RFLPA: A Robust Federated Learning Framework against Poisoning Attacks with Secure Aggregation

Peihua Mai **Ran Yan** **Yan Pang** *
National University of Singapore

Abstract

Federated learning (FL) allows multiple devices to train a model collaboratively without sharing their data. Despite its benefits, FL is vulnerable to privacy leakage and poisoning attacks. To address the privacy concern, secure aggregation (SecAgg) is often used to obtain the aggregation of gradients on sever without inspecting individual user updates. Unfortunately, existing defense strategies against poisoning attacks rely on the analysis of local updates in plaintext, making them incompatible with SecAgg. To reconcile the conflicts, we propose a robust federated learning framework against poisoning attacks (RFLPA) based on SecAgg protocol. Our framework computes the cosine similarity between local updates and server updates to conduct robust aggregation. Furthermore, we leverage verifiable packed Shamir secret sharing to achieve reduced communication cost of $O(M + N)$ per user, and design a novel dot product aggregation algorithm to resolve the issue of increased information leakage. Our experimental results show that RFLPA significantly reduces communication and computation overhead by over 75% compared to the state-of-the-art secret sharing method, BREA, while maintaining competitive accuracy.

1 Introduction

Federated learning (FL) [1–5] is a promising machine learning technique that has been gaining attention in recent years. It enables numerous devices to collaborate on building a machine learning model without sharing their data with each other. Compared with traditional centralized machine learning, FL preserves the data privacy by ensuring that sensitive data remain on local devices.

Despite its benefits, FL still has two key concerns to be addressed. Firstly, there is a threat of privacy leakage from local update. Recent works have demonstrated that the individual updates could reveal sensitive information, such as properties of the training data [6, 7], or even allows the server to reconstruct the training data [8, 9]. The second issue is that FL is vulnerable to poisoning attacks. Indeed, malicious users could send manipulated updates to corrupt the global model at their will [10]. The poisoning attacks may degrade the performance of the model, in the case of *untargeted attacks*, or bias the model's prediction towards a specific target labels, in the case of *targeted attacks* [11].

Secure aggregation (SecAgg) has become a potential solution to address the privacy concern. Under SecAgg protocol, the server could obtain the sum of gradients without inspecting individual user updates [12, 13]. However, this protocol poses a significant challenge in resisting poisoning attacks in FL. Most defense strategies [14, 15] require the server to access local updates to detect the attackers, which increases the risk of privacy leakage. The contradiction makes it difficult to develop a FL framework that simultaneously resolves the privacy and robustness concerns.

To our best knowledge, BREA is the state-of-the-art FL framework that defends against poisoning attacks using secret sharing-based SecAgg protocol [16]. Based on verifiable secret sharing, their

*Correspondence to bizpyj@nus.edu.sg

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-3314

framework leverages pairwise distances to remove outliers. However, their work is limited by the scaling concerns arising from computation and communication complexity. For a model with dimension M and N selected clients, the framework incurs $O(MN + N)$ communication per user, and $O((N^2 + MN)\log^2 N \log\log N)$ computation for the server due to the costly aggregation rule. Furthermore, BREA makes unrealistic assumptions that the users could establish direct communication channels with other mobile devices.

To address the above challenge, we propose a robust federated learning framework against poisoning attacks (RFLPA) based on SecAgg protocol. We leverage verifiable packed Shamir secret sharing to compute the cosine similarity and aggregate gradients in a secure manner with reduced communication cost of $O(M + N)$ per user. To resolve the increased information leakage from packed secret sharing, we design a dot product aggregation protocol that only reveals a single value of the dot product to the server. Our framework requires the server to store a small and clean root dataset as the benchmark. Each user relies on the server to communicate the secret with each other, and utilizes encryption and signature techniques to ensure the secrecy and integrity of messages. The implementation is available at https://github.com/NusIoraPrivacy/RFLPA.

Our main contributions involves the following:

(1) We propose a federated learning framework that overcomes privacy and robustness issues with reduced communication cost, especially for high-dimensional models. The convergence analysis and empirical results show that our framework maintains competitive accuracy while reducing communication cost significantly.

(2) To protect the privacy of local gradients, we propose a novel dot product aggregation protocol. Directly using packed Shamir secret sharing for dot product calculation can result in information leakage. Our dot product aggregation algorithm addresses this issue by ensuring that the server only learns the single value of the dot product and not other information about the local updates. Furthermore, the proposed protocol enables degree reduction by converting the degree-2d partial dot product shares into degree-d final product shares.

(3) Our framework guarantees the secrecy and integrity of secret shares for a server-mediated network model using encryption and signature techniques.

2 Literature Review

2.1 Defense against Poisoning Attacks.

Various robust aggregation rules have been proposed to defend against poisoning attacks. KRUM selects the benign updates based on the pairwise Euclidean distances between the gradients [14]. Yin et al. [17] proposes two robust coordinate-wise aggregation rules that computes the median and trimmed mean at each dimension, respectively. Bulyan [18] selects a set of gradients using Byzantine–resilient algorithm such as KRUM, and then aggregates the updates with trimmed mean. RSA [19] adds a regularization term to the objective function such that the local models are encouraged to be similar to the global model. In FLTrust [15], the server maintains a model on its clean root dataset and computes the cosine similarity to detect the malicious users. The aforementioned defense strategies analyze the individual gradients in plaintext, and thus are susceptible to privacy leakage.

2.2 Robust Privacy-Preserving FL.

To enhance privacy and resist poisoning attacks, several frameworks have integrated homomorphic encryption (HE) with existing defense techniques. Based on Paillier cryptosystem, PEFL [20] calculates the Pearson correlation coefficient between coordinate-wise medians and local gradients to detect malicious users. PBFL [21] uses cosine similarity to identify poisonous gradients and adopted fully homomorphic encryption (FHE) to ensure security. ShieldFL [22] computes cosine similarity between encrypted gradients with poisonous baseline for Byzantine-tolerance aggregation. The above approaches inherit the costly computation overhead of HE. Furthermore, they rely two non-colluding parties to perform secure computation and thus might be vulnerable to privacy leakage. Secure Multi-party Computation (SMC) is an alternative to address the privacy concern. To the best of our knowledge, BREA [16] is the first work that developed Byzantine robust FL framework using verifiable Shamir secret sharing. However, their method suffers high communication complexity of

SMC and high computation complexity of KRUM aggregation protocol. Refer to Appendix K.4 for a comprehensive comparison among existing protocols.

This paper explores the integration of SMC with defense strategy against poisoning attacks. We develop a framework that reduces communication cost, employs a more efficient aggregation rule and guarantees the security for a server-mediated model.

3 Problem Formulation and Background

3.1 Problem Statement

We assume that the server trains a model $\mathbf{w}$ with N mobile clients in a federated learning setting. All parties are assumed to be computationally bounded. Each client holds a local dataset $\{D_i\}_{i\in[N]}$, and the server owns a small, clean root dataset D_0. The objective is to optimize the expected risk function:

$$F(\mathbf{w}) = \min_{\mathbf{w}} \mathbb{E}_{D\sim\chi} L(D, \mathbf{w}), \tag{1}$$

where $L(D, \mathbf{w})$ is a empirical loss function given dataset D.

In federated learning, the server aggregates local gradients $\mathbf{g}_i^t$ to obtain global gradient $\mathbf{g}^t$ for model update:

$$\mathbf{g}^t = \sum_{i\in S} \eta_i^t \mathbf{g}_i^t,\ \mathbf{w}^t = \mathbf{w}^{t-1} - \gamma^t \mathbf{g}^t, \tag{2}$$

where η_i is the weight of client i, γ^t is the learning rate, and S is the set of selected clients.

3.2 Adversary Model

We consider two types of users, i.e., honest users and malicious users. The definitions of honest and malicious users are given as follows.

Definition 3.1 (Honest Users)**.** A user u is honest if and only if u honestly submits its local gradient g_u, where g_u is the true gradients trained on its local dataset D_u.

Definition 3.2 (Malicious Users)**.** A user u is malicious if and only if u is manipulated by an adversary who launches model poisoning attack by submitting poisonous gradients g_u^*.

Server aims to infer users' information with two types of attacks, i.e., passive inference and active inference attack. In passive inference attack, the server tries to infer users' sensitive information by the intermediate result it receives from the user or eardrops during communication. In active inference attack, the server would manipulate certain users' messages to obtain the private values of targeted users.

3.3 Design Goals

We aim to design a federated learning system with three goals.

Privacy. Under federated learning, users might still be concerned about the information leakage from individual gradients. To protect privacy, the server shouldn't have access to local update of any user. Instead, the server learns only the aggregation weights and global gradients, ensuring that individual user data remains protected.

Robustness. We aim to design a method resilient to model poisonous attack, meaning that the model accuracy should be within a reasonable range under malicious clients.

Efficiency. Our framework should maintain computation and communication efficiency even if it is operated on high dimensional vectors.

3.4 Cryptographic Primitives

In this section we briefly describe cryptographic primitives for our framework. For more details refer to Appendix B.

 https://doi.org/10.52202/079017-3314

Packed Shamir Secret Sharing. This study uses a generalization of Shamir secret sharing scheme [23], known as "packed secret-sharing" that allows to represent multiple secrets by a single polynomial [24]. A degree-d ($d \geq l-1$) packed Shamir sharing of $\mathbf{s} = (s_1, s_2, ..., s_l)$ stores the l secrets at a polynomial $f(\cdot)$ of degree at most d. The secret sharing scheme requires $d+1$ shares for reconstruction, and any $d-l+1$ shares reveals no information of the secret.

Key Exchange. The framework relies on Diffie–Hellman key exchange protocol [25] that allows two parties to establish a secret key securely.

Symmetric Encryption. Symmetric encryption guarantees the secrecy for communication between two parties [26]. The encryption and decryption are conducted with the same key shared by both communication partners.

Signature Scheme. To ensure the integrity and authenticity of message, we adopt a UF-CMA secure signature scheme [27, 28].

4 Framework

4.1 Overview

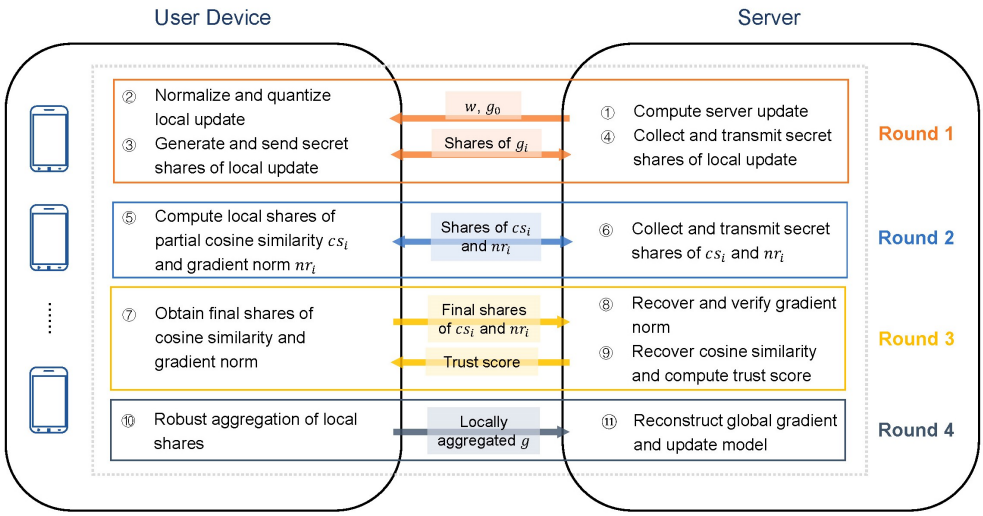

Figure 1: Overall framework

Figure 1 depicts the overall framework of our robust federated learning algorithm. The algorithm consists of four rounds:

Round 1: each client receives the server update g_0, computes their updates normalized by g_0, and distributes the secret shares of their updates to other clients.

Round 2: each client computes the local shares of partial dot product for gradient norm and cosine similarity, and conducts secret re-sharing on the local shares.

Round 3: each client obtains final shares of partial dot product for gradient norm and cosine similarity, and transmits the shares to server. Then the server would verify the gradient norm, recover cosine similarity, and compute the trust score for each client.

Round 4: on receiving the trust score from the server, each client conducts robust aggregation on the secret shares locally, and transmits the secret shares of aggregated gradient to the server. The server finally reconstructs the aggregation on the secret shares.

