SuperFL: Privacy-Preserving Federated Learning with Efficiency and Robustness

Yulin Zhao, Hualin Zhou, Zhiguo Wan, *Member, IEEE*

Abstract

Federated Learning (FL) accomplishes collaborative model training without the need to share local training data. However, existing FL aggregation approaches suffer from inefficiency, privacy vulnerabilities, and neglect of poisoning attacks, severely impacting the overall performance and reliability of model training. In order to address these challenges, we propose SuperFL, an efficient two-server aggregation scheme that is both privacy preserving and secure against poisoning attacks. The two semi-honest servers $\mathcal{S}_0$ and $\mathcal{S}_1$ collaborate with each other, with a shuffle server $\mathcal{S}_0$ in charge of privacy-preserving random clustering, while an analysis server $\mathcal{S}_1$ responsible for robustness detection, identifying and filtering malicious model updates. Our scheme employs a novel combination of homomorphic encryption and proxy re-encryption to realize secure server-to-server collaboration. We also utilize a novel sparse matrix projection compression technique to enhance communication efficiency and significantly reduce communication overhead. To resist poisoning attacks, we introduce a dual-filter algorithm based on trusted root, combine dimensionality reduction and norm calculation to identify malicious model updates.

Extensive experiments validate the efficiency and robustness of our scheme. SuperFL achieves impressive compression ratios, ranging from 5-40x, under different models while maintaining comparable model accuracy as the baseline. Notably, our solution demonstrates a maximal model accuracy decrease of no more than 2% and 6% on the MNIST and CIFAR-10 datasets respectively, under specific compression ratios and the presence of malicious clients.

I. Introduction

In recent years, the field of machine learning has witnessed a remarkable success story in federated learning [1] – a groundbreaking paradigm that allows multiple clients to collaboratively train a model without the need to share their raw training data. This innovative approach has not only advanced the frontiers of collaborative machine learning but has also emerged as a promising solution to address the long-standing issue of data isolation, often referred to as the "isolated islands problem."

Federated learning achieves its collaborative learning goals by sharing model updates from clients. While ensuring the confidentiality of the original training data, this strategy introduces three new concerns: privacy leakage, poisoning attack and communication overhead. The first problem is that federated learning can cause privacy leakage, because the model updates uploaded by clients in federated learning can cause privacy security. Some previous work has proved that it is possible to extract private training data from publicly shared model updates [2]. The second problem is that federated learning is vulnerable to poisoning attacks [3], [4] due to its distributed environment. Specifically, malicious clients upload malicious model updates to destroy the global model accuracy of training. The third issue is the significant communication overhead of federated learning, as a substantial number of model parameters need to be exchanged between the various participants and the central server, resulting in a substantial amount of data transmission.

In order to solve the above problems, numerous related studies has been conducted, however, most of the existing schemes either provide privacy protection [5], [6], [7], or resist poisoning attacks [8], [9], [10], or reduce communication overhead [11], [12]. Some attempts have since been made to achieve these the properties [13], [14], [15]. However, their schemes are limited and each lacks comprehensive consideration.

From the perspective of most schemes, it is still difficult to solve the three properties at the same time. One major challenge is balancing privacy and robustness. Many privacy-preserving schemes do not support robust aggregation, because under privacy protection, it is difficult for the server to access and analyze the model updates of any client. The second obstacle is achieving robust aggregation under model compression, because model compression will cause changes in structure and accuracy, which makes it difficult for robust algorithms to filter malicious model update detection. The third difficulty is how to combine compression and privacy protection technique. Many model compression methods rely on specific operations and structures, which are not applicable to privacy protection techniques such as homomorphic encryption and secret sharing. Therefore, it is crucial to identify a solution that facilitates robust aggregation against malicious model updates while also

Manuscript created January, 2024. (Yulin Zhao and Hualin Zhou contributed equally to this work.) (Corresponding author: Zhiguo Wan.)

Yulin Zhao and Hualin Zhou are with Institute of Software Chinese Academy of Sciences, Beijing 100190, China, and also with Hangzhou Institute for Advanced Study, University of Chinese Academy of Sciences, Hangzhou 310024, China (e-mail: zhaoyulin22@mails.ucas.ac.cn; zhouhualin22@mails.ucas.ac.cn).

Zhiguo Wan is with Zhejiang Laboratory, Hangzhou, Zhejiang 311121, China (e-mail: wanzhiguo@zhejianglab.com).

considering privacy preservation and reducing communication overhead.

Recently, a few studies have implemented the above three properties of federated learning schemes, including RoFL [16], ELSA [17]. However, RoFL uses expensive zero-knowledge proof to enforce norm bounds so that defenses attacks, which makes the efficiency very low. ELSA [17] is not suitable for running other robust aggregation methods such as krum, median, because it is expensive to run these defenses under secure multi-party computation.

For these reasons, we propose SuperFL. Recent studies have provided us with ideas for proposing SuperFL. The CKKS is a fully homomorphic encryption technique proposed by Cheon et al [18], which supports floating-point operations and is more efficient. Therefore, it is considered as a privacy-preserving technique. In order to better adapt to homomorphic encryption, we consider a linear additive compression method [19] rather than a sparse method with indexed transmission [11], [12]. For the consideration of robust aggregation, we introduce a proxy re-encryption in our scheme. Prior privacy-preserving schemes have struggled with robust aggregation due to the inability of servers to access model updates while encrypted, thereby hindering their ability to perform corresponding robust operations. Our novel approach involves randomly dividing all encrypted model updates into multiple clusters and then aggregating them according to the additivity of homomorphic encryption. Cluster aggregation offers the advantage of protecting individual model update, such that even upon decryption, the server only sees the average aggregated model update, thus ensuring privacy. We employ proxy re-encryption to transform cluster aggregated ciphertexts prior to decryption, facilitating the implementation of subsequent robust aggregation. In terms of robust aggregation, we first introduce a trusted root (with minimal data collection) as a benchmark, given that random cluster aggregation can result in more than half of the clusters being malicious. We utilize a dual filter combining norm and PCA defense to screen for malicious updates. Norm filtering can identify malicious model updates with larger update values, while PCA effectively discriminates between compressed update values. Finally, TABLE I gives a comparison of the our's SuperFL scheme with previous FL schemes. The results indicate that our scheme improves communication efficiency while achieving privacy protection, and resists poisoning attacks with the support of dual filters.

The principal contributions of our work are delineated as follows:

- We propose a novel collaborative two-server aggregation mode, comprising a shuffle server $\mathcal{S}_0$ and an analysis server $\mathcal{S}_1$. $\mathcal{S}_0$ handles random clustering, while $\mathcal{S}_1$ manages robust aggregation. We combine fully homomorphic encryption CKKS and proxy re-encryption to realize a secure collaborative aggregation framework. This paradigm supports Byzantine robustness while protecting the privacy of clients. The key idea is to resolve the conflict between robustness and privacy protection from different perspectives of two non-colluding servers.

TABLE I
A COMPARA TIVE SUMMARY BETWEEN OUR SCHEME AND PREVIOUS SCHEMES

FL Schemes	**Compression**	**Privacy Preserve**	**Poisoning Resilience**	**Defense Mechanism**
FedAvg	×	×	×	No
PBFL [14]	×	✓	✓	Single
CPFed [15]	✓	✓	×	No
lp-proj [13]	✓	×	✓	Single
ELSA [17]	×	✓	✓	Single
RoFL [16]	✓	✓	✓	Single
SuperFL	✓	✓	✓	Double

Note: Compression:Whether compressing model updates or not. Privacy Preserve:Whether possessing privacy-presserving capacity or not. Poisoning Resilience:Whether possessing defense method against malicious model updates or not. Defense Mechanism:One defense mechanism or multiple defense mechanisms.

- We optimize and implement an efficient sketch compression method utilizing coordinate embedding [20] and sparse embedding [21]. This method effectively reduces communication overhead and integrates seamlessly with homomorphic encryption schemes.
- Based on the assumption of a trusted root data set, we design an effective defense method against Byzantine attacks. This method combines feature dimension reduction and norm detection to eliminate malicious model updates, providing resistance against poisoning attacks. Our robust method exhibits strong defense capabilities against both targeted and untargeted attacks. It is also compatible with our compression and secure aggregation methods.
- We conduct extensive experiments using well-established datasets, comparing our scheme with previous state-of-the-art approaches. The results illustrate the superior performance of our scheme in terms of both robustness and efficiency.

The remainder of this paper is structured as follows: Section II introduces relevant works in the field related to our paper. Section III introduces preliminary knowledge. Section IV introduces the system model, design goals. The specific details of our scheme are described in Section V. Section VI presents the security analysis and discussion. Section VII introduces our experiment results. Finally, section VIII wraps up with a conclusion.

II. Related work

Focusing on several challenges to be solved in our scheme, we will introduce the corresponding aspects of federated learning related work.

A. *Secure Aggregation of Federated Learning*

To date, primary secure aggregation methods in federated learning involve homomorphic encryption [22] [23], secure multi-party computation [24] [5] [25], and differential privacy [26] [27] [6]. However, these methods fall short in resisting poisoning attacks due to privacy techniques safeguarding the uploaded model updates. Server can only access aggregated values, concealing malicious updates within, making it challenging to counter poisoning attacks. To align with secure aggregation, Aramoon et al. [28] and Li et al. [29] propose a robust defense strategy after the server collects model updates from groups. Aramoon et al.'s scheme, based on secret sharing, sacrifices system security during group collection. Li et al.'s scheme, using homomorphic encryption, requires numerous edge servers and demands the central server to maintain verification data at scale, making it impractical.

B. Federated Learning Against Poisoning Attacks

Poisoning attacks are common in federated learning, with adversaries employing data poisoning attacks (e.g., label flipping) and model poisoning attacks (e.g., gauss attacks) to compromise the global model. Data poisoning attacks indirectly affect the global model by contaminating local data, while model poisoning attacks manipulate model updates during communication, impacting global model accuracy. The widely-used FedAvg [30] employs a simple average aggregation strategy, making it vulnerable to poisoning attacks. Robust aggregation schemes, including Krum/Multi-Krum [8], Trimmed-Mean/Median [31], FLTrust [32], Norm Bounding [33], signGaurd [10], and Bulyan [34], aim to address these issues. However, they analyze model updates in plaintext, compromising client privacy. Miao et al. [14] proposed a privacy-protected robust aggregation using homomorphic encryption and cosine similarity. However, the computation cost under ciphertext is high, especially for large models. Lycklama's RoFL [16] resists poisoning attacks by imposing constraints on norms using Bulletproofs [35], but its collective verification by all clients and central servers poses efficiency challenges. ELSA [17] uses MPC for norm constraints, offering more efficiency, but secure multi-party computation limits support for complex defense methods like Krum [8] and Median [31] due to cost constraints.

