Federated Learning System

Weizhao Jin[*]
University of Southern California
weizhaoj@usc.edu

Yuhang Yao[*]
Carnegie Mellon University
yuhangya@andrew.cmu.edu

Shanshan Han
University of California Irvine
shanshan.han@uci.edu

Carlee Joe-Wong
Carnegie Mellon University
cjoewong@andrew.cmu.edu

Srivatsan Ravi
University of Southern California
sravi@isi.edu

Salman Avestimehr
FedML Inc.
avestime@usc.edu

Chaoyang He
FedML Inc.
ch@fedml.ai

Abstract

Federated Learning trains machine learning models on distributed devices by aggregating local model updates instead of local data. However, privacy concerns arise as the aggregated local models on the server may reveal sensitive personal information by inversion attacks. Privacy-preserving methods, such as homomorphic encryption (HE), then become necessary for FL training. Despite HE's privacy advantages, its applications suffer from impractical overheads, especially for foundation models. In this paper, we present FedML-HE, *the first practical federated learning system with efficient HE-based secure model aggregation*. FedML-HE proposes to selectively encrypt sensitive parameters, significantly reducing both computation and communication overheads during training while providing customizable privacy preservation. Our optimized system demonstrates considerable overhead reduction, particularly for large foundation models (e.g., $\sim$10x reduction for ResNet-50, and up to $\sim$40x reduction for BERT), demonstrating the potential for scalable HE-based FL deployment.

1 Introduction

Federated learning (FL) is increasingly popular in contemporary machine learning practices due to its ability to allow distributed clients to collectively train a global model without directly sharing data. Privacy preservation in standard federated learning systems depends on the distributed training process and the model aggregation function, such as FedAvg McMahan et al. (2017), FedSGD Shokri & Shmatikov (2015), and FedGAN Rasouli et al. (2020). In FL, instead of uploading raw data to a central server for training, clients train models locally and share their models with the server, where the local models are aggregated based on the aggregation functions. While FL ensures that local raw data do not leave their original locations, it remains vulnerable to eavesdroppers and malicious FL servers that might exploit plaintext local models (or model updates) to reconstruct sensitive training data, i.e., data reconstruction attacks or gradient inversion attacks in literature Zhu et al. (2019);

[*]Equal Contribution

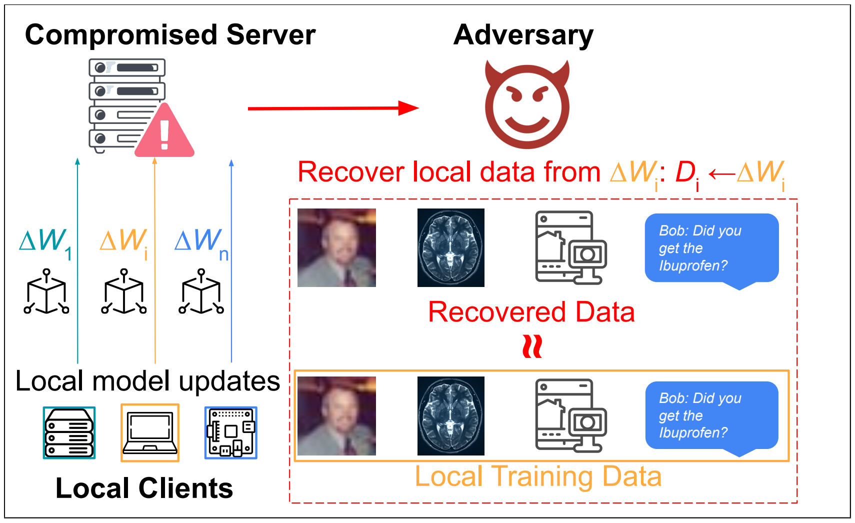


Figure 1: Data Reconstruction Attacks: an adversarial server can recover local training data from local model updates.

Criswell et al. (2014); Bhowmick et al. (2018); Hitaj et al. (2017); Han et al. (2023); Hatamizadeh et al. (2022); Fowl et al. (2022), as shown in Figure 1. This poses a privacy vulnerability especially when local models are trained on small local datasets, a common scenario in real-world applications such as smartphone text data for LLMs. Local models derived from these small datasets inherently contain fine-grained information, making it easier for adversaries to extract sensitive information from small model updates.

Existing defense methods that prevent privacy leakage from plaintext local models include differential privacy (DP) Truex et al. (2019a); Byrd & Polychroniadou (2020) and secure aggregation Bonawitz et al. (2017); So et al. (2022). DP adds noise to original models but may result in model performance degradation due to the privacy noises introduced. On the other hand, secure aggregation employs zero-sum masks to shield local model updates, ensuring that the details of each update remain private. However, secure aggregation demands additional interactive synchronization steps and is sensitive to client dropout, making it less practical in real-world FL applications, where the unstable environments of clients face challenges such as unreliable internet connections, and software crashes.

	Model Degradation	Overheads	Client Dropout	Interactive Sync	Model Visible To Server
Differential Privacy	With noise	**Light**	**Robust**	**No**	Yes
Secure Aggregation	**Exact**	Medium	Susceptible	Yes	Yes
Homomorphic Encryption	**Exact**	Large	**Robust**	**No**	**No**

Table 1: Comparison of Differential Privacy, Secure Aggregation, and Homomorphic Encryption

As shown in Table 1, compared to the non-HE FL solutions above, homomorphic encryption (HE) Paillier (1999); Gentry (2009); Fan & Vercauteren (2012); Brakerski et al. (2014); Cheon et al. (2017) offers a robust post-quantum secure solution that protects local models against attacks and *provides stronger privacy guarantee while keeping the model aggregation with exact gradients*. HE-based federated learning (HE-FL) encrypts local models on clients and performs model aggregation over ciphertexts on the server. This approach enables secure federated learning deployments with exactly the same model performance as vanilla FL and has been adopted by several FL systems Roth et al. (2022); IBM (2022); Zhang et al. (2020); Du et al. (2023) and a few domain-specific applications Stripelis et al. (2021); Yao et al. (2023).

Despite the advantages of homomorphic encryption, HE remains a powerful but complex cryptographic foundation with impractical overheads (as shown in Figure 2) for most real-world applications. Prior FL-HE solutions mainly employ existing generic HE methods without sufficient optimization for large-scale FL deployment Roth et al. (2022); IBM (2022); Zhang et al. (2020); Du et al. (2023). The scalability of encrypted computation and communication during federated training then becomes

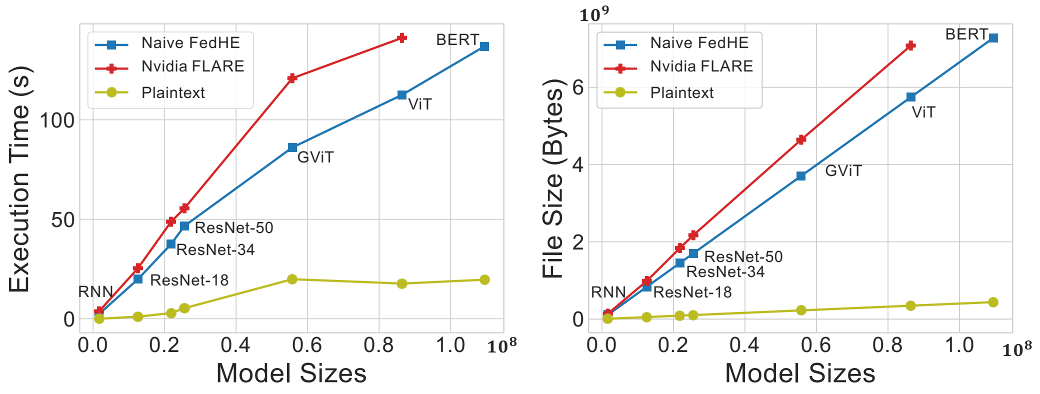


Figure 2: Computational (left) and Computation (right) Overhead Comparison for Models of Different Sizes: Naive FedML-HE vs. Nvidia FLARE vs. Plaintext Aggregation. Due to TenSeal's larger file sizes, FLARE did not finish the run on BERT on our 32GB memory machine.

a bottleneck, restricting its feasibility for real-world scenarios. This HE overhead limitation is particularly noticeable (*commonly ∼15x increase in both computation and communication* Gouert et al. (2022)) when training large foundation models across resource-constrained devices, where encrypted computing and communication of large models might take considerably longer than the actual model training. It is widely known that HE inevitably introduces large overheads regarding both computation and communication Gouert et al. (2022). To verify this, we evaluate the vanilla HE implementation to pinpoint the overhead bottlenecks.

Observation: As shown by the evaluation results in Figure 2, the computational and communication (package size) overheads introduced by HE is $O(n)$, both growing linearly with the input size n, which in our case the sizes of the models for aggregation. Although the unoptimized system is faster than Nvidia FLARE, the execution time and file size are still impractical, especially for large models.

To address these challenges, we propose FedML-HE, an efficient Homomorphic Encryption-based privacy-preserving FL system with *Selective Parameter Encryption*, designed for practical deployment across distributed edge devices. Our system significantly reduces communication and computation overheads, enabling HE-based federated learning to be more accessible and efficient in real-world scenarios (comparison with other popular HE-based FL work can be found in Table 2).

Features	IBMFL (8c8ab11)	Nvidia FLARE (9a1b226)	Ours
Homomorphic Encryption	✓	✓	✓
Threshold Key Management	✗	✗	✓
Selective Parameter Encryption	✗	◯	✓
Encrypted Foundation Model Training	◯	◯	✓

Table 2: Comparison with Existing HE-Based FL Systems. ◯ implies limited support: for Selective Parameter Encryption, FLARE offers the (random) partial encryption option which does not have clear indications on privacy impacts; for Encrypted Foundation Model Training, the other two platforms require massive resources to train foundation models in encrypted federated learning.