To address increased information leakage caused by packed secret sharing, we design a dot product aggregation protocol to sum up the dot product over sub-groups of elements. Refer to Appendix D for the algorithm to perform robust federated learning.

4.2 Normalization and Quantization

To limit the impact of attackers, we follow [15] to normalize each local gradient based on the server model update:

$$\bar{\mathbf{g}}_i = \frac{\|\mathbf{g}_0\|}{\|\mathbf{g}_i\|} \cdot \mathbf{g}_i, \tag{3}$$

where $\mathbf{g}_i$ is the local gradient of the ith client, and $\mathbf{g}_0$ is the server gradient obtained from clean root data.

Each client performs local gradient normalization, and the server validates if the updates are truly normalized. The secret sharing scheme operates over finite field $\mathbb{F}_p$ for some large prime number p, and thus the user should quantize their normalized update $\bar{\mathbf{g}}_i$. The quantization poses challenge on normalization verification, as $\|\bar{\mathbf{g}}_i\|$ might not be exactly equal to $\|\mathbf{g}_0\|$ after being converted into finite field.

To address this issue, we define the following rounding function:

$$Q(x) = \begin{cases} \lfloor qx \rfloor / q, & x \geq 0 \\ (\lfloor qx \rfloor + 1)/q, & x < 0 \end{cases}, \tag{4}$$

where $\lfloor qx \rfloor$ is the largest integer less than or equal to qx.

Therefore, the server could verify that $\|\bar{\mathbf{g}}_i\| \leq \|\mathbf{g}_0\|$, which is ensured by the quantization method.

4.3 Robust Aggregation Rule

Consistent with FLTrust[15], our framework conducts robust aggregation using the cosine similarity between users' and server's updates. The trust score of user i is:

$$TS_i = \max\left(0, \frac{\langle \mathbf{g}_i, \mathbf{g}_0 \rangle}{\|\mathbf{g}_i\| \|\mathbf{g}_0\|}\right) = \max\left(0, \frac{\langle \bar{\mathbf{g}}_i, \mathbf{g}_0 \rangle}{\|\mathbf{g}_0\|^2}\right), \tag{5}$$

where we clip the negative cosine similarity to zero to avoid the impact of malicious clients.

The global gradient is then aggregated by:

$$\mathbf{g} = \frac{1}{\sum_{i=1}^{N} TS_i} \sum_{i=1}^{N} TS_i \cdot \bar{\mathbf{g}}_i. \tag{6}$$

Finally, we use the gradient to update the global model:

$$\mathbf{w} \leftarrow \mathbf{w} - \gamma \mathbf{g}. \tag{7}$$

Our framework leverages the robust aggregation rule consistent with FLTrust due to its advantages including low computation cost, the absence of a requirement for prior knowledge about number of poisoners, defend against majority number of poisoners, and compatibility with Shamir Secret Sharing. Appendix C details the comparison between FLTrust and existing robust aggregation rules.

4.4 Verifiable Packed Secret Sharing

The core idea of packed secret sharing is to encode l secrets within a single polynomial. Consequently, the secret shares of local updates generated by each user would reduce from NM to NM/l. By selecting $l = O(N)$, the per-user communication cost at secret sharing stage can be decreased to $O(M+N)$. We assume that the prime number P is large enough such that $P > \max\{N\|\mathbf{g}_0\|, \|\mathbf{g}_0\|^2\}$ to avoid overflow.

One issue with secret sharing is that a malicious client may send invalid secret shares, i.e., shares that are not evaluated at the same polynomial function, to break the training process. To address this

 https://doi.org/10.52202/079017-3314

issue, the framework utilizes the verifiable secret sharing scheme from [29], which generates constant size commitment to improve communication efficiency. We construct the verifiable secret shares for both local gradients and partial dot products described in Section 4.5. During verifiable packed secret sharing, the user would send the secret shares $\mathbf{s}$, commitment $\mathcal{C}$, and witness w_l to other users. A commitment is a value binding to a polynomial function $\phi(x)$, i.e., the underlying generator of the secret shares, without revealing it. A witness allows others to verify that the secret share s_l is generated at l of the polynomial (see Appendix E for more details).

4.5 Dot Product Aggregation

Directly applying packed secret sharing may increase the risk of information leakage when calculating cosine similarity and gradient norm. In the example provided by Figure 2, the gradient vectors are created as secret shares by packing l secret into a polynomial function. Following the local similarity computations by each client, the server can reconstruct the element-wise product between the two gradients, which makes it easy to recover the user's gradient $\tilde{g}_i$ from the reconstructed metric. On the other hand, our proposed protocol ensures that only the single value of dot product is released to the server. Based on this, we introduce a term *partial dot product*, or *partial cosine similarity (norm square)* depending on the input vectors, defined as follows:

Partial dot product *represents the multiple dot products of several subgroups of elements from input vectors rather than a single dot product value.*

Another related concept is *final dot product*, referring to the single value of dot products between two vectors. For example, given two vectors $\boldsymbol{v}_1 = (2, -1, 4, 5, 6, 3)$ and $\boldsymbol{v}_2 = (1, 2, 0, 3, -2, 1)$, the reconstructed *partial dot product* could be $(0, 15, -9)$ if we pack 2 elements into a secret share, while the *final dot product* is 6. If each client directly uploads the shares from local dot product computation, the server would reconstruct a vector of partial cosine similarity (norm square) and thus learn more gradient information. To ensure that the server only has access to final cosine similarity

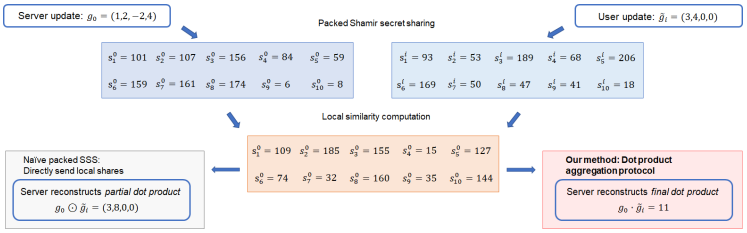

Figure 2: Cosine similarity computation on packed secret sharing

(norm square), we design a dot product aggregation algorithm based on secret re-sharing that allows the users to sum up the dot products over subgroups.

Suppose that the user i creates a packed secret sharing $\mathbf{V}^i = \{v_{jk}^i\}_{j\in[N],k\in[\lceil m/l\rceil]}$ of $\tilde{\mathbf{g}}_i = (g_1^i, g_2^i, ..., g_M^i)$, by packing each l elements into a secret. On receiving the secret shares, each user i can compute the vectors $\mathbf{cs}^i = (cs_1^i, cs_2^i, ..., cs_N^i)$ and $\mathbf{nr}^i = (nr_1^i, nr_2^i, ..., nr_N^i)$:

$$cs_j^i = \sum_l v_{il}^j \cdot v_{il}^0,\ nr_j^i = \sum_l v_{il}^j \cdot v_{il}^j, \tag{8}$$

where cs_j^i and nr_j^i denotes the i^{th} share of partial cosine similarity and partial gradient norm square for user j's gradient.

The partial cosine similarity (or gradient norm square) could be further aggregated by the procedure below in four steps.

Step 1: Secret resharing of partial dot product. Each user i could construct the verifiable packed secret shares of $\mathbf{cs}^i$ (or $\mathbf{nr}^i$) by representing p secrets on a polynomial:

$$\mathbf{S}^i = \begin{pmatrix} s_{11}^i & \cdots & s_{1\lceil N/p\rceil}^i \\ \vdots & \ddots & \vdots \\ s_{N1}^i & \cdots & s_{N\lceil N/p\rceil}^i \end{pmatrix}, \tag{9}$$

where s_{jk}^i denotes the share sent to user j for the k^{th} group of elements in vector $\mathbf{cs}^i$ (or $\mathbf{nr}^i$). By choosing $p = O(N)$, each user will generate $O(N)$ secret shares.

Step 2: Disaggregation on re-combination vector. After distributing the secret shares, each user i receives a re-combination vector $\mathbf{s}_{ik} = (s_{ik}^1, s_{ik}^2, ..., s_{ik}^N)$ for $k \in [\lceil N/p \rceil]$. Since we pack l elements for the secret shares of partial dot product, this step aims to transform the $\mathbf{s}_{ik}$ into l vectors, with each vector representing one element. For each $j \in [l]$, user i locally computes:

$$\tilde{\mathbf{h}}_{jk}^i = \mathbf{s}_{ik} B_{e_j}^{-1} Chop_d, \tag{10}$$

where B_{e_j} is an n by n matrix whose (i,k) entry is $(\alpha_k - e_j)^{i-1}$, and $Chop_d$ is an n by n matrix whose (i,k) entry is 1 if $1 \leq i = k \leq d$ and 0 otherwise. After this operation, the degree-2d partial dot product shares are transformed into degree-d shares.

Step 3: Aggregation along packed index. The new secrets are summed up along $j \in [l]$ at client side:

$$\mathbf{h}_k^i = \sum_{j=1}^{l} \tilde{\mathbf{h}}_{jk}^i. \tag{11}$$

Step 4: Decoding for final secret shares. User i can derive the final secret shares x_k^i by recovering from $\mathbf{h}_k^i = (h_{k1}^i, h_{k2}^i, ..., h_{kN}^i)$ using Reed-Solomon decoding. Noted that $\{x_k^i\}_{k \in [\lceil N/p \rceil]}$ becomes a packed secret share of dot products of degree d (see Appendix F). Therefore, the server could recover the cosine similarity (or gradient norm square) for all users on receiving the final shares from sufficient users.

4.6 Secret Sharing over Insecure Channel

This framework relies on a server-mediated communication channel for the following reasons: (1) it's challenging for mobile clients to establish direct communication with each other and authenticate other devices; (2) a server could act as central coordinator to ensure that all clients have access to the latest model. On the other hand, the secret sharing stage requires to maintain the privacy and integrity of secret shares.

To protect the secrecy of message, we utilize key agreement and symmetric encryption protocol. The clients establish the secret keys with each other through Diffie–Hellman key exchange protocol. During secret sharing, each client u uses the common key k_{uv} to encrypt the message sent to client v, and client v could decrypt the cyphertext with the same key.

Another concern is that the server may falsify the messages transmitted between clients. Signature scheme is adopted to prevent the active attack from server. We assume that all clients receive their private signing key and public signing keys of all other clients from a trusted third party. Each client i generates a signature σ_i along with the message m, and other clients verify the message using client i's public key d_i^{PK}.

5 Theoretical Analysis

5.1 Complexity Analysis

In this section, we analyze the per iteration complexity for N selected clients, and model dimension of M, and summarize the complexity in Table 1. Further details of the complexity analysis are available in Appendix G. One important observation is that the communication complexity of our protocol reduces from $O(MN + N)$ to $O(M + N)$. Furthermore, the server-side computation overhead is reduced to $O((M + N) \log^2 N \log \log N)$, benefiting from the efficient aggregation rule and packed secret sharing. It should be noted that while the BERA protocol has similar server communication complexity, it makes an unrealistic assumption that users can share secrets directly with each other, thereby saving the server's overhead.

5.2 Security Analysis

The security analysis is conducted for Algorithm 3. Given a security parameter κ, a server S, and any subsets of users $\mathcal{U}$, let $\text{REAL}_{\mathcal{C}}^{\mathcal{U},t,\kappa}$ be a random variable representing the joint view of parties

 https://doi.org/10.52202/079017-3314

Table 1: Complexity summary of RFLPA and BERA

	RFLPA		BERA	
	Computation	Communication	Computation	Communication
Server	$O((M+N)\log^2 N \log\log N)$	$O((M+N)N)$	$O((N^2+MN)\log^2 N \log\log N)$	$O(MN+N^2)$
User	$O((M+N^2)\log^2 N)$	$O((M+N))$	$O(MN\log^2 N + MN^2)$	$O(MN+N)$

in $\mathcal{C} \subseteq \mathcal{U} \cup S$ where the threshold is set to t, and $\mathcal{U}_i$ be the subset of respondents at round i such that $\mathcal{U} \supseteq \mathcal{U}_1 \supseteq \mathcal{U}_2 \supseteq \mathcal{U}_3 \supseteq \mathcal{U}_4$. We show that the joint view of any group of parties from $\mathcal{C}$ with users less than t can be simulated given the inputs of clients in that group, trust score $\{TS_j\}_{j\in\mathcal{U}_1}$, and global gradient $\mathbf{g}$. In other words, *the server learns no information about clients' input except the global gradient and trust score*.

