ClusterGuard: Secure Clustered Aggregation for Federated Learning with Robustness

Yulin Zhao, Zhiguo Wan, *Member, IEEE*, Zhangshuang Guan

***Abstract*—Federated Learning (FL) enables collaborative model training while preserving data privacy by avoiding the sharing of raw data. However, in large-scale FL systems, efficient secure aggregation and dropout handling remain critical challenges. Existing state-of-the-art methods, such as those proposed by Liu et al. (UAI'22) and Li et al. (ASIACRYPT'23), suffer from prohibitive communication overhead, implementation complexity, and vulnerability to poisoning attacks. Alternative approaches that utilize partially connected graph structures (resembling client grouping) to reduce communication costs, such as Bell et al. (CCS'20) and ACORN (USENIX Sec'23), face the risk of adversarial manipulation during the graph construction process.**

To address these issues, we propose ClusterGuard, a secure clustered aggregation scheme for federated learning. ClusterGuard leverages Verifiable Random Functions (VRF) to ensure fair and transparent cluster selection and employs a lightweight key-homomorphic masking mechanism, combined with efficient dropout handling, to achieve secure clustered aggregation. Furthermore, ClusterGuard incorporates a dual filtering mechanism based on cosine similarity and norm to effectively detect and mitigate poisoning attacks.

Extensive experiments on standard datasets demonstrate that ClusterGuard achieves over 2x efficiency improvement compared to advanced secure aggregation methods. Even with 20% of clients being malicious, the trained model maintains accuracy comparable to the original model, outperforming state-of-the-art robustness solutions. ClusterGuard provides a more efficient, secure, and robust solution for practical federated learning.

I. Introduction

In recent years, amidst the rising importance of data privacy and security, federated learning (FL) has emerged as a solution for collaborative model training among multiple data custodians without sharing raw data. While FL prevents direct data exposure to third parties, it still carries significant privacy risks, as highlighted in [1]. Protecting update data in FL training is paramount, underscoring the urgent need for a secure aggregation framework.

Currently, quite a few schemes have been proposed to address this pressing need. The pioneering work is due to Bonawitz et al. from Google [2], along with a series of subsequent works [3]–[6]. Their key idea is to establish pairwise opposite masks for each pair of clients to hide their model parameters, and these masks can be cancelled completely in aggregation. Assume a set of clients $\mathcal{U}$ perform federate learning over their local data, and each client $u \in \mathcal{U}$ obtains a local model update $\mathbf{x}_u$, where the elements of $\mathbf{x}_u$ and $\sum_{u\in\mathcal{U}} \mathbf{x}_u$ are in $\mathbb{Z}_R$ for some R. Then we can add the mask as follows to each $\mathbf{x}_u$:

$$\mathbf{y}_u = \mathbf{x}_u + \mathbf{PRG}(b_u) + \sum_{v\in\mathcal{U}:u<v} \mathbf{PRG}(s_{u,v}) - \sum_{v\in\mathcal{U}:u>v} \mathbf{PRG}(s_{v,u}) \mod R \quad (1)$$

where **PRG** is a pseudorandom generator, $s_{u,v}$ is the shared seed between clients u and v used to generate pairwise masks, and b_u is another random seed for the client itself used to generate individual masks. Clients secretly share all their seeds with other clients. Later, when the masked models $\mathbf{y}_u$ are aggregated, the individual masks of online clients are recovered, and the pairwise masks of offline clients are recovered. At this point, the pairwise opposite masks will be successfully eliminated, and the individual masks can also be correctly subtracted. If only pairwise masks are used, clients may experience delays in coming online, which could lead to model leakage if the server uses online clients to recover its secret.

However, each client needs to compute and share pairwise masks with other clients, which leads to increased complexity and resource demands. Additionally, in the event of client dropouts, the server must reconstruct numerous pairwise masks, further hindering its practicality in large-scale federated learning (FL).

Liu et al. [7], SASH [8], and LERNA [9] propose utilizing homomorphic pseudorandom generators to optimize the process into a single mask. However, their constructions face significant practical challenges: Liu et al.'s approach requires brute-forcing discrete logarithms, resulting in a substantial computational burden; SASH introduces additional communication rounds, with computation results containing unavoidable errors; and LERNA relies on maintaining committees with a large number of members to support its scheme, which is impractical in real-world scenarios. Bell et al. [3] and ACORN [10] adopted the Distributed Graph Generation approach to effectively reduce the communication overhead required by clients. However, they overlooked the risk of collusion between clients and the server, which could manipulate neighbor relationships. This makes it challenging to ensure truly random neighbor sampling, posing potential security threats to the system.

Corresponding author: Zhiguo Wan.

Yulin Zhao is 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).

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

Zhangshuang Guan is with the College of Computer Science and Technology, Zhejiang University, Hangzhou, 310027, China (e-mail: guanzs@zju.edu.cn).

In this work, we introduce a novel federated learning aggregation scheme, ClusterGuard, to address the aforementioned issues. Our key idea leverages Verifiable Random Functions (VRF) for clustering, ensuring randomness in partition results. We then devise a concise clustering aggregation scheme using lattice-based key-homomorphic pseudorandom random function (LKH-PRF) and design corresponding strategies for dropout handling and robust aggregation. We also propose two variants of the scheme, enabling our approach to be flexibly applied in both client-server and decentralized scenarios. We compare ClusterGuard with the relevant schemes listed in Table I in terms of complexity and properties. The contributions of this paper are as follows:

- We design a VRF-based cluster election strategy to ensure fairness and transparency. Clients jointly compute the election result using their own keys, preventing manipulation by either the server or clients, while safeguarding client data security. Building on cluster aggregation, we devise a concise, secure model aggregation scheme based on LKH-PRF. It is efficient, error-free, and supports dropout.
- We design a countermeasure against poisoning attacks based on norm and cosine similarity for ClusterGuard, preserving client privacy while detecting malicious clients efficiently.
- We conduct a rigorous security and theoretical analysis, accompanied by detailed performance assessments, to validate our scheme's theoretical security guarantees and feasibility.
- Extensive experiments on common datasets demonstrate over 2x performance improvement compared to classical secure aggregation methods. Additionally, under attacks from 20% of malicious clients, the training model maintains a high level of accuracy comparable to the original model, surpassing current state-of-the-art robustness solutions.

II. Models and Preliminaries

Before introducing the models and preliminaries, we list the notations used in our scheme in Table II.

A. Threat Model

In line with prior research [2], [11], we investigate two distinct threat models: 1) the semi-honest setting and 2) the malicious setting.

1) The semi-honest setting: it is assumed that the server and all clients will faithfully execute the protocol as prescribed, but some of them are curious to uncover the privacy of other parties, and they may share their information with each other to accomplish this aim;
2) The malicious setting: it is assumed that the server and a subset of clients may act maliciously, such as deviating from the designated protocol or sending altered messages to honest parties.

We consider two different types of attackers, as referenced by Zhang et al [12]. The first type of attacker aims to steal the privacy of model updates from honest clients, while the second type seeks to disrupt the normal training of federated learning. We assume that the server and the second type of attackers are adversarial, as the server may also benefit from an accurate and robust global model.

1) Type I attackers: Potential type I attackers include the server and some clients. It is important to note that collusion may occur both between clients and between the server and clients, with the aim of compromising the model privacy of honest clients. Clients may be malicious, while the server is assumed to be honest but curious.
2) Type II attackers: For type II attackers, the attackers consist of one or more malicious clients, who can upload arbitrary gradients to perform poisoning attacks, with the goal of disrupting the training of the entire global model. The attackers cannot influence the behavior of benign users, nor can they observe the local models of benign users. In the cluster aggregation scheme, we assume that the proportion of Type II attackers (malicious or compromised users) is less than $\frac{k}{n}$ to ensure that at least one cluster consists entirely of benign clients. Additionally, we assume that the attackers can fully cooperate with each other.

B. Federated Learning

Federated learning (FL) is a distributed training framework where multiple clients collaboratively train a machine learning model. The classic FL algorithm, FedAvg [13], involves n clients and a server. Each client $u_i \in \mathcal{U}$ holds its local dataset $\mathcal{D}_i$. The server initializes the global model parameter θ_0 and broadcasts it to all clients. Participants repeat the following steps for each round r until convergence:

Step 1. Each client receives the global model θ_r and calculates local model updates $\mathbf{w}^{(u_i)} = \nabla F_i(\theta_r, \xi_i)$ by performing multiple steps of Stochastic Gradient Descent (SGD) on the local available data ($\xi_i \sim \mathcal{D}_i$), where F_i is a loss function.

Step 2. Clients upload weight updates $\mathbf{w}_r^{(u_i)}$ to the server.

Step 3. The server aggregates updates and sends back $\theta_{r+1} \leftarrow \theta_r - \eta \frac{1}{n} \sum_{u_i \in \mathcal{U}} \mathbf{w}_r^{(u_i)}$, where η is the learning rate.

C. Verifiable Secret Sharing

Based on Feldman's VSS [14] , Verifiable Secret Sharing (VSS) enables secure distribution of a secret among multiple parties without a dealer. It ensures share integrity through commitments to polynomial coefficients, facilitating secret reconstruction by collaborating participants.

- $\mathbf{VSS.Setup}(1^\lambda) \rightarrow (G, q, g)$, where g is a generator of group G with prime order q.
- $\mathbf{VSS.Share}(s_u, t, \mathcal{U}) \rightarrow (\{v, s_{u,v}\}_{v \in \mathcal{U}}, \{g^{s_{u,v}}\}_{v \in \mathcal{U}}, \{g^{a_{u,d}}\}_{d \in [0,t-1]})$. On input $s_u \in \mathbb{Z}_q$, client u chooses $a_{u,1}, \cdots, a_{u,t-1} \xleftarrow{\$} \mathbb{Z}_q$, constructs a polynomial $f_u(x) = a_{u,0} + a_{u,1}x + \ldots + a_{u,t-1}x^{t-1}$ where $a_{u,0} = s_u$ and $a_{u,t-1} \neq 0$, computes $\{g^{a_{u,d}}\}$ for $\forall d \in [0, t-1]$, $s_{u,v} := f_u(v)$ and $g^{s_{u,v}}$ for $\forall v \in [1, |\mathcal{U}|]$.

TABLE I
COMMUNICATION AND COMPUTATION OVERHEAD COMPARED WITH OTHER SECURE AGGREGATION SCHEMES.

Setting	Bonawitz [2]	Bell [3]	FLDP [11]	ACORN [10]	LERNA [9]	Ours
Client Communication	$O(n+m)$	$O(\log n+m)$	$O(m+k'+n)$	$O(m+\log n)$	$O(m)$	$O(\ell l+m+n)$
Client Computation	$O(n^2+nm)$	$O(\log^2 n+m\log n)$	$O(mk'+n\log n)$	$O(m\log n)$	$O(mlog\ell+M\ell)$	$O(\ell l^2+m\ell+n)$
Server Communication	$O(n^2+nm)$	$O(n\log n+nm)$	$O(mn+k')$	$O(nm+n\log n)$	$O((M+n)m)$	$O(mn+\ell n)$
Server Computation	$O(mn^2)$	$O(n\log^2 n+nm\log n)$	$O(mk'+mn+n\log n)$	$O(nm\log n)$	$O(Mt)$	$O(nt+\frac{nm\ell}{l})$
Pairwise Masking	Yes	Yes	No	Yes	No	No
Mask Error	No	No	Yes	No	No	No
Clustering Mechanism	✗	✓	✗	✓	✗	✓
Robustness Mechanism	No	No	No	Yes	No	Yes

* n is the number of clients, m is the input size or the model vector size, k' is the key size in FLDP [11], ℓ is the key size of our scheme, and t is the threshold of secret sharing, l is the number of members in each cluster, M is the commitee size of LERNA. For fair comparison, the computation and communication of LERNA is for non-member clients (including flat secret sharing).

TABLE II
NOTATIONS

$\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{k}, \mathbf{s}, \mathbf{s}_u$	Vector format
s, s_u, a_i	Scalar format
$\langle \mathbf{x} \cdot \mathbf{y} \rangle$	Inner product
$\{s_u\}_{u\in\mathcal{U}}$	A set of scalar values
$\mathcal{U}=\{u_1, u_2, \cdots, u_n\}$	A set of users (clients)*
$\mathcal{C}=\{\mathcal{C}_1, \mathcal{C}_2, \cdots \cup \mathcal{C}_k\}$	Clients split into clusters of equal size
$\mathcal{D}_u$	Data set held by u
$N_u=\lvert\mathcal{D}_u\rvert$	Size of data sets
$N=\sum_{u\in\mathcal{U}}\lvert\mathcal{D}_u\rvert$	Total size of data sets
$x \xleftarrow{R} \mathcal{X}$	Choose x uniformly at random from $\mathcal{X}$
k	the number of clusters
l	the number of members in each cluster
n	the number of clients

- $\mathbf{VSS.Verify}(v, g^{s_{u,v}}, \{g^{a_{u,d}}\}_{d\in[0,t-1]}) \to b$. This algorithm outputs 1 if and only if the following equation holds:
$$g^{s_{u,v}} \stackrel{?}{=} \prod_{d=0}^{t-1} (g^{a_{u,d}})^{v^d} \mod q$$
- $\mathbf{VSS.Accum}(\{v, s_{u,v}\}_{u\in\mathcal{U}}) \to (v, s_v^*)$. On input $|\mathcal{U}|$ inputs of the form $(v, s_{u,v})_{u\in\mathcal{U}}$, this algorithm enables client v to accumulate $|\mathcal{U}|$ shares by computing $s_v^* = \sum_{u\in\mathcal{U}} s_{u,v}$ and $g^{s_v^*}$.
- $\mathbf{VSS.VrfyAccum}(v, g^{s_v^*}, \{g^{s_{u,v}}\}_{u\in\mathcal{U}}) \to b$. This algorithm outputs 1 if and only if the following equation holds: $g^{s_v^*} \stackrel{?}{=} \prod_{u\in\mathcal{U}} g^{s_{u,v}} \mod q$.
- $\mathbf{VSS.Recon}(\{v, s_v^*\}_{v\in\mathcal{U}}) \to s$. On input at least t inputs of the form (v, s_v^*), this algorithm computes and outputs $s = f(0) = \sum_{u\in\mathcal{U}} s_u$.

D. Lattice-based Key-homomorphic PRF

Given spaces $\mathcal{X}, \mathcal{Y}, \mathcal{K}$ over $\{0,1\}^*$, let $(\mathcal{K}, \star)$ and $(\mathcal{Y}, \bullet)$ be groups and have a group homomorphism between the space $\mathcal{K}$ and $\mathcal{Y}$ [15]. For $\forall x \in \mathcal{X}$ and $k_1, k_2 \in \mathcal{K}$, a pseudorandom function $\mathrm{PRF}_k : \mathcal{X} \to \mathcal{Y}$ where $k \in \mathcal{K}$ denotes the key is key-homomorphic, if the following equation holds: $\mathrm{PRF}_{k_1}(x) \bullet \mathrm{PRF}_{k_2}(x) = \mathrm{PRF}_{k_1 \star k_2}(x)$.