Key contributions:

- We propose FedML-HE, the first practical Homomorphic Encryption-based privacy-preserving FL system that supports encryption key management, encrypted FL platform deployment, encryption optimizations to reduce overhead, and is designed to support efficient foundation model federated training.
- We propose **Selective Parameter Encryption** that selectively encrypts the most privacy-sensitive parameters to minimize the size of encrypted model updates while providing customizable privacy preservation.

- Theoretical privacy analysis shows the HE system can ensure privacy under single-key and threshold adversaries and encrypting most sensitivity parameters provides orders-of-magnitude better privacy guarantees.
- Extensive experiments show that the optimized system achieves significant overhead reduction while preserving privacy against state-of-the-art ML privacy attacks, particularly for large models (e.g., ~10x reduction for HE-federated training ResNet-50 and up to ~40x reduction for BERT), demonstrating the potential for real-world HE-based FL deployments.

2 FedML-HE System Design

In this section, we first provide the overview of FedML-HE system in §2.1, define the threat model in §2.2, describe the algorithmic design of FedML-HE in §2.3, propose our efficient optimization method **Selective Parameter Encryption** after pinpointing the overhead bottleneck in §2.4, and explain how we integrate homomorphic encryption in federated learning from a software framework perspective in §2.5.

2.1 System Overview

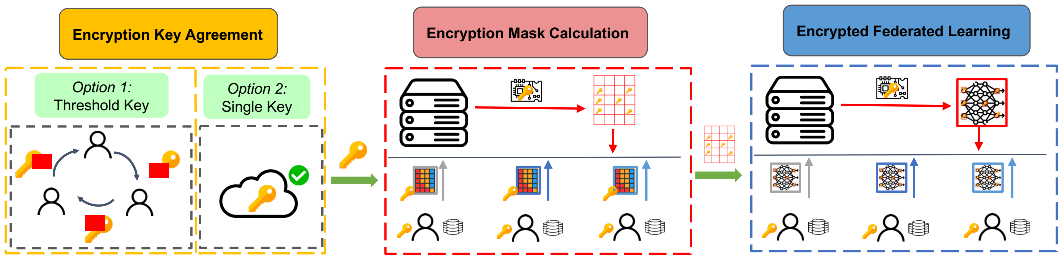


Figure 3: FedML-HE System Pipeline: in the **Encryption Key Agreement** stage, clients can either use distributed threshold key agreement protocol or outsource a trusted key authority. We simplify the illustration here by abstracting the key pair of the public key and secret key (partial secret keys if using threshold protocol) as one key; in the **Encryption Mask Calculation** stage, clients use local datasets to calculate local model sensitivity maps which are homomorphically aggregated at the server to generate an encryption mask; in the **Encrypted Federated Learning** stage, clients use homomorphic encryption with encryption mask to protect local model updates where the server aggregates them but does not have access to sensitive local models.

As shown in Figure 3, our efficient HE-based federated training process at a high level goes through three major stages: *(1)* Encryption key agreement: the clients either use threshold HE key agreement protocol or trusted key authority to generate HE keys; *(2)* Encryption mask calculation: the clients and the server apply **Selective Parameter Encryption** method using homomorphic encryption to agree on a selective encryption mask; *(3)* Encrypted federated learning: the clients selectively encrypt local model updates using the homomorphic encryption key and the encryption mask for efficient privacy-preserving training.

2.2 Threat Model

We define a semi-honest adversary $\mathcal{A}$ that can corrupt the aggregation server or any subset of local clients. $\mathcal{A}$ follows the protocol but tries to learn as much information as possible. Loosely speaking, under such an adversary, the security definition requires that only the private information in local models from the corrupted clients will be learned when $\mathcal{A}$ corrupts a subset of clients; no private information from local models nor global models will be learned by $\mathcal{A}$ when $\mathcal{A}$ corrupts the aggregation server.

When $\mathcal{A}$ corrupts both the aggregation server and a number of clients, the default setup where the private key is shared with all clients (also with corrupted clients) will allow $\mathcal{A}$ to decrypt local models from benign clients (by combining encrypted local models received by the corrupted server and the private key received by any corrupted client). This issue can be mitigated by adopting the threshold or multi-key variant of HE where decryption must be collaboratively performed by a certain number of clients Aloufi et al. (2021); Ma et al. (2022); Du et al. (2023). Since the multi-party homomorphic encryption issue is not the focus of this work, in the rest of the paper we default to a single-key homomorphic encryption setup, but details on threshold homomorphic encryption federated learning setup and microbenchmarks are provided in the appendix.

2.3 Algorithm for HE-Based Federated Aggregation

Privacy-preserving federated learning systems utilize homomorphic encryption to enable the aggregation server to combine local model parameters without viewing them in their unencrypted form by designing homomorphic encrypted aggregation functions. We primarily focus on FedAvg McMahan et al. (2017), which has been proved as still one of the most robust federated aggregation strategies while maintaining computational simplicity Wang et al. (2022).

Algorithm 1 HE-Based Federated Aggregation

- $[\![\mathbf{W}]\!]$: the fully encrypted model | $[\mathbf{W}]$: the partially encrypted model;
- p: the ratio of parameters for selective encryption;
- b: (optional) differential privacy parameter.

```
// Key Authority Generate Key
$(pk, sk) \leftarrow HE.KeyGen(\lambda)$;
// Local Sensitivity Map Calculation
for each client $i \in [N]$ do in parallel
    $\mathbf{W}_i \leftarrow Init(\mathbf{W})$;
    $\mathbf{S}_i \leftarrow Sensitivity(\mathbf{W}, \mathcal{D}_i)$;
    $[\![\mathbf{S}_i]\!] \leftarrow Enc(pk, \mathbf{S}_i)$;
    Send $[\![\mathbf{S}_i]\!]$ to server;
end
// Server Encryption Mask Aggregation
$[\![\mathbf{M}]\!] \leftarrow Select(\sum_{i=1}^{N} \alpha_i [\![\mathbf{S}_i]\!], p)$;
// Training
for $t = 1, 2, \ldots, T$ do
    for each client $i \in [N]$ do in parallel
        if $t = 1$ then
            Receive $[\![\mathbf{M}]\!]$ from server;
            $\mathbf{M} \leftarrow HE.Dec(sk, [\![\mathbf{M}]\!])$;
        end
        if $t > 1$ then
            Receive $[\mathbf{W}_{\text{glob}}]$ from server;
            $\mathbf{W}_i \leftarrow HE.Dec(sk, \mathbf{M} \odot [\mathbf{W}_{\text{glob}}]) + (\mathbf{1} - \mathbf{M}) \odot [\mathbf{W}_{\text{glob}}]$;
        end
        $\mathbf{W}_i \leftarrow Train(\mathbf{W}_i, \mathcal{D}_i)$;
        // Additional Differential Privacy
        if Add DP then
            $\mathbf{W}_i \leftarrow \mathbf{W}_i + Noise(b)$;
        end
        $[\mathbf{W}_i] \leftarrow HE.Enc(pk, \mathbf{M} \odot \mathbf{W}_i) + (\mathbf{1} - \mathbf{M}) \odot \mathbf{W}_i$;
        Send $[\mathbf{W}_i]$ to server $\mathcal{S}$;
    end
    // Server Model Aggregation
    $[\mathbf{W}_{\text{glob}}] \leftarrow \sum_{i=1}^{N} \alpha_i [\![\mathbf{M} \odot \mathbf{W}_i]\!] + \sum_{i=1}^{N} \alpha_i ((\mathbf{1} - \mathbf{M}) \odot \mathbf{W}_i)$;
end
```

Our HE-based secure aggregation algorithm, as illustrated in Algorithm 1, can be summarized as: given an aggregation server and N clients, each client $i \in [N]$ owns a local dataset $\mathcal{D}_i$ and initializes a local model $\mathbf{W}_i$ with the aggregation weighing factor α_i; the key authority or the distributed threshold key agreement protocol generates a key pair (pk, sk) and the crypto context, then distributes it to clients and server (except the server only gets the crypto context which is public configuration). The clients and the server then collectively calculate the encryption mask $\mathbf{M}$ for **Selective Parameter Encryption** also using homomorphic encryption. At every communication round $t \in [T]$, the server performs the aggregation

$$[\mathbf{W}_{\text{glob}}] = \sum_{i=1}^{N} \alpha_i [\![\mathbf{M} \odot \mathbf{W}_i]\!] + \sum_{i=1}^{N} \alpha_i ((\mathbf{1} - \mathbf{M}) \odot \mathbf{W}_i),$$

where $[\mathbf{W}_{\text{glob}}]$ is the partially-encrypted global model, $\mathbf{W}_i$ is the i-th plaintext local model where $[\![]\!]$ indicates the portion of the model that is fully encrypted, α_i is the aggregation weight for client i, and $\mathbf{M}$ is the model encryption mask.

Note that the aggregation weights can be either encrypted or in plaintext depending on whether the aggregation server is trustworthy enough to obtain that information. In our system, we set the aggregation weights to be plaintext by default. We only need one multiplicative depth of HE multiplication in our algorithm for weighting, which is preferred to reduce HE multiplication operations. Our system can also be easily extended to support more FL aggregation functions with HE by encrypting and computing the new parameters in these algorithms (e.g. FedProx Li et al. (2020)). Additionally, in Algorithm 1, optional local differential privacy noise can be easily added after local models are trained if there is an extra desire for differential privacy.

We will explain in detail how the encryption mask $\mathbf{M}$ is formalized in §2.4.