Theorem 5.1 (Security against active server and clients)**.** *There exists a PPT simulator* SIM *such that for all $t \leq K-L$, $|\mathcal{C}\backslash\{S\}| < t$, the output of* SIM *is computationally indistinguishable from the output of* $\mathrm{REAL}_{\mathcal{C}}^{\mathcal{U},t,\kappa}$*:*

$$\mathrm{REAL}_{\mathcal{C}}^{\mathcal{U},t,\kappa}(\mathbf{x}_{\mathcal{U}}) \equiv \mathrm{SIM}_{\mathcal{C}}^{\mathcal{U},t,\kappa}(\mathbf{x}_{\mathcal{U}}) \tag{12}$$

where "$\equiv$" represents computationally indistinguishable.

5.3 Correctness against Malicious Users

In this section, we show that our protocol executes correctly under the following attacks of malicious users: (1) sending invalid secret shares; (2) sending shares from incorrect computation of 6, 8, 10, or 11. Note that adversaries may also create shares from arbitrary gradients, and we left the discussion of such attack to Section 5.4.

The first attack arises when the user doesn't generate shares from the same polynomial. Such attempt is prevented by verifiable secret sharing that allows for the verification of share validity by testing 18.

The second attack could be addressed by Reed-Solomon codes. For a degree-d packed Shamir secret sharing with n shares, the Reed-Solomon decoding algorithm could recover the correct result with E errors and S erasures as long as $S + 2E + d + 1 \leq n$.

5.4 Convergence Analysis

Theorem 5.2. *Suppose Assumption J.1, J.2, J.3 in Appendix J hold. For arbitrary number of malicioius clients, the difference between the global model $\mathbf{w}^t$ learnt by our algorithm and the optimal $\mathbf{w}^*$ is bounded. Formally, we have the following inequality with probability at least $1-\delta$:*

$$\|\mathbf{w}^t - \mathbf{w}^*\| \leq (1-\rho)^t \|\mathbf{w}^0 - \mathbf{w}^*\| + 12\gamma\Delta_1 + \frac{\gamma\sqrt{d}}{q} \tag{13}$$

where $\rho = 1 - (\sqrt{1-\mu^2/(4L_g^2)} + 24\gamma\Delta_2 + 2\gamma L)$, $\Delta_1 = \nu_1\sqrt{\frac{2}{|D_0|}}\sqrt{d\log 6 + \log(3/\delta)}$, $\Delta_2 = \nu_2\sqrt{\frac{2}{|D_0|}}\sqrt{d\log\frac{18L_2}{\nu_2} + \frac{1}{2}d\log\frac{|D_0|}{d} + \log\left(\frac{6\nu_2^2 r\sqrt{D_0}}{\alpha_2\nu_1\delta}\right)}$, $L_2 = \max\{L, L_1\}$.

Remark 5.3. $\gamma\sqrt{d}/q$ is the noise caused by the quantization process in our algorithm.

6 Experiments

6.1 Experimental Setup

Dataset. We use three standard datasets to evaluation the performance of RFLPA: MNIST [30], FashionMNIST (F-MNIST) [31], and CIFAR-10 [32]. MNIST and F-MNIST are trained on the neural network classification model composed of two convolutional layers and two fully connected layers, while CIFAR-10 is trained and evaluated with a ResNet-9 [33] model.

s **Attacks.** We simulate two types of poisoning attacks: gradient manipulation attack (untargeted) and label flipping attack (targeted). Under gradient manipulation attack, the malicious users generate

arbitrary gradients from normal distribution of mean 0 and standard deviation 200. For label flipping attack, the adversaries flip the label from l to $P - l - 1$, where P is the number of classes. We consider the proportion of attackers from 0% to 30%.

6.2 Experiment Results

6.2.1 Accuracy Evaluation

We compare our proposed method with several FL frameworks: FedAvg [34], Bulyan [18], Trim-mean [17], local differential privacy (LDP) [35], central differential privacy (CDP) [35], and BREA [16]. Refer to Table 5 for the corse-grained comparison between RFLPA and the baselines. Noted that several baselines are not included in the accuracy comparison because: (i) The security of the some schemes relies on the assumption of two non-colluding parties, which is vulnerable in real life. (ii) Some frameworks entail significant computation costs, rendering their implementation in real-life scenarios impractical (see Appendix K.8.1). Table 2 summarizes the accuracies for different methods under the two attacks.

When defense strategy is not implemented, the accuracies of FedAvg decrease as the proportion of attackers increases, with a more significant performance drop observed under gradient manipulation attacks. Benefited from the trust benchmark, our proposed framework, RFLPA, demonstrates more stable performance for up to 30% adversaries compared to other baselines. In the absence of attackers, our method achieves slightly lower accuracies than FedAvg, with an average decrease of 2.84%, $4.38\%$and 3.46%, respectively, for MNIST, F-MNIST, and CIFAR-10 dataset.

6.2.2 Overhead Analysis

To verify the effectiveness of our framework on reducing overhead, we compare the per-iteration communication and computation cost for BREA and RFLPA in Figure 3. For each experiment we set the degree as $0.4N$ and encode $0.1N$ elements within a polynomial.

The left-most graph presents the overhead with different participating client size using the 1.6M parameter model described in Section 6.1. For $M \gg N$, the per-client communication complexity for RFLPA remains stable at around 82.5MB, regardless of user size. Conversely, BREA exhibits linear scalability with the number of participating clients. Our framework reduces the communication cost by over 75% compared with BREA.

The second left graph examines the communication overhead for varying model dimensions with 2,000 participating clients. RFLPA achieves a much lower per-client cost than BREA by leveraging packed secret sharing, leading to a 99.3% reduction in overhead.

The right two figures presents the computation cost under varying client size using a MNIST classifier with 1.6M parameters. Benefiting from the packed VSS, RFLPA reduces both the user and server computation overhead by over 80% compared with BREA.

Table 2: Accuracy under different proportions of attackers. The values denote the mean $\pm$ standard deviation of the performance.

Proportion of Attackers		Gradient Manipulation				Label Flipping			
		No	10%	20%	30%	No	10%	20%	30%
FedAvg	MNIST	**0.98 ±0.0**	0.46 ±0.1	0.40 ±0.1	0.32 ±0.0	**0.98 ±0.0**	**0.96 ±0.0**	0.92 ±0.0	0.82 ±0.0
	F-MNIST	**0.88 ±0.0**	0.55 ±0.0	0.51 ±0.0	0.45 ±0.1	**0.88 ±0.0**	0.82 ±0.0	0.73 ±0.0	0.69 ±0.0
	CIFAR-10	0.76 ±0.3	0.14 ±0.2	0.13 ±0.8	0.13 ±0.2	0.76 ±0.3	0.72 ±1.1	0.68 ±2.7	0.59 ±0.8
Bulyan	MNIST	0.98 ±0.0	0.92 ±0.0	0.89 ±0.0	0.87 ±0.0	0.98 ±0.0	0.91 ±0.0	0.90 ±0.0	0.87 ±0.0
	F-MNIST	0.86 ±0.0	0.73 ±0.0	0.71 ±0.1	0.69 ±0.0	0.86 ±0.0	0.76 ±0.0	0.70 ±0.1	0.68 ±0.0
	CIFAR-10	**0.77 ±1.0**	**0.73 ±0.8**	0.45 ±1.2	0.27 ±0.6	**0.77 ±1.0**	**0.72 ±0.2**	0.62 ±1.8	0.40 ±0.9
Trim-mean	MNIST	0.98 ±0.0	0.95 ±0.0	0.93 ±0.0	0.91 ±0.0	0.98 ±0.0	0.95 ±0.0	0.92 ±0.0	0.90 ±0.0
	F-MNIST	0.86 ±0.0	0.81 ±0.0	0.74 ±0.0	0.71 ±0.0	0.86 ±0.0	0.78 ±0.0	0.74 ±0.0	0.73 ±0.0
	CIFAR-10	0.76 ±1.0	0.57 ±2.1	0.51 ±1.1	0.47 ±2.2	0.76 ±1.0	0.71 ±1.3	0.68 ±0.7	0.56 ±1.1
LDP	MNIST	0.87 ±0.1	0.13 ±0.0	0.10 ±0.0	0.10 ±0.0	0.87 ±0.1	0.87 ±0.3	0.83 ±1.2	0.77 ±2.1
	F-MNIST	0.74 ±0.1	0.59 ±0.4	0.53 ±1.2	0.12 ±0.0	0.74 ±0.1	0.63 ±0.5	0.62 ±0.2	0.59 ±1.2
	CIFAR-10	0.14 ±0.2	0.14 ±0.2	0.12 ±0.3	0.12 ±0.1	0.14 ±0.2	0.14 ±0.2	0.14 ±0.3	0.13 ±0.1
CDP	MNIST	0.96 ±0.0	0.96 ±0.0	0.95 ±0.0	0.94 ±0.0	0.96 ±0.0	0.96 ±0.0	0.95 ±0.3	0.91 ±0.2
	F-MNIST	0.83 ±0.1	0.51 ±0.1	0.41 ±0.0	0.34 ±0.1	0.83 ±0.1	0.81 ±0.5	0.79 ±0.0	0.78 ±0.7
	CIFAR-10	0.71 ±1.2	0.12 ±0.5	0.12 ±0.3	0.12 ±0.3	0.71 ±1.2	0.68 ±0.7	0.66 ±1.5	0.63 ±1.3
BREA	MNIST	0.94 ±0.0	0.93 ±0.0	0.93 ±0.0	0.93 ±0.0	0.94 ±0.0	0.94 ±0.0	0.93 ±0.0	0.93 ±0.0
	F-MNIST	0.84 ±0.0	0.83 ±0.0	0.82 ±0.0	0.81 ±0.0	0.84 ±0.0	0.84 ±0.0	0.82 ±0.0	0.81 ±0.0
	CIFAR-10	0.70 ±1.0	0.69 ±1.1	0.68 ±1.9	0.68 ±0.7	0.70 ±1.0	0.70 ±2.2	0.67 ±0.9	0.65 ±2.7
RFLPA	MNIST	0.96 ±0.0	**0.96 ±0.0**	**0.95 ±0.0**	**0.95 ±0.0**	0.96 ±0.0	0.96 ±0.0	**0.95 ±0.0**	**0.95 ±0.0**
	F-MNIST	0.84 ±0.0	**0.84 ±0.0**	**0.83 ±0.0**	**0.82 ±0.0**	0.84 ±0.0	**0.83 ±0.0**	**0.83 ±0.0**	**0.82 ±0.0**
	CIFAR-10	0.74 ±2.3	0.70 ±1.8	**0.70 ±1.9**	**0.69 ±1.8**	0.74 ±2.3	0.71 ±1.7	**0.70 ±1.6**	**0.69 ±0.8**

https://doi.org/10.52202/079017-3314

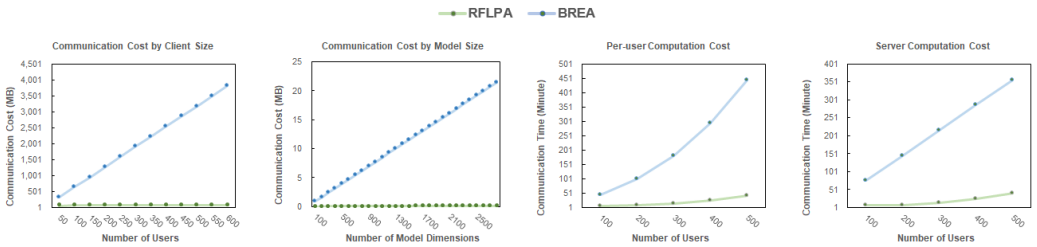

Figure 3: Per-iteration communication (left two) and computation cost (right two).

6.2.3 Other studies

For other studies, we analyze the impact of iterations on accuracy (see Appendix K.5), evaluate our protocol against additional attacks (see Appendix K.6), conduct further overhead analysis (see K.8), and examine the performance under non-iid setting (see Appendix K.9).

7 Conclusion

This paper proposes RFLPA, a robust privacy-preserving FL framework with SecAgg. Our framework leverages verifiable packed Shamir secret sharing to compute the cosine similarity between user and server update and conduct robust aggregation. We design a secret re-sharing algorithm to address the increased information leakage concern, and utilize encryption and signature techniques to ensure the security over server-mediated channel. Our approach achieves the reduced per-user communication overhead of $O(M + N)$. The empirical study demonstrates that: (1) RFLPA achieves competitive accuracies for up to 30% poisoning adversaries compared with state-of-the-art defense methods. (2) The communication cost and computation cost for RFLPA is significantly lower than BERA by over 75% under the same FL settings.