C. Model Compression of Federated Learning

The considerable communication overhead in federated learning presents a major challenge. To tackle this, compression methods like random-k sparsification [11] and Top-K sparsification [12] have been proposed, selecting model update parameters randomly or based on the k largest absolute values. However, ensuring alignment of these selected parameters across diverse models for federated aggregation proves challenging, especially when integrating with privacy protection techniques like homomorphic encryption. Lu et al. [36] proposed a scheme compatible with secure aggregation and Top-K, but it introduces an additional communication round, causing significant time overhead. Sketching is an effective model compression method, like FetchSGD [37]. Despite supporting linear addition, its efficiency in multiple random projection sketch calculations is relatively low. Zhao et al.'s compression method integrates coordinate embedding [20], eliminating redundant projection operations. However, their scheme [19] lacks privacy protection, and attempts to introduce differential privacy result in a trade-off between privacy and data availability, impacting model performance and complicating malicious update detection by the server. Lin et al. [13] proposed a method combining subspace learning model compression and robustness, but it inadequately protects client privacy.

III. Preliminaries

This section briefly introduces the knowledge of federated learning, poisoing attack, byzantine-robust federated learning, threshold fully homomorphic encryption and proxy re-encryption.

A. Federated Learning

Federated Learning (FL) is a machine learning in which multiple clients collaborate to solve machine learning problems under the coordination of a central server or service provider. The original data of each client is stored locally and will not be exchanged or transferred. Finally, all clients train a global model together.

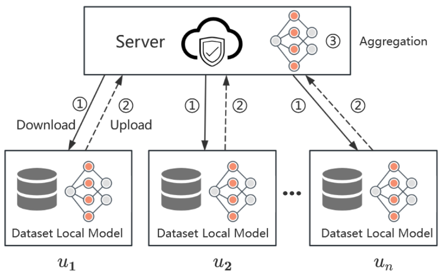


Fig. 1. The system architecture of federated learning

In federated learning setting, Assuming that n clients $\{u_1, u_2, \ldots, u_n\}$ and one central parametric aggregation server in federated learning, each client u_i has a local dataset D_i, and each communication round $t, t = 1, 2, 3, \ldots, T$. The federated learning algorithm runs as follows:

step①: each client u_i downloads the latest global model from the server.

step②: each client u_i trains based on its own local dataset D_i, then obtain local model update $\mathbf{g}_i^r$ and submit it to the server.

step③: the server aggregates local model updates $\mathbf{g}_i^r$ from all the clients, $i = 1, 2, \ldots, n$ and gets a new global model $\theta_{r+1} = \theta_r - G_r$, where G_r is $\frac{1}{n}\sum_{i=1}^{n} \mathbf{g}_i^r$.

The above steps are repeated until convergence or number of training count reached.

B. Fully Homomorphic Encryption

Fully homomorphic encryption perform any number of addition and multiplication operations on encrypted data without decrypting it. In 2009, Craig Gentry first proposed an fully homomorphic algorithm based on ideal lattices [38]. A typical FHE scheme is Cheon-Kim-Kim-Song(CKKS) [18] which supports floating point-like Single Instruction Multiple Data (SIMD) multiplication. Let N be a power-of-two, a positive integer $M = 2N$, a real vector $\mathbf{z}$, the polynomial ring $R = \mathbb{Z}[X]/(X^N + 1)$ is the M-th cyclotomic field's ring of integers, cyclotomic ring $S = \mathbb{R}[X]/(X^N + 1)$. Let $R_Q = R/QR$ for the residue ring of R modulo an integer Q. For convenience, give a fix integer $p > 0$, a fixed integer k, a modulus Q_0. For a level $0<l \leq L$, a ciphertext of level l is a vector in $R^k_{Q_l}$, where $Q_l = p^l Q_0$. For encode and decode, first define the space $\mathbb{H} = \{\mathbf{z} = (z_j)_{j\in\mathbb{Z}^*_M} \in \mathbb{C}^N : z_j = \overline{z_{-j}},\ \forall j \in \mathbb{Z}^*_M\}$, let T be a multiplicative subgroup of Z^*_M satisfying $Z^*_M/T = \{\pm 1\}$, where $\mathbb{Z}^*_M = \{x \in \mathbb{Z}_M : \gcd(x, M) = 1\}$ for the multiplicative group of units in $\mathbb{Z}_M$. Canonical embedding map $\sigma : \mathcal{S} \to \mathbb{C}^N$, space $\mathbb{H}$ can be identified with $\mathbb{C}^{N/2}$ via the natural projection π, which can be represented by $(z_j)_{j\in\mathbb{Z}^*_M} \mapsto (z_j)_{j\in T}$. The encoding algorithm transforms a vector $\mathbf{z} = (z_i)_{i\in T}$ into a polynomial $m(X) \in \mathcal{R}$ through $\sigma^{-1}\circ\pi^{-1}$, where $\pi^{-1}(\mathbf{z})[j]$ is z_j if $j \in T$, the decoding algorithm is to transform an arbitrary polynomial $m(X) \in \mathcal{R}$ into a complex vector $\mathbf{z}$ such that $\mathbf{z} = \pi \circ \sigma(m) \in \mathbb{C}^{N/2}$.

CKKS consists of the following algorithms:

KeyGen(1^λ). For a security parameter λ, generate a secret value s_k for decryption, a public information p_k for encryption, and a evaluation key evk.

Ecd$(\mathbf{z}, \Delta) \to m$. For a $(N/2)$-dimensional vector $\mathbf{z} = (z_j)_{j\in T}$ of complex numbers and a scaling factor Δ. Firstly, expands $\mathbf{z}$ into $\pi^{-1}(\mathbf{z}) \in \mathbb{H}$, then multiply the scaling factor Δ, and finally computes its discretization to $\sigma(\mathcal{R})$. Return the corresponding integral polynomial $m = \sigma^{-1}(\lfloor \Delta \cdot \pi^{-1}(z) \rceil_{\sigma(\mathcal{R})})$.

Dcd$(m, \Delta) \to \mathbf{z}$. For an input polynomial $m \in \mathcal{R}$, output the vector $\mathbf{z} = \pi \circ \sigma(\Delta^{-1} \cdot m)$.

Enc(m, pk). For a given polynomial $m \in \mathcal{R}$, output a ciphertext $c \in R^k_{Ql}$.

Dec$(\mathbf{c}, sk) \to m$. For a ciphertext $\mathbf{c}$ at level l, this algorithm outputs a polynomial $m' \leftarrow \langle \mathbf{c}, sk \rangle \pmod{Q_l}$ using the secret key sk.

Add$(\mathbf{c}_1, \mathbf{c}_2)$. For given encrypts of m_1 and m_2, output an encryption of $m_1 + m_2$. The error in outputting ciphertext is limited by the sum of two errors in the input ciphertext.

Mult$(\mathbf{c}_1, \mathbf{c}_2, evk)$. For a pair of ciphertexts $(\mathbf{c}_1, \mathbf{c}_2)$, output a ciphertext $\mathbf{c}_{\text{mult}} \in R^k_{Q_l}$ using the evaluation key evk.

C. Proxy Re-Encryption

Assuming that there are two participants Alice and Bob, in the Proxy Re-Encryption scheme, the semi-trusted proxy is given a re-encryption key, and the proxy can convert the ciphertext under the Alice public key to the ciphertext under the Bob public key. However, the proxy cannot learn any useful information of the encrypted message under either key. A fully homomorphic encryption proxy re-encryption scheme [39] is proposed by Polyakov et al. The main idea is to use the FHE key switching to perform the proxy re-encryption of the ciphertext with the re-encryption key, which represents the encryption of the old secret key using the public key of the new secret key. The specific proxy re-encryption includes the following algorithms:

Setup$(1^\lambda) \to pp$. According to a security parameter λ, the setup algorithm outputs public parameters pp.

KeyGen$(pp, 1^\lambda) \to (pk, sk)$. For public parameters pp and security parameter 1^λ, the key generation algorithm outputs a public-secrect key pairs (pk, sk).

ReKeyGen$(pp, sk_i, pk_j) \to rk_{i\to j}$. For a secret key sk_i of the delegator party i, a public key pk_j of the delegatee party j and public parameters pp, output a re-encryption key $rk_{i\to j}$ from party i to party j.

Enc$(pp, pk, m) \to c$. For a public key pk, a plaintext m and public parameters pp, output the ciphertext c.

ReEnc$(pp, pk, c_i) \to c_j$. For a re-encryption key $rk_{i\to j}$, a ciphertext c_i of party i and public parameters pp, transforms c_i into ciphertext c_j that party j can decrypt.

Dec$(pp, sk, c) \to m$. For a secret key sk, a ciphertext c and public parameters pp, recovers message m.

D. Sketching and Coordinate-wise Embedding

Sketching proves effective in tensor decomposition by approximating the original tensor with a reduced-dimensional sketch component. This approach significantly reduces calculation and storage requirements without sacrificing much accuracy [40].

The essence of sketching involves the projection operation on the high-dimensional tensor. Traditionally, a common method employs crafting a projection matrix, mapping the high-dimensional tensor to a low-dimensional space. However, to ensure model training convergence, it is crucial that the tensor in the final low-dimensional space can be reconstructed to approximate the original high-dimensional tensor. The coordinate embedding technique [20] adeptly facilitates constructing the projection matrix, efficiently meeting this requirement.

Definition 1 (α-coordinate-wise embedding)**.** *Given the parameter $\alpha \in \mathbb{R}$, we say a randomized matrix $R \in \mathbb{R}^{b\times\ell}$ with distribution Π satisfies α-coordinate-wise embedding property if for any fixed vector $g, h \in \mathbb{R}^\ell$, we have*

1. $\mathbb{E}_{R\sim\Pi}[h^\top R^\top R g] = h^\top g$,
2. $\mathbb{E}_{R\sim\Pi}[(h^\top R^\top R g)^2] \leq (h^\top g)^2 + \dfrac{\alpha}{b}||h||_2^2 \cdot ||g||_2^2$.

In the above definition, α is a small constant for the common sketch projection matrix, if h is a one-hot vector, we can get

1. $\mathbb{E}_{R\sim\Pi}[R^\top R g] = g$,
2. $\mathbb{E}_{R\sim\Pi}[(R^\top R g)^2] \leq (1 + \dfrac{\alpha\ell}{b}) \cdot ||g||_2^2$.

Through the above derivation, we can get the method of transforming high-dimensional tensor into low-dimensional tensor, and use coordinate embedding technique to realize the unbiased estimation of the original tensor. Zhao et al. [19] used this dimension reduction method to compress the model, and gave the relevant model training convergence proof. Coordinate embedding is suitable for many projection methods, such as sparse projection [21].

Definition 2 (Sparse embedding matrix based on coordinate embedding)**.** *Let $h : [n] \times [s] \to [\frac{b}{s}]$ and $\sigma : [n] \times [s] \to \{-1, 1\}$ are random hash functions, we say $R \in \mathbb{R}^{b\times n}$ is a* *sparse embedding matrix based on coordinate embedding with parameter s if $R_{\frac{(j-1)b}{s}+h(i,j),i} = \frac{\sigma(i,j)}{\sqrt{s}}$ for all $(i,j) \in [n] \times [s]$ and all other entries to zero.*

IV. PROBLEM FORMULATION

In this section, we formalize the system model, threat model, and design goals, respectively.