2.4 Efficient Optimization by Selective Parameter Encryption

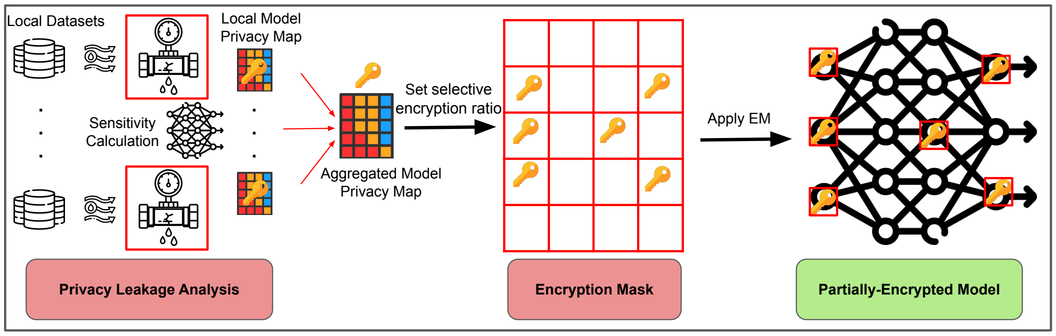


Figure 4: **Selective Parameter Encryption**: in the initialization stage, clients first calculate privacy sensitivities on the model using its own dataset and local sensitivities will be securely aggregated to a global model privacy map. The encryption mask will be then determined by the privacy map and a set selection value p per overhead requirements and privacy guarantee. Only the masked parameters will be aggregated in the encrypted form.

Fully encrypted models can guarantee no access to plaintext local models from the adversary with high overheads. However, previous work on privacy leakage analysis shows that "partial transparency", e.g. hiding parts of the models (Hatamizadeh et al., 2022; Mo et al., 2020), can limit an adversary's ability to successfully perform attacks like gradient inversion attacks (Lu et al., 2022). We therefore propose **Selective Parameter Encryption** to *selectively encrypt the most privacy-sensitive parameters* in order to reduce impractical overhead while providing customizable privacy preservation; see Figure 4.

Step 1: Privacy Leakage Analysis on Clients. Directly performing a gradient inversion attack Wei et al. (2020) and evaluating the success rate of the attack can take much more time than the model training. We then adopt sensitivity Novak et al. (2018); Sokolić et al. (2017); Mo et al. (2020) for measuring the general privacy risk on gradients w.r.t. input. Given model $\mathbf{W}$ and K data samples with input matrix $\mathbf{X}$ and ground truth label vector $\mathbf{y}$, we compute the sensitivity for each parameter w_m by $\frac{1}{K}\sum_{k=1}^{K} \|J_m(y_k)\|$, where $J_m(y_k) = \frac{\partial}{\partial y_k}\left(\frac{\partial \ell(\mathbf{X},\mathbf{y},\mathbf{W})}{\partial w_m}\right) \in R$, $\ell(\cdot)$ is the loss function given $\mathbf{X}$, $\mathbf{y}$ and $\mathbf{W}$, and $\|\cdot\|$ calculates the absolute value. The intuition is to calculate how large the gradient of the parameter will change with the true output y_k for each data point k. Each client i then sends the encrypted parameters sensitivity matrix $[\![\mathbf{S}_i]\!]$ to the server.

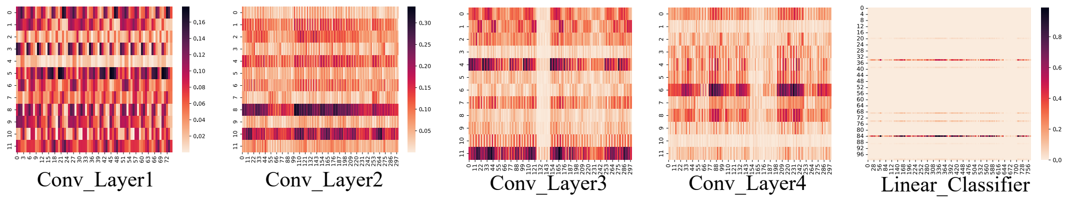


Figure 5: Model Privacy Map Calculated by Sensitivity on LeNet: darker color indicates higher sensitivity. Each subfigure shows the sensitivity of parameters of the current layer. The sensitivity of parameters is imbalanced and many parameters have very little sensitivity (its gradient is hard to be affected by tuning the data input for attack).

As shown in Figure 5, different parts of a model contribute to attacks by revealing uneven amounts of information. Using this insight, we propose to only select and encrypt parts of the model that are more important and susceptible to attacks to reduce HE overheads while preserving adequate privacy.

Step 2: Encryption Mask Agreement across Clients. The sensitivity map is dependent on the data it is processed on. With potentially heterogeneous data distributions, the server aggregates local sensitivity maps to a global privacy map $\sum_{i=1}^{N} \alpha_i [\![\mathbf{S}_i]\!]$. The global encryption mask $\mathbf{M}$ is then configured using a privacy-overhead ratio $p \in [0, 1]$ which is the ratio of selecting the most sensitive parameters for encryption. The global encryption mask is then shared among clients as part of the federated learning configuration.

2.5 Software Framework: Homomorphic Encryption In Federated Learning

In this part, we will illustrate how we design our HE-based aggregation from a software framework perspective.

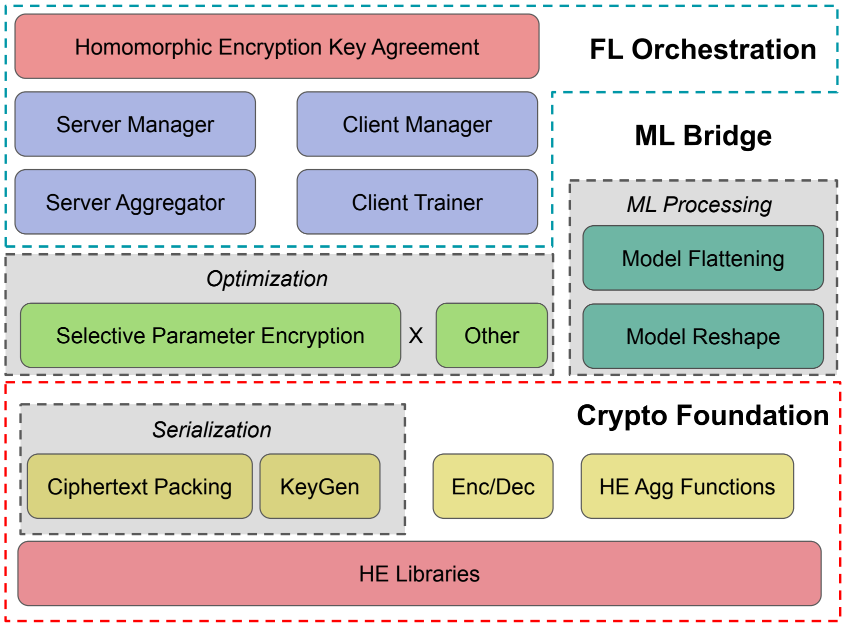


Figure 6: Framework Structure: our framework consists of a three-layer structure including Crypto Foundation to support basic HE building blocks, ML Bridge to connect crypto tools with ML functions, and FL Orchestration to coordinate different parties during a task.

Figure 6 provides a high-level design of our framework, which consists of three major layers:

- **Crypto Foundation.** The foundation layer is where Python wrappers are built to realize HE functions including key generation, encryption/decryption, secure aggregation, and ciphertext serialization using open-sourced HE libraries;

- **ML Bridge.** The bridging layer connects the FL system orchestration and cryptographic functions. Specifically, we have ML processing APIs to process inputs to HE functions from local training processes and outputs vice versa. Additionally, we realize the optimization module here to mitigate the HE overheads;
- **FL Orchestration.** The FL system layer is where the key authority server manages the key distribution and the (server/client) managers and task executors orchestrate participants.

Our layered design makes the HE crypto foundation and the optimization module *semi-independent*, allowing different HE libraries to be easily switched into FedML-HE and further FL optimization techniques to be easily added to the system.

3 Privacy By Selective Parameter Encryption

In this section, we first provide proof to analyze the privacy of fully encrypted federated learning and then analyze the privacy guarantee of Selective Parameter Encryption.

3.1 Proof of Base Protocol

In this subsection, we prove the privacy of base protocol where homomorphic-encryption-based federated learning utilizes the full model parameter encryption (i.e., the selective parameter encryption rate is set to be *1*). We define the adversary in Definition 3.1 and privacy in Definition 3.3.

Definition 3.1 (Single-Key Adversary)**.** *A semi-honest adversary $\mathcal{A}$ can corrupt (at the same time) any subset of n learners and the aggregation server, but not at the same time.*

Note that the ref of the proof assumes the single-key setup and the privacy of the threshold variant of HE-FL (as shown in Definition 3.2) can be easily proved by extending the proofs of threshold homomorphic encryption Boneh et al. (2006); Laud & Ngo (2008); Asharov et al. (2012).

Definition 3.2 (Threshold Adversary)**.** *A semi-honest adversary $\mathcal{A}_{\mathcal{T}}\langle$ can corrupt (at the same time) any subset of $n - k$ learners and the aggregation server.*

Definition 3.3 (Privacy)**.** *A homomorphic-encryption federated learning protocol π is simulation secure in the presence of a semi-honest adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ in the ideal world that also corrupts the same set of parties and produces an output identically distributed to $\mathcal{A}$'s output in the real world.*

Ideal World. Our ideal world functionality $\mathcal{F}$ interacts with learners and the aggregation server as follows:

- Each learner sends a registration message to $\mathcal{F}$ for a federated training model task $\mathbf{W}_{\text{glob}}$. $\mathcal{F}$ determines a subset $N' \subset N$ of learners whose data can be used to compute the global model $\mathbf{W}_{\text{glob}}$.
- Both honest and corrupted learners upload their local models to $\mathcal{F}$.
- If local models $\vec{\mathbf{W}}$ of learners in N' are enough to compute $\mathbf{W}_{\text{glob}}$, $\mathcal{F}$ sends $\mathbf{W}_{\text{glob}} \leftarrow \sum_{i=1}^{N'} \alpha_i \mathbf{W}_i$ to all learners in N', otherwise $\mathcal{F}$ sends empty message $\perp$.

Real World. In real world, $\mathcal{F}$ is replaced by our protocol described in Algorithm 1 with full model parameter encryption.

We describe a simulator $\mathcal{S}$ that simulates the view of the $\mathcal{A}$ in the real-world execution of our protocol. Our privacy definition 3.3 and the simulator $\mathcal{S}$ prove both confidentiality and correctness. We omit the simulation of the view of $\mathcal{A}$ that corrupts the aggregation server here since the learners will not receive the ciphertexts of other learners' local models in the execution of π thus such a simulation is immediate and trivial.