8 Discussion and Future Work

Collection of server data. One important assumption is that the server is required to collect a small, clean root dataset. Such collection is affordable for most organizations as the required dataset is of small size, e.g., 200 samples. According to theoretical analysis, the convergence is guaranteed when the root dataset is representative of the overall training data. Empirical evidence presented in [15] suggests that the performance of the global model is robust even when the root dataset diverges slightly from the overall training data distribution. Furthermore, Appendix K.10 proposes several alternative robust aggregation modules, such as KRUM and comparison with global model, to circumvent the assumption.

Compatibility with other defense strategies. RFLPA adopts a robust aggregation rule that computes the cosine similarity with server update. The framework can be easily generalized to distance-based method such as KRUM or multi-KRUM by substituting the robust aggregation module. However, extending the framework to rank-based defense methods may be more challenging. Existing SMC techniques for rank-based statistics requires $\log M$ rounds of communication, where M is the range of input values [36]. We leave the problem of communication-efficient rank-based robust FL to future work.

Differential privacy guarantee. Differential privacy (DP) [37, 38] provides formal privacy guarantees to prevent information leakage. The combination of SMC and DP, also known as Distributed DP [39], reduces the magnitude of noise added by each user compared with pure local DP. However, adopting DP in the privacy-preserving robust FL framework is non-trivial, especially when bounding the privacy leakage of robustness metrics such as cosine similarity may sacrifice utility. We leave the problem of incorporating DP into the privacy-preserving robust FL framework to future work.

References

[1] H Brendan McMahan et al. “Federated learning of deep networks using model averaging”. In: *arXiv preprint arXiv:1602.05629* 2 (2016).

[2] Rui Ye et al. “Feddisco: Federated learning with discrepancy-aware collaboration”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 39879–39902.

[3] Rui Ye et al. “Openfedllm: Training large language models on decentralized private data via federated learning”. In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2024, pp. 6137–6147.

[4] Jianyu Wang et al. “Tackling the objective inconsistency problem in heterogeneous federated optimization”. In: *Advances in neural information processing systems* 33 (2020), pp. 7611–7623.

[5] Rui Ye et al. “Fake It Till Make It: Federated Learning with Consensus-Oriented Generation”. In: *The Twelfth International Conference on Learning Representations*.

[6] Luca Melis et al. “Exploiting unintended feature leakage in collaborative learning”. In: *2019 IEEE symposium on security and privacy (SP)*. IEEE. 2019, pp. 691–706.

[7] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. “Model inversion attacks that exploit confidence information and basic countermeasures”. In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 2015, pp. 1322–1333.

[8] Ligeng Zhu, Zhijian Liu, and Song Han. “Deep leakage from gradients”. In: *Advances in neural information processing systems* 32 (2019).

[9] Di Chai et al. “Secure federated matrix factorization”. In: *IEEE Intelligent Systems* 36.5 (2020), pp. 11–20.

[10] Eugene Bagdasaryan et al. “How to backdoor federated learning”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2938–2948.

[11] Ling Huang et al. “Adversarial machine learning”. In: *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. 2011, pp. 43–58.

[12] Keith Bonawitz et al. “Practical secure aggregation for privacy-preserving machine learning”. In: *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 1175–1191.

[13] James Henry Bell et al. “Secure single-server aggregation with (poly) logarithmic overhead”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020, pp. 1253–1269.

[14] Peva Blanchard et al. “Machine learning with adversaries: Byzantine tolerant gradient descent”. In: *Advances in neural information processing systems* 30 (2017).

[15] Xiaoyu Cao et al. “FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping”. In: *ISOC Network and Distributed System Security Symposium (NDSS)*. 2021.

[16] Jinhyun So, Başak Güler, and A Salman Avestimehr. “Byzantine-resilient secure federated learning”. In: *IEEE Journal on Selected Areas in Communications* 39.7 (2020), pp. 2168–2181.

[17] Dong Yin et al. “Byzantine-robust distributed learning: Towards optimal statistical rates”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5650–5659.

[18] Rachid Guerraoui, Sébastien Rouault, et al. “The hidden vulnerability of distributed learning in byzantium”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3521–3530.

[19] Junyu Shi et al. “Challenges and approaches for mitigating byzantine attacks in federated learning”. In: *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE. 2022, pp. 139–146.

[20] Xiaoyuan Liu et al. “Privacy-enhanced federated learning against poisoning adversaries”. In: *IEEE Transactions on Information Forensics and Security* 16 (2021), pp. 4574–4588.

[21] Yinbin Miao et al. “Privacy-preserving Byzantine-robust federated learning via blockchain systems”. In: *IEEE Transactions on Information Forensics and Security* 17 (2022), pp. 2848–2861.

[22] Zhuoran Ma et al. “ShieldFL: Mitigating model poisoning attacks in privacy-preserving federated learning”. In: *IEEE Transactions on Information Forensics and Security* 17 (2022), pp. 1639–1654.

[23] Adi Shamir. “How to share a secret”. In: *Communications of the ACM* 22.11 (1979), pp. 612–613.

 https://doi.org/10.52202/079017-3314

[24] Matthew Franklin and Moti Yung. “Communication complexity of secure computation”. In: *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*. 1992, pp. 699–710.

[25] Whitfield Diffie and Martin E Hellman. “New directions in cryptography”. In: *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*. 2022, pp. 365–390.

[26] Hans Delfs et al. “Symmetric-key encryption”. In: *Introduction to cryptography: principles and applications* (2007), pp. 11–31.

[27] Ravneet Kaur and Amandeep Kaur. “Digital signature”. In: *2012 International Conference on Computing Sciences*. IEEE. 2012, pp. 295–301.

[28] Jonathan Katz. *Digital signatures*. Vol. 1. Springer, 2010.

[29] Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. “Constant-size commitments to polynomials and their applications”. In: *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*. Springer. 2010, pp. 177–194.

[30] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[31] Han Xiao, Kashif Rasul, and Roland Vollgraf. “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms”. In: *arXiv preprint arXiv:1708.07747* (2017).

[32] Alex Krizhevsky et al. “Learning multiple layers of features from tiny images”. In: (2009).

[33] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[34] Jakub Konečnỳ et al. “Federated learning: Strategies for improving communication efficiency”. In: *arXiv preprint arXiv:1610.05492* (2016).

[35] Mohammad Naseri, Jamie Hayes, and Emiliano De Cristofaro. “Local and central differential privacy for robustness and privacy in federated learning”. In: *arXiv preprint arXiv:2009.03561* (2020).

[36] Gagan Aggarwal, Nina Mishra, and Benny Pinkas. “Secure computation of the k th-ranked element”. In: *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*. Springer. 2004, pp. 40–55.

[37] Thông T Nguyên et al. “Collecting and analyzing data from smart device users with local differential privacy”. In: *arXiv preprint arXiv:1606.05053* (2016).

[38] Arnaud Berlioz et al. “Applying differential privacy to matrix factorization”. In: *Proceedings of the 9th ACM Conference on Recommender Systems*. 2015, pp. 107–114.

[39] Peter Kairouz, Ziyu Liu, and Thomas Steinke. “The distributed discrete gaussian mechanism for federated learning with secure aggregation”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 5201–5212.

[40] Mihir Bellare and Chanathip Namprempre. “Authenticated encryption: Relations among notions and analysis of the generic composition paradigm”. In: *Advances in Cryptology—ASIACRYPT 2000: 6th International Conference on the Theory and Application of Cryptology and Information Security Kyoto, Japan, December 3–7, 2000 Proceedings 6*. Springer. 2000, pp. 531–545.

[41] Menezes Alfred, Vanstone Scott, et al. *Handbook of applied cryptography*. 1997.

[42] Dan Boneh and Xavier Boyen. “Short signatures without random oracles”. In: *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*. Springer. 2004, pp. 56–73.

[43] Hsiang-Tsung Kung. *Fast evaluation and interpolation*. Carnegie-Mellon University. Department of Computer Science, 1973.

[44] Shuhong Gao. “A new algorithm for decoding Reed-Solomon codes”. In: *Communications, information and network security* (2003), pp. 55–68.

[45] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. “Robust aggregation for federated learning”. In: *IEEE Transactions on Signal Processing* 70 (2022), pp. 1142–1154.

[46] Meng Hao et al. “Efficient, private and robust federated learning”. In: *Proceedings of the 37th Annual Computer Security Applications Conference*. 2021, pp. 45–60.

[47] Hidde Lycklama et al. “Rofl: Robustness of secure federated learning”. In: *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2023, pp. 453–476.

[48] Mayank Rathee et al. “Elsa: Secure aggregation for federated learning with malicious actors”. In: *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2023, pp. 1961–1979.

[49] Minghong Fang et al. “Local model poisoning attacks to {Byzantine-Robust} federated learning”. In: *29th USENIX security symposium (USENIX Security 20)*. 2020, pp. 1605–1622.

[50] Tianyu Gu et al. “Badnets: Evaluating backdooring attacks on deep neural networks”. In: *IEEE Access* 7 (2019), pp. 47230–47244.

[51] Luisa Bentivogli et al. “The Fifth PASCAL Recognizing Textual Entailment Challenge.” In: *TAC* 7.8 (2009), p. 1.

[52] Hector Levesque, Ernest Davis, and Leora Morgenstern. “The winograd schema challenge”. In: *Thirteenth international conference on the principles of knowledge representation and reasoning*. 2012.

[53] V Sanh. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.” In: *Proceedings of Thirty-third Conference on Neural Information Processing Systems (NIPS2019)*. 2019.

[54] Brendan McMahan et al. “Communication-efficient learning of deep networks from decentralized data”. In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282.

[55] Mi Luo et al. “No fear of heterogeneity: Classifier calibration for federated learning with non-iid data”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 5972–5984.

[56] Duygu Nur Yaldiz, Tuo Zhang, and Salman Avestimehr. “Secure Federated Learning against Model Poisoning Attacks via Client Filtering”. In: *ICLR 2023 Workshop on Backdoor Attacks and Defenses in Machine Learning*.

 https://doi.org/10.52202/079017-3314

A Notation Table

Table 3: Notation table.

NOTATION	DESCRIPTION	NOTATION	DESCRIPTION
$\mathbf{w}$	Model parameter	$\mathbf{g}$	Gradients
D, D_0, D_i	Dataset	η_i	Aggregation weight
γ^t	Learning rate	S	Set of participation clients
N	Participating client size	M	Model dimension
$\mathbf{V}^i$, v^i_{jk}	Packed secret shares for gradients	TS_i	Trust score
l	# of secrets packed at a polynomial for gradient	p	# of secrets packed at a polynomial for shares of partial dot product
$\mathbf{cs}^i$, cs^i_j	Shares of partial cosine similarity	$\mathbf{nr}^i$, nr^i_j	Shares of partial gradient norm square
$\mathbf{S}^i$, s^i_{jk}	Packed secret shares of $\mathbf{cs}^j$ or $\mathbf{nr}^j$	$\tilde{\mathbf{h}}^i_k$	Secret shares disaggregated along packed index
x^i_k	Packed secret share of dot product	$\mathbf{h}^i_k$	Secret shares aggregated along packed index
e_i	Pre-determined secret point	α_i	Pre-selected elements for secret sharing
B_{e_j}	n by n matrix whose (i,k) entry is $(\alpha_k - e_j)^{i-1}$	$Chop_d$	n by n matrix whose (i,k) entry is 1 if $1 \leq i = k \leq d$ and 0 otherwise

B Details of Cryptographic Primitives

B.1 Packed Shamir Secret Sharing

The operations of Packed Shamir Secret Sharing performed on a finite field $\mathbb{F}_P$ for some prime number P. Denote $\{e_i\}_{i\in[l]}$ as the pre-determined secret point, and $\{\alpha_i\}_{i\in[d]}$ as the pre-selected elements for secret sharing. To share the secrets $\mathbf{g} = (g_1, g_2, ..., g_l)$, the user can generate a degree-d polynomial function:

$$\phi(x) = q(x)\Pi_{i=1}^{l}(x - e_i) + \sum_{i=1}^{l} g_1 L_i(x), \tag{14}$$

where $q(x)$ is a random degree-$d-l$ polynomial, and $L_i(x)$ is the Lagrange polynomial $\frac{\Pi_{j\neq i}(x-e_j)}{\Pi_{j\neq i}(e_i-e_j)}$.