A. System Model

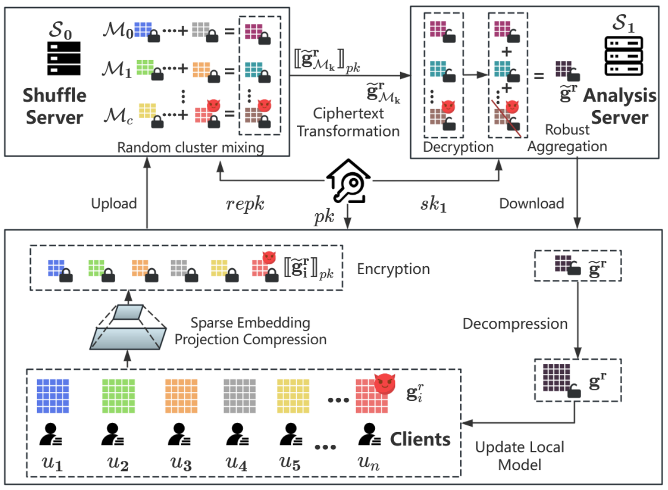

$\mathcal{S}_0$
Shuffle Server
$\mathcal{M}_0$
$\mathcal{M}_1$
$\mathcal{M}_c$
Random cluster mixing
$[\![\widetilde{\mathbf{g}}^{\mathbf{r}}_{\mathcal{M}_k}]\!]_{pk}$
$\widetilde{\mathbf{g}}^{\mathbf{r}}_{\mathcal{M}_k}$
Ciphertext Transformation
$\mathcal{S}_1$
$\widetilde{\mathbf{g}}^{\mathbf{r}}$
Analysis Server
Robust Aggregation
Decryption
Upload
$repk$
pk
sk_1
Download
$[\![\widetilde{\mathbf{g}}^{\mathbf{r}}_{\mathbf{i}}]\!]_{pk}$
Encryption
$\widetilde{\mathbf{g}}^{\mathbf{r}}$
Sparse Embedding Projection Compression
Decompression
$\mathbf{g}^{\mathbf{r}}$
$\mathbf{g}^{r}_{i}$
Clients
u_1
u_2
u_3
u_4
u_5
u_n
Update Local Model

Fig. 2. System Model.

In our paper, we consider a federated learning architecture for privacy protection, as shown in fig 2. Our proposed system model consists of follow classes:

- Key Generation Center (KGC): The KGC is an independent and trusted organization that distributes and manages public key, private key and proxy re-encrypt key.
- Clients: Clients have own local datasets. Clients only have public key. All clients' public key is same. During the learning process, each client trains its local model over private dataset and exchanges the encrypted compressed local model updates through the $\mathcal{S}_0$.
- Shuffle Server ($\mathcal{S}_0$): The $\mathcal{S}_0$ server is mainly used for random cluster aggregation and ciphertexts transformation. Firstly, $\mathcal{S}_0$ shuffles ciphertexts of model updates from all clients and aggregates them in a clustered manner. Then $\mathcal{S}_0$ uses a proxy key to transform the ciphertexts. Finally, it sends the results back to $\mathcal{S}_1$.
- Analysis Server ($\mathcal{S}_1$): Server has strong computing power and is a robust aggregator for malicious update filtering. Initially, it decrypts ciphertexts from $\mathcal{S}_0$ with sk_1. After obtaining the decryption results, the correct model update values are aggregated using a robust algorithm. Finally, the aggregated results are sent to each client.

Our system undergoes several steps. In the initialization phase, KGC generates the public security parameter pp, along with two pairs of public and secret keys (pk, sk) and (pk_1, sk_1). Additionally, it creates the proxy re-encryption key $repk \leftarrow \textbf{ReKeyGen}(pp, sk_1, pk)$. The public key pk is broadcast to each client, while $repk$ and sk_1 are sent to the server $\mathcal{S}_0$ and $\mathcal{S}_1$ respectively. Subsequently, $\mathcal{S}_1$ generates the shared projection matrix $\mathbf{P}$ required by the compression and broadcasts it to each client. In the first stage, clients locally compress the model update in accordance with $\mathbf{P}$, encrypt it using pk, and transmit the ciphertexts to the server $\mathcal{S}_0$. Moving to the second stage, $\mathcal{S}_0$ shuffles and mixes the received model updates within random clusters, resulting in a set of cluster-aggregated encryption values. Employing the proxy re-encryption key $repk$, the ciphertexts are transformed and forwarded to the server $\mathcal{S}_1$. In the third stage, $\mathcal{S}_1$ decrypts the ciphertexts using sk_1. Subsequently, malicious updates are identified and eliminated through norm filtering and principal component analysis (PCA). Finally, $\mathcal{S}_1$ obtains the aggregated compression of the model updates, sending the aggregation back to each client.

B. Threat Model

In our proposed scheme, the KGC is considered as fully trusted third party and hence would never collude with any entity. Shuffle server $\mathcal{S}_0$ and analysis server $\mathcal{S}_1$ are viewed as honest but curious, meaning that they honestly execute the established protocols but may be curious to deduce some sensitive information such as model updates uploaded by local clients. In addition, both of the servers are assumed to be non-colluding, otherwise they will obtain private information from the clients. In our system, we discuss two types of clients. One is the type of honest clients, which runs according to regulations and uploads real trained model updates. The other is the type of malicious clients, which uploads malicious model updates or uploads model updates trained with poisoned data to reduce global model accuracy.

C. Design Goals

- Privacy. In our scheme, we protect the privacy of the model updates uploaded by the clients. No third party or participant can infer the original information of the client.
- Robustness. In our scheme, we use robust aggregation algorithms to resist malicious client's attacks and select the correct model updates for aggregation.
- Efficiency. Our scheme should reduce the scale of model update transmission, thereby reducing the computational and transmission burden under ciphertext.
- Accuracy. Our scheme ensures the model achieves a high level of accuracy while concurrently safeguarding privacy and resisting poisoning attacks. The objective is to maintain the global model accuracy as closely as possible to the baseline model accuracy.

V. DESIGN OF OUR SCHEME

In this section, we will introduce a technical overview and provide a detailed description of our entire solution. The main notations are shown in TABLE II.

TABLE II
NOTATIONS

Notation	Description
D_i	Training dataset of the client u_i
D_0	The trusted root dataset collected by $\mathcal{S}_1$
$\mathcal{U}$	All clients participating in aggregation
$\mathcal{M}_k$	k-th cluster
$\mathbf{w}_i^r$	Local model of the client u_i in the r-th iteration
η	Local learning rate
$\mathbf{g}_i^r$	Model update of the client u_i in the r-th iteration
$\mathbf{g}_i^{r,l}$	Model update of the client u_i in the l-th local iteration in the r-th global iteration
$\mathbf{g}_{\mathcal{M}_k}^r$	Model update aggregation of the cluster $\mathcal{M}_k$ in the r-th iteration
$\mathbf{g}^r$	Model update aggregation in the r-th iteration
$\mathbf{g}_0^r$	Model update for trusted root dataset in the r-th iteration
θ_r	Global model in the r-th iteration
$\widetilde{\mathbf{g}}$	Compression of any model update or model update aggregation $\mathbf{g}$
b	Batch size
x	Number of clients
y	Number of clients in each cluster
t	Global iterations
e	Local iterations
c	Number of clusters
pk	Public key of clients $\mathcal{U}$
sk_1	Secret key of server $\mathcal{S}_1$
$repk$	Re-encrypt key of $\mathcal{S}_0$
$\mathcal{Q}$	The set of compressed model update aggregation values for all clusters
$[\![\widetilde{\mathbf{g}}]\!]_{key}$	Encryption of any compressed model update or model update aggregation $\tilde{\mathbf{g}}$ using key

A. Technical Overview

Inspired by recent studies, our approach is anchored in the utilization of CKKS fully homomorphic encryption [18], due to its support for floating-point operations and superior efficiency in privacy protection.

In order to optimize compatibility with homomorphic encryption, SuperFL adopts a compression method that supports linear operation. At initialization, the server only needs to broadcast a public sparse matrix $\mathbf{P}$ to all clients, and each client can project and compress its own model update $\mathbf{g}_i^r$ based on $\mathbf{P}$ in r round.

A pivotal innovation within SuperFL is the introduction of a novel proxy re-encryption scheme to address the challenges posed by robust aggregation. Recognizing the limitations of conventional privacy protection schemes, particularly the encrypted model update inaccessibility during encryption, SuperFL uses a random cluster aggregation mode with two non-colluding servers (shuffle server $\mathcal{S}_0$ and analysis server $\mathcal{S}_1$). Proxy re-encryption serves as a perfect bridge for communication between two servers. Server $\mathcal{S}_0$ holds the proxy key $repk$ while $\mathcal{S}_1$ holds its own secret key sk. $\mathcal{S}_1$ can only wait for $\mathcal{S}_0$ to complete cluster aggregation and then complete ciphertext conversion before decrypting. This approach safeguards each individual model update $\mathbf{g}_i^r$, the server $\mathcal{S}_1$ only perceives the average aggregation value $\mathbf{g}_{\mathcal{M}_k}^r$, adapting to our subsequent robustness method.

To further enhance robust aggregation, SuperFL introduces a trusted root as a stringent criterion. In light of the potential risks associated with random cluster aggregation leading to malicious clusters, SuperFL employs a dual filter mechanism. This mechanism combines norm and principal component analysis (PCA) defense to effectively filter out malicious updates. Norm filters model updates with larger update values, while PCA distinguishes differences between compressed update values, providing a comprehensive defense against poisoning attacks.

In summary, SuperFL is a comprehensive FL framework that balances efficiency and security through advanced homomorphic encryption, innovative compression techniques, and robust aggregation strategies, as shown in Figure 2.

B. Model Update Computation

1) Client local model update computation: In the r-th iterations of the global training, each client u_i, $i \in [1, x]$ get the latest global model $\mathbf{w}_i^r$. Then, each client u_i trains the model with its own local dataset D_i to obtain a model update $\mathbf{g}_i^r$. The equation for updating the local model is as Eq. 1:

$$\theta_i^{r,l} = \theta_i^{r,l-1} - \eta \nabla f(D_i, \theta_i^{r,l-1}) \tag{1}$$

where $f(D_i, \theta_i^{r,l-1})$ is the loss for client u_i in the l-th local iteration, ∇ is the derivation operation, η is local learning rate.

2) Server trusted root model update computation: The server's trusted root dataset's model update process is consistent with the client's local model update. Both of which input the global model from the previous round of update. The difference is the dataset. Above update called ModelUpdate is in Algorithm 1.

Algorithm 1 ModelUpdate

```
Input: local training datasets $D$, local learning rate $\eta$, batch
    size $b$, local iterations $e$, model $\mathbf{w}$.
Output: Client model update
1: $\theta_i^{r,0} \leftarrow \mathbf{w}$
2: for $l = 1, \cdots, e$
3:     Sample data from dataset $D$ with batch size b
4:     Calculate $\nabla f(D, \theta_i^{r,l-1})$
5:     $\theta_i^{r,l} \leftarrow \theta_i^{r,l-1}$ - $\eta \nabla f(D, \theta_i^{r,l-1})$
6: $\mathbf{g}_i^r \leftarrow \theta_i^{r,e}$ - $\mathbf{w}$
7: return $\mathbf{g}_i^r$
```

C. Sparse Embedding Sketch Compression

In federated learning, communication cost emerges as a critical issue, particularly with the substantial overhead associated with transmitting high-dimensional models. Model compression becomes imperative to alleviate this burden. In the context of privacy protection, the server needs to securely aggregate model parameters in the ciphertext state, which means that the compression results must maintain computability, which is a significant challenge. To achieve efficient model compression and transmission, we propose sparse embedding sketch compression to further optimize our federated learning aggregation scheme. This is a model compression method based on random projection sketch. Subsequent sections will detail the steps.