Simulator. In the ideal world, $\mathcal{S}$ receives λ and 1^n from $\mathcal{F}$ and executes the following steps:

1. $\mathcal{S}$ chooses a uniformly distributed random tape r.
2. $\mathcal{S}$ runs the key generation function to sample pk: $(pk, sk) \leftarrow HE.KeyGen(\lambda)$.
3. For a chosen ith learner, $\mathcal{S}$ runs the encryption function to sample: $(c_i) \leftarrow HE.Enc(pk, r^{|\mathbf{W}_i|})$.

4. $\mathcal{S}$ repeats Step 3 for all other learners to obtain $\vec{c}$, and runs the federated aggregation function f to sample: $(c_{\text{glob}}) \leftarrow HE.Eval(\vec{\mathbf{c}}, f)$.

The execution of $\mathcal{S}$ implies that:

$$\{(c_i, c_{\text{glob}})\} \overset{\text{s}}{\equiv} \left\{ \left(HE.Enc(pk, \mathbf{W}_i), HE.Eval(\vec{\mathbf{W}}, f) \right) \right\}$$

Thus, we conclude that $\mathcal{S}$'s output in the ideal world is computationally indistinguishable from the view of $\mathcal{A}$ in a real world execution:

$$\{\mathcal{S}(1^n, (\lambda))\} \overset{\text{s}}{\equiv} \{\text{view}^{\pi}(\lambda)\}$$

,

where view is the view of $\mathcal{A}$ in the real execution of π.

3.2 Proof of Encrypted Learning by DP Theory

Definition 3.4 (Adjacent Datasets)**.** Two datasets D_1 and D_2 are said to be adjacent if they differ in the data of exactly one individual. Formally, they are adjacent if:

$$|D_1 \Delta D_2| = 1$$

Definition 3.5 (ϵ-Differential Privacy)**.** A randomized algorithm $\mathcal{M}$ satisfies ϵ-differential privacy if for any two adjacent datasets D_1 and D_2, and for any possible output $O \subseteq \text{Range}(\mathcal{F})$, the following inequality holds:

$$\frac{\Pr[\mathcal{M}(D_1) \in O]}{\Pr[\mathcal{M}(D_2) \in O]} \leq e^{\epsilon}$$

Smaller values of the privacy parameter ϵ imply stronger privacy guarantees.

Definition 3.6 (Laplace mechanism)**.** Given a function $f : \mathcal{D} \to \mathbb{R}$,

where $\mathcal{D}$ is the domain of the dataset and d is the dimension of the output, the Laplace mechanism adds Laplace noise to the output of f.

Let b be the scale parameter of the Laplace distribution, which is given by:

$$\text{Lap}(x \mid b) = \frac{1}{2b} e^{-\frac{|x|}{b}}$$

Given a dataset D, the Laplace mechanism $\mathcal{F}$ is defined as:

$$\mathcal{M}(D) = f(D) + \text{Lap}(0 \mid b)^d$$

Definition 3.7 (Sensitivity)**.** To ensure ϵ-differential privacy, we need to determine the appropriate scale parameter b. This is where the sensitivity of the function f comes into play. The sensitivity Δf of a function f is the maximum difference in the output of f when applied to any two adjacent datasets:

$$\Delta f = \max_{D_1, D_2 : |D_1 \Delta D_2| = 1} \|f(D_1) - f(D_2)\|_1$$

Based on Definition 3.4, 3.5, 3.6 and 3.7 we have

Lemma 3.8 (Achieving ϵ-Differential Privacy by Laplace Mechanism Dwork (2008); Abadi et al. (2016))**.** *To achieve ϵ-differential privacy, we choose the scale parameter b as:*

$$b = \frac{\Delta f}{\epsilon}$$

With this choice of b, the Laplace mechanism $\mathcal{F}$ satisfies ϵ-differential privacy.

By adding noise $\text{Lap}(0 \mid b)^d$ on one parameter in the model gradient where $b = \frac{\Delta f}{\epsilon}$, we can achieve ϵ-differential privacy. We then show homomorphic encryption provides a much stronger differential privacy guarantee.

Theorem 3.9 (Achieving 0-Differential Privacy by Homomorphic Encryption)**.** *For any two adjacent datasets D_1 and D_2, since $\mathcal{M}(D)$ is computationally indistinguishable, we have*

$$\frac{\Pr\left[\mathcal{M}\left(D_1\right) \in O\right]}{\Pr\left[\mathcal{M}\left(D_2\right) \in O\right]} \leq e^{\epsilon}.$$

We then have $\epsilon = 0$ if O is encrypted.

In other words, $\mathcal{A}$ cannot retrieve sensitive information from encrypted parameters.

3.3 Proof of Selective Parameter Selection

Lemma 3.10 (Sequential Composition Dwork (2008),)**.** *If $\mathcal{M}_1(x)$ satisfies ϵ_1-differential privacy and $\mathcal{M}_2(x)$ satisfies ϵ_2-differential privacy, then the mechanism $\mathcal{G}(x) = (\mathcal{M}_1(x), \mathcal{M}_2(x))$ which releases both results satisfies $(\epsilon_1 + \epsilon_2)$-differential privacy*

Based on Lemma 3.8, 3.10 and Theorem 3.9, we can now analyze the privacy of Selective Parameter Encryption

Theorem 3.11 (Achieving $\sum_{i \in [N]/\mathcal{S}} \frac{\Delta f_i}{b}$-Differential Privacy by Partial Encryption)**.** *If we apply Homomorphic Encryption on partial model parameters $\mathcal{S}$ and Laplace Mechanism on remaining model parameters $[N]/\mathcal{S}$ with fixed noise scale b. For each parameter $i \in [N]/\mathcal{S}$, we have $\epsilon_i = \frac{\Delta f_i}{b}$. Such partial encryption satisfies $\sum_{i \in [N]/\mathcal{S}} \frac{\Delta f_i}{b}$-differential privacy.*

Let $J = \sum_{i=1}^{N} \frac{\Delta f_i}{b}$ and assume $\Delta f \sim \mathcal{U}(0, 1)$ where $\mathcal{U}$ represents the uniform distribution, we can then show the privacy cost of adding Laplace noise on all parameters, random parameter encryption, and selective parameter encryption.

Remark 3.12 (Achieving J-Differential Privacy by Laplace Mechanism on All Model Parameters). If we add Laplace noise on all parameters with fixed noise scale b, it satisfies J-differential privacy.

Remark 3.13 (Achieving $(1-p)J$-Differential Privacy by Random Selection). If we randomly select model parameters with probability p and homomorphically encrypt the remaining parameters, it satisfies $(1-p)J$-differential privacy.

Remark 3.14 (Achieving $(1-p)^2 J$-Differential Privacy by Sensitive Parameter Selection). If we select the most sensitive parameters with ratio p and homomorphically encrypt the remaining parameters, it satisfies $(1-p)^2 J$-differential privacy.

Key Observation: Selective Parameter Encryption requires $(1-p)^2$ *times less privacy budget* than random selection and complete differential privacy with the same privacy preservation.

4 Evaluation

In this section, we focus on the evaluation results to show how our proposed universal optimization scheme largely mitigates these overheads for real-world deployment but still guarantees adequate defense against privacy attacks. Note that additional experimental results regarding other FL system aspects are included in in the appendix.

4.1 Experiment Setup

Models. We test our framework on models in different ML domains with different sizes including Llama-2 (7 billion) (more details in in the appendix).

HE Libraries. We implement our HE core using both PALISADE and TenSEAL. Unless otherwise specified, our results show the evaluation of the PALISADE version.

Default Crypto Parameters. Unless otherwise specified, we choose the multiplicative depth of 1, the scaling factor bit digit of 52, an HE packing batch size of 4096, and a security level of 128 as our default HE cryptographic parameters during the evaluation.

Microbenchmark. For microbenchmarking HE overheads, we use an Intel 8-core 3.60GHz i7-7700 CPU with 32 GB memory and an NVIDIA Tesla T4 GPU on Ubuntu 18.04.6.

4.2 Optimizations

To mitigate the HE overhead surge, our optimization scheme **Selective Parameter Encryption** works by selecting sensitive portions of parameters for encrypted computation while leaving the rest in plaintext per desired overhead expectations and privacy promise. In this section, we first evaluate the overhead optimization from **Selective Parameter Encryption** and then use the state-of-the-art privacy attacks to evaluate the effectiveness of our selection defense during FL training.

Note that other parameter efficiency techniques Tang et al. (2019); Hu et al. (2021) for both training-from-scratch and fine-tuning scenarios can also be applied in our system before **Selective Parameter Encryption** and efficiently reducing the sizes of shared models directly helps with HE computation and communication efficiency (we also include preliminary results on this part in the appendix.

4.2.1 Optimized Overheads

We first examine the overhead optimization gains from **Selective Parameter Encryption**. We examine the overhead change when parameters with high privacy importance are selected and encrypted. Figure 7 shows the overhead reduction from only encrypting certain parts of models, where both overheads are nearly proportional to the size of encrypted model parameters, which is coherent with the general relationship between HE overheads and input sizes. Note that after 10% encryption per our **Selective Parameter Encryption**, the overheads are close to the ones of plaintext aggregation.