The shares sent to player j is generated by:

$$s_j = \phi(\alpha_j). \tag{15}$$

We use $\langle \mathbf{g} \rangle_d$ to denote the degree-d packed secret shares of vector $\mathbf{g}$. The following properties holds for the packed sharing scheme:

- $\langle \alpha\mathbf{x} + \beta\mathbf{y} \rangle_d = \alpha\langle \mathbf{x} \rangle_d + \beta\langle \mathbf{y} \rangle_d$
- $\langle \mathbf{x} * \mathbf{y} \rangle_{d_1+d_2} = \langle \mathbf{x} \rangle_{d_1} * \langle \mathbf{y} \rangle_{d_2}$

B.2 Key Exchange

Diffie–Hellman key exchange protocol consists of the following algorithms:

- *Generate parameters*: $pp = \mathbf{GenParam}(sp)$ set up the parameters, including prime number and primitive root, according to the security parameter.
- *Key generation*: $(s_i^{SK}, s_i^{PK}) = \mathbf{KEGen}(pp)$ generates the private-public key pairs for user i.
- *Key derivation*: $s_{ij} = \mathbf{KEAgree}(s_i^{SK}, s_j^{PK})$ outputs the shared secret key between user i and j.

B.3 Symmetric Encryption

The smmetric encryption scheme consists of the following algorithms:

- *Encryption*: $c = \mathbf{Enc}(m, k)$ encrypts message m to cyphertext c using key k.
- *Decryption*: $m = \mathbf{Dec}(c, k)$ reverses cyphertext c to message m using key k.

To ensure correctness, we require that $m = \mathbf{Dec}(\mathbf{Enc}(m,k),k)$. For security, the encryption scheme should be indistinguishability under a chosen plaintext attack (IND-DPA) and integrity under ciphertext-only attack (INT-CTXT) [40].

B.4 Signature Scheme

The UF-CMA secure signature scheme that consists of a tuple of algorithms $(\mathbf{Gen}, \mathbf{Sign}, \mathbf{Verify})$:

- *Key generation*: Based on the security parameter sp, $(d^{SK}, d^{PK}) = \mathbf{SigGen}(sp)$ returns the private-public key pairs.
- *Signing algorithm*: $\sigma = \mathbf{Sign}(d^{SK}, m)$ generates a signature σ with secret key and message as input.
- *Signature verification*: $\mathbf{Verify}(d^{PK}, m, \sigma)$ takes as input the public key, a message and a signature, and returns 1 if the signature is valid and 0 otherwise.

To proof the security of the signature scheme, we show that no adversary can forge a valid signature on an arbitrary message. Denote a UF-CMA secure signature scheme as $\mathrm{DS} = (\mathrm{k}, \mathrm{Sign}, \mathrm{Verify})$, where k is the security parameter. The UF-CMA advantage of an adversary A is defined as $\mathrm{Adv}_{\mathrm{DS}}(\mathrm{A}, \mathrm{k}) = \mathbb{P}(\mathrm{Exp}_{\mathrm{DS}}^{\mathrm{uf-cma}}(\mathrm{A}, \mathrm{k}) = 1)$, where $\mathrm{Exp}_{\mathrm{DS}}^{\mathrm{uf-cma}}(\mathrm{A}, \mathrm{k})$ represents the experiments conducted by adversary A to produce a signature, and $\mathrm{Exp}_{\mathrm{DS}}^{\mathrm{uf-cma}}(\mathrm{A}, \mathrm{k}) = 1$ means that A produced a valid signature. In a UF-CMA secure signature scheme, no probabilistic polynomial time (PPT) adversary is able to produce a valid signature on an arbitrary message with more than negligible probability. In other words, for all PPT adversaries A, there exists a negligible function ϵ such that $\mathrm{Adv}_{\mathrm{DS}}(\mathrm{A}, \mathrm{k}) \leq \epsilon(\mathrm{k})$.

C Comparison between Byzantine-robust aggregation rules

To provide justification for our algorithm's utilization of FLTrust as the aggregation rule, we summarize the existing Byzantine-robust aggregation rules along four dimensions: (i) computation complexity, (ii) whether the algorithm needs prior knowledge about the number of poisoners, (iii) maximum number of poisoners, (iv) whether the algorithm is compatible with Shamir Secret Sharing (SSS).

Table 4: Comparison between Byzantine-robust aggregation rules.

	Computation complexity	**Need prior knowledge about # of poisoners**	**# of poisoners**	**Compatible with SSS**
KRUM	$O(N^2(M + \log N))$	Yes	$< 50\%$	Yes
Bulyan	$O(N^2M)$	Yes	$< 25\%$	No
Trim-mean	$O(MN \log N)$	Yes	$< 50\%$	No
FLTrust	$\mathbf{O(MN)}$	**No**	$\mathbf{< 100\%}$	**Yes**

Among these dimensions, FLTrust demonstrates clear advantages over other robust aggregation rules:

- *Low computation cost:* for a system with N users and M model size, the computation cost of FLTrust is $O(MN)$, lower than existing methods that grow quadratically with N.
- *No need of prior knowledge about number of poisoners:* the server does not need to know the number of malicious clients in advance to conduct robust aggregation.
- *Defend against majority number of poisoners:* benefiting from the trusted root of clean dataset at the server, the aggregation rule we adopted can return robust result even when the number of poisoners is above 50%.
- *Compatible with Shamir Secret Sharing (SSS):* the method we adopted is compatible with the SSS algorithm. While for Bulyan and Trim-mean, there are some non-linear operations not supported by SSS.

https://doi.org/10.52202/079017-3314

D Algorithm of RFLPA

This section presents the our algorithm to conduct robust federated learning with secure aggregation.

Algorithm 1 RFLPA

Input: Local dataset D_i of clients $i \in [N]$, root dataset D_0 at server, number of iterations T, security parameter κ.
Output: Global model $\mathbf{w}^T$
Clients set up encryption and signature key pairs (c_i^{PK}, c_i^{SK}), $(d_i^{PK}, d_i^{SK}) \leftarrow$ SetupKeys(N, κ) for $i \in [N]$.
Server initialize global model $\mathbf{w}^0$
for $t \in [1, T]$ **do**
  Server conduct local update with root data, compute update norm $\|\mathbf{g}_0\|$, and create packed secret shares $\mathbf{v}_0$.
  Each clients from $\mathcal{U}_t$ download global model $\mathbf{w}^{t-1}$, corresponding shares of $\mathbf{v}_0$, and $\|\mathbf{g}_0\|$.
  Server obtain gradients $\mathbf{g} \leftarrow$ RobustSecAgg($\mathcal{U}_t$, $\mathbf{w}^{t-1}$, $\mathbf{v}_0$, $\|\mathbf{g}_0\|$)
  Server update global model $\mathbf{w}^t \leftarrow \mathbf{w}^{t-1} - \gamma^t \mathbf{g}$
end for

Algorithm 2 SetupKeys

Input: number of clients N, security parameter κ.
Output: key pairs $\{(d_i^{PK}, d_i^{SK})\}_{i\in[N]}$;
secret keys $\{k_{ij}\}_{i,j\in[N]}$.
Each user $i \in [N]$ receive their signing key d_i^{SK} from the trusted third party, as well as the verification keys d_j^{PK} of all users $j \in [N]$.
Each user $i \in [N]$ generate key pairs $(s_i^{SK}, s_i^{PK}) = \mathbf{KEGen}(sp)$, and create signature $\sigma_i = \mathbf{Sign}(d_i^{PK}, s_i^{PK})$.
Users $i \in [N]$ send $(s_i^{PK}||\sigma_i)$, public key along with signature, to the server.
Server distribute $\{(s_i^{PK}||\sigma_i)\}_{i\in[N]}$ to all users.
Each user i asserts that $\mathbf{Verify}(d^{PK}, s_j^{PK}, \sigma_j) = 1$, and compute $k_{ij} = KEAgree(s_i^{SK}, s_j^{PK})$ for $j \in [N]\backslash i$.

Suppose that user i create a packed secret shares $\mathbf{s}$ of $\mathbf{g}$ with polynomial $\phi(x)$. Providing κ security, the user sets up generator ψ and secret key α, and also outputs the public key $(\psi, \psi^\alpha, ..., \psi^{\alpha^d})$ for a degree d polynomial. To make the secret shares verifiable, the user broadcasts a commitment to the function:

$$\mathcal{C} = \psi^{\phi(\alpha)}. \tag{16}$$

E Verifiable Packed Secret Sharing

For each secret s_l, user i computes a witness sent to the corresponding client in a private channel:

$$w_l = \psi^{(\phi(\alpha)-\phi(l))/(\alpha-l)}. \tag{17}$$

After receiving the commitment and witness, user l can verify the secret by checking:

$$e(\mathcal{C}, \psi) = e(w_l, \psi^\alpha/\psi^l)e(\psi, \psi)^{\phi(l)}, \tag{18}$$

where $e(\cdot)$ denotes a symmetric bilinear pairing.

The correctness and secrecy of the protocol are guarantee by the discrete logarithm (DL) [41], t-polynomial Diffie-Hellman (t-polyDH) [29], and t-Strong Diffie-Hellman (t-SDH) [42] assumptions.

F Explanation of Secret Re-sharing

For $m \in [\lceil N/p \rceil]$, the shares of secret $cs^i_{(m-1)p+k}$ for some $k \in [p]$ can be represented as:

$$\left(s^i_{1m} \ \dots \ s^i_{Nm}\right) = \left(cs^i_{(m-1)p+k} \ \theta_1 \ \dots \ \theta_d \ 0 \ \dots \ 0\right) \times B_{e_k}, \tag{19}$$