The above properties can be realized by the lattice cryptography. In fact, the lattice-based KH-PRF is an approximate homomorphic function, formulated as [16]: $\text{LKH-PRF}_{k_1+k_2}(x) = \text{LKH-PRF}_{k_1}(x) + \text{LKH-PRF}_{k_2}(x) + e$, where $e \in \mathbb{N}$ is small. Among existing lattice-based KH-PRF schemes, Boneh et al.' s construction [16] based on LWR is simpler and more efficient, and its security is based on ROM. Let $H : \mathcal{X} \to \mathbb{Z}_{\mathfrak{q}}^{\ell}$ be a hash function modeled as a random oracle, $\mathbf{k} \in \mathbb{Z}_{\mathfrak{q}}^{\ell}$ be a vector key, the lattice-based KH-PRF is defined as follows:

$$\text{LKH-PRF}_{\mathbf{k}}(x) = \lceil \langle H(x), \mathbf{k} \rangle \rfloor_{\mathfrak{p}} \tag{2}$$

where $\lceil x \rfloor_{\mathfrak{p}}$ denotes the rounding operation as $\lceil x \rfloor_{\mathfrak{p}} = \lfloor x \cdot (\mathfrak{p}/\mathfrak{q}) \rfloor \mod \mathfrak{p}$. Ernst et al. [15] complete the security proof of the LKH-PRF, provide a simple construction of $H(x)$, and introduce a rounding error $e \in \{0,1\}$ in their construction. Let $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ where $x_i \in \mathcal{X}$, we can extend LKH-PRF for m-dimensional vectors as follows: $\text{LKH-PRF}_{\mathbf{k}}(\mathbf{x}) = [\text{LKH-PRF}_{\mathbf{k}}(x_1), \cdots, \text{LKH-PRF}_{\mathbf{k}}(x_m)]$.

E. Verifiable Random Function

Verifiable Random Functions (VRFs) [17] are cryptographic primitives that extend the concept of random functions by enabling third parties to verify the correctness of the outputs without knowing the secret keys. A Formal Definition is as follows:

- **KeyGen**$(1^\lambda) \to (sk_u, pk_u)$ Generate the corresponding key pair (sk_u, pk_u) based on the public security parameter λ.
- **Eval**$(sk_u, m_u) \to (r_u, \pi_u)$ Given the secret key sk_u, an input $m_u \in \mathbb{N}$, the evaluation algorithm outputs $f_{sk}(x)$ containing $r_u \in \mathbb{N}$ along with a proof π_u, such that the correctness of the output can be verified by pk_u.
- **Check**$(pk_u, m_u, r_u, \pi_u) \to \{0,1\}$ Given pk_u, m_u, r_u, and π_u, the verification algorithm outputs 1 if r_u is correct, and 0 otherwise.

III. RELATED WORK

A. Secure aggregation based on MPC

MPC enables joint computation over private inputs, suitable for secure federated learning aggregation. SecureML [18] employs two-party computation for privacy, but only for basic neural networks. Google's scheme [2] utilizes double masking and secret sharing with high computational overhead. New approaches like FastSecAgg [4] and Laf [19] optimize secret sharing but overlook double mask complexity. CCESA [20],

SecAgg+ [3], and Turbo-Aggregate [5] alter client communication to reduce complexity. However, they still rely on double masks, leading to increased computation with dropped clients. As dropped clients rise, both server and client computation and communication increase significantly. Liu et al. [7] used homomorphic PRG based on Diffie-Hellman assumption to construct a single mask protocol for the secure aggregation, but the whole protocol requires additional discrete logarithm calculation when recovering model aggregation, which hinders its application in the actual scenario. SASH [8] and LERNA [9] proposed using (R)LWR to construct homomorphic masks for single-server aggregation, which is similar to our paper, but our scheme requires fewer public parameters. SASH did not consider how to solve the error problem caused by homomorphic masks. And LERNA uses large-scale heavy clients to serve as committee members to address errors and mask reconstruction issues, which increases the actual deployment difficulty of the scheme. At the same time, their aggregation scheme cannot provide security against poisoning attacks.

B. Robust aggregation of federated learning

. Poisoning attacks, divided into data and model poisoning, pose threats to machine learning models. Data poisoning involves adding misleading samples to training data, leading to incorrect predictions [21], [22]. Model poisoning entails sending false updates to global models, impacting performance in distributed settings like federated learning [23], [24]. [25]–[29] proposed defenses using Euclidean distance, median, and cosine similarity, but they require servers to access plaintext updates, risking privacy. SecureFL [30], PEFL [31], FLOD [32], PBFL [33], ShieldFL [34], and RFed [35] enable secure computation of similar metrics but rely on two or more non-colluding servers to ensure privacy, which is challenging in practice. ARCON [10] and Hao et al. [36] proposed single server robust aggregation schemes, with the former leveraging zero-knowledge proof and the latter employing multi-key homomorphic encryption, both incurring substantial communication overhead. Cluster aggregation addresses these challenges, allowing the server to aggregate cluster values without revealing individual updates. Several related works [37]–[40] explore this approach, but these studies either require the server has a certain scale of verification data set, or can not guarantee that the performance of the model is not damaged.

IV. Our Scheme

We consider the following federated learning scenario: (i) a server, responsible for aggregating model updates from participating clients, and (ii) a set of clients $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ with the training data $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n\}$, carrying out the model training locally, and uploading the encrypted model.

A. Overview

Our approach focuses on the inherent need for secure and efficient collaboration in federated learning settings. Initially, the ClusterGuard architecture strategically organizes clients into clusters based on Verifiable Random Function (VRF), ensuring a fair and transparent partitioning process.

Within each cluster, secure aggregation is employed to integrate model updates in a privacy-preserving manner. Each client generates a model mask using its private key, ensuring that individual contributions are securely incorporated into the aggregated model without compromising confidentiality. This approach not only enhances data privacy but also reduces the scale of secure computations.

The collector plays a pivotal role in the aggregation process by directly calculating the overall model mask based on the total secret key derived from all participating clients (we can easily achieve $\text{KH-PRF}_{\mathbf{k}}(\mathbf{x}) = \sum_{u\in\mathcal{U}} \text{KH-PRF}_{\mathbf{k}_u}(\mathbf{x})$ through the key homomorphic pseudorandom function). By eliminating the need for individual client contributions during aggregation, the server streamlines the process and ensures computational efficiency without sacrificing security.

Furthermore, robust aggregation are employed to validate the integrity of the aggregated model update, ensuring its correctness and reliability. This comprehensive approach to aggregation guarantees the accuracy of the final model update, even in the presence of adversarial attacks or client dropouts.

Taking real-world application scenarios into account, we propose two variants of our scheme: one designed for the single server-client model and the other suitable for a decentralized setting. For consistency and ease of explanation, we focus on the single server-client model in the main description, while the decentralized variant is presented as an extension in the discussion section. We give the overall architecture of our solution in Figure 1, which is divided into four rounds. The specific protocol can be found in Protocol 3.

B. LKH-PRF Model Mask and Aggregation

In HomAgg scheme, each user $u \in \mathcal{U}$ holds the m-dimensional model update $\mathbf{x}_u \in \mathbb{R}^m$, while the outputs of LKH-PRF are elements of $\mathbb{Z}_{\mathfrak{p}}$, thus we define an encoding scheme $(\text{Encode}, \text{Decode})$ for HomAgg as follows:

$$\begin{aligned} &\text{Encode} : \mathbb{R}^m \to \mathbb{Z}^m : \mathbf{x}_u \mapsto \hat{\mathbf{x}_u} = \lfloor (\mathbf{x}_u + \mathbf{b}) \cdot 2^{prec} \rfloor \\ &\text{Decode} : \mathbb{Z}^m \to \mathbb{R}^m : \hat{\mathbf{x}_u} \mapsto \mathbf{x}_u = \frac{\hat{\mathbf{x}_u}}{2^{prec}} - \mathbf{b} \end{aligned} \tag{3}$$

where $prec$ is the number of bits of precision and $\mathbf{b}$ is a constant offset to ensure encoding as unsigned integers. Note that scalar multiplication, modulo and comparison operations on vectors are applied individually each of their elements.

As described in Equation (II-D), let $pp = \{\ell, \mathfrak{p}, \mathfrak{q}, \mathbf{c}\}$ be public parameters of $\text{LKH-PRF}_{\mathbf{k}}$ where PRF is basd on $\text{LWR}_{\ell,\mathfrak{q},\mathfrak{p}}$, $\mathbf{k} \in \mathbb{Z}_{\mathfrak{q}}^{\ell}$ and $\mathbf{c} \in \mathbb{Z}_{\mathfrak{q}}^{m}$ is a non-repetitive sequence representing label values for various parameters of the global model, e.g., $\mathbf{c} = (c||1, c||2, \cdots, c||m)$ and $c \in \mathbb{Z}_{\mathfrak{q}}$. In the setup phase, the public parameter $\mathbf{c}$ only requires an additional scalar $c \in \mathbb{Z}_q$, rendering it simpler compared to the matrix construction utilized in scheme [8], [9].

Each user $u \in \mathcal{U}$ holds a secret key $\mathbf{s_u}$, and he can compute his masked model as follows:

$$\mathbf{y}_u = \Delta \cdot \text{Encode}(\mathbf{x}_u) + \text{LKH-PRF}_{\mathbf{s}_u}(\mathbf{c}) \tag{4}$$

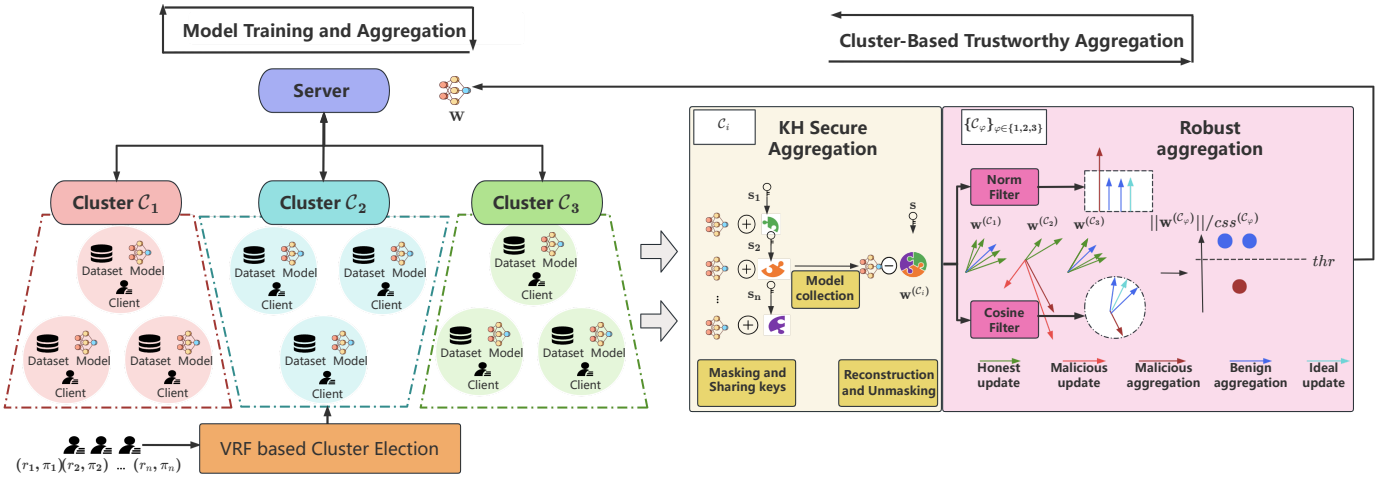


Fig. 1. The architecture of ClusterGuard. Initially, the clients are divided into clusters, with the grouping determined based on the VRFs, and secure aggregation is performed within each cluster. Each user calculates the corresponding model mask according to its secret key $\mathbf{s_i}$, and uploads the masked model. The server directly calculates the model mask and eliminates the secure aggregation model according to the total secret key $\mathbf{s}$. After conducting robust aggregation, the server obtains the correct aggregated model update.

where $\Delta > n$ is a scaling factor. The server holds the secret key $\mathbf{s} = \sum_{u\in\mathcal{U}} \mathbf{s_u}$ constructed with key shares and computes the aggregate masks as follows:

$$\text{LKH-PRF}_{\mathbf{s}}(\mathbf{c}) = \sum_{u\in\mathcal{U}} \text{LKH-PRF}_{\mathbf{s}_u}(\mathbf{c}) + \mathbf{e} \mod \mathfrak{p} \quad (5)$$

where $\mathbf{e} = (e_1, \ldots, e_m)$ is a vector composed of errors and $0 \le e_i < |\mathcal{U}| = n$ since the summation of LKH-PRF causes the accumulation of its error from $\{0, 1\}$. With reasonable parameter selection, we assume that $n\Delta < \mathfrak{p}$, the secure aggregation can be done as follows:

$$\begin{aligned}\hat{\mathbf{z}} &= \frac{1}{n\Delta}\Big(\sum_{u\in\mathcal{U}} \mathbf{y}_u - \text{LKH-PRF}_{\mathbf{s}}(\mathbf{c})\Big) \\ &= \frac{1}{n\Delta}\sum_{u\in\mathcal{U}} \Delta\cdot\hat{\mathbf{x}_u} - \frac{\mathbf{e}}{\Delta} \mod \mathfrak{p} \\ &= \sum_{u\in\mathcal{U}} \frac{1}{n}\cdot \text{Encode}(\mathbf{x}_u)\end{aligned} \quad (6)$$

It can be seen that the solution to the error introduces a scaling factor. Although the error issue is resolved, a new problem arises: the scaling factor must be at least greater than the number of clients. As the number of clients increases, for a fixed modulus $\mathfrak{p}$, this will result in a decrease in the maximum number of bits that the target vector can add a mask to. Since the server will compute $\sum_{u\in\mathcal{U}} \Delta\cdot\text{Encode}(\mathbf{x}_u)$, and it is necessary to ensure no overflow occurs, each client can use at most $\lceil \log_2(\frac{\mathfrak{p}}{n\cdot\Delta})\rceil$ bits to transmit its original data. Therefore, the number of clients must be limited. A fixed cluster can guarantee this. After decoding the aggregation results, it can be easily deduced that we can obtain the final model updates aggregation:

$$\mathbf{z} = \text{Decode}(\hat{\mathbf{z}}) = \text{Decode}\Big(\sum_{u\in\mathcal{U}} \frac{1}{n}\hat{\mathbf{x}_u}\Big) = \frac{1}{n}\sum_{u\in\mathcal{U}} \cdot\mathbf{x}_u \quad (7)$$

C. VRF Based Clustering

Consider a bilinear pairing $e : G_1 \times G_2 \to G_T$, where G_1 and G_2 are multiplicative cyclic groups of prime order p, and g_1, g_2 are the generators of G_1 and G_2 respectively. Let $H : \{0,1\}^* \to G_1$ be a hash function that maps a message m to an element in G_1.

Assume there are n clients, each holding a private key $sk_i \in \mathbb{Z}_p$ and a corresponding public key $pk_i = g_2^{sk_i}$. The protocol proceeds as follows:

Each client u_i generates a random value $r_i \in \{0,1\}^*$ and computes its proof:

$$\pi_i = H(r_i)^{sk_i}. \quad (8)$$

The aggregate proof from all n clients is computed as:

$$\pi = \prod_{i=1}^{n} \pi_i = H(r_1)^{sk_1}\cdot H(r_2)^{sk_2}\cdots H(r_n)^{sk_n}. \quad (9)$$

The BLS aggregate verification is performed by checking the following equation:

$$e(\pi, g_2) \stackrel{?}{=} \prod_{i=1}^{n} e(H(r_i), pk_i). \quad (10)$$

Correctness:

$$\begin{aligned} e(\pi, g_2) &= e\left(\prod_{i=1}^{n} H(r_i)^{sk_i}, g_2\right) \\ &= \prod_{i=1}^{n} e(H(r_i)^{sk_i}, g_2) \\ &= \prod_{i=1}^{n} e(H(r_i), g_2^{sk_i}) \\ &= \prod_{i=1}^{n} e(H(r_i), pk_i). \end{aligned}$$

Note:

For $A, B \in G_1$, we have $e(A \cdot B, g_2) = e(A, g_2) \cdot e(B, g_2)$.
Let $A = g_1^{\alpha}, B = g_1^{\beta}$, then
$e(AB, g_2) = e(g_1^{\alpha} \cdot g_1^{\beta}, g_2) = e(g_1, g_2)^{\alpha+\beta}$.
$e(A, g_2) \cdot e(B, g_2) = e(g_1^{\alpha}, g_2) \cdot e(g_1^{\beta}, g_2) =$
$e(g_1, g_2)^{\alpha} \cdot e(g_1, g_2)^{\beta} = e(g_1, g_2)^{\alpha+\beta}$.

Each client possesses a key pair (pk_i, sk_i) and generates a tuple (r_i, π_i), broadcasting these values to the server and other clients. Upon receiving the broadcasts, other clients perform batch verification. If the verification is successful, they compute their cluster assignments $\{\mathcal{C}_\varphi\}_{\varphi \in [0,k-1]}$ as follows:

1) Compute the aggregated randomness:

$$R_{\text{total}} = \bigoplus_{i=1}^{n} r_i$$

where $\bigoplus$ denotes the bitwise XOR operation.

2) Determine the cluster index for each client:

$$\varphi_i = \text{Cluster}(u_i) = \text{Hash}(i \| R_{\text{total}} \| r) \mod k$$

where φ_i indicates the cluster $\mathcal{C}_\varphi$ to which client u_i is assigned, r is current federated learning round.

The secure aggregation protocol is then executed within each cluster to collect model updates. After the aggregation process, the server compiles a trusted root dataset to aid in anomaly detection, filtering out abnormal updates to ensure the integrity of the results.

D. Collection

Secrect sharing Setup: All clients are randomly divided into k clusters, i.e., $\mathcal{U} = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \ldots \cup \mathcal{C}_k$, and the size of each cluster is fixed at $t = |\mathcal{C}_\varphi|$ where $\varphi \in [1, k]$. Clients perform the following steps to complete the key sharing:

Step 1. For each cluster $\mathcal{C}_\varphi \subset \mathcal{U}$, each client $u \in \mathcal{C}_\varphi$ selects a random ℓ-dimensional vector $\mathbf{s}_u$ and calculates its own model mask in advance.

Step 2. Each client $u \in \mathcal{C}_\varphi$ utilizes $\mathbf{VSS.Share}$ to generate t shares for each element s_u in $\mathbf{s}_u$ and sends them to other clients $v \in \mathcal{C}_\varphi$ respectively. Meanwhile, u utilizes $\mathbf{VSS.Accum}$ to accumulate secret key shares received from other clients v.

Step 3. For each cluster $\mathcal{C}_\varphi \subset \mathcal{U}$, the server/leader can collect t shares accumulated by each client $u \in \mathcal{C}_\varphi$ and call $\mathbf{VSS.Recon}$ to recover $s_{\mathcal{C}_\varphi} = \sum_{u \in \mathcal{C}_\varphi} s_u$ and $\mathbf{s}_{\mathcal{C}_\varphi} = \sum_{u \in \mathcal{C}_\varphi} \mathbf{s}_u$, finally obtain $\mathbf{s} = \sum_{\mathcal{C}_\varphi \subset \mathcal{U}} \mathbf{s}_{\mathcal{C}_\varphi}$. After receiving all the masked model parameters, the server can unmask and decode to obtain the aggregated model parameter as follows:

$$\mathbf{z} = \frac{1}{k} \sum_{\mathcal{C}_\varphi \subset \mathcal{U}} \left(\frac{1}{l} \sum_{u \in \mathcal{C}_\varphi} \mathbf{x}_u \right) = \frac{1}{n} \sum_{u \in \mathcal{U}} \mathbf{x}_u \tag{11}$$

where $\frac{1}{l} \sum_{u \in \mathcal{C}_\varphi} \mathbf{x}_u$ denotes the secure aggregation value of clients in $\mathcal{C}_\varphi$. The detailed secret sharing and reconstruction process can be found in Figure 2.

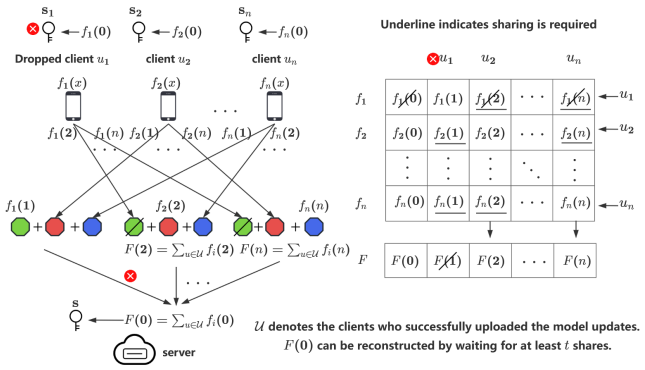


Fig. 2. Each cluster performs secret sharing internally. After collecting all masked model updates, the server notifies each client $u \in \mathcal{U}$ that needs to calculate the sum of secret shares (i.e., all clients that have successfully uploaded masked model updates in a cluster). Then the server only needs to wait for t clients within the cluster to upload the sum of shares to reconstruct the cluster's total secret key $\mathbf{s}$.

E. Handling Dropped Clients

The dropped clients affect the elimination of masks, we consider how to deal with this problem under the setting of cluster aggregation and (l, t)-secret sharing. When at most $(l - t)$ clients are dropped in a cluster, the server can adopt t clients' shares to recover the sum of secrets of $|\mathcal{C}_\varphi| = l$ clients, and then eliminate the masks associated with the received masked models from clients.

By collecting masked models, the server can record client $v \in \mathcal{C}_\varphi$ who has successfully uploaded masked models to form a new cluster $\mathcal{C}'_\varphi$, and then send to each client $u \in \mathcal{C}'_\varphi$ its corresponding cluster's online client list. Each client $u \in \mathcal{C}'_\varphi$ accumulates shares $s_{v,u}$ received from $v \in \mathcal{C}_\varphi$ and send them to the server. Finally, the server collects at least t shares accumulated by each client $u \in \mathcal{C}'_\varphi$ to recover the secret of $\mathcal{C}_\varphi$.

F. Robust Aggregation

To counter poisoning attacks, we use robust aggregation during cluster aggregation to safeguard the model from adversaries. Similar to FLtrust [28], the server collects a small, clean root dataset to compute the ideal model update. It then calculates the cosine similarity between the ideal update and the cluster's model update to gauge the security of the cluster aggregation. Let $\mathbf{w}_r^{(\mathcal{C}_\varphi)}$ denote an updated value of the aggregation model of the cluster $\mathcal{C}_\varphi$ at a round r, g_r^* denote ideal model update, define the cosine similarity score css [28] for the cluster model update of $\mathcal{C}_\varphi$ as

$$css_{\mathcal{C}_\varphi} = \frac{\langle g_r^*, \mathbf{w}_r^{(\mathcal{C}_\varphi)} \rangle}{\|g_r^*\| \cdot \|\mathbf{w}_r^{(\mathcal{C}_\varphi)}\|} \tag{12}$$

where $css_{\mathcal{C}_\varphi}$ reflects the positivity and negativity of the contribution of the model update, as shown in Figure 1.

Under the cluster collection, the cluster is divided into three situations: (1) all benign clients, (2) mixed malicious clients and benign clients, and (3) all malicious clients. Given a threshold thr, the cluster whose css is less than thr is labeled as malicious, and its update cannot be used in the

final aggregation operation. Therefore, the final result with the robustness mechanism should be modified as follows:

$$\begin{aligned} \mathcal{U}' &\leftarrow \{\mathcal{C}_\varphi \mid \mathcal{C}_\varphi \subset \mathcal{U} \ \ \& \ \ css_{\mathcal{C}_\varphi} > thr\} \\ \mathbf{z}' &= \frac{1}{k'} \sum_{\mathcal{C}_\varphi \subset \mathcal{U}'} (\frac{1}{l} \sum_{u \in \mathcal{C}_\varphi} \mathbf{x}_u) = \frac{1}{k'l} \sum_{u \in \mathcal{U}'} \mathbf{x}_u \end{aligned} \tag{13}$$

Those with abnormal model update amplitude will also be eliminated, and we uses L_∞-norm to judge, when the updated norm is significantly different from the overall median norm, it is considered a malicious update. Additionally, beyond this, when the model update scores for all clusters are too low (less than the update threshold), it is essential to reject all updates. This can be referred to as the "baffle mechanism", which further safeguards the overall model performance against potentially harmful malicious updates. Let $Max(\cdot)$ return the maximum value, $Med(\cdot)$ returns the median value, $Stdev(\cdot)$ return the standard deviation, and β, bl_1, bl_2 are hyperparameters, the specific robust aggregation is shown in Algorithm 1.

Algorithm 1 RobustAgg

Input: Global model θ_r, model update aggregations of all clusters $\mathcal{Q} = \{\mathbf{w}_r^{(\mathcal{C}_1)}, \mathbf{w}_r^{(\mathcal{C}_2)}, \cdots, \mathbf{w}_r^{(\mathcal{C}_k)}\}$ (recover the model structure from the aggregated vectors $\{\mathbf{z}_{\mathcal{C}_\varphi}\}_{\mathcal{C}_\varphi \subset \mathcal{U}}$), root data set $\mathcal{D}_0$, local learning rate η.

Output: Robustness aggregation $\mathbf{w}_r$

Initialization: Generate $dis_1 = dis_2 = \emptyset, \mathcal{Q}_1' = \mathcal{Q}_2' = \emptyset$, the weight factor $\beta > 1$
Step 1: Trusted update calculation
Sample the root dataset for stochastic gradient descent to obtain the trusted model update g_r^*.
Step 2: Norm-based Filtering
for each cluster $\mathcal{C}_\varphi$ **do**
  $ndis_r^{(\mathcal{C}_\varphi)} \leftarrow |\frac{\|\mathbf{w}_r^{(\mathcal{C}_\varphi)}\|_\infty}{\|\mathbf{g}_r^*\|_\infty} - 1|$
  $dis_1 \leftarrow dis_1 \cup ndis_r^{(\mathcal{C}_\varphi)}$
$thr_1 \leftarrow \textbf{Min}(dis_1) \cdot \beta$
if $\textbf{Min}(dis_1) > bl_1$:
  $\mathcal{Q}_1' = \emptyset$
else:
  $\mathcal{Q}_1' = \{\mathbf{w}_r^{(\mathcal{C}_\varphi)} | ndis_r^{(\mathcal{C}_\varphi)} < thr_1\}$
Step 3: Cosine-based Filtering
for each cluster $\mathcal{C}_\varphi$ **do**
  $css_r^{(\mathcal{C}_\varphi)} = \frac{\langle g_r^*, \mathbf{w}_r^{(\mathcal{C}_\varphi)} \rangle}{\|g_r^*\| \cdot \|\mathbf{w}_r^{(\mathcal{C}_\varphi)}\|}$
  $dis_2 \leftarrow dis_2 \cup css_r^{(\mathcal{C}_\varphi)}$
$thr_2 \leftarrow \textbf{Max}(\{dis_2\}) - \beta \cdot Stdev(dis_2)$
if $\textbf{Max}(\{dis_2\}) < bl_2$:
  $\mathcal{Q}_2' = \emptyset$
else:
  $\mathcal{Q}_2' = \{\mathbf{w}_r^{(\mathcal{C}_\varphi)} | css_r^{(\mathcal{C}_\varphi)} < thr_2\}$
Step 4: Aggregation
Get trusted set: $\mathcal{Q}' = \mathcal{Q}_1' \cap \mathcal{Q}_2'$
If $Q' = \emptyset$, then $\mathbf{w}^r$ is a full 0 tensor with the same shape as the original model update, otherwise calculate the aggregate value $\mathbf{w}^r \leftarrow \frac{1}{|\mathcal{Q}'|} \sum \mathcal{Q}'$.
return $\mathbf{w}^r$

G. Putting it all Together

We give the overall architecture of our solution in Figure 1, which is divided into four rounds. The specific protocol can be found in Protocol 3.

In **Round 0**, the clients broadcast their VRFs and proofs, which are forwarded by the server. Upon receiving the information, clients perform verification. Subsequently, the clients are randomly partitioned into clusters, and the collection of model updates is carried out based on the cluster information. In **Round 1**, each client generates its own secret key and performs secret sharing with other clients in its cluster. In **Round 2**, each client adopts LKH-PRF to compute its own masked model update and uploads it to the server. The server records the cluster information of dropped clients and dispatches rescue clients as shown in Figure 2. In **Round 3**, the server waits for a threshold number of secret shares to be received, allowing it to begin reconstructing the total secret key for each cluster. Then server calculates the final model update aggregation of all clients. Note that execution within a malicious setting requires additional verification operations. We construct a complete protocol as shown in Figure 3.

V. Security and Convergence Analysis

A. Security in the Semi-Honest Setting

It is straightforward to see that our protocol achieves correctness for all participants as all of them honestly execute the protocol as prescribed. With regard to the privacy property, we establish the security of our protocol as a secure multiparty computation. This is demonstrated by showing that the joint view of the server and any subset of fewer than t clients reveals no information about the inputs of other clients, except for what can be inferred from the output of the computation.

Let $\mathcal{U}$ be the set of clients, and $\mathcal{C} \subset \mathcal{U} \cup \{\mathcal{S}\}$ be the set of honest-but-curious parties, x_u be the input of client u. Suppose the simulator SIM has access to an oracle $\mathsf{IDEAL}(t, x_u)_{u \in \mathcal{U} \setminus \mathcal{C}}$ as defined below:

$$\mathsf{IDEAL}(t, x_u)_{u \in \mathcal{U} \setminus \mathcal{C}} = \begin{cases} \sum\limits_{u \in \mathcal{U} \setminus \mathcal{C}} x_u & |\mathcal{U} \setminus \mathcal{C}| > t \\ \perp & \text{otherwise} \end{cases}$$

Please remember that in the traditional SecAgg scheme, the pairwise seeds used to generate masks are shared among users. For the (t, n) Shamir scheme with a shared secret s, the secret can only be reconstructed if more than t or more shares are combined. Similarly, in the cluster aggregation scheme, we require that there should not be too many type I attackers in each cluster to prevent the reconstruction of honest users' secrets. When analyzing the security of our scheme, we need to provide a precondition to ensure its security.

Theorem 1. *Let n be the number of clients, k be the number of clusters, l be the number of members in each cluster, t be the Shamir threshold, and q_1 be the number of type I attackers. Thus, given the security parameter λ, the constraint on k can be expressed as:*

$$k < \frac{2^{-\lambda}}{\sum_{i=t}^{min\{q_1, l\}} \frac{q_1!(n-q_1)!l!(n-l)!}{i!(q_1-i)!(l-i)!(n-q_1-l+i)!n!}} \tag{14}$$

ClusterGuard

System Setup:

- All parities (n clients and a server) are given (G, q, g) by calling $\mathbf{VSS.Setup}(1^\lambda)$
- Assign a unique identity and private key sk_u and pk_u for each client $u \in \mathcal{U}$, i.e., $1 \leq u \leq |\mathcal{U}|$
- Generate public parameters for LKH-PRF$_\mathbf{k}$: $\{\text{LWR}_{\ell,\mathfrak{q},\mathfrak{p}}, \mathbf{c}\}$, where $\mathbf{c} \in \mathbb{Z}_\mathfrak{q}^m$ and $\mathbf{k} \in \mathbb{Z}_\mathfrak{q}^\ell$ (Note that $n \cdot q < \mathfrak{q}$). The current number of training rounds is r.
- All parties also have a private authenticated channel with each other based on PKI.

* **Round 0 - Initialization:**

Client u:

- Each client u_i computes its VRF and proof $(r_u, \pi_u) \leftarrow \mathbf{Eval}(sk_u, r)$, then broadcasts it to other clients through server.
- Upon receiving $\{(r_i, \pi_i)\}_{i \in \mathcal{U}/u}$ broadcasted by other clients, perform batch verification, and if the verification fails, reject participation in this training round.
- Compute $R_{\text{total}} = \bigoplus_{i=1}^n r_i$, then determine the cluster assignment $\mathcal{C}_\varphi$ as $\varphi = \text{Cluster}(u_i) = \text{Hash}(i||R_{\text{total}}) \mod k$.
- Select a random ℓ-dimensional vector $\mathbf{s}_u \in \mathbb{Z}_\mathfrak{q}^\ell$.
- Store all messages received and values generated in this round, and move to the next round.

* **Round 1 - Sharing key:**

Client u:

- Generate $|\mathcal{C}_\varphi|$ key shares for each element s_u in $\mathbf{s}_u$: $(\{v, s_{u,v}\}_{v \in \mathcal{C}_\varphi}, \{g^{s_{u,v}}\}_{v \in \mathcal{C}_\varphi}, \{g^{a_{u,d}}\}_{d \in [0,t-1]}) \leftarrow \mathbf{VSS.Share}(s_u, t, \mathcal{C}_\varphi)$.
- Send $(v, s_{u,v})$ to client $v \in \mathcal{C}_\varphi \backslash \{u\}$, broadcast $\{g^{s_{u,v}}\}_{v \in \mathcal{C}_\varphi}$ among $\mathcal{C}_\varphi$, and broadcast $\{g^{a_{u,d}}\}_{d \in [0,t-1]}$ among $\mathcal{C}_\varphi$.
- Receive shares $(u, s_{v,u})$ and verifiable information $(g^{s_{v,u}}, \{g^{a_{v,d}}\}_{d \in [0,t-1]})$ from other clients $v \in \mathcal{C}_\varphi$, and check all shares received: $\mathbf{VSS.Verify}(u, g^{s_{v,u}}, \{g^{a_{v,d}}\}_{d \in [0,t-1]}) = 1$.
- Store all messages received, and if any of the above operations (share, check) fails, abort.

Server:

- For each cluster $\mathcal{C}_\varphi \subset \mathcal{U}$, receive $(u, \{g^{s_{u,v}}\}_{v \in \mathcal{C}_\varphi})$ from at least t clients (denote with $u \in \mathcal{C}_\varphi^{(1)} \subseteq \mathcal{C}_\varphi$ and $\mathcal{U}^{(1)} \subseteq \mathcal{U}$)
- Store all messages received.

* **Round 2 - Masked model update collection:**

Client u:

- Compute the masked input vector: $\mathbf{y}_u = \Delta \cdot \text{Encode}(\mathbf{x}_u) + \text{LKH-PRF}_{\mathbf{s}_u}(\mathbf{c})$, and send $\mathbf{y}_u$ to server.

Server:

- For each cluster $\mathcal{C}_\varphi^{(1)} \subset \mathcal{U}^{(1)}$, collect the masked model $\mathbf{y}_u$ from as least t clients (denote with $\mathcal{C}_\varphi^{(2)} \subseteq \mathcal{C}_\varphi^{(1)}$ and $\mathcal{U}^{(2)} \subseteq \mathcal{U}^{(1)}$).
- For each cluster $\mathcal{C}_\varphi^{(2)} \subset \mathcal{U}^{(2)}$, if $|\mathcal{C}_\varphi^{(2)}| \geq t$, compute $\mathbf{y}_{\mathcal{C}_\varphi^{(2)}} = \sum_{u \in \mathcal{C}_\varphi^{(2)}} \mathbf{y}_u$, and send to each client $u \in \mathcal{C}_\varphi^{(2)}$ its cluster's information, i.e., an online client list. Otherwise, remove $\mathcal{C}_\varphi^{(2)}$ from $\mathcal{U}^{(2)}$.

* **Round 3 - Unmasking:**

Client u:

- Receive from the server its new cluster's information, i.e., an online client list $\{v\}_{v \in \mathcal{C}_\varphi^{(2)}}$.
- Accumulate shares received from $v \in \mathcal{C}_\varphi^{(2)}$: $(u, s_u^*) \leftarrow \mathbf{VSS.Accum}(\{u, s_{v,u}\}_{v \in \mathcal{C}_\varphi^{(2)}})$, and send $(u, s_u^*, g^{s_u^*})$ to the server.

Server:

- For each cluster $\mathcal{C}_\varphi^{(2)} \subset \mathcal{U}^{(2)}$, receive $(u, s_u^*, g^{s_u^*})$ from at least t clients (denote with $\mathcal{C}_\varphi^{(3)} \subseteq \mathcal{C}_\varphi^{(2)}$ and $\mathcal{U}^{(3)} \subseteq \mathcal{U}^{(2)}$).
- For each client $u \in \mathcal{C}_\varphi^{(3)}$, check shares: $\mathbf{VSS.VrfyAccum}(u, g^{s_u^*}, \{g^{s_{v,u}}\}_{v \in \mathcal{C}_\varphi^{(1)}}) = 1$.
- For each cluster $\mathcal{C}_\varphi^{(3)} \subset \mathcal{U}^{(3)}$, if $|\mathcal{C}_\varphi^{(3)}| \geq t$, recover $s_{\mathcal{C}_\varphi^{(2)}} \leftarrow \mathbf{VSS.Recon}(\{u, s_u^*\}_{u \in \mathcal{C}_\varphi^{(3)}})$. Otherwise, remove $\mathcal{C}_\varphi^{(2)}$ from $\mathcal{U}^{(2)}$.
- For each cluster $\mathcal{C}_\varphi^{(2)} \subset \mathcal{U}^{(2)}$, compute each element s in a ℓ-dimensional vector $\mathbf{s}$ where $s = \sum_{\mathcal{C}_\varphi^{(2)} \subset \mathcal{U}^{(2)}} s_{\mathcal{C}_\varphi^{(2)}} = \sum_{u \in \mathcal{U}^{(2)}} s_u$.
- Unmask the aggregated model $\hat{\mathbf{z}} = \frac{1}{n'\Delta}(\sum_{\mathcal{C}_\varphi^{(2)} \subset \mathcal{U}^{(2)}} \mathbf{y}_{\mathcal{C}_\varphi^{(2)}} - \text{LKH-PRF}_\mathbf{s}(\mathbf{c}))$.
- If any of the above operation (check, recon, LKH-PRF, unmask) fails, abort.
- Decode $\hat{\mathbf{z}}$ to output $\mathbf{z} = \frac{1}{n'} \sum_{u \in \mathcal{U}^{(2)}} \mathbf{x}_u$ where $n' = |\{u \in \mathcal{U}^{(2)}\}|$.

Fig. 3. The description of ClusterGuard protocol for one FL round. Red, underlined parts are optional to guarantee public verification.

The full proof of the above theorem, is given in Appendix. Then the security of our protocol is given by the following theorem:

Theorem 2 (Privacy under Semi-Honest Setting)**.** *There exists a PPT simulator SIM such that for all λ, t, $\mathcal{U}$, x_u, and $\mathcal{C}$ such that $\mathcal{C} \subseteq \mathcal{U} \cup \mathcal{S}$ and $|\mathcal{C} \setminus \mathcal{S}| < t$, the output of SIM is computationally indistinguishable from the output of $\mathsf{REAL}_{\mathcal{C}}^{\mathcal{U},t,\lambda}$:*

$$\mathsf{REAL}_{\mathcal{C}}^{\mathcal{U},t,\lambda}(\mathcal{A}_{\mathcal{C}}, x_{\mathcal{U}\setminus\mathcal{C}}) \approx_c SIM_{\mathcal{C}}^{\mathcal{U},t,\lambda,\mathsf{IDEAL}(t,x_u)_{u\in\mathcal{U}\setminus\mathcal{C}}}(\mathcal{A}_{\mathcal{C}})$$

This theorem along with the next one is proved with the proof presented in Appendix.

B. Privacy in the Malicious Setting

Similar to the semi-honest case, we also prove privacy under the malicious setting by showing that the joint view of the server and any subset of fewer than t clients reveals no information about the inputs of other clients, except for what can be inferred from the output of the computation.

Theorem 3 (Privacy under Malicious Setting)**.** *There exists a PPT simulator SIM such that for all PPT adversaries $\mathcal{A}_{\mathcal{C}}$, all λ, t, $\mathcal{U}$, x_u, and $\mathcal{C}$ such that $\mathcal{C} \subseteq \mathcal{U} \cup \mathcal{S}$ and $|\mathcal{C} \setminus \mathcal{S}| < t$, the output of SIM is computationally indistinguishable from the output of $\mathsf{REAL}_{\mathcal{C}}^{\mathcal{U},t,\lambda}$:*

$$\mathsf{REAL}_{\mathcal{C}}^{\mathcal{U},t,\lambda}(\mathcal{A}_{\mathcal{C}}, x_{\mathcal{U}\setminus\mathcal{C}}) \approx_c SIM_{\mathcal{C}}^{\mathcal{U},t,\lambda,\mathsf{IDEAL}(t,x_u)_{u\in\mathcal{U}\setminus\mathcal{C}}}(\mathcal{A}_{\mathcal{C}})$$

The full proof of the above theorem, which also proves Theorem 1, is given in Appendix.

C. Convergence Analysis

The goal of federated learning is to collaboratively train a global model across multiple clients such that the expected loss function over all clients is minimized. The formal definition is as follows:

$$\theta^* = \min_{\theta} \mathbb{E}_{\xi\sim\mathcal{D}}\left[\mathcal{L}(\theta, \xi)\right] \tag{15}$$

where θ^* denotes the optimal global model, λ represents the training data sampled from $\mathcal{D} = \cup_{u\in\mathcal{U}}\mathcal{D}_u$, $\mathcal{L}(\theta, \xi)$ is the loss function of the model θ under the data ξ.

We use $F(\theta)$ to represent $\mathbb{E}_{\xi\sim\mathcal{D}}\left[\mathcal{L}(\theta, \xi)\right]$. Before providing the convergence proof, we follow the common assumptions for distributed optimization convergence analysis given in [41].

Assumption 1. *$L-Lipschitz$ continuity of the gradient: the gradient of $F(\theta)$ is Lipschitz continuous with constant $L > 0$, for all $\theta, \theta' \in \mathbb{R}^d$,*

$$\|\nabla F(\theta') - \nabla F(\theta)\| \leq L\|\theta' - \theta\| \tag{16}$$

It can also be expressed in the following form:

$$F(\theta') \leq F(\theta) + \nabla F(\theta)^\top(\theta' - \theta) + \frac{L}{2}\|\theta' - \theta\|^2 \tag{17}$$

Assumption 2. *Variance between local and global model updates: There exists a constant σ_g^2 such that for all clients u_i, the variance between the client model updates $\nabla F_i(\theta)$ and the global model update $\nabla F(\theta)$ is bounded:*

$$\mathbb{E}\left[\|\nabla F_i(\theta) - \nabla F(\theta)\|^2\right] \leq \sigma_g^2 \tag{18}$$

The stochastic gradient of each worker (client) has a bounded variance uniformly, satisfying:

$$\mathbb{E}_{\xi_i\sim\mathcal{D}_i}\left[\|\nabla F_i(\theta; \xi_i) - \nabla F_i(\theta)\|^2\right] \leq \sigma_c^2 \tag{19}$$

where $F_i(\theta; \xi_i)$ is the stochastic gradient computed by worker u_i based on the sample ξ_i, and $\nabla F_i(\theta)$ is the expesected gradient for worker u_i.

Our robustness algorithm uses model updates trained on trusted root dataset or datasets from honest clients to guide trust. It leverages the similarity between the aggregation of each cluster and this trusted model update to filter out the clusters containing malicious clients. Referencing the work [42] [41], any robust gradient aggregation rule inevitably results in an error in the average honest gradient. Therefore, we propose the following assumption:

Assumption 3. *For the problem with $(1-\beta)k$ benign clusters (without malicious clients) (denoted by $\mathcal{G}_g$) and βk Byzantine clusters (with malicious clients), suppose at most δk Byzantine clusters can bypass the robustness algorithm at each iteration. Assume the existence of positive constants c and b such that the model update output $\boldsymbol{RobustAgg}(\{\boldsymbol{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi\in\mathcal{C}})$ from the robustness algorithm satisfies the following conditions:*

Bounded Bias:

$$\begin{aligned}\mathbb{E}\left[\|\boldsymbol{RobustAgg}(\{\boldsymbol{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi\in\mathcal{C}}) - \bar{\boldsymbol{w}}\|\right]^2 \leq \\ c\delta \sup_{\mathcal{C}_i,\mathcal{C}_j\in\mathcal{G}_g} \mathbb{E}\left[\|\boldsymbol{w}_r^{(\mathcal{C}_i)} - \boldsymbol{w}_r^{(\mathcal{C}_j)}\|^2\right]\end{aligned} \tag{20}$$

where $\bar{\boldsymbol{w}}_r = \frac{1}{|\mathcal{G}_g|}\sum_{\mathcal{C}_i\in\mathcal{G}_g}\boldsymbol{w}_r^{(\mathcal{C}_i)}$.

Bounded Variance:

$$Var\left[\|\boldsymbol{RobustAgg}(\{\boldsymbol{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi\in\mathcal{C}})\|\right] \leq b^2 \tag{21}$$

Here, δ is the fraction of Byzantine clusters that can bypass the robustness algorithm, and $0 \leq \delta < \beta < 1$.

Theorem 4. *Consider the function F that satisfies assumptions 1 and 2. Assume there is a robust aggregation scheme that satisfies assumption 3. Let the number of type II attackers be q_2, and $F^* = min_\theta\nabla(\theta)$. There exists $\eta < \frac{1}{L}(1 - \frac{\sqrt{\delta}}{4} - \frac{q_2}{2n})$ for which, after R rounds, the following condition holds:*

$$\frac{1}{R}\sum_{r=0}^{R-1}\mathbb{E}\|\nabla F(\theta_r)\|^2 \leq \frac{(F(\theta_0) - F^*)}{M\eta R} + \frac{L\eta}{M}\Delta_3 + \frac{1}{M}\Delta_4 \tag{22}$$

where $M = 1 - \frac{\sqrt{\delta}}{4} - \frac{q_2}{2n} - L\eta, \Delta_4 = \frac{8c\sqrt{\delta}}{l}(\sigma_c^2 + \sigma_g^2) + \frac{nq_2\sigma_g^2}{2(n-q_2)^2}, \Delta_3 = 2\Delta_1 + 4\Delta_2 + 12\frac{n-q_2-(1-\beta)kl}{(n-q_2-1)(1-\beta)kl}(\sigma_c^2 + \sigma_g^2 + \Delta_2), \Delta_1 = \frac{8c\delta}{l}(\sigma_c^2 + \sigma_g^2) + b^2, \Delta_2 = \frac{q_2^2\sigma_g^2}{(n-q_2)^2} + \frac{2\sigma_c^2}{n-q_2}$.

Remark. The theorem provides an insightful analysis of federated learning convergence under adversarial settings, with and without attacks. In the ideal case where $\delta = \beta = 0$ and $q_2 = 0$, indicating no adversarial behavior, the convergence bound simplifies significantly. In this scenario, the parameter $M = 1 - L\eta$ depends only on the smoothness constant L and the learning rate η, with the error terms $\Delta_3 = 2b^2 + \frac{8\delta_c^2}{n}, \Delta_4 = 0$. At this point, setting a sufficiently small learning rate can ensure convergence without extra errors. Specifically,

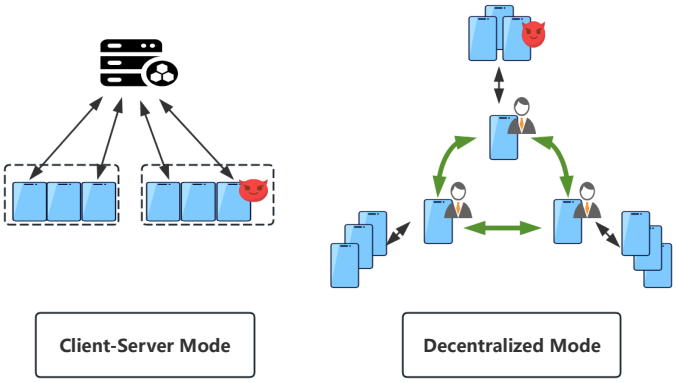


Fig. 4. The figure shows two variations of our federated learning framework: a centralized server-client model and a decentralized model.

if we set $\eta = \mathcal{O}(\frac{1}{\sqrt{R}})$, and in this case, we can obtain $\frac{1}{R}\sum_{r=0}^{R-1}\mathbb{E}\|\nabla F(\theta_r)\|^2 \leq \mathcal{O}(\frac{F(\theta_0)-F^*}{\sqrt{R}-L} + \frac{L}{\sqrt{R}-L}\Delta_3)$. At this point, the ClusterGuard scheme will converge with a rate of $\mathcal{O}(\frac{1}{\sqrt{R}})$, and the extra error terms will also be eliminated.

In the general case, where $\delta > 0$, $\beta > 0$, and $q_2 > 0$, the convergence behavior is influenced by adversarial robustness and the number of attackers. We still take $\eta = \mathcal{O}(\frac{1}{\sqrt{R}})$, and we can obtain: $\frac{1}{R}\sum_{r=0}^{R-1}\mathbb{E}\|\nabla F(\theta_r)\|^2 \leq \mathcal{O}(\frac{(F(\theta_0)-F^*)}{\left(\sqrt{R}(1-\frac{\sqrt{\delta}}{4}-\frac{q_2}{2n})-L\right)} + \frac{L\Delta_3}{\left(\sqrt{R}(1-\frac{\sqrt{\delta}}{4}-\frac{q_2}{2n})-L\right)} + \frac{\Delta_4}{1-\frac{\sqrt{\delta}}{4}-\frac{q_2}{2n}-\frac{L}{\sqrt{R}}})$. The overall convergence rate can be observed as $\mathcal{O}(\frac{1}{\sqrt{R}})$, with an additional error term $\frac{\Delta_4}{1-\frac{\sqrt{\delta}}{4}-\frac{q_2}{2n}}$. When malicious clients (i.e., q_2 increases), the robustness of the aggregation algorithm is insufficient (i.e., δ increases), or the heterogeneity among client updates increases (i.e., σ_g^2, σ_c^2 increases, see inequality 24), the extra error term will grow.

VI. Discussion

A. Decentralized Mode Extension

Our cluster-based federated learning scheme can be extended to adapt to decentralized federated learning scenarios 4. Accordingly, we need to revise the protocol as follows:

1) VRF-Based Clustering: Each client holds a key pair (pk_i, sk_i) and generates a tuple $(r_i, \pi_i, \text{round})$, broadcasting these values along with the current round number. Upon receiving the broadcasts, other clients perform batch verification. If the verification succeeds, the clients determine their clusters $\{\mathcal{C}_\varphi\}_{\varphi\in[0,k-1]}$ and leaders $\{I^\varphi_{\text{Leader}}\}_{\varphi\in[0,k-1]}$ for the current round as follows:

1) Compute the aggregated randomness:

$$R_{\text{total}} = \bigoplus_{i=1}^{n} r_i$$

where $\bigoplus$ denotes the bitwise XOR operation.

2) Assign clusters using a hash function:

$$\varphi_i = \text{Hash}(i \| R_{\text{total}} \| r) \mod k$$

where φ_i denotes the cluster index of client u_i, and $\mathcal{C}_\varphi = \{u_i \mid \varphi_i = \varphi\}$ is the set of clients in cluster φ, r is current federated learning round.

3) Select a leader for each cluster using a pseudorandom function (PRF):

$$I^\varphi_{\text{Leader}} = \text{PRF}(R_{\text{total}}, \text{round} \| \varphi) \mod |\mathcal{C}_\varphi|$$

where $I^\varphi_{\text{Leader}}$ represents the index of the leader in cluster $\mathcal{C}_\varphi$.

The leaders $\{I^\varphi_{\text{Leader}}\}$ securely aggregate the updates $\mathbf{z}_{\mathcal{C}_\varphi}$ from their respective clusters $\mathcal{C}_\varphi$. After completing the aggregation, the leaders exchange their aggregated updates $\{\mathbf{z}_{\mathcal{C}_\varphi}\}_{\varphi\in[0,k-1]}$ and then distribute the exchanged updates uniformly to all clients (refer to Figure 4).

2) Robust Aggregation: Each client refines the received updates $\{\mathbf{z}_{\mathcal{C}_\varphi}\}_{\varphi\in[0,k-1]}$ by excluding those deemed malicious, based on its local dataset $\mathcal{D}_i$:

$$\mathbf{z}^{\text{filtered}}_{\mathcal{C}_\varphi} = f(\{\mathbf{z}_{\mathcal{C}_\varphi}\}_{\varphi\in[0,k-1]}, \mathcal{D}_i),$$

where f is a filtering function (in robust aggregation) designed to identify and remove anomalous updates, referencing the robustness Algorithm 1. But unlike Algorithm 1, it relies on its own data for trust instead of a trusted root dataset maintained by the server.

3) Evaluation: In this paradigm, the security analysis against Type-I attackers remains consistent with the previous analysis, as the multiple leaders collectively assume the role of the server. For Type-II attackers, we consider two scenarios:

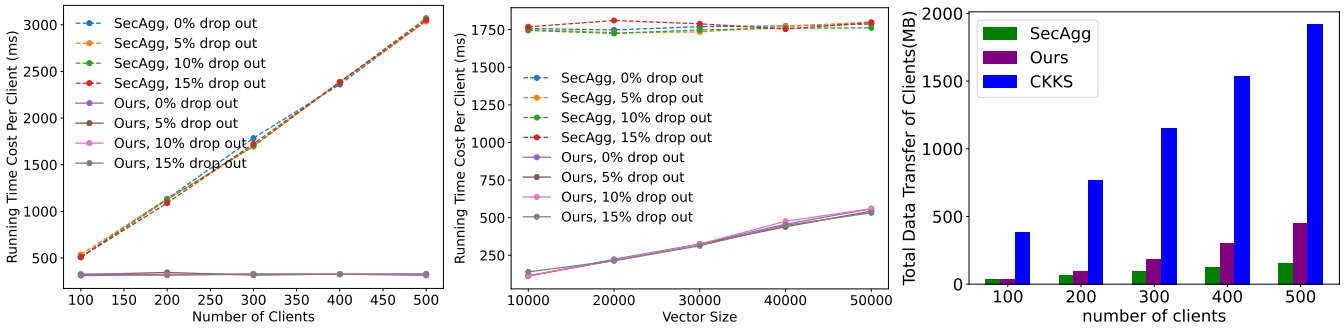
1. Non-leader clients: In this case, malicious attacks can still be countered using the previously established defense mechanisms. However, the trust in this scenario does not stem from a trusted dataset collected by the server but rather from the client's own local dataset.

2. Leader clients: Leaders may attempt to send entirely incorrect model updates to the clients. Fortunately, the baffle mechanism ensures that if all the received model updates exhibit extremely low similarity to the client's own updates, the client can reject these updates.

Detailed experimental results for these scenarios are illustrated in appendix.

VII. Evaluation

We analyze the robust aggregation of ClusterGuard with n clients and a single server, where each client holds a data vector of size m and a secret key of size ℓ. For clients, the computation cost is $O(\ell l^2 + m\ell + n)$, which includes generating a secret vector, creating l secret shares, computing the model update mask and verifying all VRFs. The communication cost is $O(\ell l + m + n)$, covering the exchange of secret shares, verification, masked updates, and VRFs. For the server, the computation cost is $O(nl + \frac{nm\ell}{l})$, dominated by reconstructing cluster secret keys and calculating masks, which can reduce to $O(nt + m\ell + \frac{n\ell}{l})$ without robustness detection. If some clients drop out, the cost further decreases. The communication cost is $O(mn + \ell n)$ for receiving masked updates, secret shares, and sending cluster information.

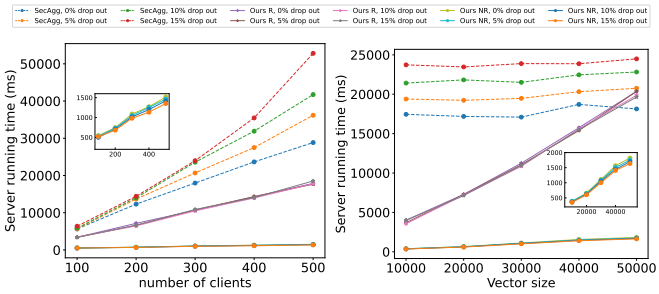


(a) Running time cost, as the number of clients increases.

(b) Running time cost, as the size of model update vector increases.

(c) Total data transfer, as the number of clients increases.

Fig. 5. Client protocol performance comparison. The data vector size remains fixed at 30K in (a) and (c), while the number of clients is fixed at 300 in (b).



(a) Server time cost vs number of clients, the vector size is fixed at 30K.

(b) Server time cost vs vector size, the number of clients is fixed at 300.

Fig. 6. Server protocol performance comparison. R denotes the server with robust aggregation, while NR denotes non-robust aggregation.

A. Prototype Performance

Our experimental setup consists of two parts. First, we evaluate the performance of the secure aggregation protocol without considering users' local training processes. Second, we assess federated learning accuracy and robustness using the secure aggregation protocol, involving complete federated learning training.

1) Experiment Setup: We implement the aggregation protocol in Python. The lattice based key-homomorphic pseudorandom function (LKH-PRF) is implemented using the go program accelerated by goroutine. Model updates are set to 16-bit accuracy. For $\mathrm{LWR}_{\ell,\mathfrak{q},\mathfrak{p}}$ of LKH-PRF, we set $\ell = 512$, $\mathfrak{p} = 2^{32}$, and $\mathfrak{q}$ as a 64-bit prime close to 2^{64}. Using the LWE estimator [43] [8], we estimate that the hardness exceeds 2^{128}. For each cluster, we set the default values as $l = 10$, $t = 0.8l$. For the cluster election based on VRF, we implement it using GoFE [1]. The experiments are conducted on a Windows desktop with AMD Ryzen 9 5900HX (3.3 GHz), with 32 GB of RAM and NVIDIA GeForce RTX 3080 Laptop GPU.

2) Experimental Performance: Figures 5 and 6 depict our experimental performance results. In comparison with SecAgg's (Bonawitz et al. [2]) specific performance results in Figure 5 (a), we observe a significant improvement in client running time. Each client's running time remains nearly constant as the number of clients increases, primarily dependent on the time taken for single mask computation.

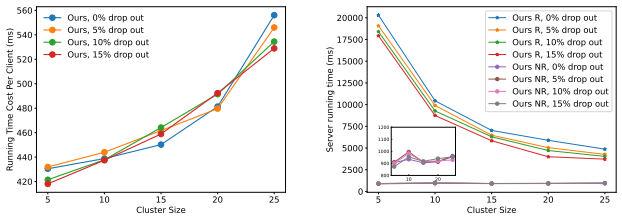
[1]GoFE can be referred to: https://github.com/fentec-project/gofe

In Figure 5 (b), we note that our protocol exhibits faster growth in client running time as the aggregation vector size increases, albeit still outperforming SecAgg. This increase is attributed to the fact that calculating a single mask entails m ℓ-dimensional vector inner product operations, with computation time directly proportional to vector size. In contrast, SecAgg employs pseudorandom number calculation under AES CTR mode (utilizing one-time filling), which does not significantly vary with vector size as the number of clients is fixed. Figure 5 (c) illustrates the total data transfer of our protocol compared to SecAgg and CKKS based homomorphic encryption schemes [44]. Given that our communication units are secrect vectors, the total data transfer is higher than SecAgg, but still significantly lower than the communication overhead associated with homomorphic encryption-based aggregation protocols.

Figure 6 (a) compares the server running time between the two protocols as the number of clients increases. Our protocol demonstrates a clear advantage both when robustness detection is considered and when it is not. When robustness detection is taken into account, it shows increased server running time due to the total mask calculation for all clusters. However, as dropped clients increase, the server time for SecAgg rise, while our server time remains unaffected. This is because SecAgg requires additional masking recovery operations for dropped clients, unlike our protocol, which only needs limited client communication to reconstruct the total key.

Figure 6 (b) demonstrates the relationship between server running time and model update size. Without considering robustness detection, our protocol only needs to recover and accumulate cluster keys once, resulting in significantly better running times compared to SecAgg. When robustness is considered, the total running time of our protocol increases linearly with the size of the model update. Lastly, Figure 7 shows the impact of cluster size on runtime. Reducing the cluster size decreases client runtimes, while increasing it reduces server runtime with robustness detection. Adjusting the cluster size allows flexible control over server and client complexity.

We assessed ClusterGuard's defense against DLG attacks using LeNet and non-i.i.d. data, with a batch size of 8. Figure 9 shows reconstruction results across 500 epochs for



(a) Running time per client, as the cluster size increases.

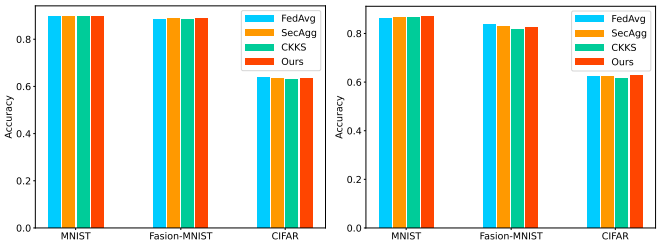
(b) Total server running time, as the cluster size increases.

Fig. 7. The influence of cluster size on the performance of the protocol. The data vector size remains fixed at 30K and the number of clients is fixed at 300. R denotes the server with robust aggregation, while NR denotes non-robust aggregation.

various cluster sizes. Without protection (FedAvg), recognizable images emerged after 100 epochs (Figures 910(a), 10(c)), showing vulnerability to attacks. In contrast, ClusterGuard, tested with cluster sizes 2–20, produced noisy, unrecognizable reconstructions (e.g., size 4 in Figures 9(b), 10(d)), preventing data leakage. These results confirm ClusterGuard effectively resists DLG attacks and enhances privacy.

B. Accuracy and Robustness Performance

1) Datasets and model configurations: We implement a complete federated learning process using PyTorch, based on three neural networks with varying architectures and sizes. We evaluate model accuracy and resistance to poisoning attacks using our ClusterGuard protocol on three datasets: MNIST, Fashion-MNIST, and CIFAR. For MNIST, we utilize a Logistic Regression model with 7,850 parameters. For Fashion-MNIST, we employ the standard LeNet [45] architecture, which contains 61,706 parameters. Finally, for CIFAR, we use a large-scale CNN with 160,856 parameters.

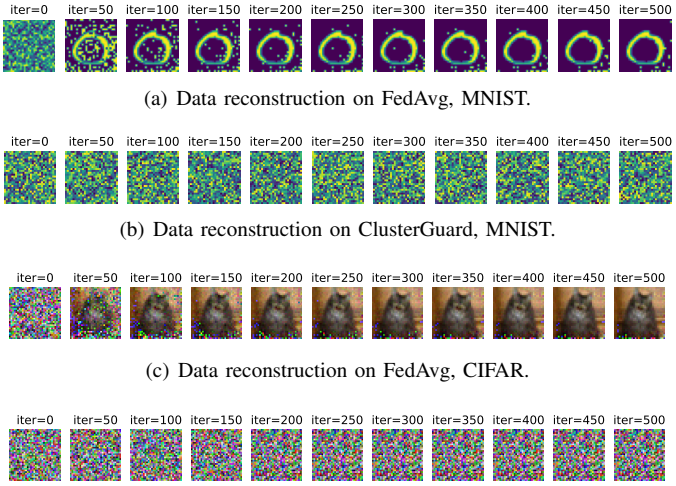


(a) IID.

(b) Non-IID.

Fig. 8. Accuracy comparison under the different datasets with two different distributions.

2) Federated Learning Settings: We consider two scenarios: IID and non-IID data distribution. For IID data, the dataset is evenly distributed among clients, while for non-IID data, we use a Dirichlet distribution with a hyperparameter $\alpha = 0.5$. In our experiments, we adopt a default non-IID setting. We involve $[100, 100, 60]$ clients for MNIST, Fashion-MNIST and CIFAR-10. Each client's local learning rate is $[0.01, 0.01, 0.05]$, with 5 local iterations and a batch size of 32. By default, we set $l = t = 4$.



(a) Data reconstruction on FedAvg, MNIST.

(b) Data reconstruction on ClusterGuard, MNIST.

(c) Data reconstruction on FedAvg, CIFAR.

(d) Data reconstruction on ClusterGuard, CIFAR.

Fig. 9. The reconstruction results of the DLG attack.

3) Poisoning Attacks and Defense: In our algorithm experiment, we uniformly extract 500 samples as the root data set $\mathcal{D}_0$, like FLtrust [28]. For our robust algorithm, we set parameters β, bl_1, bl_2 to $1.5, 0.05, 2$ respectively. We consider the following common poisoning attacks.

Label flipping attack [21], [22]: specifically, malicious clients are trained using incorrect labels, i.e., any label from $\{0, ..., 9\}$ is transformed into $9-$label.

Noise attack: malicious clients perturb their gradients by adding Gaussian noise with mean 0 and standard deviation , i.e., $\mathbf{w}_m = \mathbf{w} + \mathcal{N}(\mu, \sigma^2)$. We take $\mu = 0, \sigma = 0.5$.

Sign flipping attack [46]: malicious clients send updates in the opposite direction to those of benign clients, i.e., $\mathbf{w}_m = -\delta\mathbf{w}$. We take $\delta = 2$.

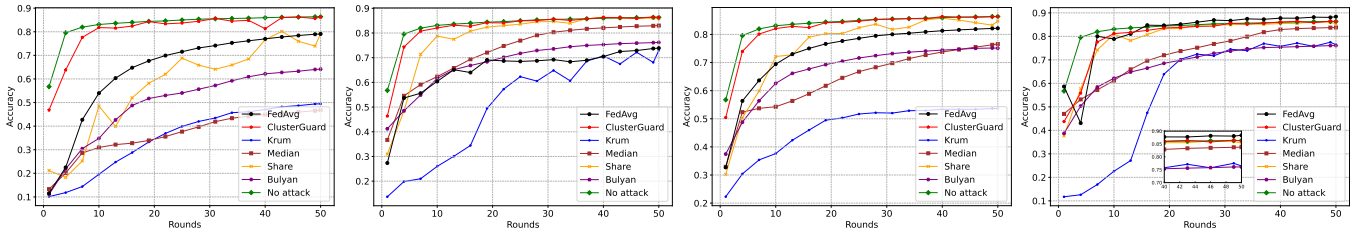
Scale attack: We consider an attacker uploading a scaled version of the benign model update. Typically, the scaling factor satisfies $\gg 1$. In our experiment, we set $\gamma = 20$.

4) Experimental Performance: We first discuss the comparison of the model accuracy of our protocol with others. Since the error of LKH-PRF has been processed, the influence of the model accuracy only comes from the encoding and decoding of the model updates. As shown in Figure 8, our protocol has the same model accuracy and convergence speed as plaintext aggregation (FedAvg [13]), and is also consistent with other secure aggregation protocols.

TABLE III
THE IMPACT OF ALGORITHM-RELATED PARAMETERS ON MODEL ACCURACY (%).

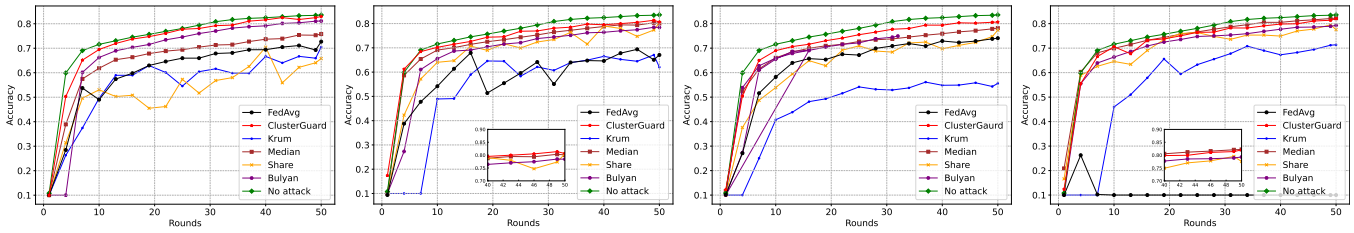
Dataset	Scheme	Cluster size				
		3	4	5	6	10
Fasion-MNIST	Share	66.09	71.55	70.92	67.26	56.06
	ClusterGuard	80.25	82.39	82.20	82.30	78.94
CIFAR-10	Share	33.62	51.49	44.11	44.42	28.35
	ClusterGuard	65.39	64.11	64.27	63.42	62.09
Dataset	**Scheme**	**Threshold parameter β**				
		1	1.5	2	2.5	3
Fasion-MNIST	ClusterGuard	82.06	82.19	82.09	77.98	77.26
CIFAR-10	ClusterGuard	63.98	64.11	63.77	60.54	60.38

Figures 10, 11, and 12 illustrate the comparison of different



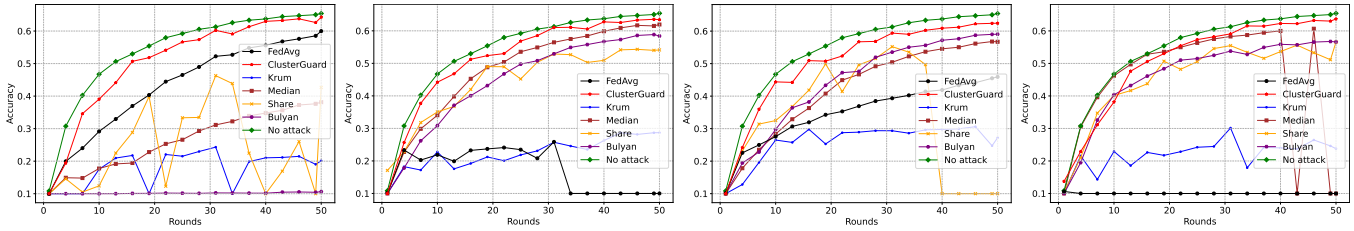
(a) Label Flipping Attack. (b) Noise Attack. (c) Sign flipping Attack. (d) Scale Attack.

Fig. 10. Accuracy comparison. Assuming 20% of the clients are malicious. The experimental setup involves the MNIST dataset and LR model.



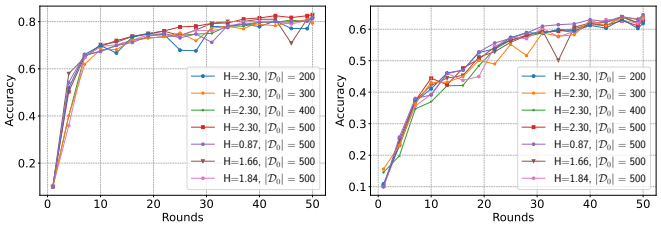
(a) Label Flipping Attack. (b) Noise Attack. (c) Sign flipping Attack. (d) Scale Attack.

Fig. 11. Accuracy comparison. Assuming 20% of the clients are malicious. The experimental setup involves the Fasion-MNIST dataset and LeNet model.



(a) Label Flipping Attack. (b) Noise Attack. (c) Sign flipping Attack. (d) Scale Attack.

Fig. 12. Accuracy comparison. Assuming 20% of the clients are malicious. The experimental setup involves the CIFAR-10 dataset and CNN model.



(a) Fasion-MNIST. (b) CIFAR-10.

Fig. 13. The impact of root dataset with different distributions on the accuracy of global model.

defense algorithms against various attacks when the proportion of malicious clients is 20%. In our experiments, we default to using a non-IID setting for defense evaluation. It can be observed that our aggregation method achieves significantly better convergence accuracy compared to other robust defense methods for various poisoning attacks.

We further discuss the relationship between parameter selection and robustness of our proposed scheme in Table III, with a default setting of 20% malicious clients performing label-flipping attacks. It shows that our ClusterGuard scheme consistently outperforms the baseline. However, when the number of attackers is significantly lower than the number of clusters, reducing the cluster size does not provide substantial benefits, as the server can already accurately locate the clusters containing adversaries. Conversely, increasing the cluster size, which causes the number of attackers to exceed the number of clusters, results in a loss of model performance. At the same time, the threshold parameter needs to be selected appropriately. When the threshold parameter is large enough, the algorithm degenerates into the FedAvg [13]. Figure 13 discusses the impact of the root dataset on the effectiveness of our robust defense approach. We conducted experiments on root datasets of varying scales and distributions, using label-flipping attacks as the evaluation scenario. We define the uniformity of the root dataset using distribution entropy, specifically, $H(\mathcal{D}) = -\sum_{i=1}^{n'} p_i \log p_i$ where p_i is the probability of the i-th class, n' is the total number of classes. The results demonstrate that differences in data distribution within the root dataset $\mathcal{D}_0$ do not affect the correctness of the algorithm. The influence of data distribution differences on the results can be relatively ignored, which further highlights the robustness and practicality of our approach. The server does not need to obtain $\mathcal{D}_0$ with a large number of samples, and the data distribution of does not need to be fully aligned with

that of the clients. This reduces the difficulty of acquiring the root dataset for the server.

VIII. Conclusion

In summary, this paper focused on the critical role of secure aggregation in federated learning. We highlighted the limitations of existing protocols like SecAgg, particularly in terms of computational efficiency, network stability, and security against malicious attacks.

To address these concerns, we introduced ClusterGaurd, a novel secure aggregation scheme that emphasizes Byzantine robustness. ClusterGaurd leverages lattice-based key-homomorphic pseudorandom functions for efficient and secure aggregation while countering malicious poisoning attacks with a VRF based clustered aggregation. Our extensive experiments validate ClusterGaurd's adaptability and resilience.

References

[1] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems*, vol. 32, 2019.

[2] K. Bonawitz, V. Ivanov, B. Kreuter *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.

[3] J. H. Bell, K. A. Bonawitz, A. Gascón *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.

[4] S. Kadhe, N. Rajaraman, O. O. Koyluoglu *et al.*, "Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning," *arXiv preprint arXiv:2009.11248*, 2020.

[5] J. So, B. Güler, and A. S. Avestimehr, "Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 479–489, 2021.

[6] Y. Ma, J. Woods, S. Angel, A. Polychroniadou, and T. Rabin, "Flamingo: Multi-round single-server secure aggregation with applications to private federated learning," in *2023 IEEE Symposium on Security and Privacy (SP)*, 2023, pp. 477–496.

[7] Z. Liu, J. Guo, K.-Y. Lam *et al.*, "Efficient dropout-resilient aggregation for privacy-preserving machine learning," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1839–1854, 2022.

[8] Z. Liu, S. Chen, J. Ye *et al.*, "Sash: efficient secure aggregation based on shprg for federated learning," in *Uncertainty in Artificial Intelligence*. PMLR, 2022, pp. 1243–1252.

[9] H. Li, H. Lin, A. Polychroniadou *et al.*, "Lerna: Secure single-server aggregation via key-homomorphic masking," Cryptology ePrint Archive, Paper 2023/1936, 2023.

[10] J. Bell, A. Gascón, T. Lepoint *et al.*, "{ACORN}: Input validation for secure aggregation," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 4805–4822.

[11] T. Stevens, C. Skalka, C. Vincent *et al.*, "Efficient differentially private secure aggregation for federated learning via hardness of learning with errors," in *31st USENIX Security Symposium*, 2022, pp. 1379–1395.

[12] Z. Zhang, J. Li, S. Yu, and C. Makaya, "Safelearning: Secure aggregation in federated learning with backdoor detectability," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3289–3304, 2023.

[13] B. McMahan, E. Moore, D. Ramage *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[14] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *28th Annual Symposium on Foundations of Computer Science*. IEEE, 1987, pp. 427–438.

[15] J. Ernst and A. Koch, "Private stream aggregation with labels in the standard model." *Proc. Priv. Enhancing Technol.*, vol. 2021, no. 4, pp. 117–138, 2021.

[16] D. Boneh, K. Lewi, H. Montgomery *et al.*, "Key homomorphic prfs and their applications," in *Annual Cryptology Conference*. Springer, 2013, pp. 410–428.

[17] Y. Shi, T. Luo, J. Liang, M. H. Au, and X. Luo, "Obfuscating verifiable random functions for proof-of-stake blockchains," *IEEE Transactions on Dependable and Secure Computing*, 2023.

[18] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 19–38.

[19] P. Xu, M. Hu, T. Chen *et al.*, "Laf: Lattice-based and communication-efficient federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2483–2496, 2022.

[20] B. Choi, J.-y. Sohn, D.-J. Han *et al.*, "Communication-computation efficient secure aggregation for federated learning," *arXiv preprint arXiv:2012.05433*, 2020.

[21] V. Tolpegin, S. Truex, M. E. Gursoy *et al.*, "Data poisoning attacks against federated learning systems," in *25th European Symposium on Research in Computer Security, ESORICS 2020*. Springer, 2020, pp. 480–501.

[22] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," *Advances in neural information processing systems*, vol. 30, 2017.

[23] J. Hou, F. Wang, C. Wei *et al.*, "Credibility assessment based byzantine-resilient decentralized learning," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[24] M. Fang, X. Cao, J. Jia *et al.*, "Local model poisoning attacks to Byzantine-Robust federated learning," in *29th USENIX Security Symposium*. USENIX Association, Aug. 2020, pp. 1605–1622.

[25] P. Blanchard, E. M. El Mhamdi, R. Guerraoui *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.

[26] D. Yin, Y. Chen, R. Kannan *et al.*, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[27] R. Guerraoui, S. Rouault *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3521–3530.

[28] X. Cao, M. Fang, J. Liu *et al.*, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," *arXiv preprint arXiv:2012.13995*, 2020.

[29] D. N. Yaldiz, T. Zhang, and S. Avestimehr, "Secure federated learning against model poisoning attacks via client filtering," *arXiv preprint arXiv:2304.00160*, 2023.

[30] M. Hao, H. Li, G. Xu, H. Chen, and T. Zhang, "Efficient, private and robust federated learning," 2021, pp. 45–60.

[31] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, "Privacy-enhanced federated learning against poisoning adversaries," vol. 16, pp. 4574–4588, 2021.

[32] Y. Dong, X. Chen, K. Li, D. Wang, and S. Zeng, "Flod: Oblivious defender for private byzantine-robust federated learning with dishonest-majority," 2021, pp. 497–518.

[33] Y. Miao, Z. Liu, H. Li *et al.*, "Privacy-preserving byzantine-robust federated learning via blockchain systems," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2848–2861, 2022.

[34] Z. Ma, J. Ma, Y. Miao, Y. Li, and R. H. Deng, "Shieldfl: Mitigating model poisoning attacks in privacy-preserving federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1639–1654, 2022.

[35] Y. Miao, X. Yan, X. Li, S. Xu, X. Liu, H. Li, and R. H. Deng, "Rfed: Robustness-enhanced privacy-preserving federated learning against poisoning attack," vol. 19, pp. 5814–5827, 2024.

[36] X. Hao, C. Lin, W. Dong, X. Huang, and H. Xiong, "Robust and secure federated learning against hybrid attacks: A generic architecture," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 1576–1588, 2024.

[37] O. Aramoon, P.-Y. Chen, G. Qu *et al.*, "Meta federated learning," *arXiv preprint arXiv:2102.05561*, 2021.

[38] Y. Li, X. Wang, R. Sun *et al.*, "Trustiness-based hierarchical decentralized federated learning," *Knowledge-Based Systems*, vol. 276, p. 110763, 2023.

[39] Y.-R. Yang, K. Wang, and W.-J. Li, "Fedrep: A byzantine-robust, communication-efficient and privacy-preserving framework for federated learning," *arXiv preprint arXiv:2303.05206*, 2023.

[40] Z. Zhang, J. Li, S. Yu, and C. Makaya, "Safelearning: Secure aggregation in federated learning with backdoor detectability," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3289–3304, 2023.

[41] J. Xu, S.-L. Huang, L. Song, and T. Lan, "Byzantine-robust federated learning through collaborative malicious gradient filtering," in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2022, pp. 1223–1235.

[42] S. P. Karimireddy, L. He, and M. Jaggi, "Learning from history for byzantine robust optimization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5311–5319.
[43] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *Journal of Mathematical Cryptology*, vol. 9, no. 3, pp. 169–203, 2015.
[44] J. H. Cheon, A. Kim, M. Kim *et al.*, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology–ASIACRYPT*. Springer, 2017, pp. 409–437.
[45] Y. LeCun, L. Bottou, Y. Bengio *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[46] C. Xie, O. Koyejo, and I. Gupta, "Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation," in *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, ser. Proceedings of Machine Learning Research, R. P. Adams and V. Gogate, Eds., vol. 115. PMLR, 22–25 Jul 2020, pp. 261–270. [Online]. Available: https://proceedings.mlr.press/v115/xie20a.html
[47] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *arXiv preprint arXiv:2003.00295*, 2020.
[48] L. Lei, C. Ju, J. Chen, and M. I. Jordan, "Non-convex finite-sum optimization via scsg methods," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

APPENDIX

Proof for Theorem 1

Theorem 1. *Let n be the number of clients, k be the number of clusters, l be the number of members in each cluster, t be the Shamir threshold, and q_1 be the number of type I attackers. Thus, given the security parameter λ, the constraint on k can be expressed as:*

$$k < \frac{2^{-\lambda}}{\sum_{i=t}^{min\{q_1,l\}} \frac{q_1!(n-q_1)!l!(n-l)!}{i!(q_1-i)!(l-i)!(n-q_1-l+i)!n!}}$$

Proof. **Case 1:** $q_1 < t$. In this case, the number of malicious clients in a cluster cannot exceed t, since there are fewer than t malicious clients in total. Therefore, the probability that a cluster contains more than t malicious clients is zero:

$$\Pr(X_i \geq t) = 0$$

Thus, for this case, the privacy constraint is trivially satisfied, and no further constraint on k is required.

Case 2: $q_1 \geq t$. To prove this, we begin by defining the number of malicious clients in a single group. Suppose that $X_i \sim H(n, q_1, l)$ represents the number of malicious clients in a cluster $\mathcal{C}_i$, where X_i follows a hypergeometric distribution with parameters n, q_1, l. The probability that a cluster contains more than t malicious clients is then:

$$\Pr(X_i \geq t) = \sum_{i=t}^{min\{l,q_1\}} \frac{\binom{q_1}{i}\binom{n-q_1}{l-i}}{\binom{n}{l}}$$

We now expand the binomial coefficients for the hypergeometric distribution. After simplifying the expression, we get:

$$\Pr(X_i \geq t) = \sum_{i=t}^{min\{l,q_1\}} \frac{q_1!(n-q_1)!l!(n-l)!}{i!(q_1-i)!(l-i)!(n-q_1-l+i)!n!}$$

Now, let A represent the event that at least one group contains more than t malicious clients. In this case, the corrupted clients can collaboratively infer the model privacy of the remaining honest clients. Assuming the groups are independent, the probability that at least one group contains more than t malicious clients is:

$$P(A) \leq k \cdot \Pr(X_i \geq t)$$

To ensure that the probability of compromising privacy is less than $2^{-\xi}$, we obtain the condition:

$$k \cdot \Pr(X_i \geq t) < 2^{-\lambda}$$

This condition leads to the following constraint on k:

$$k < \frac{2^{-\lambda}}{\sum_{i=t}^{min\{l,q_1\}} \frac{q_1!(n-q_1)!l!(n-l)!}{i!(q_1-i)!(l-i)!(n-q_1-l+i)!n!}}$$

In this case, the probability of the privacy security of the scheme being compromised at this point is given by:

$$\Pr(R_{success} = 1) \leq 2^{-\lambda}$$

□

Proof for Theorem 3

Theorem 3 (Privacy under Malicious Setting)**.** *There exists a PPT simulator SIM such that for all PPT adversaries $\mathcal{A}_\mathcal{C}$, all λ, t, $\mathcal{U}$, x_u, and $\mathcal{C}$ such that $\mathcal{C} \subseteq \mathcal{U} \cup \mathcal{S}$ and $|\mathcal{C} \setminus \mathcal{S}| < t$, the output of SIM is computationally indistinguishable from the output of* $\mathsf{REAL}_\mathcal{C}^{\mathcal{U},t,\lambda}$*:*

$$\mathsf{REAL}_\mathcal{C}^{\mathcal{U},t,\lambda}(\mathcal{A}_\mathcal{C}, x_{\mathcal{U}\setminus\mathcal{C}}) \approx_c SIM_\mathcal{C}^{\mathcal{U},t,\lambda,\mathsf{IDEAL}(t,x_u)_{u\in\mathcal{U}\setminus\mathcal{C}}}(\mathcal{A}_\mathcal{C})$$

The proof is based on a standard hybrid argument. We define a series of subsequent modifications to the real execution of our protocol, such that the views of corrupt parties in any two subsequent executions are computationally indistinguishable.

For fixed n, t, λ and a set $\mathcal{C}$ of corrupt parties, $\mathcal{A}_\mathcal{C}$ denotes the PPT adversary algorithm that represents the "next-message" function of parties in $\mathcal{C}$, i.e. $\mathcal{A}_\mathcal{C}$ outputs the message for party $c \in \mathcal{C}$, and also chooses the honest clients to abort due to failure in a round.

H_0: This random variable represents the combined views of all parties in $\mathcal{C}$ in the real execution REAL of our protocol.

H_1: In this hybrid, SIM runs a full execution of the protocol with $\mathcal{A}_\mathcal{C}$, and simulates LKH-PRF as a random oracle (using a dynamically generated table), PKI and the rest of the setup phase. As a result, the view of the adversary is the same as the previous hybrid.

H_2: This hybrid is identical to H_1, except additionally, SIM substitutes all the encrypted shares sent between pairs of honest clients with encryptions of 0 (in Round 1 - Sharing key).
Note that, we assume the encryption scheme used for encrypting $u_{u,v}$ has IND-CPA security, which guarantees this hybrid is indistinguishable from the previous one.

H_3: This hybrid is identical to H_2 except that SIM aborts if $\mathcal{A}_\mathcal{C}$ queries the random oracle LKH-PRF on input s_u for some honest user u before the adversary receives the responses from honest clients in Round 3 - Unmasking. Because s_u is information theoretically hidden from $\mathcal{A}_\mathcal{C}$,

SIM will only abort if $\mathcal{A}_{\mathcal{C}}$ can make a correct guess of s_u, which is of negligible probability.

H_4: This hybrid is identical to H_3 except that the values of $\mathbf{y}_u$ computed by SIM on behalf of the honest clients and sent to $\mathcal{A}_{\mathcal{C}}$ (in Round 2 Masked model update collection), are substituted with uniformly sampled values, and the output of some random oracle queries for the LKH-PRF, i.e. $\mathbf{mask}_{s_u}$, is modified to ensure consistency. That is, $\mathbf{mask}_{s_u} \leftarrow \mathbf{y}_u - \mathbf{x}_u$. Because $\mathcal{A}_{\mathcal{C}}$ cannot query the random oracle on input s_u, the value $\mathbf{y}_u$ looks randomly for the adversary. So the view of the adversary in this hybrid is statistically indistinguishable from the previous one.

H_5: This hybrid is identical to H_4 except that for all honest clients, SIM programs the random oracle to set the output of LKH-PRF as $\mathbf{mask}_{s_u} \leftarrow \mathbf{y}_u - \mathbf{r}_u$, where $\mathbf{r}_u$ is a set of randomly chosen values satisfying $\sum_{u\in\mathcal{U}\setminus\mathcal{C}} \mathbf{x}_u = \sum_{u\in\mathcal{U}\setminus\mathcal{C}} \mathbf{r}_u$. Since $\mathcal{A}_{\mathcal{C}}$ cannot query the random oracle for LKH-PRF on s_u of honest clients, this hybrid is indistinguishable from the previous one.

H_6: This hybrid is identical to H_5 except that SIM does not receive the inputs of the honest clients, but makes a query to the functionality $\mathsf{IDEAL}(t, x_u)_{u\in\mathcal{U}\setminus\mathcal{C}}$ in Round 3 Unmasking. This modification does not change the view seen by the adversary, and so this hybrid is indistinguishable from the previous one. In addition, this hybrid does not use inputs of the honest parties, and this concludes the proof.

Proof for Theorem 4

Theorem 4. *Consider the function F that satisfies assumptions 1 and 2. Assume there is a robust aggregation scheme that satisfies assumption 3. Let the number of type II attackers be q_2, and $F^* = min_\theta \nabla(\theta)$. There exists $\eta < \frac{1}{L}(1-\frac{\sqrt{\delta}}{4}-\frac{q_2}{2n})$ for which, after R rounds, the following condition holds:*

$$\frac{1}{R}\sum_{r=0}^{R-1}\mathbb{E}\|\nabla F(\theta_r)\|^2 \leq \frac{(F(\theta_0)-F^*)}{M\eta R} + \frac{L\eta}{M}\Delta_3 + \frac{1}{M}\Delta_4 \tag{23}$$

where $M = 1-\frac{\sqrt{\delta}}{4}-\frac{q_2}{2n}-L\eta, \Delta_4 = \frac{8c\sqrt{\delta}}{l}(\sigma_c^2+\sigma_g^2) + \frac{nq_2\sigma_g^2}{2(n-q_2)^2}, \Delta_3 = 2\Delta_1 + 4\Delta_2 + 12\frac{n-q_2-(1-\beta)kl}{(n-q_2-1)(1-\beta)kl}(\sigma_c^2+\sigma_g^2+\Delta_2), \Delta_1 = \frac{8c\delta}{l}(\sigma_c^2+\sigma_g^2)+b^2, \Delta_2 = \frac{q_2^2\sigma_g^2}{(n-q_2)^2} + \frac{2\sigma_c^2}{n-q_2}$.

Proof. Let the remaining benign clients be $\mathcal{U}_b$. Define $\bar{\mathbf{w}}_r^{(\mathcal{U}_b)}$ as the average model update of the benign clients. Note that the number of clients in the benign clusters is less than or equal to the number of benign clients ($|\{u_i\}_{u_i\in\mathcal{G}_g}| \leq |\mathcal{U}_b|$), since the Byzantine clusters may also contain benign clients.

For $\forall u_i, u_j \in \mathcal{U}_b$, we have

$$\begin{aligned}
&\mathbb{E}\left[\|\mathbf{w}_r^{(u_i)} - \mathbf{w}_r^{(u_j)}\|^2\right] \\
&= \mathbb{E}[\|\mathbf{w}_r^{(u_i)} - \nabla F_i(\theta_r) + F_i(\theta_r) - \nabla F(\theta_r) \\
&+ \nabla F(\theta_r) - \nabla F_j(\theta_r) + \nabla F_j(\theta_r) - \mathbf{w}_r^{(u_j)}\|^2] \\
&\overset{(a)}{\leq} 4(\mathbb{E}\left[\|\mathbf{w}_r^{(u_i)} - \nabla F_i(\theta_r)\|^2\right] + \mathbb{E}\left[\|\nabla F_i(\theta_r) - \nabla F(\theta_r)\|^2\right] \\
&+ \mathbb{E}\left[\|\nabla F(\theta_r) - \nabla F_j(\theta_r)\|^2\right] + \mathbb{E}\left[\|\nabla F_j(\theta_r) - \mathbf{w}_r^{(u_j)}\|^2\right]) \\
&\overset{(b)}{\leq} 8(\sigma_c^2+\sigma_g^2)
\end{aligned} \tag{24}$$

where (a) follows Reddi et al. [47] (A.4, Lemma 6), while (b) is based on Assumptions 2.

We first prove that the difference between the robust aggregation update and the benign cluster average update is bounded:

$$\begin{aligned}
&\mathbb{E}[\|\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi\in\mathcal{C}}) - \bar{\mathbf{w}}_r\|^2] \\
&\overset{a}{=} \mathbb{E}[\|\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi\in\mathcal{C}}) - \bar{\mathbf{w}}_r\|]^2 \\
&+ \mathrm{Var}\left[\|\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi\in\mathcal{C}})\|\right] \\
&\overset{(b)}{\leq} c\delta \sup_{\mathcal{C}_i,\mathcal{C}_j\in\mathcal{G}_g} \mathbb{E}\left[\|\mathbf{w}_r^{(\mathcal{C}_i)} - \mathbf{w}_r^{(\mathcal{C}_j)}\|^2\right] + b^2 \\
&= \frac{c\delta}{l^2} \sup_{\mathcal{C}_i,\mathcal{C}_j\in\mathcal{G}_g} \mathbb{E}\left[\|\sum_{u_i\in\mathcal{C}_i}\mathbf{w}_r^{(u_i)} - \sum_{u_j\in\mathcal{C}_j}\mathbf{w}_r^{(u_j)}\|^2\right] + b^2 \\
&\overset{(c)}{\leq} \underbrace{\frac{8c\delta}{l}(\sigma_c^2+\sigma_g^2)+b^2}_{\Delta_1}
\end{aligned} \tag{25}$$

where (a) follows from the basic relationship between expectation and variance, (b) is based on Assumption 3, and (c) uses (24) and Reddi et al. [47] (A.4, Lemma 6).

Next, we prove that the difference between the benign cluster average update and the global true update is bounded:

$$\begin{aligned}
&\mathbb{E}\left[\|\bar{\mathbf{w}}_r - \nabla F(\theta_r)\|^2\right] \\
&= \mathbb{E}\left[\|\bar{\mathbf{w}}_r - \bar{\mathbf{w}}_r^{(\mathcal{U}_b)} + \bar{\mathbf{w}}_r^{(\mathcal{U}_b)} - \nabla F(\theta_r)\|^2\right] \\
&\leq 2\mathbb{E}\left[\|\bar{\mathbf{w}}_r - \bar{\mathbf{w}}_r^{(\mathcal{U}_b)}\|^2\right] + 2\mathbb{E}\left[\|\bar{\mathbf{w}}_r^{(\mathcal{U}_b)} - \nabla F(\theta_r)\|^2\right] \\
&\overset{(a)}{\leq} 2\mathbb{E}\left[\|\bar{\mathbf{w}}_r - \bar{\mathbf{w}}_r^{(\mathcal{U}_b)}\|^2\right] + 2\underbrace{\frac{q_2^2\sigma_g^2}{(n-q_2)^2} + \frac{2\sigma_c^2}{n-q_2}}_{\Delta_2}
\end{aligned} \tag{26}$$

where (a) follows Xu et al. [41] (Lemma 1).

We further prove that the difference between the benign cluster average update and the average update of all benign clients is bounded:

$$\begin{aligned}
&\mathbb{E}\left[\|\bar{\mathbf{w}}_r - \bar{\mathbf{w}}_r^{(\mathcal{U}_b)}\|^2\right] \\
&= \mathbb{E}\left[\|\frac{1}{(1-\beta)kl}\sum_{u_i\in\mathcal{G}_g}(\mathbf{w}_r^{(u_i)} - \bar{\mathbf{w}}_r^{(\mathcal{U}_b)})\|^2\right] \\
&\overset{(a)}{\leq} \frac{n-q_2-(1-\beta)kl}{(n-q_2-1)(1-\beta)kl}\cdot\frac{1}{n-q_2}\sum_{u_i\in\mathcal{U}_b}\mathbb{E}\left[\|\mathbf{w}_r^{(u_i)} - \bar{\mathbf{w}}_r^{(\mathcal{U}_b)}\|^2\right]
\end{aligned} \tag{27}$$

where (a) follows Lei et al. [48] (Lemma A.1), because the clustering is completely random, $\{u_i\}_{u_i \in \mathcal{G}_g}$ can be considered a random sample from the set $\mathcal{U}_b$.

Note that

$$\begin{aligned}
&\mathbb{E}\left[\|\mathbf{w}_r^{(u_i)} - \bar{\mathbf{w}}_r^{(\mathcal{U}_b)}\|^2\right] \\
&= \mathbb{E}[\|\mathbf{w}_r^{(u_i)} - \nabla F_i(\theta_r) + \nabla F_i(\theta_r) \\
&- \nabla F(\theta_r) + \nabla F(\theta_r) - \bar{\mathbf{w}}_r^{(\mathcal{U}_b)}\|^2] \\
&\overset{(a)}{\leq} 3\mathbb{E}\left[\|\mathbf{w}_r^{(u_i)} - \nabla F_i(\theta_r)\|^2\right] + 3\mathbb{E}\left[\|\nabla F_i(\theta_r) - \nabla F(\theta_r)\|^2\right] \\
&+ 3\mathbb{E}\left[\nabla F(\theta_r) - \bar{\mathbf{w}}_r^{(\mathcal{U}_b)}\|^2\right] \\
&\overset{(b)}{\leq} 3\sigma_c^2 + 3\sigma_g^2 + 3\Delta_2
\end{aligned} \tag{28}$$

where (a) refers to Reddi et al. [47] (A.4, Lemma 6), and (b) is based on (26) and Assumption 2.

So, we can get from (27) and (28)

$$\begin{aligned}
&\mathbb{E}\left[\|\bar{\mathbf{w}}_r - \bar{\mathbf{w}}_r^{(\mathcal{U}_b)}\|^2\right] \\
&\leq 3\frac{n - q_2 - (1-\beta)kl}{(n - q_2 - 1)(1-\beta)kl}(\sigma_c^2 + \sigma_g^2 + \Delta_2)
\end{aligned} \tag{29}$$

Finally, we aim to prove that the difference between the robust aggregation update and the global true update is bounded:

$$\begin{aligned}
&\mathbb{E}_r[\|\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}}) - \nabla F(\theta_r)\|^2] \\
&= \mathbb{E}_r[\|\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}}) - \bar{\mathbf{w}}_r + \bar{\mathbf{w}}_r - \nabla F(\theta_r)\|^2] \\
&\overset{(a)}{\leq} 2\mathbb{E}_r[\|\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}}) - \bar{\mathbf{w}}_r\|^2] + 2\mathbb{E}_r[\|\bar{\mathbf{w}}_r - \nabla F(\theta_r)\|^2] \\
&\overset{(b)}{\leq} \underbrace{2\Delta_1 + 4\Delta_2 + 12\frac{n - q_2 - (1-\beta)kl}{(n - q_2 - 1)(1-\beta)kl}(\sigma_c^2 + \sigma_g^2 + \Delta_2)}_{\Delta_3}
\end{aligned} \tag{30}$$

where (a) follows Reddi et al. [47] (A.4, Lemma 6), while (b) utilizes (25), (26), and (29).

We will now proceed with the derivation of the convergence

$$\begin{aligned}
&\mathbb{E}_r[F(\theta_{r+1})] - F(\theta_r) \\
&\overset{(a)}{\leq} -\langle \nabla F(\theta_r), \mathbb{E}_r[\theta_{r+1} - \theta_r]\rangle + \frac{L}{2}\mathbb{E}_r[\|\theta_r - \theta_{r+1}\|^2] \\
&= -\eta\langle \nabla F(\theta_r), \mathbb{E}_r\left[\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}})\right]\rangle \\
&+ \frac{L\eta^2}{2}\mathbb{E}_r\left[\|\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}})\|^2\right] \\
&= -\eta\langle \nabla F(\theta_r), \mathbb{E}_r\left[\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}})\right] - \nabla F(\theta_r) + \\
&\nabla F(\theta_r)\rangle + \frac{L\eta^2}{2}\mathbb{E}_r[\|\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}}) \\
&- \nabla F(\theta_r) + \nabla F(\theta_r)\|^2] \\
&\overset{(b)}{\leq} -\eta\langle \nabla F(\theta_r), \mathbb{E}_r\left[\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}})] - \nabla F(\theta_r)\right]\rangle \\
&+ L\eta^2\mathbb{E}_r\left[\|\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}}) - \nabla F(\theta_r)\|^2\right] \\
&+ (L\eta^2 - \eta)\|\nabla F(\theta_r)\|^2
\end{aligned} \tag{31}$$

where (a) applies the property of $L - Lipschitz$ continuity, and (b) relies on Reddi et al. [47] (A.4, Lemma 6).

Note that

$$\begin{aligned}
&-\langle \nabla F(\theta_r), \mathbb{E}_r\left[\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}})] - \nabla F(\theta_r)\right]\rangle \\
&= -\langle \nabla F(\theta_r), \mathbb{E}_r\left[\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}})] - \nabla F(\theta_r)\right]\rangle \\
&= -\langle \nabla F(\theta_r), \mathbb{E}_r\left[\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}})] - \bar{\mathbf{w}}_r\right]\rangle \\
&- \langle \nabla F(\theta_r), \mathbb{E}_r\left[\bar{\mathbf{w}}_r - \bar{\mathbf{w}}_r^{(\mathcal{U}_b)}\right]\rangle \\
&- \langle \nabla F(\theta_r), \mathbb{E}_r\left[\bar{\mathbf{w}}_r^{(\mathcal{U}_b)} - \nabla F(\theta_r)\right]\rangle \\
&\overset{(a)}{=} -\langle \nabla F(\theta_r), \mathbb{E}_r\left[\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}})] - \bar{\mathbf{w}}_r\right]\rangle \\
&- \langle \nabla F(\theta_r), \mathbb{E}_r\left[\bar{\mathbf{w}}_r^{(\mathcal{U}_b)} - \nabla F(\theta_r)\right]\rangle
\end{aligned} \tag{32}$$

where (a) is due to the fact that $\{u_i\}_{u_i \in \mathcal{G}_g}$ is a random sample from $\mathcal{U}_b$, and the overall expectation equals the sample expectation, i.e., $\mathbb{E}_r[\bar{\mathbf{w}}_r] = \mathbb{E}_r[\bar{\mathbf{w}}_r^{(\mathcal{U}_b)}]$.

We will first analyze the first term:

$$\begin{aligned}
&-\langle \nabla F(\theta_r), \mathbb{E}_r\left[\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}})] - \bar{\mathbf{w}}_r\right]\rangle \\
&\overset{(a)}{\leq} \|\nabla F(\theta_r)\| \cdot \mathbb{E}_r\left[\|\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}})] - \bar{\mathbf{w}}_r\|\right] \\
&\overset{(b)}{\leq} \frac{1}{2}\Big(\frac{2}{\sqrt{\delta}}\mathbb{E}_r\left[\|\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}})] - \bar{\mathbf{w}}_r\|\right]^2 \\
&+ \frac{\sqrt{\delta}}{2}\|\nabla F(\theta_r)\|^2\Big) \\
&\overset{(c)}{\leq} \frac{\sqrt{\delta}}{4}\|\nabla F(\theta_r)\|^2 + \frac{8c\sqrt{\delta}}{l}(\sigma_c^2 + \sigma_g^2)
\end{aligned} \tag{33}$$

where (a) applies the Cauchy-Schwarz inequality, (b) utilizes the AM-GM inequality, and (c) follows (25).

Next, we analyze the second term:

$$\begin{aligned}
&-\langle \nabla F(\theta_r), \mathbb{E}_r\left[\bar{\mathbf{w}}_r^{(\mathcal{U}_b)} - \nabla F(\theta_r)\right]\rangle \\
&\overset{(a)}{\leq} \|\nabla F(\theta_r)\| \cdot \mathbb{E}_r\left[\|\bar{\mathbf{w}}_r^{(\mathcal{U}_b)} - \nabla F(\theta_r)\|\right] \\
&\overset{(b)}{\leq} \frac{1}{2}\Big(\frac{q_2}{n}\|\nabla F(\theta_r)\|^2 + \frac{n}{q_2}\mathbb{E}_r\left[\|\bar{\mathbf{w}}_r^{(\mathcal{U}_b)} - \nabla F(\theta_r)\|\right]^2\Big) \\
&\overset{(c)}{\leq} \frac{q_2}{2n}\|\nabla F(\theta_r)\|^2 + \frac{nq_2\sigma_g^2}{2(n - q_2)^2}
\end{aligned} \tag{34}$$

where (a) applies the Cauchy-Schwarz inequality, (b) utilizes the AM-GM inequality, and (c) is based on the derivation from Xu et al. [41] (Lemma 1).

Combining (32), (33), and (34), we have

$$\begin{aligned}
&-\langle \nabla F(\theta_r), \mathbb{E}_r\left[\mathbf{RobustAgg}(\{\mathbf{w}_r^{(\mathcal{C}_\varphi)}\}_{\mathcal{C}_\varphi \in \mathcal{C}})] - \nabla F(\theta_r)\right]\rangle \\
&\leq \Big(\frac{\sqrt{\delta}}{4} + \frac{q_2}{2n}\Big)\|\nabla F(\theta_r)\|^2 + \underbrace{\frac{8c\sqrt{\delta}}{l}(\sigma_c^2 + \sigma_g^2) + \frac{nq_2\sigma_g^2}{2(n - q_2)^2}}_{\Delta_4}
\end{aligned} \tag{35}$$

Substituting (35) into (31) and reorganizing, we obtain

$$\begin{aligned}
&\eta\Big(1 - \frac{\sqrt{\delta}}{4} - \frac{q_2}{2n} - L\eta\Big)\|\nabla F(\theta_r)\|^2 \leq \mathbb{E}_r[F(\theta_r) - F(\theta_{r+1})] + \\
&+ L\eta^2\Delta_3 + \eta\Delta_4
\end{aligned} \tag{36}$$

Assuming $\eta < \frac{1}{L}(1 - \frac{\sqrt{\delta}}{4} - \frac{q_2}{2n})$, we can get $M = 1 - \frac{\sqrt{\delta}}{4} - \frac{q_2}{2n} - L\eta > 0$, sum over the federated learning rounds from $0 \to R$, and we obtain

$$\frac{1}{R}\sum_{r=0}^{R-1}\mathbb{E}\|\nabla F(\theta_r)\|^2 \leq \frac{(F(\theta_0) - F^*)}{M\eta R} + \frac{L\eta}{M}\Delta_3 + \frac{1}{M}\Delta_4 \tag{37}$$

□

Experimental Evaluation: Further Results

We present the evaluation of the ClusterGuard federated learning scheme in a decentralized setting with 20 clients. The training configuration, as well as the model and dataset settings, align with the default setup described earlier. We use the average model accuracy of honest clients to estimate the model performance. We consider two situations: the first case is the attack from non-leader clients, and the second case is the attack from leader clients.

First case: the attack setup is the same as described earlier, where we still consider the different attack strategies mentioned above. The number of adversaries is set to 20% of the total clients.

TABLE IV
PERFORMANCE COMPARISON OF CLUSTERGUARD AND OTHER SCHEMES UNDER MALICIOUS ATTACKS IN DECENTRALIZED SETTING

Datasets	Scheme	Attacks			
		Label flipping	Noise	Sign flipping	Scaling
MNIST	FedAvg	80.81	66.25	33.02	0.31/0.35
	Krum	63.28	66.11	60.75	72.43
	Median	69.30	79.37	68.21	86.73
	Share	73.59	87.14	29.97	34.95
	Bulyan	69.35	68.77	69.41	77.69
	ClusterGuard	86.96	85.54	82.03	85.11
Fasion-MNIST	FedAvg	83.11	54.35	59.96	10.00
	Krum	69.32	67.32	65.61	69.92
	Median	83.43	87.67	79.34	87.96
	Share	73.39	86.85	51.60	26.94
	Bulyan	84.17	85.24	85.23	84.85
	ClusterGuard	85.11	86.91	85.29	85.28
CIFAR-10	FedAvg	58.99	10.00	10.00	10.00
	Krum	29.60	37.83	36.66	36.78
	Median	41.85	64.19	53.84	67.89
	Share	43.52	64.38	62.58	11.65
	Bulyan	14.91	58.35	60.19	60.88
	ClusterGuard	64.67	65.30	63.44	64.33

Table IV presents a comprehensive evaluation of ClusterGuard in comparison to other robust aggregation methods (including Median, Krum, Share, and Bulyan) across various malicious attack strategies, including label flipping, noise injection, sign flipping, and scaling, on the MNIST, Fashion MNIST, and CIFAR-10 datasets. The experimental results consistently demonstrate that ClusterGuard outperforms the other defense strategies across all three datasets. Methods such as Krum and Share are susceptible to carefully crafted attacks, whereas ClusterGuard maintains consistently high accuracy and resilience. Moreover, in contrast to Median and Bulyan, which are limited to simpler tasks such as MNIST digit classification, ClusterGuard demonstrates superior performance on the more complex image classification task of CIFAR-10, where other methods show significant performance degradation. These findings establish ClusterGuard as a robust and reliable solution for decentralized federated learning, capable of defending against a wide range of diverse and sophisticated attack strategies.

Second case: we utilize the Random Attack method to simulate the behavior of adversarial clients. Byzantine clients send gradients with randomized values, generated from a multi-dimensional Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. In our experiments, we set $\mu = 0$ and $\sigma = 0.5$ to simulate random attacks. Let τ denote the attack interval for all leader clients. In practice, such attacks are challenging to sustain, as the leader selection process is fully random, based on a VRF-based election mechanism. This part of the experiment will be conducted in the future.