1) Initialize projection matrix: To ensure compression doesn't hinder federated learning convergence, we must guarantee that decompressed model updates approximate pre-compression updates. Using the coordinate embedding technique [20], we construct a public projection matrix meeting the criteria. Server $\mathcal{S}_1$ broadcasts this matrix to all clients, who then use it for model compression across multiple rounds.

We combine coordinate embedding [20] and sparse embedding [21] to directly compute the Compressed Sparse Row format (CSR) of the projection matrix. This enhances transmission and computational efficiency. CSR sparse matrix format supports various arithmetic operations like matrix multiplication and transposition. Algorithm 2 provides detailed steps.

2) Compression and decompression: Assuming the model update is a matrix $\mathbf{g} \in \mathbb{R}^{n\times s}$ and the public projection matrix is $\mathbf{P}$, the following compression and decompression operations $sec/desec$ can be obtained:

$$\begin{aligned} \mathrm{sec}(\mathbf{g}) &= \mathbf{P}^\top \mathbf{g}\ (\mathbb{R}^{n\times s} \to \mathbb{R}^{d\times s}) \\ \mathrm{desec}(\widetilde{\mathbf{g}}) &= \mathbf{P}\widetilde{\mathbf{g}}\ (\mathbb{R}^{d\times s} \to \mathbb{R}^{n\times s}) \end{aligned} \tag{1}$$

Since all clients share the same projection matrix, the compression operation of model update sec can satisfy the linear property and can be aggregated directly on the server.

$$\sum_{u_i\in\mathcal{U}} sec(\mathbf{g}_i^r) = sec(\sum_{u_i\in\mathcal{U}} \mathbf{g}_i^r) = sec(\mathbf{g}^r) \tag{2}$$

Since the model update after the decompression operation $desec$ is an unbiased estimate of the original model update, we can obtain the approximate value of the original high-dimensional model.

$$desec(\sum_{u_i\in\mathcal{U}} sec(\mathbf{g}_i^r)) \approx \mathbf{g}^r \tag{3}$$

Algorithm 2 GetProjectMatrix

```
Input: Federated learning model dimension $\ell = ns$, target
    compressed model dimension $\ell' = ds$
Output: Public projection sparse matrix $\mathbf{P}$ in CSR format
    (corresponding to the matrix of $\mathbb{R}^{n\times d}$)
1: Initialization: Server $\mathcal{S}_1$ generate CSR arrays $values = [\ ], col\_indices = [\ ], row\_ptr = [0]$ and random hash function $h : [n] \times [s] \to [\frac{d}{s}], \sigma : [n] \times [s] \to \{-1, 1\}$
2: for $x = 1 \to n$
3:     for $y = 1 \to s$
4:         Add $\frac{\sigma(x,y)}{\sqrt{s}}$ into the end of $values$
5:         Add $\frac{(y-1)d}{s} + h(x, y)$ into the end of $col\_indices$
6:     Add $row\_ptr[x-1] + s$ into the end of $row\_ptr$
7: $\mathbf{P} \leftarrow (values, col\_indices, row\_ptr)$
8: return $\mathbf{P}$
```

D. Secure Cluster Aggregation

To identify malicious clients, we choose the cluster aggregation method. For aggregation, homomorphic encryption facilitates addition between ciphertexts. Since the matrix projection based compression operation satisfies linearity, homomorphic addition can be directly performed after encryption. Once the shuffle server $\mathcal{S}_0$ receives all ciphertexts sent by clients, it randomly partitions the ciphertexts corresponding to different clients into clusters, performs an average aggregation on the ciphertexts within each cluster, and then transforms the aggregated ciphertexts of each cluster using the proxy key $repk$. For decryption, the analysis server $\mathcal{S}_1$ receives ciphertexts from $\mathcal{S}_0$ and decrypts the ciphertexts using its own secret key sk_1. $repk$ and sk_1 are generated during initialization. We detail the cluster aggregation method by SCAgg in Algorithm 3.

Algorithm 3 SCAgg

```
Input: local training datasets $D_i$, local learning rate $\eta$, batch
    size $b$, local iterations $e$, local model $\mathbf{w}_i^r$ for the $i$-th
    client's $r$-th global iteration, re-encrypt key $repk$ of server
    $\mathcal{S}_0$, secret key $sk_1$ of server $\mathcal{S}_1$
Output: The compressed model update aggregations of all
    clusters $\mathcal{Q}$.
1: Server $\mathcal{S}_0$ excutes:
2: Assign clients to clusters $\mathcal{U} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \cdots \cup \mathcal{M}_c$ (Ensure randomness)
3: $\mathcal{G}_1 \leftarrow \emptyset$
4: for each group $\mathcal{M}_k$ do
5:     $[\![\widetilde{\mathbf{g}}_{\mathcal{M}_k}^r]\!]_{pk} \leftarrow \frac{1}{|\mathcal{M}_k|} \cdot \sum_{u_i\in\mathcal{M}_k} [\![\widetilde{\mathbf{g}}_i^r]\!]_{pk}$
6:     $\mathcal{G}_1 \leftarrow \mathcal{G}_1 \cup [\![\widetilde{\mathbf{g}}_{\mathcal{M}_k}^r]\!]_{pk}$
7: $\mathcal{G}_2 \leftarrow \emptyset$
8: for each $[\![\widetilde{\mathbf{g}}_{\mathcal{M}_k}^r]\!]_{pk} \in GA$ do
9:     $[\![\widetilde{\mathbf{g}}_{\mathcal{M}_k}^r]\!]_{pk_1} \leftarrow$ Re-encrypt $[\![\widetilde{\mathbf{g}}_{\mathcal{M}_k}^r]\!]_{pk}$ using $repk$.
10:    $\mathcal{G}_2 \leftarrow \mathcal{G}_2 \cup [\![\widetilde{\mathbf{g}}_{\mathcal{M}_k}^r]\!]_{pk_1}$
11: $\mathcal{S}_0$ send $\mathcal{G}_2$ to $\mathcal{S}_1$
12: Server $\mathcal{S}_1$ excutes:
13: Receive $\mathcal{G}_2$ from server $S_0$
14: $\mathcal{Q} \leftarrow \emptyset$
15: for each $[\![\widetilde{\mathbf{g}}_{\mathcal{M}_k}^r]\!]_{pk_1} \in \mathcal{G}_2$ do
16:     $\widetilde{\mathbf{g}}_{\mathcal{M}_k}^r \leftarrow$ Decrypt $[\![\widetilde{\mathbf{g}}_{\mathcal{M}_k}^r]\!]_{pk_1}$ using $sk_1$
17:     $\mathcal{Q} \leftarrow \mathcal{Q} \cup \widetilde{\mathbf{g}}_{\mathcal{M}_k}^r$
18: return $\mathcal{Q}$
```

E. Robust Aggregation

Our goal is to devise a defense mechanism that can exclude model updates from clusters containing malicious clients during compressed state aggregation. Our defense relies on the insight that model updates from benign clients exhibit different characteristics compared to those from malicious clients, and random projection retains these fundamental characteristics. This allows us to use feature dimension reduction techniques like Principal Component Analysis (PCA) for effective feature extraction, offering a solution to the issue at hand. We guide trust using the self-collected root dataset of server $\mathcal{S}_1$ and consider model updates with eigenvalues similar to those derived from the trusted root dataset as benign.

After obtaining the compressed updated aggregations of c clusters, $\mathcal{S}_1$ calculates a standard model update based on its trusted root data set, embedding it into the same dimension using a shared sparse matrix. All compressed model updates undergo PCA analysis, and those with a distance below the dynamic threshold from the standard model updates are deemed safe. Updates with L_2 norm exceptions relative to the standard model update are excluded to resist scaling attacks by malicious clients.

The root dataset collection is akin to FLTrust [32], requiring only a small number of clean samples. Let **Min**$(\cdot)$ return the minimum value, **PCA**$(\cdot)$ represent component analysis, and **Sign**$(\cdot)$ denote the symbol function. Figure 3 illustrates the main concept of our robust algorithm. Algorithm 4 outlines the specifics of our robust defense algorithm.

Algorithm 4 RobustAgg

```
Input: Global model $\theta_r$, the compressed model update aggregations of all clusters $\mathcal{Q} = \{\widetilde{\mathbf{g}}^r_{\mathcal{M}_1}, \widetilde{\mathbf{g}}^r_{\mathcal{M}_2}, \cdots, \widetilde{\mathbf{g}}^r_{\mathcal{M}_c}\}$, root data set $\mathcal{D}_0$, local learning rate $\eta$, batch size $b$, local iterations $e$
Output: Robustness aggregation $\widetilde{\mathbf{g}}^r$
1:  Initialization: Generate $dis_1 = dis_2 = \emptyset, \mathcal{Q}'_1 = \mathcal{Q}'_2 = \emptyset$, the weight factor $\tau < 1, \beta > 1$
2:  Server $\mathcal{S}_0$ excutes:
3:  Step 1: Trusted update calculation
4:  $\theta^r_0 \leftarrow$ ModelUpdate$(D_0, \eta, b, e, \theta_r)$
5:  $\mathbf{g}^r_0 \leftarrow \theta^r_0 - \theta_r$
6:  $\widetilde{\mathbf{g}}^r_0 \leftarrow sec(\mathbf{g}^r_0)$
7:  Step 2: Norm-based Filtering
8:  for each group $\mathcal{M}_k$ do
9:      $ndis^r_k \leftarrow |\|\widetilde{\mathbf{g}}^r_{\mathcal{M}_i}\| - \|\widetilde{\mathbf{g}}^r_0\||$
10:     $dis_1 \leftarrow dis_1 \cup ndis^r_k$
11: $thr_1 \leftarrow$ Min$(dis_1) \cdot \beta$
12: Choose the compressed model updates that satisfies $ndis^r_k < thr_1$ as $\mathcal{Q}'_1$
13: Step 3: PCA-based Selection
14: $\{ps_i\}_{i\in[0,c]} \leftarrow$ PCA$(\{\widetilde{\mathbf{g}}^r_0\} \cup \mathcal{Q}, components = 1)$
15: for each group $\mathcal{M}_k$ do
16:     if Sign$(ps_0)$ == Sign$(ps_k)$
17:         $pdis^r_k \leftarrow \tau|ps_k - ps_0|$
18:     else
19:         $pdis^r_k \leftarrow |ps_k - ps_0|$
20:     $dis_2 \leftarrow dis_2 \cup pdis^r_k$
21: $thr_2 \leftarrow$ Min$(dis_2) \cdot \beta$
22: Choose the compressed model updates that satisfies $pdis^r_k < thr_2$ as $\mathcal{Q}'_2$
23: Step 4: Aggregation
24: Get trusted set: $\mathcal{Q}' = \mathcal{Q}'_1 \cap \mathcal{Q}'_2$
25: If $\mathcal{Q}' = \emptyset$, then $\widetilde{\mathbf{g}}^r$ is a full 0 tensor with the same shape as the original compression model update, otherwise calculate the aggregate value $\widetilde{\mathbf{g}}^r \leftarrow \frac{1}{|\mathcal{Q}'|}\sum \mathcal{Q}'$.
26: return $\widetilde{\mathbf{g}}^r$
```

F. Putting It All Together

The overall process algorithm of our scheme, termed CPBFL, involves initializing parameters, keys, global model, and projection matrix before training. During training, each client locally trains with its dataset, compresses, encrypts, and

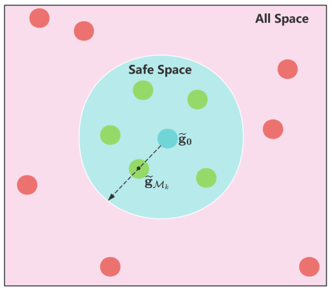

All Space
Safe Space
$\widetilde{\mathbf{g}}_0$
$\widetilde{\mathbf{g}}_{\mathcal{M}_k}$

Fig. 3. Illustration of our robust algorithm: green, red and blue points denote benign updates, malicious updates and the standard update by server $\mathcal{S}_1$ from trusted root data D_0 respectively. In our algorithm, we gauge update security using feature distance from the standard. Feature distance is calculated from update norm and PCA. The threshold radius is determined by the minimum feature distance multiplied by expansion factor β. Updates within this radius are considered benign.

uploads results to shuffle server $\mathcal{S}_0$. $\mathcal{S}_0$ performs clustering aggregation and ciphertext conversion. Analysis server $\mathcal{S}_1$ decrypts ciphertexts with key sk_1 and performs robust aggregation to obtain the final global model update. This update is then sent back to clients.

Algorithm 5 CPBFL

```
Input: Global model $\theta_r$, $n$ clients $\{u_1, u_2, ..., u_n\}$ with local training datasets $D = \{D_1, D_2, ..., D_n\}$, trusted root dataset $D_0$, local learing rate $\eta$, global iterations $t$, local iterations $e$, batch size $b$, the compression ratio $\rho$.
1:  Initialization: KGC distributes public key $pk$ to clients, $repk$ to server $\mathcal{S}_0$ and $(pk_1, sk_1)$ to server $\mathcal{S}_1$. Server $\mathcal{S}_1$ initialization global model $\theta_0$. $\mathcal{S}_1$ generate a projection matrix $\mathbf{P} \leftarrow$ GetProjectMatrix$(\ell, \rho\ell)$ according to the dimension $\ell$ of the global model and the compression ratio $\rho$ and broadcast $\mathbf{P}$ to all clients.
2:  for $r = 1 \to t$ do
3:      Client $u_i$ excutes :
4:      if $r = 1$
5:          $\mathbf{w}^r_i \leftarrow \theta_0$
6:      else
7:          $\mathbf{w}^r_i \leftarrow \mathbf{w}^{r-1}_i + desec(\widetilde{\mathbf{g}}^{r-1}_i)$
8:      $\mathbf{g}^r_i \leftarrow$ ModelUpdate$(\mathbf{w}^r_i, D_i, \eta, e, b)$
9:      $\widetilde{\mathbf{g}}^r_i \leftarrow sec(\mathbf{g}^r_i)$
10:     $u_i$ encrypt compressed model update $[\![\widetilde{\mathbf{g}}^{\mathbf{r}}_{\mathbf{i}}]\!]_{pk}$ using $pk$
11:     Client $u_i$ send $[\![\widetilde{\mathbf{g}}^{\mathbf{r}}_{\mathbf{i}}]\!]_{pk}$ to Server $\mathcal{S}_0$
12:     Server $\mathcal{S}_0, \mathcal{S}_1$ excutes :
13:     $\mathcal{Q} \leftarrow$ SCAgg$([\![\widetilde{\mathbf{g}}^{\mathbf{r}}_1]\!]_{pk}, [\![\widetilde{\mathbf{g}}^{\mathbf{r}}_2]\!]_{pk}, \cdots, [\![\widetilde{\mathbf{g}}^{\mathbf{r}}_x]\!]_{pk})$
14:     $\widetilde{\mathbf{g}}_r \leftarrow$ RobustAgg$(\theta_{r-1}, \mathcal{Q}, D_0, \eta, b, e)$
15:     $\theta_r \leftarrow \theta_{r-1} + desec(\widetilde{\mathbf{g}}^r)$
16:     $\mathcal{S}_1$ send $\widetilde{\mathbf{g}}_r$ to all clients
```

VI. Security Analysis and Discussion

A. Security Analysis

In this section, we present a security proof for the scheme. Reflecting on our threat model, it encompasses two servers $\mathcal{S}_0$ and $\mathcal{S}_1$, along with clients $\mathcal{U}$ in federated learning. Servers

$\mathcal{S}_0$ and $\mathcal{S}_1$ are semi-honest, permitting active malicious adversaries in all clients. The underlying cryptographic building blocks are instantiated with the security parameter λ. Let $X_{\mathcal{P}}$ represent the inputs of any subset of all participants $\mathcal{P} \subseteq \mathcal{U} \cup \{\mathcal{S}_0, \mathcal{S}_1\}$. Furthermore, servers $\mathcal{S}_0$ and $\mathcal{S}_1$ do not collude. In the ensuing proof, the view of each party comprises its internal state (including input and randomness) and all messages received from other parties. For any subset $\mathcal{P} \subseteq \mathcal{U} \cup \{\mathcal{S}_0, \mathcal{S}_1\}$, considering any adversary denoted by $\mathcal{A}_{\mathcal{P}}$ let $\mathsf{REAL}_{\mathcal{P}}^{\mathcal{U},\lambda}(\mathcal{A}_{\mathcal{P}}, X_{\mathcal{U}\cup\{\mathcal{S}_0,\mathcal{S}_1\}\setminus\mathcal{P}})$ be a random variable representing the joint views of participants in $\mathcal{P}$ during the actual execution of our scheme in the real world, and $\mathsf{SIM}_{\mathcal{P}}^{\mathcal{U},\lambda}(\mathcal{A}_{\mathcal{P}}, X_{\mathcal{P}})$ be another combined view of participants in $\mathcal{P}$ simulating the scheme in the ideal world, with the inputs of honest participants randomly and uniformly selected and denoted as $X_{\mathcal{P}}$. To satisfy security requirements, $\mathsf{SIM}_{\mathcal{P}}^{\mathcal{U},\lambda}(\mathcal{A}_{\mathcal{P}}, X_{\mathcal{P}})$ and $\mathsf{REAL}_{\mathcal{P}}^{\mathcal{U},\lambda}(\mathcal{A}_{\mathcal{P}}, X_{\mathcal{U}\cup\{\mathcal{S}_0,\mathcal{S}_1\}\setminus\mathcal{P}})$ must be indistinguishable.

Theorem 1 (Privacy against actively adversarial users, with semi-honest server)**.** *Considering all $\mathcal{U}, \lambda$ and $\mathcal{P}$ where $\mathcal{P} \subseteq \mathcal{U} \cup \{\mathcal{S}_0, \mathcal{S}_1\}$, for every non-uniform probabilistic polynomial-time (PPT) adversary $\mathcal{A}_{\mathcal{P}}$ (malicious adversary $\mathcal{A}_{\mathcal{U}'}$ or semi-honest adversary $\mathcal{A}_{\mathcal{S}_0}, \mathcal{A}_{\mathcal{S}_1}$), where $\mathcal{U}' \subset \mathcal{U}$ and $|\mathcal{U}'| < x-1$, there exists a probabilistic polynomial-time (PPT) simulator* SIM *such that*

$$\mathsf{REAL}_{\mathcal{P}}^{\mathcal{U},\lambda}(\mathcal{A}_{\mathcal{P}}, X_{\mathcal{U}\cup\{\mathcal{S}_0,\mathcal{S}_1\}\setminus\mathcal{P}}) \approx_c \mathsf{SIM}_{\mathcal{P}}^{\mathcal{U},\lambda}(\mathcal{A}_{\mathcal{P}}, X_{\mathcal{P}})$$

where $\approx_c$ denotes that the outputs are computationally indistinguishable.

Proof. In this case, we prove the security of our scheme without collusion between the clients and the server. We will use standard hybrid argument to prove our theory. We use the simulator $\mathsf{SIM}_{\mathcal{P}}^{\mathcal{U},\lambda}$ to simulate the behavior of the adversary $\mathcal{A}_{\mathcal{P}}$. Starting from the random variable $\mathsf{REAL}_{\mathcal{P}}^{\mathcal{U},\lambda}$, we define our simulator $\mathsf{SIM}_{\mathcal{P}}^{\mathcal{U},\lambda}$ through a series of (polynomially many) subsequent modifications so that any two subsequent random variables are computationally indistinguishable.

Hyb_0 We initialize a series of random variables that are indistinguishable from $\mathsf{REAL}_{\mathcal{P}}^{\mathcal{U},\lambda}$ in the real execution of our scheme.

Hyb_1 In the process of uploading the model updates, we use the simulator $\mathsf{SIM}_{u_i}^{\mathcal{U},\lambda}$ ($u_i \in \mathcal{U} \setminus \mathcal{P}$) to simulate the above operation. Each simulator of user u_i will generate a random vector $\mathbf{X_i}$ instead of $\mathbf{g}_i^r$ and apply the compression and encryption algorithm to abtain the ciphertext $[\![\widetilde{\mathbf{X}}_\mathbf{i}]\!]_{pk}$ using a uniformly random public key pk. The views of $\mathcal{A}_{\mathcal{S}_0}$ include all the ciphertexts. This hybrid is indistinguishable from the previous one, since the encryption scheme CKKS satisfies IND-CPA security and only the contents of the ciphertexts have changed.

Hyb_2: In this hybrid, we alter the input of the SCAgg algorithm with encryptions of uniformly random vectors (of appropriate length) $[\![\mathbf{R_i}]\!]_{pk}$ instead of $[\![\widetilde{\mathbf{g}}_\mathbf{i}^\mathbf{r}]\!]_{pk}$ ($i \in [1, x]$). $\mathsf{SIM}_{\mathcal{S}_0}^{\mathcal{U},\lambda}$ randomly shuffles and aggregates these ciphertexts within clusters, yielding $[\![\mathbf{R}_{\mathcal{M}_k}]\!]_{pk}(k \in [1, c])$. Subsequently, it calculates $[\![\mathbf{R}_{\mathcal{M}_k}]\!]_{pk_1}$ using the re-encryption algorithm. Given that both proxy re-encryption and CKKS homomorphic encryption adhere to the IND-CPA security property, the ideal views of $\mathcal{A}_{\mathcal{S}_0}$ are computationally indistinguishable from $\mathsf{REAL}_{\mathcal{S}_0}^{\mathcal{U},\lambda}$.

Hyb_3 In this hybrid, $\mathsf{SIM}_{\mathcal{S}_1}^{\mathcal{U},\lambda}$ simulates the preparation for robustness algorithm. For server $\mathcal{S}_1$, all cluster aggregations $\mathcal{Q} = \{\widetilde{g}_{\mathcal{M}_k}^r\}$ are exposed, where $k \in [1, c]$. As $\mathcal{S}_0$ and $\mathcal{S}_1$ are non-colluding servers, since the inputs $\{[\![\widetilde{\mathbf{g}}_\mathbf{i}^\mathbf{r}]\!]_{pk}\}$ ($i \in [1, x]$) are encrypted and the intermediate results are protected by noise in CKKS, cluster aggregations cannot leak any user's data privacy without knowing any information about the input and intermediate calculations. We can modify the inputs to be a set of uniformly random vectors instead of $\{\widetilde{g}_{\mathcal{M}_k}^r\}$, the real view of the interactive scheme and ideal view by $\mathcal{A}_{\mathcal{S}_1}$ are simulatable and computationally distinguishable.