Algorithm 3 RobustSecAgg

Input: Set of active clients in current iteration $\mathcal{U}_0$, global parameters $\mathbf{w}$ downloaded from server, packed secret shares of server update $\mathbf{v}_0$, norm of server update $\|\mathbf{g}_0\|$.
Output: Global aggregated gradient $\mathbf{g}$
Round 1:
Client i:
• Generate local gradient $\mathbf{g}_i$
• Generate packed secrets $\{\mathbf{v}_{ij}\}_{j\in\mathcal{U}_0}$, commitments $\mathcal{C}$ and witness $\{\omega_{ij}\}_{j\in\mathcal{U}_0}$ for $\mathbf{g}_i$ from 15, 16, and 17, encrypt $\mathbf{c}_{ij} = \mathbf{Enc}(\mathbf{v}_{ij}||\omega_{ij}, k_{ij})$, and create signature $\sigma_{ij} = \mathbf{Sign}(d_i^{SK}, \mathbf{c}_{ij}||\mathcal{C})$ for $j \in [N]\backslash i$
• Send $(\mathcal{C}||\{\mathbf{c}_{ij}\}_{j\in[N]\backslash i}||\{\sigma_{ij}\}_{j\in[N]\backslash i})$ to the server
Server:
• Collect messages from at least K clients (denote $\mathcal{U}_1$ the set of all respondents).
• Send $(\mathcal{C}||\{\mathbf{c}_{ij}\}_{i\in\mathcal{U}_1\backslash j}||\{\sigma_{ij}\}_{i\in\mathcal{U}_1\backslash j})$ to client j for $j \in \mathcal{U}_1$.
Round 2:
Client i:
• Receive $(\mathcal{C}||\{\mathbf{c}_{ji}\}_{j\in\mathcal{U}_1\backslash i}||\{\sigma_{ji}\}_{j\in\mathcal{U}_1\backslash i})$ from server, and assert that $\mathbf{Verify}(d_j^{PK}, \mathbf{c}_{ji}||\mathcal{C}, \sigma_{ji}) = 1$.
• Recover $(\{\mathbf{v}_{ji}\}_{j\in\mathcal{U}_1\backslash i}, \{\omega_{ji}\}_{j\in\mathcal{U}_1\backslash i}) = \mathbf{Dec}(\mathbf{c}_{ji}, k_{ji})$, and verify the secret shares $\{\mathbf{v}_{ji}\}_{j\in\mathcal{U}_1\backslash i}$ by testing 18.
• Compute local shares of partial norm $\{nr_j^i\}_{j\in\mathcal{U}_1}$ and partial cosine similarity $\{cs_j^i\}_{j\in\mathcal{U}_1}$ from 8.
• Construct packed secret shares $\{\mathbf{s}_{ik}\}_{k\in\mathcal{U}_1}$, commitments $\mathcal{C}$, and witness $\{\omega'_{ik}\}_{k\in\mathcal{U}_1}$ for $(\{nr_j^i\}_{j\in\mathcal{U}_1}, \{cs_j^i\}_{j\in\mathcal{U}_1})$, encrypt $\mathbf{c}'_{ik} = \mathbf{Enc}(\mathbf{s}_{ik}||\omega'_{ik}, k_{ik})$, and create signature $\sigma'_{ik} = \mathbf{Sign}(d_i^{SK}, \mathbf{c}'_{ik}||\mathcal{C})$ for $k \in [N]\backslash i$
• Send $(\mathcal{C}||\mathbf{c}'_{ij}||\sigma'_{ij})$ for $j \in [N]\backslash i$ to the server
Server:
• Collect messages from at least K clients (denote $\mathcal{U}_2$ the set of all respondents).
• Send $(\mathcal{C}||\{\mathbf{c}'_{ij}\}_{i\in\mathcal{U}_2\backslash j}||\{\sigma'_{ij}\}_{i\in\mathcal{U}_2\backslash j})$ to client j for $j \in \mathcal{U}_2$.
Round 3:
Client i:
• Receive $(\mathcal{C}||\{\mathbf{c}'_{ji}\}_{j\in\mathcal{U}_2\backslash i}||\{\sigma'_{ji}\}_{j\in\mathcal{U}_2\backslash i})$ from server, and assert that $\mathbf{Verify}(d_j^{PK}, \mathbf{c}'_{ji}||\mathcal{C}, \sigma'_{ji}) = 1$.
• Recover $(\{\mathbf{s}_{ji}\}_{j\in\mathcal{U}_2\backslash i}, \{\omega'_{ji}\}_{j\in\mathcal{U}_2\backslash i}) = \mathbf{Dec}(\mathbf{c}'_{ji}, k_{ji})$, and verify the secret shares $\{\mathbf{s}'_{ji}\}_{j\in\mathcal{U}_2\backslash i}$ by testing 18.
• Obtain the final share of norm $\{\overline{nr}_j^i\}_{j\in|\mathcal{U}_1|/p}$ and cosine similarity $\{\overline{cs}_j^i\}_{j\in|\mathcal{U}_1|/p}$ from 10, 11, and Reed-Solomon decoding.
• Send $(\{\overline{nr}_j^i\}_{j\in|\mathcal{U}_1|/p}, \{\overline{cs}_j^i\}_{j\in|\mathcal{U}_1|/p}$ to the server.
Server:
• Collect messages from at least K clients (denote $\mathcal{U}_3$ the set of all respondents).
• Recover $\{\|\mathbf{g}_j\|^2\}_{j\in\mathcal{U}_1}$ using Reed-Solomon decoding, and assert that $\|\mathbf{g}_j\|^2 \leq \|\mathbf{g}_0\|^2, \forall j \in \mathcal{U}_1$.
• Recover $\{\langle\bar{\mathbf{g}}_i, \mathbf{g}_0\rangle\}_{j\in\mathcal{U}_1}$ using Reed-Solomon decoding, and compute the trust score $\{TS_j\}_{j\in\mathcal{U}_1}$ from 5.
• Broadcast the trust score $\{TS_j\}_{j\in\mathcal{U}_1}$ to all users $i \in \mathcal{U}_3$.
Round 4:
Client i:
• Compute local aggregation $\langle\mathbf{g}\rangle_i$ from 6, and send to the server.
Server:
• Collect messages from at least K clients.
• Recover $\mathbf{g}$ using Reed-Solomon decoding.

where $\{\theta_j\}_{j\in[d]}$ are random integers.

https://doi.org/10.52202/079017-3314

Hence, the user side computation of 10 is the same as:

$$\begin{pmatrix} s^1_{1,m} & \cdots & s^N_{1,m} \\ \vdots & \ddots & \vdots \\ s^1_{Nm} & \cdots & s^N_{Nm} \end{pmatrix} B^{-1}_{e_j} Chop_d B_{e'_j} = B^T_{e_k} \\ \times \begin{pmatrix} cs^1_{(m-1)p+k} & \cdots & cs^N_{(m-1)p+k} \\ \vdots & \ddots & \vdots \end{pmatrix} B^{-1}_{e_j} Chop_d. \tag{20}$$

The aggregation of new secret and reconstruction of $\{x^j_m\}$ is equivalent to taking the first column of:

$$B^T_{e_k} \begin{pmatrix} cs^1_{(m-1)p+k} & \cdots & cs^N_{(m-1)p+k} \\ \vdots & \ddots & \vdots \end{pmatrix} \\ \times \left(B^{-1}_{e_1} + \cdots + B^{-1}_{e_l}\right) Chop_d. \tag{21}$$

Since $\mathbf{cs}^j$ is a packed secret share of the partial cosine similarity, it follows that:

$$\left(cs^1_h \ \ldots \ cs^N_h\right) B^{-1}_{e_j} Chop_d = \left(\textstyle\sum_{(j-1)l<i\leq jl} \bar{g}_{hi} g_{0i} \ \ldots\right), \tag{22}$$

meaning that the first elements gives the partial cosine similarity.

Therefore, the final shares sent to server $\{x^j_m\}$ can be formulated as:

$$\left(x^1_m \ \ldots \ x^N_m\right) = \left(\textstyle\sum_i \bar{g}_{m(p-1)+h,i} g_{0i} \ \theta_1 \ \ldots \ \theta_d \ 0 \ \ldots\right) B_{e_h}, \tag{23}$$

for $h \in (m(p-1), mp]$. Therefore, the server could retrieve the dot product by Reed-Solomon decoding, which is equivalent to multiplying $\{B^{-1}_{e_h}\}_{h\in(m(p-1),mp]}$ and obtaining the first element.

G Details of Complexity Analysis

User computation: User's computation cost can be broken as: (1) generating packed secret shares of update ($O(M+N)\log^2 N$) complexity [43]); (2) computing shares of partial gradient norm square and cosine similarity ($O(M+N)$) complexity); (3) creating packed secret shares of partial gradient norm square and cosine similarity ($O(N\log^2 N)$ complexity); (4) deriving final secret shares of gradient norm square and cosine similarity ($O(N^2\log^2 N)$ complexity). Therefore, each user's computation cost is $O((M+N^2)\log^2 N)$.

User communication: User's communication cost can be broken as: (1) downloading parameters from server ($O(M)$ messages); (2) sending and receiving secret shares of gradient ($O((M,N))$ messages); (3) sending and receiving secret shares of partial gradient norm square and cosine similarity ($O(N)$ messages); (4) sending final shares of gradient norm square and cosine similarity ($O(1)$ messages); (5) receiving trust scores from the server ($O(N)$ messages); (6) sending shares of aggregated update to the server ($O(M/N+1)$ messages). Hence, each user's communication cost is $O(M+N)$.

Server computation: The server's computation cost can be broken as: (1) recovering gradient norm square and cosine similarity by Reed-Solomon decoding ($O(N\log^2 N\log\log N)$ complexity [44]); (2) computing the trust score of each user ($O(N)$ complexity); (3) decoding the aggregated global gradient ($O(M+N)\log^2 N\log\log N$) complexity). Therefore, the server's computation cost is $O((M+N)\log^2 N\log\log N)$.

Server communication: The server's communication cost can be broken as: (1) distributing parameters to clients ($O(MN)$ messages); (2) sending and receiving secret shares of user update ($O((M+N)N\}$) messages); (3) sending and receiving secret shares of partial gradient norm square and cosine similarity ($O(N^2)$ messages); (4) receiving final shares of gradient norm square and cosine similarity ($O(N)$ messages); (5) broadcasting trust scores to clients ($O(N^2)$ messages); (6) receiving shares of aggregated update from clients ($O(M+N)$ messages). Overall, the server's communication cost is $O((M+N)N)$.

H Proof of Theorem 5.1

Proof. We utilize the standard hybrid argument to prove the theorem. we define a PPT simulator SIM through a series of (polynomially many) subsequent to $\mathrm{REAL}_{\mathcal{C}}^{\mathcal{U},t,\kappa}$, so that the view of $\mathcal{C}$ in SIM is computationally indistinguishable from that in $\mathrm{REAL}_{\mathcal{C}}^{\mathcal{U},t,\kappa}$.

Hyb_1: In the hybrid, each honest user from $\mathcal{U}_1 \backslash \mathcal{C}$ encrypts shares of a uniformly random vector, instead of the raw gradients. The properties of Shamir's secret sharing ensure that the distribution of any $|\mathcal{C} \backslash \{S\}| < t$ shares of raw gradients is identical to that of any equivalent length vector, and IND-CPA security guarantees that the view of server is indistinguishable in both cases. Hence, this hybrid is identical from the previous one.

Hyb_2: In the hybrid, the simulator aborts if $\mathcal{C}$ provides any of the honest user i with a signature on j's message, $\mathbf{c}_{ji}$, but the user couldn't produce the same signature given the public key (in round 2). The security of the signature scheme guarantees that this hybrid is indistinguishable from the previous one.

Hyb_3: In this hybrid, SIM aborts if any of the honest user i fails to verify the secret shares $\mathbf{s}_{ji}$ from user j by checking 18. The the DL, t-polyDH, and t-SDH assumptions guarantee that this hybrid is identical from the previous one.

Hyb_4: In the hybrid, each honest user from $\mathcal{U}_2 \backslash \mathcal{C}$ encrypts shares of a uniformly random vector rather than partial norm and cosine similarity. The properties of Shamir's secret and IND-CPA security ensure that this hybrid is indistinguishable from the previous one.

Hyb_5: In the hybrid, the simulator aborts if $\mathcal{C}$ provides any of the honest user i with a signature on j's message, $\mathbf{c}'_{ji}$, but the user couldn't produce the same signature given the j's key (in round 3). Because of the security of the signature scheme, this hybrid is indistinguishable from the previous one.

Hyb_6: This hybrid is defined as Hyb_3, with the only difference that SIM verify the secret shares $\mathbf{s}'_{ji}$ in round 3. This hybrid is indistinguishable from the previous one under DL, t-polyDH, and t-SDH assumptions.

The above changes do not modify the views seen by the colluding parties, and the hybrid doesn't make use of the honest users' input. Therefore, the output of SIM is computationally indistinguishable from the output of $\mathrm{REAL}_{\mathcal{C}}^{\mathcal{U},t,\kappa}$, and this concludes the proof.

□

I Proof of Theorem 5.2

Denote $\bar{\mathbf{g}}^t = \sum_i \eta_i \bar{\mathbf{g}}_i$ be the aggregated gradients at iteration t.

Lemma I.1. *For arbitrary number of adversarial clients, the distance between $\bar{\mathbf{g}}^t$ and $\nabla F(\mathbf{w}^t)$ is bounded by:*

$$\|\bar{\mathbf{g}}^t - \nabla F(\mathbf{w}^t)\| \leq 3\|\mathbf{g}_0^t - \nabla F(\mathbf{w}^t)\| + 2\|\nabla F(\mathbf{w}^t)\| + \frac{\sqrt{d}}{q}. \tag{24}$$

https://doi.org/10.52202/079017-3314

Proof. It follows that:

$$\begin{aligned}
\|\bar{\mathbf{g}}^t - \nabla L^t(\mathbf{w})\| &= \|\sum_i \eta_i \bar{\mathbf{g}}_i - \nabla F^t(\mathbf{w})\| \\
&= \|\sum_i \eta_i \bar{\mathbf{g}}_i - \bar{\mathbf{g}}_0 + \bar{\mathbf{g}}_0 - \mathbf{g}_0 + \mathbf{g}_0 - \nabla F^t(\mathbf{w})\| \\
&\leq \|\sum_i \eta_i \bar{\mathbf{g}}_i - \bar{\mathbf{g}}_0\| + \|\bar{\mathbf{g}}_0 - \mathbf{g}_0\| + \|\mathbf{g}_0 - \nabla F^t(\mathbf{w})\| \\
&\leq \sum_i \eta_i \|\bar{\mathbf{g}}_i\| + \|\bar{\mathbf{g}}_0\| + \|\bar{\mathbf{g}}_0 - \mathbf{g}_0\| + \|\mathbf{g}_0 - \nabla F^t(\mathbf{w})\| \\
&\overset{(a)}{\leq} 2\|\mathbf{g}_0\| + \frac{\sqrt{d}}{q} + \|\mathbf{g}_0 - \nabla F^t(\mathbf{w})\| \\
&\leq 3\|\mathbf{g}_0 - \nabla F^t(\mathbf{w})\| + 2\|\nabla F^t(\mathbf{w})\| + \frac{\sqrt{d}}{q},
\end{aligned} \tag{25}$$

where (a) is because $\sum_i \eta_i = 1$, $\|\bar{\mathbf{g}}_i\| \leq \|\mathbf{g}_0\|$, and$\|\bar{\mathbf{g}}_0\| \leq \|\mathbf{g}_0\|$. □

Lemma I.2. *Under Assumption J.1, we have the following bound at iteration t:*

$$\|\mathbf{w}^t - \mathbf{w}^* - \gamma \nabla F(\mathbf{w}^t)\| \leq \sqrt{1 - \mu^2/(4L_g^2)}\|\mathbf{w}^t - \mathbf{w}^*\|. \tag{26}$$

Proof. Refer to lemma 2 in [15] for the proof. □

Lemma I.3. *Suppose Assumption J.1, J.2, J.3 holds. For any $\delta \in (0,1)$, if $\Delta_1 \leq \nu_1^2/\alpha_1$, $\Delta_2 \leq \nu_2^2/\alpha_2$, we have:*

$$P\left\{\|\mathbf{g}_0 - \nabla F(\mathbf{w})\| \leq 8\Delta_2\|\mathbf{w} - \mathbf{w}^* + 4\Delta_1\|\right\} \geq 1 - \delta, \tag{27}$$

for any $\mathbf{w} \in \Theta \subset \left\{\mathbf{w} : \|\mathbf{w} - \mathbf{w}^*\| \leq r\sqrt{d}\right\}$ given some positive number r.