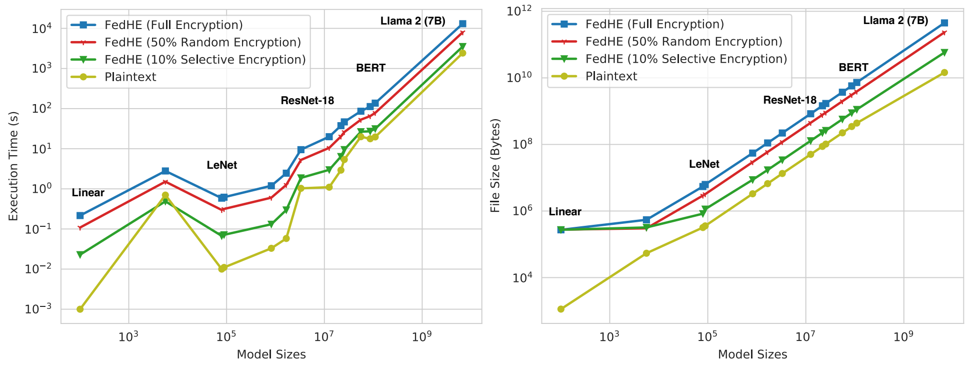


Figure 7: Computational (up) and Computation (down) Overhead Comparison For Models of Different Sizes (logarithmic scale): 10% Encryption is based on our selection strategy and 50% encryption is based on random selection.

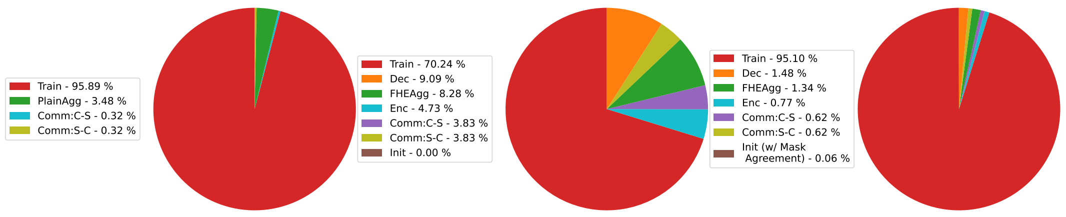


Figure 8: Time Distribution of A Training Cycle on ResNet-50: with a single AWS region bandwidth of 200 MB/s for plaintext FL (left), HE w/o optimization (middle), and HE w/ optimization (right). Optimization setup uses *DoubleSqueeze* Tang et al. (2019) with $k = 1,000,000$ and encryption mask with an encrypted ratio $s = 30\%$.

Figure 8 provides a perspective of overhead distribution to dissect the training cycle composition for the HE framework (both with and without optimizations) and the plaintext framework respectively with a single AWS region bandwidth. For a medium-sized model, the overheads (both computation and communication) from HE shift some portion of the local training procedure to aggregation-related steps compared to Non-HE, but not with an infeasible margin relatively speaking. Though generally smaller models require shorter training time, the overheads of the HE-based aggregation also drop proportionally.

4.2.2 Effectiveness of Selection Defense

To evaluate the defense effectiveness of **Selective Parameter Encryption**, we first use privacy sensitivity to generate a privacy map (Figure 5) and then verify the effectiveness of selection by performing gradient inversion (DLG Zhu et al. (2019)). We also provide defense results with Language Model Inversion Attacks Fowl et al. (2022) on Bert.

Defense effectiveness on CV tasks. We use image samples from CIFAR-100 to calculate the parameter sensitivities of the model. In the DLG attack experiments, we use Multi-scale Structural Similarity Index (MSSSIM), Visual Information Fidelity (VIF), and Universal Quality Image Index (UQI) as metrics to measure the similarity between recovered images and original training images to measure the attack quality hence the privacy leakage[2]. In Figure 9, compared to random encryption selection where encrypting 42.5% of the parameters can start to protect against attacks, our top-10% encryption selection according to the model privacy map only alone can defend against the attacks, meaning lower overall overhead with the same amount of privacy protection.

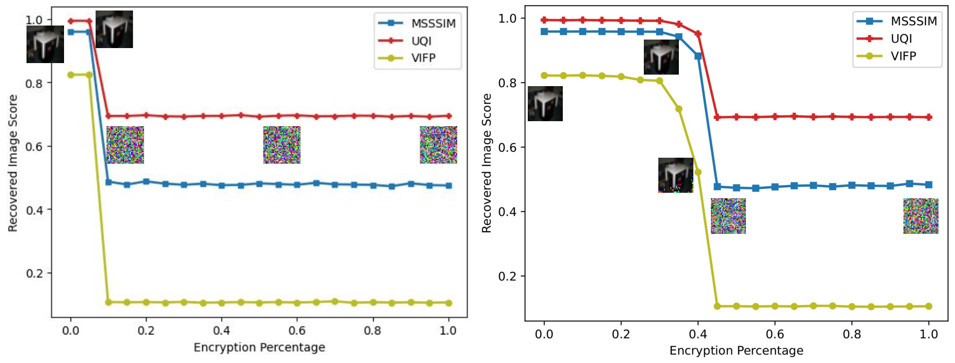
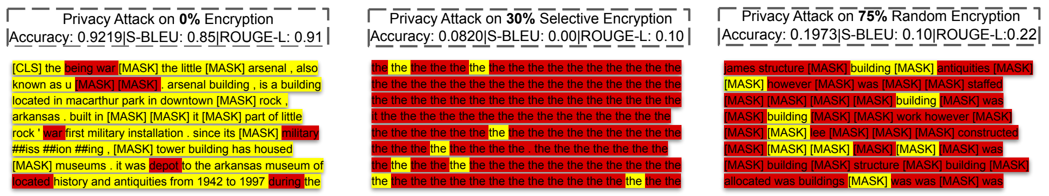


Figure 9: Selection Protection Against Gradient Inversion Attack Zhu et al. (2019) On LeNet with the CIFAR-100 Dataset: attack results when protecting top-s sensitive parameters (left) vs protecting random parameters (right). Each configuration is attacked 10 times and the best-recovered image is selected.

Defense effectiveness on NLP tasks. We use language samples from wikitext dataset in our experiment. As shown in Figure 10, with our sensitivity map indicating the top 30% privacy-sensitive parameters, our encryption mask can prevent inversion attacks that yields better defense results than randomly encrypting 75% of the model parameters.


Privacy Attack on **0%** Encryption
Accuracy: 0.9219|S-BLEU: 0.85|ROUGE-L: 0.91

[CLS] the being war [MASK] the little [MASK] arsenal , also known as u [MASK] [MASK] . arsenal building , is a building located in macarthur park in downtown [MASK] rock , arkansas . built in [MASK] [MASK] it [MASK] part of little rock ' war first military installation . since its [MASK] military ##iss ##ion ##ing , [MASK] tower building has housed [MASK] museums . it was depot to the arkansas museum of located history and antiquities from 1942 to 1997 during the

Privacy Attack on **30%** Selective Encryption
Accuracy: 0.0820|S-BLEU: 0.00|ROUGE-L: 0.10

the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the it the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the . the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the the

Privacy Attack on **75%** Random Encryption
Accuracy: 0.1973|S-BLEU: 0.10|ROUGE-L:0.22

james structure [MASK] building [MASK] antiquities [MASK] [MASK] however [MASK] was [MASK] [MASK] staffed [MASK] [MASK] [MASK] [MASK] building [MASK] was [MASK] building [MASK] [MASK] work however [MASK] [MASK] [MASK] lee [MASK] [MASK] [MASK] constructed [MASK] [MASK] [MASK] [MASK] [MASK] [MASK] was [MASK] building [MASK] structure [MASK] building [MASK] allocated was buildings [MASK] was was [MASK] was

Figure 10: Language Model Inversion Attacks Fowl et al. (2022) on Bert with the wikitext Dataset: Red indicates falsely-inverted words and Yellow indicates correctly-inverted words.

Empirical Selection Recipe. Our selection strategy works by first encrypting more important model parameters. Empirically, from our experimental investigation, encrypting top-30% most sensitive parameters, as well as the first and last model layers, tends to be robust to avoid information leakage Hatamizadeh et al. (2022) and attack defense (e.g. Figure 5), which can be used as a general guideline on top of model privacy maps.

[2]The image similarity metric library used is at `https://pypi.org/project/sewar/`.

5 Related Work

Existing Privacy Attacks On FL. Threats and attacks on privacy in the domain of Federated Learning have been studied in recent years Mothukuri et al. (2021). General FL privacy attacks can be categorized into two types: inference attacks Nasr et al. (2019); Wang et al. (2019); Truex et al. (2019b) and data leakage/reconstruction Criswell et al. (2014); Bhowmick et al. (2018); Hitaj et al. (2017). Attacks are usually carried out on the models to retrieve certain properties of data providers or even reconstruct the data in the training datasets. With direct access to more fine-grained local models trained on a smaller dataset Wang et al. (2019), the adversary can have a higher chance of a successful attack. Moreover, further attacks can be performed using GAN-based attacks to even fully recover the original data Hitaj et al. (2017). The majority of the privacy attacks can be traced back to the direct exposure of plaintext accesses to local models to other parties (usually the server).

Existing Non-HE Defense Mechanism. Local differential privacy has been adopted to protect local model updates by adding differential noise on the client side before the server-side aggregation Truex et al. (2019a); Byrd & Polychroniadou (2020) where privacy guarantee requires large-scale statistical noise on fine-grained local updates that generally degrades model performance by a large margin. On the other hand, other work proposes to apply zero-sum masks (usually pair-wise) to mask local model updates such that any individual local update is indistinguishable to the server Bonawitz et al. (2017); So et al. (2022). However, such a strategy introduces several challenges including key/mask synchronization requirements and federated learner dropouts. Compared to these solutions providing privacy protection in FL, HE is non-interactive and dropout-resilient (vs. general secure aggregation protocols Bonawitz et al. (2017); So et al. (2022)) and it introduces negligible model performance degradation (vs. noise-based differential privacy solutions Truex et al. (2019a); Byrd & Polychroniadou (2020)).

Existing HE-based FL Work. Existing HE-based FL work either apply restricted HE schemes (e.g. additive scheme Paillier) Zhang et al. (2020); Fang & Qian (2021); Jiang et al. (2021) without extensibility to further FL aggregation functions as well as sufficient performance and security guarantee (due to Paillier) or provide a generic HE implementation on FL aggregation Roth et al. (2022); IBM (2022); Jiang et al. (2021); Du et al. (2023); Ma et al. (2022). However, previous work still leaves the HE overhead increase issue as an open question. In our work, we propose a universal optimization scheme to largely reduce the overhead while providing promised privacy guarantees in a both systematic and algorithmic fashion, which makes HE-based FL viable in practical deployments.