Hyb_4 In this hybrid, after $\mathcal{S}_1$ receives all cluster aggregations from $\mathcal{S}0$, $\mathsf{SIM}_{\mathcal{S}_1}^{\mathcal{U},\lambda}$ simulates the robustness aggregation algorithm. The view of $\mathcal{A}_{\mathcal{S}_1}$ encompasses all intermediate results of the robustness algorithm, specifically referring to the norms and PCA's dimension reduction values of all cluster aggregations. These intermediate values, derived from aggregated values of each cluster, do not leak individual model update information from any users. The final aggregated model update $sum = \sum_{\mathcal{M}_k \in \mathcal{Q}'} \widetilde{g}_{\mathcal{M}_k}^r$ is calculated, where $\mathcal{Q}'$ is the set of aggregations of compressed model updates within clusters after $\mathcal{S}_1$ completes the robust algorithm. The views of $\mathcal{A}_{\mathcal{S}0}$ and $\mathcal{A}_{\mathcal{U}}$ include the final aggregation. The final aggregate value in real views remains indistinguishable from that of an equivalent set of random vectors $sum' = \sum_{\mathcal{M}_k \in \mathcal{Q}'} \mathcal{R}_k$ in ideal views of $\mathcal{A}_{\mathcal{S}_1}$ and $\mathcal{A}_{\mathcal{U}}$.

Therefore, we can define a PPT simulator, denoted as $\mathsf{SIM}_{\mathcal{P}}^{\mathcal{U},\lambda}$, which samples from the distribution in the last hybrid. The preceding argument conclusively demonstrates that the output of the simulator is computationally indistinguishable from the output of $\mathsf{REAL}_{\mathcal{P}}^{\mathcal{U},\lambda}$, thereby completing the proof. □

Theorem 2 (Security against a server colluding with clients)**.** *Considering all $\mathcal{U}, \lambda$ and $\mathcal{P}$ where $\mathcal{P} \subseteq \mathcal{U} \cup \{\mathcal{S}_0, \mathcal{S}_1\}$, for every non-uniform probabilistic polynomial-time (PPT) adversary $\mathcal{A}_{\mathcal{P}}$ controlling a set of malicious clients and having access to the view of at most one (semi-honest) server, there exists a probabilistic polynomial-time (PPT) simulator* SIM *such that*

$$\mathsf{REAL}_{\mathcal{P}}^{\mathcal{U},\lambda}(\mathcal{A}_{\mathcal{P}}, X_{\mathcal{U}\cup\{\mathcal{S}_0,\mathcal{S}_1\}\setminus\mathcal{P}}) \approx_c \mathsf{SIM}_{\mathcal{P}}^{\mathcal{U},\lambda}(\mathcal{A}_{\mathcal{P}}, X_{\mathcal{P}})$$

where $\approx_c$ denotes that the outputs are computationally indistinguishable.

Proof. We assumed in the threat model that there is no collude between servers, but we do not rule out the possibility of clients colluding with one of the servers. We will analyze the above situation. We assume κ malicious clients $\mathcal{U}_M = \{u_{1'}, u_{2'} \cdots, u_{\kappa'}\}(|\mathcal{U}_M| = \kappa)$, we allow these malicious clients to conspire to achieve the attack target.

Case 1: $\mathcal{U}_M$ collude with shuffle server $\mathcal{S}_0$

We assume that these clients $\mathcal{U}_M$ and $\mathcal{S}_0$ are simultaneously corrupted by the adversary $\mathcal{A}_{(\mathcal{U}_M, \mathcal{S}_0)}$. The views of

$\mathcal{A}_{(\mathcal{U}_M,\mathcal{S}_0)}$ contains the ciphertext of local model updates for all clients, the ciphertexts of model updates for each cluster collected by random clusters, as well as the total public key and proxy re encryption key. However, because there are no secret keys in the views, the adversary $\mathcal{A}_{(\mathcal{U}_M,\mathcal{S}_0)}$ is unable to decrypt and obtain model updates from other honest clients. Due to the fact that homomorphic encryption CKKS and proxy re encryption algorithms satisfy the security properties of IND-CPA, the joint view of adversary $\mathcal{A}_{(\mathcal{U}_M,\mathcal{S}_0)}$ in ideal and real worlds are computationally indistinguishable, so $\mathsf{REAL}^{\mathcal{U},\lambda}_{(\mathcal{U}_M,\mathcal{S}_0)}(\mathcal{A}_{(\mathcal{U}_M,\mathcal{S}_0)}, X_{\mathcal{U}\cup\{\mathcal{S}_0,\mathcal{S}_1\}\setminus\mathcal{P}}) \approx_c \mathsf{SIM}^{\mathcal{U},\lambda}_{\mathcal{P}}(\mathcal{A}_{(\mathcal{U}_M,\mathcal{S}_0)}, X_{(\mathcal{U}_M,\mathcal{S}_0)})$.

Case 2: $\mathcal{U}_M$ collude with analysis server $\mathcal{S}_1$

We assume that these clients $\mathcal{U}_M$ and $\mathcal{S}_1$ are simultaneously corrupted by the adversary $\mathcal{A}_{(\mathcal{U}_M,\mathcal{S}_1)}$. The views of $\mathcal{A}_{(\mathcal{U}_M,\mathcal{S}_1)}$ contains all cluster aggregations $\mathcal{Q} = \{\widetilde{g}^r_{\mathcal{M}_k}\}$ and model updates of clients in the set $\mathcal{U}_M$. Due to the random shuffling and combination of $\mathcal{S}_0$, the plaintexts still have a random distribution in the joint view of $\mathcal{A}_{(\mathcal{U}_M,\mathcal{S}_1)}$. So the behavior of the adversary $\mathcal{A}_{(\mathcal{U}_M,\mathcal{S}_1)}$ remains indistinguishable in both the real and ideal worlds. □

Theorem 3. *Shuffle server $\mathcal{S}_0$, analysis server $\mathcal{S}_1$ and malicious clients can get nothing about the sensitive information of honest clients theoretically.*

Proof. For the server $\mathcal{S}_0$, throughout the entire process, only the ciphertexts of all model updates $[\![\widetilde{\mathbf{g}}^{\mathbf{r}}_{\mathbf{i}}]\!]_{pk}$ and $[\![\widetilde{\mathbf{g}}^{\mathbf{r}}_{\mathbf{i}}]\!]_{pk_1}$ are visible, and the exact value of the model update $\mathbf{g}^r_i$ for any client u_i cannot be obtained, where $i \in [1,x]$. For server $\mathcal{S}_1$, all cluster aggregations $\mathcal{Q} = \{\widetilde{g}^r_{\mathcal{M}_k}\}$ are exposed, where $k \in [1,c]$. As $\mathcal{S}_0$ and $\mathcal{S}_1$ are non-colluding servers, $\mathcal{S}_1$ is also unable to acquire the exact value of any client's model update. For κ malicious clients $\{u_{1'}, u_{2'} \cdots, u_{\kappa'}\}(|\{1', 2' \cdots, \kappa'\}| = \kappa)$, we allow these malicious clients to conspire to achieve the attack target, they can get the final cluster aggregation $sum = \sum_{\mathcal{M}_k \in \mathcal{Q}'} \widetilde{g}^r_{\mathcal{M}_k}$, where $\mathcal{Q}'$ is the set of aggregations of the compressed model updates within clusters after $\mathcal{S}_1$ complete the robust algorithm. Assume that $\mathcal{Q}'$ contains the compression model update of users $u' = \bigcup_{k\in c'} \mathcal{M}_k = \{u_{1''}, \cdots, u_{\kappa''}\} \cup \mathcal{U}_p$, where c' denotes the clusters selected by the robust algorithm, $\{u_{1''}, \cdots, u_{\kappa''}\}$ and $\mathcal{U}_p$ denote the set of malicious clients and honest clients selected respectively. For malicious clients, the privacy data of any client cannot be obtained. The analysis is as follows: (1) Random clustering is performed on the $\mathcal{S}_0$ side, and robust filtering is performed on the $\mathcal{S}_1$ side. The malicious client cannot obtain the user list u' of the final aggregation, and cannot perform effective collusion attacks to obtain the model update of the honest client, when $\{u_{1''}, \cdots, u_{\kappa''}\} \neq \{u_{1'}, \cdots, u'_{\kappa}\}$. (2) When $|\mathcal{U}_p| > 1$, malicious clients can theoretically only obtain the sum of model updates of multiple honest clients rather than model updates of any single client $u_i \in \mathcal{U}_p$. For the case of $|\mathcal{U}_p| = 1$, it is unrealistic when the number of clients reaches a certain scale. □

B. Discussion

Our method involves cluster aggregation, where one or more malicious model updates in the cluster are considered malicious. Also, we need a trusted root as a guide because malicious updates will exceed half of the number of clusters, and we also require one less cluster to include all benign model updates. In addition, if the proportion of malicious updates is theoretically less than 50% without clustering aggregation, norm and PCA dimensionality reduction can be directly used for malicious update filtering, without the need for trusted roots as guidance.

Certainly, the cluster size is linked to the resistible number of malicious clients. The relationship between the proportion of malicious clients with maximum tolerance (Att denoted) and the cluster size (y denoted) is expressed as $Att = \frac{\lfloor \frac{x}{y} \rfloor - 1}{x}$, where x is the number of clients.

Moreover, multiple rounds of random clusters can introduce additional privacy risks, as the server can clearly see multiple averages of clients. We introduce a variable R, which denotes the number of rounds of random cluster. In our experimental setting, we conduct random cluster in every round, thus $R = t$. At least $R \geq y = \lfloor \frac{x}{c} \rfloor$ is required for the server to recognize all client updates. Consider a scenario with 4 clients $\{u_1, u_2, u_3, u_4\}$, the cluster size is 2. We assume that the global Federated Learning (FL) model is nearing convergence, and the locally trained models exhibit minimal changes across different rounds, so we can get $\mathbf{w}_i^{t_0} \approx \mathbf{w}_i^{t_0+1}, i \in [1,4]$. In round t_0, the server obtains the model aggregation $z_0^{t_0} = \mathbf{w}_1^{t_0} + \mathbf{w}_2^{t_0}$ and $z_1^{t_0} = \mathbf{w}_3^{t_0} + \mathbf{w}_4^{t_0}$, while in round t_0+1, it obtains $z_0^{t_0+1} = \mathbf{w}_1^{t_0+1} + \mathbf{w}_3^{t_0+1}$ and $z_1^{t_0+1} = \mathbf{w}_2^{t_0+1} + \mathbf{w}_4^{t_0+1}$. The server may attempt to consolidate all client models by combining these aggregation equations. However, in practice, overlapping clusters may occur for different rounds. So $R \geq y = \lfloor \frac{x}{c} \rfloor$ also cannot serve as an assurance that the server can compromise the models of clients.

VII. Experimnet

We evaluate the effectiveness of our scheme in compressing and resisting poisoning attacks and show our experimental results.

A. Experimental Setup

a) Datasets: We use multiple datasets from different domains in our experiments, including two image classification datasets.

- MNIST: 10-class handwritten digit image classification dataset consisting of 60000 training images and 10000 testing images.
- CIFAR-10: The CIFAR-10 dataset has a total of 60000 samples, each of which is a 32 × 32 pixel RGB image (color image). These 60000 samples were divided into 50000 training samples and 10000 testing samples.

b) Models: We train various global models on the above two datasets to show the generality of. we train a LeNet-5 and a CNN as global model. The parameters of the model are shown in TABLE III.

TABLE III
MODEL SUMMARY

Network	Layer					
	Conv1	Conv2	Conv3	FC1	FC2	FC3
LeNet-5	156	2416	-	48120	10164	850
CNN	896	18496	73856	524544	21588	850

c) FL Settings: We set the number of clients as $N = 20$, we train the global models for 40 communication round. In each round, each client locally trains $R = 5$ rounds via mini-batch SGD with a batch size of $b = 32$, the number of clusters is 5. The training datasets are independent identically distributed (iid) and Dirichlet distributed (non-iid) with parameter alpha = 0.5. Additionally, it should be noted that the model updates aggregation method in our experiment is average aggregation rather than weighted average aggregation.

d) Adversary Parameters: We set the proportion of malicious clients to 0.05 to 0.2.

e) Evaluated Poisoning Attacks: We implement both untargeted poisoning attacks and targeted poisoning attacks. The above includes:

- Label Flipping Attack: Label Flipping (LF) attack simply flips the label of each training sample. we flip its initial Label l to l+1 mod L, where L is the total number of classes and $l = 0, 1, ..., L-1$. It is a classic targeted attack.
- Gaussian Attack: Malicious clients add Gaussian noise to the updated local model parameters. For simplicity sake, the mean and variance of the noise are 0 and 0.5 in our experiment.
- Scaling Attack: Scaling attack is a untargeted local model poisoning attack. Specifically, we consider the attacker uploading the inverse value of the benign model update and scaling it. Usually, the scaling factor γ satisfies $\gamma \gg 1$. In the experiment, we set it to 10.

f) Experimental Configuration: We conduct federated learning simulation experiments on a laptop, which is configured with AMD Ryzen 9 5900HX and RTX3080 independent graphics. Also we use OpenFHE library to deploy CKKS and Proxy Re-Encryption [41]. Finally, we use Intel Math Kernel Library (MKL) to support and accelerate matrix operations in the CSR format for sparse matrixs.

B. Experimental Results

We compare the performance of Top-K [12], FetchSGD [37], Guassi RP (Guassi Random Projection) [42] and SuperFL when the communication round reaches the preset final round. We use the LeNet-5 model and CNN model to train 40 rounds on the MNIST dataset and CIFAR-10 dataset respectively. Also, the data distribution is the Dirichlet distribution. Besides, Our compression ratio is calculated by dividing the number of model parameters by the number of compressed model parameters. We observe the accuracy of the model test and give the performance comparison between SuperFL and other compression methods in Figure 4 and in Figure 5.

We verify the model accuracy of the four compression methods under different compression ratios. The compression ratio of 1 is our Fedavg accuracy. In Figure 4(a) and in Figure 5(a), it can be seen that SuperFL has similar model accuracy with Gaussin RP under a certain compression ratio. In additon, our SuperFL has higher model accuracy than FetchSGD under the same compression ratio. In Figure 4(b) and in Figure 5(b), we fix different compression ratios for the four compression methods to verify the accuracy of the model.

In TABLE IV, we set target accuracy 95% for MNIST + LeNet-5 and target accuracy 64% for CIFAR-10 + CNN learning task respectively. It can be seen that our scheme achieves the target accuracy in fewer rounds at same compression ratio than the Guassin RP. Figure 6(a) shows the execution time of model compression using different projection compression methods. It can be seen that our model compression is more efficient than other model compression algorithms, due to the use of sparse matrix multiplication optimization techniques.

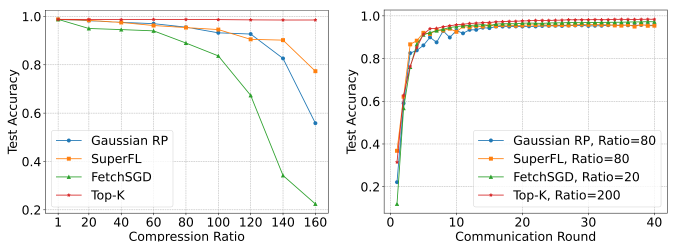


Fig. 4. Test accuracy on MNIST: (a) Test accuracy on MNIST in different upload compression ratio, (b) Test accuracy on MNIST in different rounds

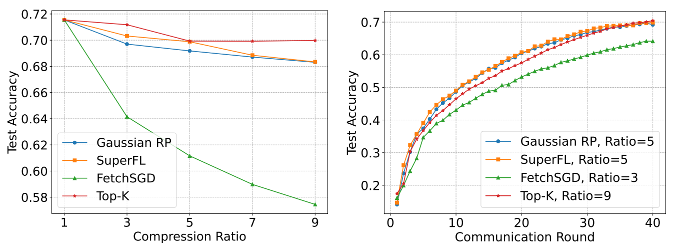
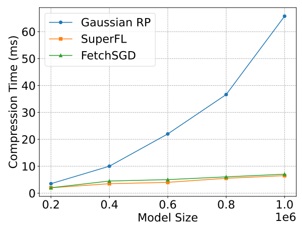


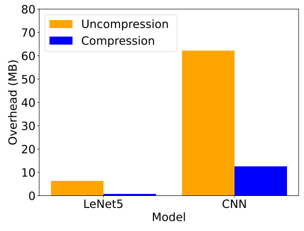
Fig. 5. Test accuracy on CIFAR-10: (a) Test accuracy on CIFAR-10 in different upload compression ratio, (b) Test accuracy on CIFAR-10 in different rounds.

TABLE IV
UPLOAD COMPRESSION RATIO OF EACH METHOD ON MNIST DATASET WITH THE TARGET TEST ACCURACY 95%, UPLOAD COMPRESSION RATIO OF EACH METHOD ON CIFAR-10 DATASET WITH THE TARGET TEST ACCURACY 64%

Dataset	Method	Round	Compression ratio
MNIST	Gaussian RP	16	80
	SuperFL	11	80
	FetchSGD	10	20
	Top-K	7	160
CIFAR-10	Gaussian RP	26	5
	SuperFL	24	5
	FetchSGD	40	3
	Top-K	29	9



(a) Time performance



(b) Overhead performance

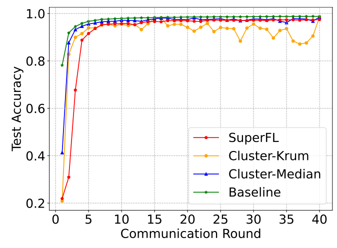
Fig. 6. (a) Comparison of execution time of compression algorithms for models of different sizes (Fixed compression ratio of 100), (b) The communication overhead of ciphertexts for uncompressed models and compressed models in a single round of upload.

Before embarking on our robust defense experiments, we initially assessed the model test error rate of our defense algorithm under attack-free conditions, as shown in TABLE V. The compression ratios for training models on the MNIST and CIFAR-10 datasets are set to 40 and 5, respectively. The findings show that our defense algorithm exhibits comparable model performance to FedAvg in the absence of attacks. Then

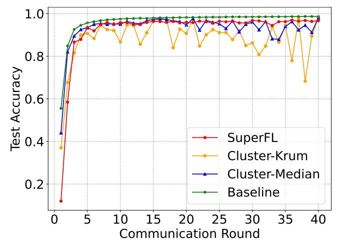
TABLE V
TEST ERROR RATE OF SUPERFL AND OTHER DEFENSE ALGORITHMS UNDER THE ABSENCE OF ATTACK ON DIFFERENT DATASETS

Dataset	**Scheme**	**Data Distribution**	
		IID	Non-IID
MNIST	FedAvg	0.01	0.01
	SuperFL	0.02	0.02
	Krum	0.01	0.01
	Median	0.01	0.01
CIFAR-10	FedAvg	0.27	0.29
	SuperFL	0.28	0.34
	Krum	0.28	0.36
	Median	0.27	0.30

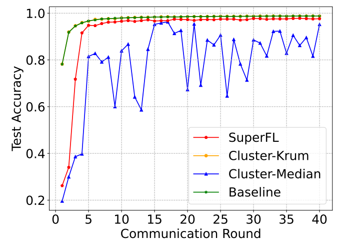
we evaluate the robustness of our scheme against poisoning attacks. Figure 7 shows the comparison of the training results of our robustness scheme under 40 rounds with other schemes under the MNIST data set. TABLE VI shows the comparison of model accuracy with other schemes under different number of malicious clients. The results indicate that our approach consistently preserves accuracy across various types of poisoning attacks, even when subjected to different numbers of malicious clients. This performance surpasses that of other algorithms. Note that our robustness test is performed at a compression ratio of 40, which also indicates that our robustness algorithm can be well adapted to our model compression algorithm. Figure 8 and TABLE VII show the robustness of superFL on CIFAR-10, where we set the compression ratio of our scheme to 5. It can be seen that our scheme still has a comprehensive defense effect in the case of IID and Non-IID. In particular, when the proportion of malicious clients exceeds or equals 15%, a scenario where over half of the groups potentially harbor malicious nodes, conventional defense mechanisms such as Krum and Median falter. In contrast, our algorithm continues to exhibit robust performance under such circumstances.



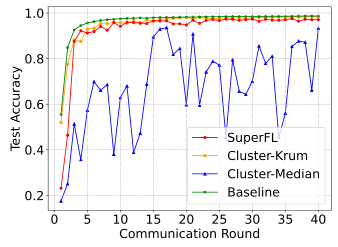
(a) Label Flipping attack, IID, Att = 20%



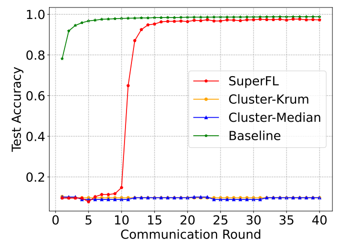
(b) Label Flipping attack, Non-IID, Att = 20%



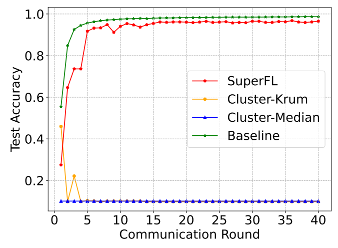
(c) Gauss attack, IID, Att = 20%



(d) Gauss attack, Non-IID, Att = 20%



(e) Scale attack, IID, Att = 20%



(f) Scale attack, Non-IID, Att = 20%

Fig. 7. Test accuracy on MNIST, where Att denotes percentage of malicious clients.

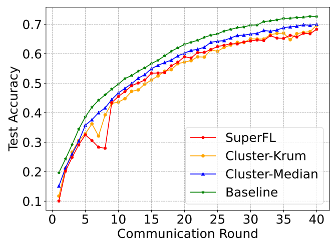
TABLE VI
TEST ERROR RATE OF SUPERFL AND OTHER DEFENSE ALGORITHMS UNDER DIFFERENT NUMBER OF MALICIOUS ATTACKS ON MNIST

Attacks	**Scheme**	**Proportion of malicious clients**			
		5%	10%	15%	20%
LF attack	SuperFL	0.02/0.02	0.02/0.03	0.02/0.03	0.03/0.03
	Krum	0.01/0.02	0.01/0.02	0.05/0.05	0.09/0.13
	Median	0.01/0.01	0.01/0.02	0.02/0.02	0.03/0.07
Gauss attack	SuperFL	0.02/0.03	0.02/0.03	0.02/0.03	0.03/0.03
	Krum	0.01/0.02	0.01/0.02	0.01/0.02	0.01/0.02
	Median	0.01/0.01	0.02/0.02	0.06/0.11	0.12/0.24
Scaling attack	SuperFL	0.02/0.03	0.02/0.03	0.02/0.03	0.03/0.03
	Krum	0.01/0.02	0.01/0.02	0.01/0.02	0.90/0.90
	Median	0.01/0.02	0.01/0.02	0.90/0.90	0.91/0.90

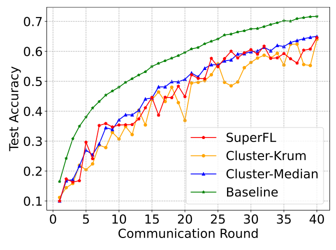
TABLE VII
TEST ERROR RATE OF SUPERFL AND OTHER DEFENSE ALGORITHMS UNDER DIFFERENT NUMBER OF MALICIOUS ATTACKS ON CIFAR-10

Attacks	**Scheme**	**Proportion of malicious clients**			
		5%	10%	15%	20%
LF attack	SuperFL	0.29/0.34	0.29/0.34	0.31/0.34	0.33/0.35
	Krum	0.29/0.36	0.30/0.39	0.32/0.41	0.32/0.41
	Median	0.28/0.32	0.28/0.32	0.29/0.33	0.30/0.35
Gauss attack	SuperFL	0.30/0.34	0.30/0.37	0.29/0.37	0.29/0.37
	Krum	0.28/0.37	0.28/0.37	0.28/0.38	0.28/0.39
	Median	0.27/0.31	0.27/0.32	0.68/0.78	0.83/0.90
Scaling attack	SuperFL	0.28/0.34	0.28/0.35	0.29/0.36	0.30/0.36
	Krum	0.28/0.36	0.28/0.36	0.90/0.90	0.90/0.90
	Median	0.29/0.33	0.29/0.40	0.90/0.90	0.90/0.90

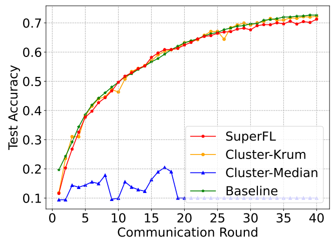
we compare the communication overhead of ciphertexts for uncompressed models and compressed models in a single



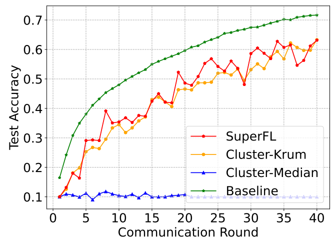
(a) LF attack, IID, Att = 20%



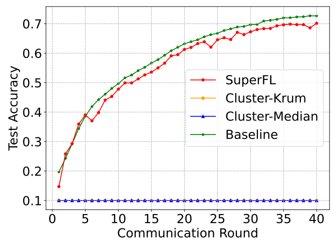
(b) LF attack, Non-IID, Att = 20%



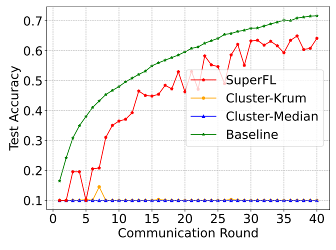
(c) Gauss attack, IID, Att = 20%



(d) Gauss attack, Non-IID, Att = 20%



(e) Scaling attack, IID, Att = 20%



(f) Scaling attack, Non-IID, Att = 20%

Fig. 8. Test accuracy on CIFAR-10, where Att denotes percentage of malicious clients.

round of upload as shown in Figure 6(b). It is evident that compression can reduce transmission overhead.

TABLE VIII
TEST ERROR RATE OF SUPERFL ALGORITHMS UNDER DIFFERENT NUMBER OF MALICIOUS LF ATTACKS AND DIFFERENT COMPRESSION RATIO ON MNIST

Compression Ratio	**Proportion of malicious clients**			
	5%	10%	15%	20%
40	2.66%	2.74%	3.23%	3.56%
80	4.79%	4.88%	5.55%	5.62%
120	14.37%	16.30%	18.30%	8.08%
160	15.49%	16.83%	23.73%	25.25%

TABLE VIII presents the testing error rate of our scheme under various compression ratios in the presence of label-flipping attacks. As evidenced by TABLE VIII, it is clear that the testing error rate rises as the compression ratio increases. Nevertheless, it is worth noting that our approach maintains a satisfactory defensive performance when the compression ratio $\leq$ 80.

VIII. Conclusion

In this paper, we propose SuperFL, a groundbreaking solution that effectively addresses critical challenges in privacy preservation, robust aggregation, and protection against poisoning attacks. By utilizing classic fully homomorphic encryption techniques called CKKS and proxy re-encryption techniques, we establish a secure framework that guarantees byzantine robustness while safeguarding customer privacy. This framework incorporates a unique two-server collaborative aggregation model. To enhance communication efficiency, we integrate efficient sketch compression methods and employ a multiple-defense strategy based on trusted root datasets. This approach not only enhances communication efficiency but also effectively defends against poisoning attacks. Extensive experiments and comparisons with cutting-edge solutions demonstrate the outstanding performance of our method in terms of robustness and efficiency.

References

[1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[2] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Pro. Int. Conf. Adv. Neural Inf. Process. Syst. (NIPS)*, 2019, pp. 14 774–14 784.

[3] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Proc. 25th Eur. Symp. Res. Comput. Secur. (ESORICS)*, 2020, pp. 480–501.

[4] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *Proc. 29th USENIX Secur. Symp. (USENIX Secur.)*, 2020, pp. 1605–1622.

[5] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly) logarithmic overhead," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2020, pp. 1253–1269.

[6] T. Stevens *et al.*, "Efficient differentially private secure aggregation for federated learning via hardness of learning with errors," in *Proc. 31th USENIX Secur. Symp. (USENIX Secur.)*, 2022, pp. 1379–1395.

[7] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep lerning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1333–1345, 2018.

[8] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Pro. Int. Conf. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 118–128.

[9] B. Zhao, P. Sun, T. Wang, and K. Jiang, "Fedinv: Byzantine-robust federated learning by inversing local model updates," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2022, pp. 9171–9179.

[10] J. Xu, S.-L. Huang, L. Song, and T. Lan, "Byzantine-robust federated learning through collaborative malicious gradient filtering," in *Proc. IEEE 42th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2022, pp. 1223–1235.

[11] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," in *Pro. Int. Conf. Adv. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 4452–4463.

[12] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in *Proc. Conf. Empir. Methods Nat. Lang. Process. (EMNLP)*, Sep. 2017, pp. 440–445.

[13] S. Lin, Y. Han, X. Li, and Z. Zhang, "Personalized federated learning towards communication efficiency, robustness and fairness," in *Pro. Int. Conf. Adv. Neural Inf. Process. Syst. (NIPS)*, 2022, pp. 30 471–30 485.

[14] Y. Miao *et al.*, "Privacy-preserving byzantine-robust federated learning via blockchain systems," *IEEE Trans. Inf. Forensics Security*, vol. 17, p. 2848–2861, 2022.

[15] R. Hu, Y. Gong, and Y. Guo, "Federated learning with sparsification-amplified privacy and adaptive optimization," in *Proc. 30th Int. Joint Conf. Artif. Intell. (IJCAI)*, 8 2021, pp. 1463–1469.

[16] H. Lycklama, L. Burkhalter, A. Viand, N. Küchler, and A. Hithnawi, "Rofl: Robustness of secure federated learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*. IEEE, 2023, pp. 453–476.

[17] M. Rathee, C. Shen, S. Wagh, and R. A. Popa, "Elsa: Secure aggregation for federated learning with malicious actors," in *Proc. IEEE Symp. Secur. Privacy (SP)*. IEEE, 2023, pp. 1961–1979.

[18] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur. (ASIACRYPT)*, 2017, pp. 409–437.

[19] Z. Song, Y. Wang, Z. Yu, and L. Zhang, "Sketching for first order method: Efficient algorithm for low-bandwidth channel and vulnerability," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2023, pp. 32 365–32 417.

[20] Z. Song and Z. Yu, "Oblivious sketching-based central path method for linear programming," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 9835–9847.

[21] J. Nelson *et al.*, "Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci. (FOCS)*, 2013, pp. 117–126.

[22] Y. Aono *et al.*, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, pp. 1333–1345, 2017.

[23] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2020, pp. 493–506.

[24] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2017, pp. 19–38.

[25] P. Xu, M. Hu, T. Chen, W. Wang, and H. Jin, "Laf: Lattice-based and communication-efficient federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 2483–2496, 2022.

[26] Z. Huang, R. Hu, Y. Guo, E. Chan-Tin, and Y. Gong, "Dp-admm: Admm-based distributed learning with differential privacy," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1002–1012, 2019.

[27] K. Wei *et al.*, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, 2020.

[28] O. Aramoon, P.-Y. Chen, G. Qu, and Y. Tian, "Meta federated learning," *arXiv preprint arXiv:2102.05561*, 2021.

[29] Y. Li, X. Wang, R. Sun, X. Xie, S. Ying, and S. Ren, "Trustiness-based hierarchical decentralized federated learning," *Knowledge-Based Systems*, vol. 276, p. 110763, 2023.

[30] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2017, pp. 1273–1282.

[31] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 5650–5659.

[32] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2020.

[33] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.

[34] R. Guerraoui, S. Rouault *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 3521–3530.

[35] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2018, pp. 315–334.

[36] S. Lu, R. Li, W. Liu, C. Guan, and X. Yang, "Top-k sparsification with secure aggregation for privacy-preserving federated learning," *Computers & Security*, vol. 124, p. 102993, 2023.

[37] D. Rothchild *et al.*, "Fetchsgd: Communication-efficient federated learning with sketching," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020.

[38] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput*, 2009, pp. 169–178.

[39] Y. Polyakov, K. Rohloff, G. Sahu, and V. Vaikuntanathan, "Fast proxy re-encryption for publish/subscribe systems," *ACM Trans. Priv. Secur. (TOPS)*, vol. 20, no. 4, pp. 1–31, 2017.

[40] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," in *Int. Colloq. Automata Lang. Program. (ICALP)*. Springer, 2002, pp. 693–703.

[41] A. Badawi *et al.*, "Openfhe: Open-source fully homomorphic encryption library," in *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, 2022, p. 53–63.

[42] P. Li, T. J. Hastie, and K. W. Church, "Very sparse random projections," in *Proc. Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2006, p. 287–296.