Proof. Refer to lemma 4 in [15] for the proof. □

Proof of Theorem 5.2: Given the lemmas above, we can proceed to prove Theorem 5.2. We have:

$$\begin{aligned}
\|\mathbf{w} - \mathbf{w}^*\| &\leq \|\mathbf{w}^{t-1} - \gamma \nabla F(w^{t-1}) - \mathbf{w}^*\| + \gamma\|\bar{\mathbf{g}}^t - \nabla F(\mathbf{w}^t)\| \\
&\leq \|\mathbf{w}^{t-1} - \gamma \nabla F(w^{t-1}) - \mathbf{w}^*\| + 3\gamma\|\mathbf{g}_0^t - \nabla F(\mathbf{w}^t)\| \\
&\quad + 2\gamma\|\nabla F(\mathbf{w}^t)\| + \frac{\gamma\sqrt{d}}{q} \\
&\leq \left(\sqrt{1 - \mu^2/(4L^2)} + 24\gamma\Delta_2 + 2\gamma L\right)\|\mathbf{w}^{t-1} - \mathbf{w}^*\| \\
&\quad + 12\gamma\Delta_1 + \frac{\gamma\sqrt{d}}{q}.
\end{aligned} \tag{28}$$

Therefore, with probability at least $1 - \delta$, it follows that:

$$\|\mathbf{w}^t - \mathbf{w}^*\| \leq (1 - \rho)^t\|\mathbf{w}^0 - \mathbf{w}^*\| + +12\gamma\Delta_1 + \frac{\gamma\sqrt{d}}{q}. \tag{29}$$

J Assumptions for convergence analysis 5.4

Assumption J.1. The expected risk function $F(\mathbf{w})$ is μ-strongly convex and L-smooth for any $\mathbf{w}$, $\bar{\mathbf{w}}$:

$$\begin{aligned}
F(\bar{\mathbf{w}}) &\geq F(\mathbf{w}) + \langle \nabla F(\mathbf{w}), \bar{\mathbf{w}} - \mathbf{w}\rangle + \frac{\mu}{2}\|\bar{\mathbf{w}} - \mathbf{w}\|^2 \\
\|\nabla F(\mathbf{w}) - \nabla F(\bar{\mathbf{w}})\| &\leq L\|\bar{\mathbf{w}} - \mathbf{w}\|.
\end{aligned} \tag{30}$$

Moreover, the empirical loss function $L(D, \mathbf{w})$ is L_1-smooth probabilistically. For any $\delta \in (0, 1)$, there exists an L_1 such that:

$$P\left\{\sup_{\mathbf{w}\neq\bar{\mathbf{w}}} \frac{\|\nabla L(D, \mathbf{w}) - \nabla L(D, \bar{\mathbf{w}})\|}{\|\mathbf{w} - \bar{\mathbf{w}}\|} \leq L_1\right\} \geq 1 - \frac{\delta}{3}. \tag{31}$$

Assumption J.2. The root dataset D_0 and clients' local dataset $D_i (i = 1, 2, ..., n)$ are sampled independently from distribution χ.

Assumption J.3. The gradients of the empirical loss function $\nabla L(D, \mathbf{w}^*)$ at the optimal model $\mathbf{w}^*$ is bounded. Furthermore, $h(D, \mathbf{w}) = \nabla L(D, \mathbf{w}) - \nabla L(D, \mathbf{w}^*)$ is also bounded. Specifically, $\langle \nabla L(D, \mathbf{w}^*), \mathbf{v}\rangle$ and $\langle h(D, \mathbf{w}) - \mathbb{E}[h(D, \mathbf{w})], \mathbf{v}\rangle / \|\mathbf{w} - \mathbf{w}^*\|$ are sub-exponential for any unit vector $\mathbf{v}$. Formally, for $\forall|\lambda| \leq 1/\alpha_1$, $\forall|\lambda| \leq 1/\alpha_2$, $\mathbf{B} = \{\mathbf{v} : \|v\| = 1\}$, it holds that:

$$\begin{aligned} &\sup_{\mathbf{v}\in\mathbf{B}} \mathbb{E}[\exp(\lambda\langle \nabla L(D, \mathbf{w}^*), \mathbf{v}\rangle)] \leq e^{\nu_1^2\lambda^2/2} \\ &\sup_{\mathbf{v}\in\mathbf{B},\mathbf{w}} \mathbb{E}\left[\exp\left(\frac{\langle h(D, \mathbf{w}) - \mathbb{E}[h(D, \mathbf{w})], \mathbf{v}\rangle}{\|\mathbf{w} - \mathbf{w}^*\|}\right)\right] \leq e^{\nu_2^2\lambda^2/2}. \end{aligned} \tag{32}$$

K Experiments

The experiments are conducted on a 16-core Ubuntu Linux 20.04 server with 64GB RAM and A6000 driver, where the programming language is Python.

K.1 Datasets

MNIST is a collection of handwritten digits, including 60,000 training and 10,000 testing images of 28×28 pixels. F-MNIST consists of 70,000 fashion images of size 28×28 and is split into 60,000 training and 10,000 testing samples. CIFAR-10 is natural dataset that includes 60,000 32×32 colour images in 10 classes, splitting into 50,000 training and 10,000 testing images.

K.2 FL configuration

both datasets are split among 10,000 users and select 100 users in each iteration. The server stores 200 clean samples as benchmark. We allow up to 20% clients to drop out in each round, and a maximum of 30% participating clients to collaborate with each other to reveal the secret. Therefore, we construct a secret sharing of degree 40, considering the doubling of degree during dot product computation, and pack each 10 elements into a secret.

K.3 Hyper-Parameters

The parameters are updated using Adaptive Moment Estimation (Adam) method with a learning rate of 0.01. Each accuracy reported in the tables is an average of 5 experiments, and each round of experiments runs for 200 iterations. Both LDP and CDP adopt privacy parameter $\epsilon = 3$ and $\delta = 0.0001$.

K.4 Comparison among Aggregation Frameworks

In Table 5 we summarize the comparison among aggregation frameworks along four dimensions:

- *Robustness against malicious users:* most algorithms provide certain level of robustness against malicious users. Local DP is not that effective in defending malicious users according to our experiment results. Though Robust Federated Aggregation (RFA) [45] provides a robust aggregation protocol based on geometric median, the malicious users could freely manipulate the uploaded gradients for poisoning attacks.
- *Privacy Protection against server:* whether the framework protect user's plaintext gradient against server. Only PEFL, PBFL, ShieldFL, SecureFL [46], RoFL [47], ELSA [48], BREA, and RFLPA achieves the goals of robustness and privacy simultaneously.

 https://doi.org/10.52202/079017-3314

- *Collusion threshold during model training:* the server could obtain users' plaintext gradients if it colludes with more than the given level of parties. PEFL, PBFL, ShieldFL, SecureFL, and ELSA all rely on two non-colluding parties during model training to protect users' message. The collaboration between the two non-colluding parties could compromise user's privacy.
- *MPC techniques:* the main multiparty computation techniques leveraged by the framework. PEFL, PBFL, ShieldFL, SecureFL, and ELSA are based on multi-party computation (MPC) or homomorphic encryption (HE), RoFL is based on zero-knowledge proof (ZKP), and BREA and RFLPA are based on secret sharing.

Furthermore, although RoFL and ELSA could defend against malicious users, they are designed specifically for a naive robust aggragation method, norm bounding. It's completely impractical to generalize these frameworks to more advance defense method such as Krum.

Table 5: Corse-grained comparison among Aggregation Frameworks. "/" denotes non-applicable. ELSA improves on RoFL regarding the the efficiency.

	Robustness against malicious users	Privacy Protection against server	Collusion threshold during model training	MPC techniques
FedAvg	Yes	No	/	/
Bulyan	Yes	No	/	/
Trim-mean	Yes	No	/	/
KRUM	Yes	No	/	/
Central DP	Yes	No	/	/
Local DP	Not effective	Yes	/	/
RFA	No	Yes	/	/
PEFL	Yes	Yes	1	HE (Paillier)
PBFL	Yes	Yes	1	HE (CKKS)
ShieldFL	Yes	Yes	1	HE (Paillier)
SecureFL	Yes	Yes	1	MPC & HE (BFV)
RoFL	Yes	Yes	$O(N)$	ZKP
ELSA	Yes	Yes	1	MPC
BREA	Yes	Yes	$O(N)$	Secret sharing
RFLPA	Yes	Yes	$O(N)$	Secret sharing

K.5 Accuracies over Iterations

Figure 4 demonstrates the impact of different iterations on test accuracies for RFLPA, BREA and FedAvg using the MNIST dataset. The results reveal that the RFLPA algorithm displays comparable convergence regardless of the existence of attackers, while FedAvg exhibits significantly inferior convergence when 30% attackers are present.

K.6 Performance on Additional Attacks

K.6.1 Poisoning Attacks

We evaluate our protocol against several stealthier attacks: (1) KRUM attack [49], (2) BadNets [50], and (3) Scaling attack [10]. KRUM attack is untarget attack, and BadNets as well as Scaling attack are backdoor attacks that specifically degrade the performance on triggered samples. We follow the same approach as in [50] and [10] to embed triggers in the targeted images.

Table 6 compares the performance of RFLPA and FedAvg against the above attacks. For KRUM attack, RFLPA improves the accuracy on the general dataset over FedAvg by more than 1.6x. For the two backdoor attacks, RFLPA show trivial performance loss on the general and triggered dataset, as opposed to the significant degradation in accuracy for FedAvg.

K.6.2 Inference Attacks

We assess our RFLPA against passive inference attack using the Deep Leakage from Gradients (DLG) [8]. It is important to note that experiments were not conducted for active inference attacks, where

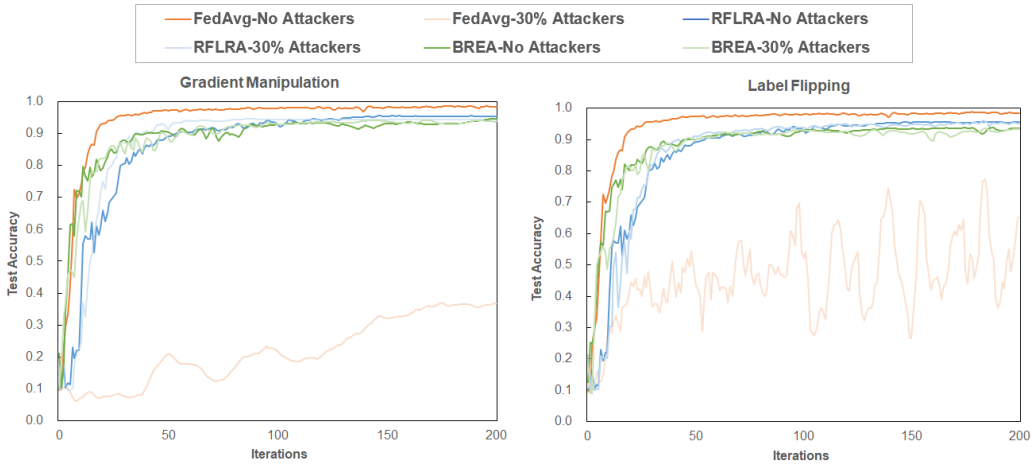

Figure 4: Test accuracy of RFLPA and FedAvg for different proportions of malicious users on MNIST dataset.