6 Conclusion

In this paper, we propose FedML-HE, the first practical Homomorphic Encryption-based privacy-preserving FL system that supports encryption key management, encrypted FL platform deployment, encryption optimizations to reduce overhead, and is designed to support efficient foundation model federated training. We design Selective Parameter Encryption that selectively encrypts the most privacy-sensitive parameters to minimize the size of encrypted model updates while providing customizable privacy preservation. Future work includes quantitative and theoretical analysis of the trade-offs among privacy guarantee, system overheads, and model performance compared to other approaches (including difference privacy and secure aggregation approaches), and improving threshold-HE's performance in the FL setting as well as supporting decentralized primitives such as Proxy Re-Encryption Ateniese et al. (2006).

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pp. 308–318, 2016.

Aharoni, E., Adir, A., Baruch, M., Drucker, N., Ezov, G., Farkash, A., Greenberg, L., Masalha, R., Moshkowich, G., Murik, D., et al. Helayers: A tile tensors framework for large neural networks on encrypted data, 2011.

Aloufi, A., Hu, P., Song, Y., and Lauter, K. Computing blindfolded on data homomorphically encrypted under multiple keys: A survey. ACM Computing Surveys (CSUR), 54(9):1–37, 2021.

Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., and Wichs, D. Multiparty computation with low communication, computation and interaction via threshold fhe. In Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31, pp. 483–501. Springer, 2012.

Ateniese, G., Fu, K., Green, M., and Hohenberger, S. Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Transactions on Information and System Security (TISSEC), 9(1):1–30, 2006.

Bhowmick, A., Duchi, J., Freudiger, J., Kapoor, G., and Rogers, R. Protection against reconstruction and its applications in private federated learning. arXiv preprint arXiv:1812.00984, 2018.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1175–1191, 2017.

Boneh, D., Boyen, X., and Halevi, S. Chosen ciphertext secure public key threshold encryption without random oracles. In Cryptographers' Track at the RSA Conference, pp. 226–243. Springer, 2006.

Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT), 6(3):1–36, 2014.

Byrd, D. and Polychroniadou, A. Differentially private secure multi-party computation for federated learning in financial applications. In Proceedings of the First ACM International Conference on AI in Finance, pp. 1–9, 2020.

Cheon, J. H., Kim, A., Kim, M., and Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23, pp. 409–437. Springer, 2017.

Criswell, J., Dautenhahn, N., and Adve, V. Kcofi: Complete control-flow integrity for commodity operating system kernels. In 2014 IEEE symposium on security and privacy, pp. 292–307. IEEE, 2014.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.

Du, W., Li, M., Wu, L., Han, Y., Zhou, T., and Yang, X. A efficient and robust privacy-preserving framework for cross-device federated learning. Complex & Intelligent Systems, pp. 1–15, 2023.

Dwork, C. Differential privacy: A survey of results. In International conference on theory and applications of models of computation, pp. 1–19. Springer, 2008.

Fan, J. and Vercauteren, F. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144, 2012. URL `https://eprint.iacr.org/2012/144`. `https://eprint.iacr.org/2012/144`.

Fang, H. and Qian, Q. Privacy preserving machine learning with homomorphic encryption and federated learning. Future Internet, 13(4):94, 2021.

Fowl, L., Geiping, J., Reich, S., Wen, Y., Czaja, W., Goldblum, M., and Goldstein, T. Decepticons: Corrupted transformers breach privacy in federated learning for language models. arXiv preprint arXiv:2201.12675, 2022.

Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the forty-first annual ACM symposium on Theory of computing, pp. 169–178, 2009.

Gouert, C., Mouris, D., and Tsoutsos, N. G. New insights into fully homomorphic encryption libraries via standardized benchmarks. Cryptology ePrint Archive, 2022.

Han, S., Buyukates, B., Hu, Z., Jin, H., Jin, W., Sun, L., Wang, X., Xie, C., Zhang, K., Zhang, Q., et al. Fedmlsecurity: A benchmark for attacks and defenses in federated learning and llms. arXiv preprint arXiv:2306.04959, 2023.

Hatamizadeh, A., Yin, H., Roth, H. R., Li, W., Kautz, J., Xu, D., and Molchanov, P. Gradvit: Gradient inversion of vision transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10021–10030, 2022.

Hitaj, B., Ateniese, G., and Perez-Cruz, F. Deep models under the gan: information leakage from collaborative deep learning. In Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, pp. 603–618, 2017.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685, 2021.

IBM. Ibmfl crypto. `https://github.com/IBM/federated-learning-lib/blob/main/Notebooks/crypto_fhe_pytorch/pytorch_classifier_aggregator.ipynb`, 2022. Accessed: 2023-1-25.

Jiang, Z., Wang, W., and Liu, Y. Flashe: Additively symmetric homomorphic encryption for cross-silo federated learning. arXiv preprint arXiv:2109.00675, 2021.

Jin, W., Krishnamachari, B., Naveed, M., Ravi, S., Sanou, E., and Wright, K.-L. Secure publish-process-subscribe system for dispersed computing. In 2022 41st International Symposium on Reliable Distributed Systems (SRDS), pp. 58–68. IEEE, 2022.

Laud, P. and Ngo, L. Threshold homomorphic encryption in the universally composable cryptographic library. In International Conference on Provable Security, pp. 298–312. Springer, 2008.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. Proceedings of Machine learning and systems, 2:429–450, 2020.

Lu, J., Zhang, X. S., Zhao, T., He, X., and Cheng, J. April: Finding the achilles' heel on privacy for vision transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10051–10060, 2022.

Ma, J., Naas, S.-A., Sigg, S., and Lyu, X. Privacy-preserving federated learning based on multi-key homomorphic encryption. International Journal of Intelligent Systems, 37(9):5880–5901, 2022.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics, pp. 1273–1282. PMLR, 2017.

Mo, F., Borovykh, A., Malekzadeh, M., Haddadi, H., and Demetriou, S. Layer-wise characterization of latent information leakage in federated learning. In ICLR Distributed and Private Machine Learning workshop, 2020.

Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A., and Srivastava, G. A survey on security and privacy of federated learning. Future Generation Computer Systems, 115:619–640, 2021.

Nasr, M., Shokri, R., and Houmansadr, A. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In 2019 IEEE symposium on security and privacy (SP), pp. 739–753. IEEE, 2019.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. Sensitivity and generalization in neural networks: an empirical study. In International Conference on Learning Representations, 2018.

Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18, pp. 223–238. Springer, 1999.

Rasouli, M., Sun, T., and Rajagopal, R. Fedgan: Federated generative adversarial networks for distributed data. arXiv preprint arXiv:2006.07228, 2020.

Roth, H. R., Cheng, Y., Wen, Y., Yang, I., Xu, Z., Hsieh, Y.-T., Kersten, K., Harouni, A., Zhao, C., Lu, K., et al. Nvidia flare: Federated learning from simulation to real-world. arXiv preprint arXiv:2210.13291, 2022.

Shamir, A. How to share a secret. Communications of the ACM, 22(11):612–613, 1979.

Shokri, R. and Shmatikov, V. Privacy-preserving deep learning. In Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, pp. 1310–1321, 2015.

So, J., Nolet, C. J., Yang, C.-S., Li, S., Yu, Q., E Ali, R., Guler, B., and Avestimehr, S. Lightsecagg: a lightweight and versatile design for secure aggregation in federated learning. Proceedings of Machine Learning and Systems, 4:694–720, 2022.

Sokolić, J., Giryes, R., Sapiro, G., and Rodrigues, M. R. Robust large margin deep neural networks. IEEE Transactions on Signal Processing, 65(16):4265–4280, 2017.

Stripelis, D., Saleem, H., Ghai, T., Dhinagar, N., Gupta, U., Anastasiou, C., Ver Steeg, G., Ravi, S., Naveed, M., Thompson, P. M., et al. Secure neuroimaging analysis using federated learning with homomorphic encryption. In 17th International Symposium on Medical Information Processing and Analysis, volume 12088, pp. 351–359. SPIE, 2021.

Tang, H., Yu, C., Lian, X., Zhang, T., and Liu, J. Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. In International Conference on Machine Learning, pp. 6155–6165. PMLR, 2019.

Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., and Zhou, Y. A hybrid approach to privacy-preserving federated learning. In Proceedings of the 12th ACM workshop on artificial intelligence and security, pp. 1–11, 2019a.

Truex, S., Liu, L., Gursoy, M. E., Yu, L., and Wei, W. Demystifying membership inference attacks in machine learning as a service. IEEE Transactions on Services Computing, 14(6):2073–2089, 2019b.

Wang, J., Das, R., Joshi, G., Kale, S., Xu, Z., and Zhang, T. On the unreasonable effectiveness of federated averaging with heterogeneous data. arXiv preprint arXiv:2206.04723, 2022.

Wang, Z., Song, M., Zhang, Z., Song, Y., Wang, Q., and Qi, H. Beyond inferring class representatives: User-level privacy leakage from federated learning. In IEEE INFOCOM 2019-IEEE conference on computer communications, pp. 2512–2520. IEEE, 2019.

Wei, W., Liu, L., Loper, M., Chow, K.-H., Gursoy, M. E., Truex, S., and Wu, Y. A framework for evaluating client privacy leakages in federated learning. In Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25, pp. 545–566. Springer, 2020.

Yao, Y., Jin, W., Ravi, S., and Joe-Wong, C. Fedgcn: Convergence and communication tradeoffs in federated training of graph convolutional networks. Advances in neural information processing systems, 2023.

Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., and Liu, Y. Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC 2020), 2020.

Zhu, L., Liu, Z., and Han, S. Deep leakage from gradients. Advances in neural information processing systems, 32, 2019.

A Preliminaries

A.1 Federated Learning

Federated learning is first proposed in McMahan et al. (2017), which builds distributed machine learning models while keeping personal data on clients. Instead of uploading data to the server for centralized training, clients process their local data and share updated local models with the server. Model parameters from a large population of clients are aggregated by the server and combined to create an improved global model.

The FedAvg McMahan et al. (2017) is commonly used on the server to combine client updates and produce a new global model. At each round, a global model $\mathbf{W}_{\text{glob}}$ is sent to N client devices. Each client i performs gradient descent on its local data with E local iterations to update the model $\mathbf{W}_i$. The server then does a weighted aggregation of the local models to obtain a new global model, $\mathbf{W}_{\text{glob}} = \sum_{i=1}^{N} \alpha_i \mathbf{W}_i$, where α_i is the weighting factor for client i.

Typically, the aggregation runs using plaintext model parameters through a central server (in some cases, via a decentralized protocol), giving the server visibility of each local client's model in plaintext.

A.2 Homomorphic Encryption

- HE.*KeyGen*(λ): given the security parameter λ, the key generation algorithm outputs a key pair (pk, sk) and the related cryptographic context.
- HE.*Enc*(pk, m):the encryption algorithm takes in pk and a plaintext message m, then outputs the ciphertext c.
- HE.*Eval*(c, f):the encrypted evaluation algorithm takes in a ciphertext message c and a function f, then outputs the computation result c'.
- HE.*Dec*(sk, c'):the encryption algorithm takes in sk and a ciphertext message c', then outputs the plaintext m'.

Figure 11: General Scheme of Homomorphic Encryption

Homomorphic Encryption is a cryptographic primitive that allows computation to be performed on encrypted data without revealing the underlying plaintext. It usually serves as a foundation for privacy-preserving outsourcing computing models. HE has generally four algorithms (*KeyGen*, *Enc*, *Eval*, *Dec*) as defined in Figure 11. The fundamental concept is to encrypt data prior to computation, perform the computation on the encrypted data without decryption, and then decrypt the resulting ciphertext to obtain the final plaintext.

Since FL model parameters are usually not integers, our method is built on the Cheon-Kim-Kim-Song (CKKS) scheme Cheon et al. (2017), a (leveled) HE variant that can work with approximate numbers.

B Key Management And Threshold HE

Our general system structure assumes the existence of a potentially compromised aggregation server, which performs the HE-based secure aggregation. Alongside this aggregation server, there also exists a trusted key authority server that generates and distributes HE keys and related crypto context files to authenticated parties (as described previously in Algorithm 1 in the main paper. We assume there is no collusion between these two servers.

Moreover, secure computation protocols for more decentralized settings without an aggregation server are also available using cryptographic primitives such as Threshold HE Aloufi et al. (2021), Multi-Key HE Aloufi et al. (2021), and Proxy Re-Encryption Ateniese et al. (2006); Jin et al. (2022). In such settings, secure computation and decryption can be collaboratively performed across multiple parties without the need for a centralized point. We plan to introduce a more decentralized version of FedML-HE in the future. Due to the collaborative nature of such secure computation, the key management will act more as a coordination point instead of a trusted source for key generation.

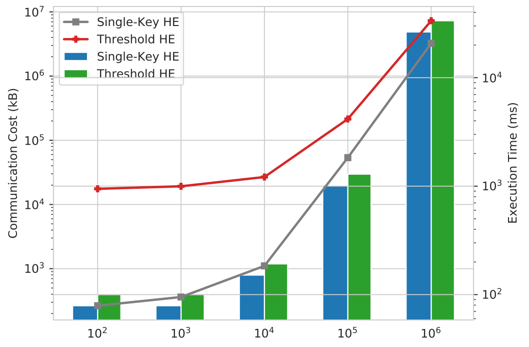


Figure 12: Microbenchmark of Threshold-HE-Based FedAvg Implementation: we use a two-party threshold setup. Both the single-key variant and the threshold variant are configured with an estimated precision of 36 for a fair comparison.

The threshold variant of HE schemes is generally based on Shamir's secret sharing Shamir (1979) (which is also implemented in PALISADE). Key generation/agreement and decryption processes are in an interactive fashion where each party shares partial responsibility of the task. Threshold key generation results in each party holding a share of the secret key and threshold decryption requires each party to partially decrypt the final ciphertext result and merge to get the final plaintext result. We provide benchmarkings of the threshold-HE-based FedAvg implementation in Figure 12.

C Framework APIs and Platform Deployment

C.1 Framework APIs

Table 3 shows the framework APIs in our system related to HE.

API Name	Description
pk, sk = **key_gen**(*params*)	Generate a pair of HE keys (public key and private key)
1d_local_model = **flatten**(*local_model*)	Flatten local trained model tensors into a 1D local model
enc_local_model = **enc**(*pk, 1d_model*)	Encrypt the 1D model
enc_global_model = **he_aggregate**(*enc_models[n], weight_factors[n]*)	Homomorphically aggregate a list of 1D local models
dec_global_model = **dec**(*sk, enc_global_model*)	Decrypt the 1D global model
global_model = **reshape**(*dec_global_model, model_shape*)	Reshape the 1D global model back to the original shape

Table 3: HE Framework APIs

C.2 Deploy Anywhere: A Deployment Platform MLOps For Edges/Cloud

We implement our deployment-friendly platform such that FedML-HE can be easily deployed across cloud and edge devices.. Before the training starts, a user uploads the configured server package and the local client package to the web platform. The server package defines the operations on the FL server, such as the aggregation function and client sampling function; the local client package defines the customized model architecture to be trained (model files will be distributed to edge devices in the first round of the training). Both packages are written in Python. The platform then builds and runs the docker image with the uploaded server package to operate as the server for the training with edge devices configured using the client package.

As shown in Figure 13, during the training, users can also keep tracking the learning procedure including device status, training progress/model performance, and FedML-HE system overheads (e.g., training time, communication time, CPU/GPU utilization, and memory utilization) via the web interface. Our platform keeps close track of overheads, which allows users to in real-time pinpoint HE overhead bottlenecks if any.

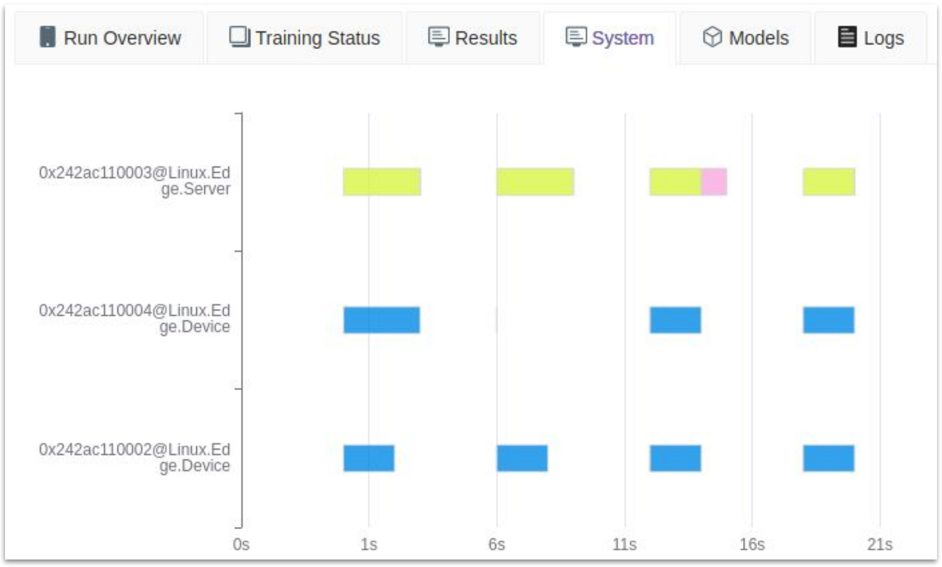

Run Overview
Training Status
Results
System
Models
Logs
0x242ac110003@Linux.Edge.Server
0x242ac110004@Linux.Edge.Device
0x242ac110002@Linux.Edge.Device
0s
1s
6s
11s
16s
21s

Figure 13: Deployment Interface Example of FedML-HE: Overhead distribution monitoring on each edge device (e.g. Desktop (Ubuntu), Laptop (MacBook), and Raspberry Pi 4), which can be used to pinpoint HE overhead bottlenecks and guide optimization.

D Additional Experiments

Model	Model Size	HE Time (s)	Non-HE Time (s)	Comp Ratio	Ciphertext	Plaintext	Comm Ratio
Linear Model	101	0.216	0.001	150.85	266.00 KB	1.10 KB	240.83
TimeSeries Transformer	5,609	2.792	0.233	12.00	532.00 KB	52.65 KB	10.10
MLP (2 FC)	79,510	0.586	0.010	60.46	5.20 MB	311.98 KB	17.05
LeNet	88,648	0.619	0.011	57.95	5.97 MB	349.52 KB	17.50
RNN(2 LSTM + 1 FC)	822,570	1.195	0.013	91.82	52.47 MB	3.14 MB	16.70
CNN (2 Conv + 2 FC)	1,663,370	2.456	0.058	42.23	103.15 MB	6.35 MB	16.66
MobileNet	3,315,428	9.481	1.031	9.20	210.41 MB	12.79 MB	16.45
ResNet-18	12,556,426	19.950	1.100	18.14	796.70 MB	47.98 MB	16.61
ResNet-34	21,797,672	37.555	2.925	12.84	1.35 GB	83.28 MB	16.60
ResNet-50	25,557,032	46.672	5.379	8.68	1.58 GB	97.79 MB	16.58
GroupViT	55,726,609	86.098	19.921	4.32	3.45 GB	212.83 MB	16.61
Vision Transformer	86,389,248	112.504	17.739	6.34	5.35 GB	329.62 MB	16.62
BERT	109,482,240	136.914	19.674	6.96	6.78 GB	417.72 MB	16.62
Llama 2	6.74 B	13067.154	2423.976	5.39	417.43 GB	13.5 GB	30.92

Table 4: Vanilla Fully-Encrypted Models of Different Sizes: with 3 clients; Comp Ratio is calculated by time costs of HE over time costs of Non-HE; Comm Ratio is calculated by file sizes of HE over file sizes of Non-HE. CKKS is configured with default crypto parameters.

We evaluate the HE-based training overheads (without our optimization in place) across various FL training scenarios and configurations. This analysis covers diverse model scales, HE cryptographic parameter configurations, client quantities involved in the task, and communication bandwidths. This helps us to identify bottlenecks in the HE process throughout the entire training cycle. We also benchmark our framework against other open-source HE solutions to demonstrate its advantages.

D.1 Parameter Efficiency Techniques in HE-Based FL

Table 5 shows the optimization gains by applying model parameter efficiency solutions in HE-Based FL.

Models	PT (MB)	CT	Opt (MB)
ResNet-18 (12 M) Tang et al. (2019)	47.98	796.70 MB	19.03
BERT (110 M) Hu et al. (2021)	417.72	6.78 GB	16.66

Table 5: Parameter Efficiency Overhead: PT means plaintext and CT means ciphertext. Communication reductions are 0.60 and 0.96.

D.2 Results on Different Scales of Models

We evaluate our framework on models with different size scales and different domains, from small models like the linear model to large foundation models such as Vision Transformer Dosovitskiy et al. (2020) and BERT Devlin et al. (2018). As Table 4 show, both computational and communicational overheads are generally proportional to model sizes.

Table 4 illustrates more clearly the overhead increase from the plaintext federated aggregation. The computation fold ratio is in general 5x $\sim$ 20x while the communication overhead can jump to a common 15x. Small models tend to have a higher computational overhead ratio increase. This is mainly due to the standard HE initialization process, which plays a more significant role when compared to the plaintext cost. The communication cost increase is significant for models with sizes smaller than 4096 (the packing batch size) numbers. Recall that the way our HE core packs encrypted numbers makes an array whose size is smaller than the packing batch size still requires a full ciphertext.

HE Batch Size	Scaling Bits	Comp (s)	Comm (MB)	Model Test Accuracy Δ (%)
1024	14	8.834	407.47	-0.28
1024	20	7.524	407.47	-0.21
1024	33	7.536	407.47	0
1024	40	7.765	407.47	0
1024	52	7.827	407.47	0
2048	14	3.449	204.50	-0.06
2048	20	3.414	204.50	-0.13
2048	33	3.499	204.50	0
2048	40	3.621	204.50	0
2048	52	3.676	204.50	0
4096	14	1.837	103.15	-1.85
4096	20	1.819	103.15	0.32
4096	33	1.886	103.15	0
4096	40	1.998	103.15	0
4096	52	1.926	103.15	0

Table 6: Computational & Communicational Overhead of Different Crypto Parameter Setups: tested with CNN (2 Conv+ 2 FC) and on 3 clients; model test accuracy Δs is the difference between the best plaintext global model and the best global encrypted global models.

D.3 Results on Different Cryptographic Parameters

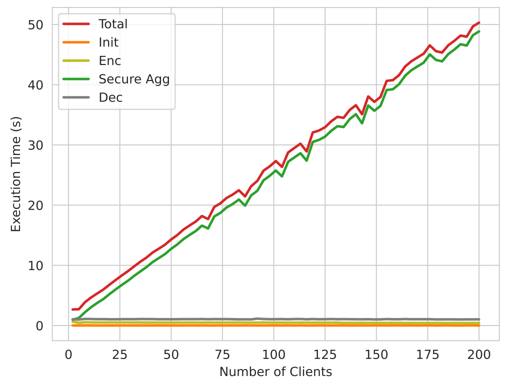
We evaluate the impacts of variously-configured cryptographic parameters. We primarily look into the packing batch size and the scaling bits. The packing batch size determines the number of slots packed in a single ciphertext while the scaling bit number affects the "accuracy" (i.e., how close the decrypted ciphertext result is to the plaintext result) of approximate numbers represented from integers.

From Table 6, the large packing batch sizes in general result in faster computation speeds and smaller overall ciphertext files attributed to the packing mechanism for more efficiency. However, the scaling factor number has an almost negligible impact on overheads.

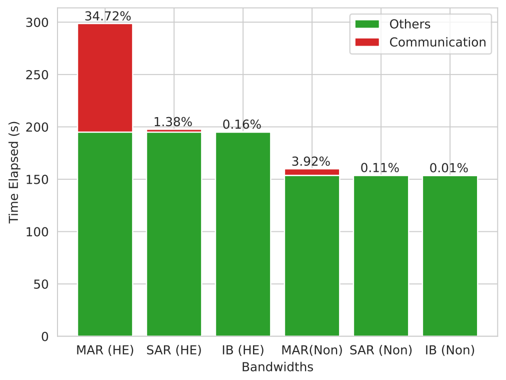
Unsurprisingly, it aligns with the intuition that the higher bit scaling number results in higher "accuracy" of the decrypted ciphertext value, which generally means the encrypted aggregated model would have a close model test performance to the plaintext aggregated model. However, it is worth mentioning that since CKKS is an approximate scheme with noises, the decrypted aggregated model can yield either positive or negative model test accuracy Δs, but usually with a negative or nearly zero Δ.

D.4 Impact from Number of Clients

As real-world systems often experience a dynamic amount of participants within the FL system, we evaluate the overhead shift over the change in the number of clients. Figure 14a breaks down the cost distribution as the number of clients increases. With a growing number of clients, it also means proportionally-added ciphertexts as inputs to the secure aggregation function thus the major impact is cast on the server. When the server is overloaded, our system also supports client selection to remove certain clients without largely degrading model performance.



(a) Step Breakdown of HE Computational Cost vs. Number of Clients (Up to 200): tested on fully-encrypted CNN



(b) Impact of Different Bandwidths on Communication and Training Cycles on Fully-Encrypted ResNet-50: HE means HE-enabled training and Non means plaintext. Others include all other procedures except communication during training. Percentages represent the portion of communication cost in the entire training cycle.

Figure 14: Results on Different Number of Clients and Communication Setup

D.5 Communication Cost on Different Bandwidths

FL parties can be allocated in different geo-locations which might result in communication bottlenecks. Typically, there are two common scenarios: (inter) data centers and (intra) data centers. In this part, we evaluate the impact of the bandwidths on communication costs and how it affects the FL training cycle. We categorize communication bandwidths using 3 cases:

- Infiniband (IB): communication between intra-center parties. 5 GB/s as the test bandwidth.
- Single AWS Region (SAR): communication between inter-center parties but within the same geo-region (within US-WEST). 592 MB/s as the test bandwidth.
- Multiple AWS Region (MAR): communication between inter-center parties but across the different geo-region (between US-WEST and EU-NORTH). 15.6 MB/s as the test bandwidth.

As shown in Figure 14b, we deploy FedML-HE on 3 different geo-distributed environments, which are operated under different bandwidths. It is obvious that the secure HE functionality has an enormous impact on low-bandwidth environments while medium-to-high-bandwidth environments suffer limited impact from increased communication overhead during training cycles, compared to Non-HE settings.

D.6 Different Encryption Selections

Table 7 shows the overhead reductions with different selective encryption rates.

Selection	Comp (s)	Comm	Comp Ratio	Comm Ratio
Enc w/ 0%	17.739	329.62 MB	1.00	1.00
Enc w/ 10%	30.874	844.49 MB	1.74	2.56
Enc w/ 30%	50.284	1.83 GB	2.83	5.69
Enc w/ 50%	70.167	2.83 GB	3.96	8.81
Enc w/ 70%	88.904	3.84 GB	5.01	11.93
Enc w/ All	112.504	5.35 GB	6.34	16.62

Table 7: Overheads With Different Parameter Selection Configs Tested on Vision Transformer: “Enc w/ 10%” means performs encrypted computation only on 10% of the parameters; all computation and communication results include overheads from plaintext aggregation for the rest of the parameters.

D.7 Comparison with Other FL-HE Frameworks

We compare our framework to the other open-sourced FL frameworks with HE capability, namely NVIDIA FLARE (NVIDIA) and IBMFL.

Both NVIDIA and IBMFL utilize Microsoft SEAL as the underlying HE core, with NVIDIA using OpenMinded’s python tensor wrapper over SEAL and TenSEAL; IBMFL using IBM’spython wrapper over SEAL and HELayers (HELayers also has an HElib version). Our HE core module can be replaced

Frameworks	HE Core	Key Management	Comp (s)	Comm (MB)	HE Multi-Party Functionalities
Ours	PALISADE	✓	2.456	105.72	PRE, ThHE
Ours (w/ Opt)	PALISADE	✓	0.874	16.37	PRE, ThHE
Ours	SEAL (TenSEAL)	✓	3.989	129.75	—
Nvidia FLARE (9a1b226)	SEAL (TenSEAL)	✓	2.826	129.75	—
IBMFL (8c8ab11)	SEAL (HELayers)	○	3.955	86.58	—
Plaintext	—	—	0.058	6.35	—

Table 8: Different Frameworks: tested with CNN (2 Conv + 2 FC) and on 3 clients; Github commit IDs are specified. For key management, our work uses a key authority server; FLARE uses a security content manager; IBMFL currently provides a local simulator.

with different available HE cores, to give a more comprehensive comparison, we also implement a TenSEAL version of our framework for evaluation.

Table 8 demonstrates the performance summary of different FedML-HE frameworks using an example of a CNN model with 3 clients. Our PALISADE-powered framework has the smallest computational overhead due to the performance of the PALISADE library. In terms of communication cost, FedML-HE (PALISADE) comes second after IBMFL's smallest file serialization results due to the efficient packing of HELayers' Tile tensors Aharoni et al. (2011).

Note that NVIDIA's TenSEAL-based realization is faster than the TenSEAL variant of our system. This is because NVIDIA scales each learner's local model parameters locally rather than weighing ciphertexts on the server. This approach reduces the need for the one multiplication operation usually performed during secure aggregation (recall that HE multiplications are expensive). However, such a setup would not suit the scenario where the central server does not want to reveal its weighing mechanism per each individual local model to learners as it reveals partial (even full in some cases) information about participants in the system.