Table 6: Accuracies on CIFAR-10 under varying proportions of attackers. For backdoor attacks, the values are presented as *overall accuracy (backdoor accuracy)*.

	FedAvg			RFLPA		
% of attackers	10%	20%	30%	10%	20%	30%
KRUM attack	0.27	0.12	0.11	0.71	0.70	0.70
BadNets	0.68 (0.54)	0.67 (0.54)	0.55 (0.28)	0.71 (0.68)	0.70 (0.68)	0.69 (0.66)
Scaling attack	0.70 (0.22)	0.68 (0.21)	0.54 (0.19)	0.70 (0.69)	0.70 (0.69)	0.69 (0.69)

the server might alter users' messages, such as secret shares, to access private data. This omission is due to the protection provided by the signature scheme, which safeguards message integrity and prevents the server from forging any user's messages.

DLG attempts to reconstruct the original image from the aggregated gradients. We conducted an attack on the CIFAR-10 dataset, using the specifications in Appendix K.2. The average peak signal-to-noise ratio (PSNR) of generated image with respect to original image is 11.27, much lower than the value of 36.5 when no secure aggregation is involved. Figure 5 shows that the inferred images are far from the raw images under DLG attack.

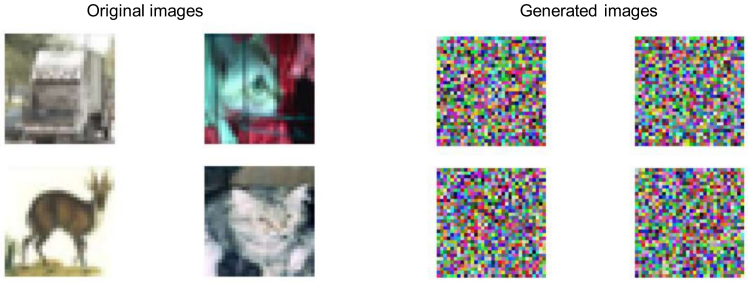

Figure 5: Original and inferred image under RFLPA.

https://doi.org/10.52202/079017-3314

K.7 Performance on Diverse Dataset

K.7.1 Performance on Natural Language Processing (NLP) Dataset

We evaluate the accuracy of our framework on two NLP datasets, Recognizing Textual Entailment (RTE) [51] and Winograd NLI (WNLI) [52], by finetuning a distillBERT model [53]. We present the performance for gradient manipulation attack in Table 7. The result demonstrates that for the two NLP datasets, RFLPA has robust accuracies in the presence of up to 30% attackers.

Table 7: Accuracies on NLP dataset under different proportions of attackers.

	RTE				WNLI			
Proportion of Attackers	No	10%	20%	30%	No	10%	20%	30%
FedAvg	**0.599**	0.509	0.487	0.462	**0.619**	0.563	0.437	0.437
BREA	0.584	**0.592**	0.570	0.567	0.592	**0.592**	0.577	0.563
RFLPA	0.596	0.582	**0.582**	**0.577**	0.619	0.592	**0.592**	**0.563**

K.7.2 Performance on CIFAR-100 Dataset

To test a more complex CV dataset, we evaluate our frameworks on CIFAR-100 [32] dataset using a ResNet-9 classifier. It can be observed in Figure 8 that RFLPA significantly enhances the accuracy over FedAvg from 10% attackers, by an average of 3.94x. Furthermore, RFLPA experiences little performance degradation in the presence of up to 30% attackers.

Table 8: Accuracy on CIFAR-100 dataset under gradient manipulation attack.

	FedAvg				RFLPA			
% of attackers	No	10%	20%	30%	No	10%	20%	30%
Accuracy	0.55 $\pm$0.2	0.11 $\pm$0.5	0.10 $\pm$0.0	0.10 $\pm$0.0	0.55 $\pm$0.2	0.54$\pm$0.1	0.50 $\pm$0.1	0.49 $\pm$0.2

K.8 Overhead Analysis

K.8.1 Computation Time between RFLPA and HE-based methods

To verify the practicability of RFLPA, we benchmark our framework with three HE-based methods, PEFL [20], PBFL [21], and ShieldFL [22]. Table 9 presents the per-iteration computation time using a MNIST classifier (1.6M parameters) for the three algorithms and RFLPA. It can be observed that it takes 1.5 to 6.5 day to run the three HE-based algorithms for only a single iteration, which renders them impractical for real-life deployment.

Table 9: Computation cost (in minutes) with varying client size.

	Per-user Cost				Server Cost			
Client size	100	200	300	400	100	200	300	400
RFLPA	3.41	11.44	24.51	42.60	6.68	8.46	15.00	26.47
PEFL	111.51	109.27	109.44	110.13	2156.20	6056.98	6785.71	9365.46
PBFL	12.65	12.58	12.73	12.63	1806.05	3598.54	5386.97	7193.64
ShieldFL	111.73	109.43	109.25	109.84	2192.48	6093.05	6809.60	9384.11

K.8.2 Ablation Study

Considering that RFLPA and BREA leverage different robust aggregation rule, we conducted ablation study to demonstrate that the reduction in overhead is attributed to the scheme design of RFLPA rather than the inherent advantages of the underlying aggregation rule. In particular, we replace the aggregation module in RFLPA with KRUM, and presents the per-iteration communication and computation cost, respectively, in Table 10 and 11. It can be observed that even with substituting the aggregation module with KRUM in our framework, there's still notable reduction in the communication cost benefiting from the design of our secret sharing algorithm.

Table 10: Communication cost (in MB) per client with varying client size with MNIST classifier (1.6M parameters). RFLPA (KRUM) replaces the aggregation rule with KRUM in RFLPA.

Client size	300	400	500	600
RFLPA	82.51	82.52	82.53	82.54
BREA	1909.92	2544.45	3178.98	3813.51
RFLPA (KRUM)	79.58	82.25	85.68	89.87

Table 11: Computation cost (in minutes) with varying client size with MNIST classifier (1.6M parameters). RFLPA (KRUM) replaces the aggregation rule with KRUM in RFLPA.

	Per-user Cost				Server Cost			
Client size	100	200	300	400	100	200	300	400
RFLPA	3.41	11.44	24.51	42.60	6.68	8.46	15.00	26.47
BREA	44.73	101.39	182.27	294.27	75.85	145.30	216.96	287.22
RFLPA (KRUM)	13.60	35.56	46.48	75.78	31.77	34.04	39.76	62.81

K.9 Non-IID Setting

K.9.1 Heterogenous Clients

The previous experiments were conducted under the assumption that the local data of clients are independent and identically distributed (IID). To simulate the non-IID dataset, we adopted the setting in [54] by sorting the data based on their labels and dividing them into 10,000 subsets. Consequently, the local data owned by most clients consist of only one label.

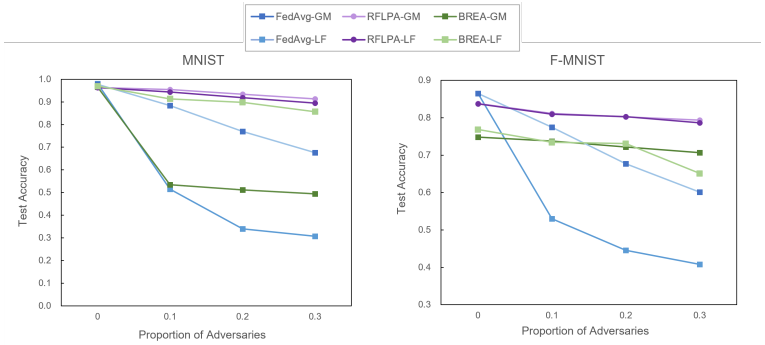

Figure 6: Test accuracy on non-IID dataset. GM stands for gradient manipulation attack, and LF stands for label flipping attack.

We compare the accuracy of RFLPA, BREA and FedAvg on non-IID dataset in Figure 6. The RFLPA demonstrates resilient performance against poisoning attacks, even when the dataset is distributed non-identically among clients.

K.9.2 Dynamic Data

For dynamic settings, we consider the case where the data of the clients change during the federated training with the arrival of new data. To simulate the setting, we leverage Dirichlet Distribution Allocation (DDA) [55] to sample non-iid dataset, and change the distribution for each client every 20 epochs. The parameter of the Dirichlet distribution is set to $\alpha = 0.1$.

Table 12 presents the accuracy against gradient manipulation attack. Our RFLPA demonstrates robust performance under the dynamic setting for up to 30% attackers. The improvement of RFLPA over FedAvg is more than 2x when there are at least 20% attackers.

https://doi.org/10.52202/079017-3314

Table 12: Accuracy under dynamic client data distribution against gradient manipulation attack.

	MNIST				F-MNIST			
% of attackers	No	10%	20%	30%	No	10%	20%	30%
FedAvg	0.98 ±0.0	0.27 ±0.2	0.29 ±0.3	0.29 ±0.1	0.86 ±0.0	0.52±0.1	0.21 ±0.2	0.18 ±0.2
RFLPA	0.96 ±0.0	0.94 ±0.0	0.92±0.0	0.90±0.0	0.83±0.0	0.79±0.0	0.79±0.1	0.77±0.0

K.10 Integration with Other Aggregation Protocols

The robust aggregation rule of RFLPA is based on FLTrust, requiring a clean root data set on server side. Suppose we cannot get any clean root dataset even if the required size is small, it is feasible to replace the aggregation protocol with other robust aggregation algorithms to circumvent the assumption.

First, our algorithm can be integrated with KRUM-based method by substituting the aggregation module with KRUM. Though KRUM incurs greater cost than the original method, Appendix K.8.2 shows that there is a notable reduction in communication and computation cost compared with BREA, benefiting from the design of our secret sharing algorithm. The accuracy of RFLPA (KRUM) is expected to be the same as BREA, as both utilize the same aggregation rule.

Another alternative is to compute the cosine similarity with global weights. Specifically, we can compute the cosine similarity between each local update and the global weights as follows [56]:

$$cos(\mathbf{w}_i^t, \mathbf{w}_G^{t-1}) = \frac{\langle \mathbf{w}_i^t, \mathbf{w}_G^{t-1} \rangle}{\|\mathbf{w}_i^t\| \|\mathbf{w}_G^{t-1}\|} \tag{33}$$

, and filter out the clients with similarity smaller than a pre-specified threshold, which is set to 0 in our evaluation.

From Table 13, we can observe that compared with FedAvg, RFLPA-GW effectively improves the accuracy in the presence of attackers. Noted that the communication and computation cost of RFLPA-GW is at the same scale of RFLPA's original level, as both compute the cosine similarity with a single baseline.

Table 13: Accuracy for defense based on global weight under different proportions of attackers. RFLPA-GW replaces the robust aggregation rule in RFLPA with the method based on cosine similarity with global weight.

	MNIST				F-MNIST			
% of attackers	No	10%	20%	30%	No	10%	20%	30%
FedAvg	**0.98** ±0.0	0.46 ±0.1	0.40 ±0.1	0.32 ±0.0	**0.88** ±0.0	0.55 ±0.0	0.51 ±0.0	0.45 ±0.1
RFLPA-GW	0.98 ±0.0	0.95 ±0.1	0.92±0.0	0.91±0.1	0.90±0.0	0.80±0.1	0.77±0.0	0.75±0.0

L Impact Statement

Our work in developing a robust federated learning framework (RFLPA) addresses significant challenges in privacy and security in federated learning (FL), presenting substantial benefits in data protection and carrying broader societal implications. The advancements in safeguarding data privacy bolster ethical standards in data handling, yet they may raise concerns in scenarios requiring data transparency. Our efforts contribute to the technical evolution of FL but also underscore the need for ongoing ethical considerations in the face of rapidly advancing machine learning technologies.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification:

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 8 Discussion and Future Work.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification:

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification:

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification:

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification:

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification:

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification:

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

https://doi.org/10.52202/079017-3314

Answer: [Yes]

Justification:

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Appendix L.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification:

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification:

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification:

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer:[NA]

    Justification: