Do We Really Need to Design New Byzantine-robust Aggregation Rules?

Minghong Fang*,✉, Seyedsina Nabavirazavi†, Zhuqing Liu¶,
Wei Sun§, Sundararaja Sitharama Iyengar†, Haibo Yang‡
*University of Louisville, †Florida International University, ¶University of North Texas,
§Wichita State University,‡Rochester Institute of Technology

***Abstract*—Federated learning (FL) allows multiple clients to collaboratively train a global machine learning model through a server, without exchanging their private training data. However, the decentralized aspect of FL makes it susceptible to poisoning attacks, where malicious clients can manipulate the global model by sending altered local model updates. To counter these attacks, a variety of aggregation rules designed to be resilient to Byzantine failures have been introduced. Nonetheless, these methods can still be vulnerable to sophisticated attacks or depend on unrealistic assumptions about the server. In this paper, we demonstrate that there is no need to design new Byzantine-robust aggregation rules; instead, FL can be secured by enhancing the robustness of well-established aggregation rules. To this end, we present FoundationFL, a novel defense mechanism against poisoning attacks. FoundationFL involves the server generating synthetic updates after receiving local model updates from clients. It then applies existing Byzantine-robust foundational aggregation rules, such as Trimmed-mean or Median, to combine clients' model updates with the synthetic ones. We theoretically establish the convergence performance of FoundationFL under Byzantine settings. Comprehensive experiments across several real-world datasets validate the efficiency of our FoundationFL method.**

I. Introduction

In recent years, federated learning (FL) has emerged as a promising approach to distributed learning [1]. It allows multiple clients to collaboratively train a global machine learning model (called *global model*) under the coordination of a central server, all while respecting the privacy of clients' sensitive training data. Essentially, in each training round, the server distributes the current global model to all clients or a subset of them. Each selected client refines its local machine learning model (called *local model*) by using this global model and its own local training data. Subsequently, the client sends its local model update back to the server. Upon receiving updates from clients, the server aggregates these updates into a global model update, which it then integrates to further update the global model. FL has been implemented across a range of practical tasks and applications, including credit risk assessment [2], predictive text input [3], and speech recognition [4].

However, the decentralized nature of FL poses distinct challenges, with one of the most significant being its susceptibility to poisoning attacks [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]. In these attacks, malicious clients, under the control of an attacker, attempt to compromise the integrity of the global model. They do this by manipulating their local training data or sending carefully crafted updates directly to the server. Depending on the attacker's objectives, poisoning attacks can be categorized as untargeted [8], [9], [10], [11] or targeted [5], [6], [15]. In untargeted attacks, the goal is to degrade the overall performance of the global model. In contrast, targeted attacks aim to induce incorrect predictions specifically on testing inputs chosen by the attacker, while leaving predictions for other inputs unaffected. It has been demonstrated that a single malicious client is sufficient to successfully compromise an FL system that uses a straightforward average aggregation strategy. In this scenario, one malicious client can arbitrarily manipulate the final aggregated result [8].

In response to poisoning attacks, researchers have shifted from using a straightforward average aggregation rule to developing robust aggregation methods to increase the resilience of FL. These include the Trimmed-mean and Median aggregation rules introduced by [16], which are termed *Byzantine-robust foundational aggregation rules*. Trimmed-mean is a coordinate-wise aggregation protocol that removes some of the largest and smallest extreme values for each dimension before averaging the remaining values. In contrast, the Median method calculates the coordinate-wise median of the clients' local model updates. Despite their robustness, recent research indicates that these rules are still susceptible to advanced poisoning attacks [9], [10]. To combat these vulnerabilities, newer and more complex aggregation rules have been developed [8], [15], [17], [18], [19], [20], [21], [22], [23], [24]. These, however, either depend heavily on the FL system having access to a clean dataset [15], [20], [21], [23], [24] or continue to be prone to sophisticated attacks [25], [26]. This ongoing escalation between attackers and defenders often results in a costly and potentially unsustainable arms race. This situation raises an essential research question: ***Do we really need to design new Byzantine-robust aggregation rules?***

In this paper, we find a promising answer to the aforementioned question. Instead of developing complex new

✉ Minghong Fang is the corresponding author.

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.241796
www.ndss-symposium.org

Byzantine-robust aggregation protocols, we aim to enhance the robustness of FL systems by employing well-established Byzantine-robust aggregation methods like Trimmed-mean and Median. The unique aspect of FL is the non-identical and non-independent (Non-IID) distribution of training data among clients, which introduces substantial diversity. This diversity allows malicious clients to manipulate their local model updates to undermine the FL system, while remaining distinct from benign clients. To mitigate these issues of heterogeneity, in our proposed FoundationFL framework, the server introduces synthetic updates in each global training round. The primary challenge then becomes determining these synthetic updates. To tackle this, the server calculates a closeness score for each client in every round to assess how their local model updates align with the most extreme updates. The server then selects the client's local model update that deviates the most from these extreme updates. This chosen update is used as the basis for the synthetic updates, which are then mixed with the clients' local model updates. Ultimately, the server applies Byzantine-robust foundational aggregation rules like Trimmed-mean or Median to combine both the clients' local and the synthetic updates.

We offer theoretical guarantees for our FoundationFL framework under poisoning attacks. Specifically, we provide a theoretical demonstration that the global model learnt through our FoundationFL, even when subjected to poisoning attacks, converges with high probability to the optimal global model that would be obtained in the absence of attacks, given certain mild assumptions. We conduct comprehensive evaluations of our FoundationFL across 6 datasets spanning various domains, against 12 different poisoning attacks, and comparing with 10 FL aggregation rules. Our findings show that our proposed FoundationFL significantly surpasses current Byzantine-robust FL methods in performance.

We summarize our main contributions in this paper as follows:

- We introduce FoundationFL, a robust aggregation framework designed to combat poisoning attacks within FL environments.
- We demonstrate theoretically that FoundationFL remains resilient to poisoning attacks under commonly accepted assumptions within the Byzantine-robust FL community.
- Our thorough experiments across diverse benchmark datasets, various poisoning attack scenarios, and practical FL setups validate the effectiveness of our FoundationFL framework.

II. Background and Related Work

Notations: In this paper, the ℓ_2-norm is denoted by $\|\cdot\|$. Furthermore, for any natural number n, the set $\{1, \ldots, n\}$ is represented as $[n]$. Additionally, matrices and vectors are specified using bold typeface.

A. Federated Learning

Federated learning (FL) usually includes a server and n participating clients that work together to train a global model without exchanging private data. We denote the local training dataset of client i as D_i, where $i \in [n]$. The goal of FL is to minimize the following global objective function:

$$L(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} L_i(\boldsymbol{\theta}), \tag{1}$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ is the model parameter, d is the dimension of $\boldsymbol{\theta}$, $L_i(\boldsymbol{\theta}) = \frac{1}{|D_i|} \sum_{z \in D_i} l(\boldsymbol{\theta}, z)$ is the local training objective (empirical loss) of client i, $|D_i|$ denotes the count of training example for the client i.

FL solves the above Problem (1) through an iterative process. Specifically, in each global training round t, the following three specific steps are executed:

- **Step I: Global model synchronization.** The server distributes the current global model $\boldsymbol{\theta}^t$ to all clients or a selected group of them.
- **Step II: Local model updating.** Each client $i \in [n]$ fine-tunes its local model $\boldsymbol{\theta}_i^t$ by leveraging the current global model $\boldsymbol{\theta}^t$ and its local training dataset D_i. Then, client i transmits its local model update $\boldsymbol{g}_i^t = \boldsymbol{\theta}_i^t - \boldsymbol{\theta}^t$ to the server.
- **Step III: Global model updating.** After collecting model updates from the clients, the server employs the aggregation rule, denoted as Agg, to merge these updates to get a global model update. The global model is then updated as $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \eta \cdot \text{Agg}\{\boldsymbol{g}_i^t : i \in [n]\}$, where η is the learning rate.

FL repeats the aforementioned three steps across multiple global training rounds until a specified convergence condition is satisfied. It's worth mentioning that various FL methods often employ distinct aggregation protocols [1], [8], [16]. For instance, in the case of the FedAvg [1] aggregation rule, the server combines the model updates as $\text{Agg}\{\boldsymbol{g}_i^t : i \in [n]\} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{g}_i^t$.

B. Poisoning Attacks to FL

Although FL's decentralized architecture offers privacy benefits, it also renders it susceptible to poisoning attacks. In a FL environment, malicious clients may attempt to manipulate the global model's performance by interfering with the training process. This could involve corrupting their local training data (known as *data poisoning attacks* [11]) or tampering with their local model updates (known as *local model poisoning attacks* [8], [9], [10]). The ultimate goal of these malicious clients is to degrade the global model's performance. For instance, the final learnt global model exhibits reduced accuracy when classifying testing examples. For instance, in a label flipping attack [11], malicious clients flip the labels associated with their training data while leaving the underlying features unchanged. In the Trim attack [9], malicious clients strategically manipulate their local model updates before sending them to the central server. This manipulation aims to exploit the vulnerabilities of Trimmed-mean [16] or Median [16] aggregation rule employed by the server, causing a significant deviation in the global model updates after the attack compared to the updates before the attack.

C. Byzantine-robust Aggregation Rules

In standard FL, the server combines the local model updates it receives by computing their averages [1]. Nevertheless, recent research [8] has revealed that this aggregation method, based on averaging, is highly susceptible to Byzantine attacks. In such attacks, a single malicious client has the ability to manipulate the final aggregated outcome in arbitrary ways. In order to safeguard FL against poisoning attacks, several Byzantine-robust foundational aggregation rules, such as Krum [8], Trimmed-mean [16] and Median [16], have been suggested. For instance, with the Krum [8] aggregation rule, once the server receives n local model updates from clients, it selects and outputs the local model update that has the smallest total distance to its $n - f - 2$ closest neighbors, where n is the total number of clients, f is the number of malicious clients. Trimmed-mean [16] aggregation rule operates on a per-coordinate basis. Specifically, for every dimension, the server starts by eliminating the largest c elements and the smallest c elements, and subsequently computes the average of the remaining values, where c is the trim parameter. Median [16] is another coordinate-wise aggregation method. In this approach, the server combines the received local model updates by determining the median value for each dimension of the updates. In recent years, several Byzantine-robust advanced aggregation rules have been proposed [15], [19], [20], [21], [23], [24], [27], [28], [29], [30], [31], [32]. For instance, in Bulyan [31] aggregation rule, the server first leverage Krum [8] aggregation rule to select a select of local model updates, then take the Median [16] of these selected updates. The authors in [15], [20], [21], [23], [24] assume that the server possesses a clean validation dataset, sourced from the same distribution as the overall training dataset. Utilizing this validation dataset, the server calculates a benchmark model update. This benchmark is then applied to determine if the local model updates received are either benign or malicious.

Limitations of existing robust aggregation rules: Firstly, current Byzantine-resistant aggregation methods are not entirely secure, as they remain susceptible to sophisticated poisoning attacks [9], [10]. Secondly, numerous robust aggregation rules rely on strong assumptions regarding the server's capabilities, such as possessing a separate validation data [15], [20], [21], [23], [24]. However, this assumption is often impractical as it is challenging for the server to accurately know the distribution of clients' local training data. Moreover, the possession of validation data by the server could infringe upon privacy concerns, contradicting the core principles of FL's design.

III. Threat Model

Attacker's goal and knowledge: In our paper, we consider the attack model as described in [9], [10], [15]. Specifically, the attacker manipulates certain malicious clients, which may either be fake clients injected by the attacker or benign clients compromised by the attacker. These controlled malicious clients send meticulously crafted updates of local models to the server to achieve the attacker's objectives. For instance, in untargeted attacks, the goal is to corrupt the resulting global model such that it incorrectly classifies a significant portion of test examples without distinction. In targeted attacks, the objective is to manipulate the global model so that it predicts specific instances chosen by the attacker to match predetermined labels. We consider the worst-case but realistic attack scenario where the attacker knows the aggregation rule used by the server, and all clients' local model updates. For example, the server may public its aggregation protocol, and the attacker may eavesdrop the communication link in order to get access to local model updates on the benign clients. We note that in our proposed FoundationFL framework, the server generates some synthetic updates. These generated synthetic updates are not access to the attacker, since the server in FL is secure, it hard and even impossible for the attacker to compromise the server in order to have access to these synthetic updates.

Defender's goal and knowledge: Our goal is to develop an effective Byzantine-robust method that accomplishes the following three objectives:

- **Competitive performance:** The proposed defense scheme for FL should also perform effectively in non-adversarial environments. Specifically, in the absence of malicious clients, the model trained using our algorithm should achieve a testing error rate comparable to that of averaging-based aggregation, which is known to deliver state-of-the-art performance in non-adversarial FL settings.
- **Byzantine robustness:** The proposed method should demonstrate robustness against Byzantine attacks, both empirically and theoretically.
- **Efficiency:** The proposed algorithm should not increase the communication costs between the server and clients, nor should it lead to significant computational demands on the server side.

Regarding the defender's knowledge, the defender (server) is unaware of the attacker's methods of conducting attacks. Additionally, the defender does not have knowledge about the distribution of the clients' local training data. Note that, following [9], [10], [25], we assume in our threat model that the majority of clients are benign.

IV. Our Method

A. Overview

In this section, we provide a formal demonstration that the server can enhance the robustness of the FL system by using established Byzantine-robust foundational aggregation rules. This indicates that creating new, complex Byzantine-robust aggregation protocols is unnecessary. In our framework, the server takes a proactive approach by generating synthetic updates upon receiving local model updates from clients. Following this, the server employs existing Byzantine-robust foundational aggregation protocols, such as Trimmed-mean or Median, to combine the local model updates from clients with the generated synthetic updates.

B. FoundationFL

Trimmed-mean [16]: Let $\boldsymbol{g}^t$ represent the global model update at the training round t, where $\boldsymbol{g}^t = \text{Agg}\{\boldsymbol{g}_i^t : i \in [n]\}$. The k-th component of vector $\boldsymbol{g}^t$, denoted $\boldsymbol{g}^t[k]$ for $k \in [d]$, is calculated using the Trimmed-mean aggregation method. In this method, for each dimension, the largest c and smallest c values are discarded, and the mean of the remaining values is computed. Specifically, $\boldsymbol{g}^t[k]$ is obtained by $\boldsymbol{g}^t[k] = \text{Trimmed-mean}\{\boldsymbol{g}_1^t[k], \ldots, \boldsymbol{g}_n^t[k]\}$.

Median [16]: The Median is another rule for aggregation that also operates on each coordinate individually. For every dimension, the server calculates the median of the values from the clients' local model updates. Specifically, for the k-th dimension, $\boldsymbol{g}^t[k]$ is determined by $\boldsymbol{g}^t[k] = \text{Median}\{\boldsymbol{g}_1^t[k], \ldots, \boldsymbol{g}_n^t[k]\}$.

In FL, a distinctive feature is the non-identical and non-independent (Non-IID) distribution of clients' training data. This diversity across different clients means that the training data is not uniformly distributed, making it significantly varied. As clients train their local models using the current global model and their unique training data, even benign local model updates exhibit notable differences. As a result, this heterogeneity can mask the activities of malicious clients, who may take advantage of these differences to manipulate their local model updates and launch poisoning attacks without detection. These malicious clients craft their updates that, while aimed at undermining the global model, appear as normal within the range of local model updates from benign clients. This subtle manipulation makes it exceedingly challenge to detect and neutralize such threats within the FL system. Moreover, this inherent heterogeneity underscores why existing Byzantine-robust foundational aggregation rules, such as Trimmed-mean [16] and Median [16] (described further above), remain vulnerable to Byzantine attacks. This vulnerability has been illustrated in [9], [10], highlighting the ongoing challenges in securing FL systems against sophisticated poisoning attacks.

To address the issue of data heterogeneity in FL and to enhance system robustness, our approach focuses on leveraging existing Byzantine-robust foundational aggregation rules, rather than developing new ones. The core concept of our proposed FoundationFL framework involves a proactive step by the server: after receiving local model updates from clients during each global training round, the server generates additional synthetic model updates. These synthetic updates, when combined with the clients' local model updates, are then aggregated using established Byzantine-robust aggregation rules like Trimmed-mean or Median. The fundamental advantage of our proposed method is that by introducing synthetic model updates, we create a more homogeneous set of updates—where the augmented model updates (comprising both clients' local model updates and the synthetic updates) exhibit much lower variance compared to the original solely client-sourced updates. This reduction in variance across the updates makes the aggregated model less susceptible to outliers and potential poisoning attacks. By feeding these more uniform updates into proven robust aggregation mechanisms, we significantly enhance the system's ability to thwart malicious interventions, thereby increasing the overall security and reliability of the FL system. This strategy leverages the strengths of existing robust aggregation frameworks while effectively countering the challenges posed by data heterogeneity in federated environments. In what follows, we demonstrate how to construct the synthetic updates.

In each global training round t, assuming the server generates m synthetic updates, represented as $\{\bar{\boldsymbol{g}}_1^t, \ldots, \bar{\boldsymbol{g}}_m^t\}$. The central challenge lies in determining these m updates effectively. Remember that Trimmed-mean and Median are robust statistical methods designed to eliminate outliers (extreme values) from each dimension of clients' local model updates. Motivated by this observation, the server first identifies a client's local model that deviates the most from the extreme updates, and augments clients' local model updates by incorporating multiple copies of this selected update. Define $\boldsymbol{g}_{\text{max}}^t \in \mathbb{R}^d$ and $\boldsymbol{g}_{\text{min}}^t \in \mathbb{R}^d$ as the vectors representing the largest and smallest updates across all dimensions, respectively. Specifically, $\boldsymbol{g}_{\text{max}}^t[k]$ is the maximum value of the set $\{\boldsymbol{g}_1^t[k], \boldsymbol{g}_2^t[k], \ldots, \boldsymbol{g}_n^t[k]\}$, and $\boldsymbol{g}_{\text{min}}^t[k]$ is the minimum value of the same set for the dimension k. The values of $\boldsymbol{g}_{\text{max}}^t[k]$ and $\boldsymbol{g}_{\text{min}}^t[k]$ are determined as:

$$\boldsymbol{g}_{\text{max}}^t[k] = \max\{\boldsymbol{g}_1^t[k], \ldots, \boldsymbol{g}_n^t[k]\}, k \in [d] \tag{2}$$

$$\boldsymbol{g}_{\text{min}}^t[k] = \min\{\boldsymbol{g}_1^t[k], \ldots, \boldsymbol{g}_n^t[k]\}, k \in [d]. \tag{3}$$

Upon deriving $\boldsymbol{g}_{\text{max}}^t$ and $\boldsymbol{g}_{\text{min}}^t$, the server assigns a score s_i^t to each client $i \in [n]$. This score quantifies how closely each client i's local model update $\boldsymbol{g}_i^t$ aligns with the extreme updates, namely $\boldsymbol{g}_{\text{max}}^t$ and $\boldsymbol{g}_{\text{min}}^t$. A higher s_i^t indicates a greater likelihood that the update $\boldsymbol{g}_i^t$ is malicious. In our proposed FoundationFL, the server calculates s_i^t by taking the lesser of the distances between $\boldsymbol{g}_i^t$ and the vectors $\boldsymbol{g}_{\text{max}}^t$ and $\boldsymbol{g}_{\text{min}}^t$, as shown below:

$$s_i^t = \min\{\|\boldsymbol{g}_i^t - \boldsymbol{g}_{\text{max}}^t\|, \|\boldsymbol{g}_i^t - \boldsymbol{g}_{\text{min}}^t\|\}. \tag{4}$$

The server then selects one local model update from the set $\{\boldsymbol{g}_1^t, \ldots, \boldsymbol{g}_n^t\}$ that exhibits the greatest deviation from the extreme updates $\boldsymbol{g}_{\text{max}}^t$ and $\boldsymbol{g}_{\text{min}}^t$. Let $i_* \in [n]$ be defined as the client with the largest score, such that $s_{i_*}^t \geq s_i^t$ for all $i \in [n]$. Formally, this selection criterion is defined as:

$$i_* = \underset{i \in [n]}{\text{argmax}}\, s_i^t. \tag{5}$$

Following that, the server designates each synthetic update as $\bar{\boldsymbol{g}}_j^t = \boldsymbol{g}_{i_*}^t$ for all $j \in [m]$. It then supplements the clients' local model updates with these m synthetic updates. To derive the global model update, the server applies Byzantine-robust foundational aggregation protocols. The resulting global model update, denoted by $\hat{\boldsymbol{g}}^t$, is computed as follows: if the Trimmed-mean protocol is employed for aggregation, each dimension $k \in [d]$ is updated according to:

$$\hat{\boldsymbol{g}}^t[k] = \text{Trimmed-mean}\{\boldsymbol{g}_1^t[k], \ldots, \boldsymbol{g}_n^t[k], \bar{\boldsymbol{g}}_1^t[k], \ldots, \bar{\boldsymbol{g}}_m^t[k]\} \tag{6}$$

$$\text{s.t. } \bar{\boldsymbol{g}}_1^t[k] = \cdots = \bar{\boldsymbol{g}}_m^t[k] = \boldsymbol{g}_{i_*}^t[k].$$

Algorithm 1 Training procedure of FoundationFL.

Input: The n clients with local training dataset; number of global training rounds T; number of synthetic updates m; learning rate η; Byzantine-robust foundational aggregation rule Agg.

Output: Global model $\boldsymbol{\theta}^T$.

Initialize $\boldsymbol{\theta}^0$.
for $t = 0, 1, \cdots, T-1$ **do**
  // Step I: Global model synchronization.
  The server send the global model $\boldsymbol{\theta}^t$ to all clients.
  // Step II: Local model updating.
  for each client $i \in [n]$ in parallel **do**
    Client i fine-tunes its local model and sends the local model update $\boldsymbol{g}_i^t$ to the server.
  end for
  // Step III: Global model updating.
  The server compute the score s_i^t for $i \in [n]$ based on Eq. (4).
  The server chooses client i_* with the highest score as defined in Eq. (5), and generates each synthetic update as $\bar{\boldsymbol{g}}_j^t = \boldsymbol{g}_{i_*}^t$ for all $j \in [m]$.
  The server computes the global model update $\hat{\boldsymbol{g}}^t$ based on Eq. (6) if using the Trimmed-mean aggregation rule, or according to Eq. (7) if using the Median aggregation rule.
  The server updates the global model as $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \eta \cdot \hat{\boldsymbol{g}}^t$.
end for

Similarly, if the Median protocol is used, the global model update for each dimension $k \in [d]$ is determined by:

$$\hat{\boldsymbol{g}}^t[k] = \text{Median}\{\boldsymbol{g}_1^t[k], \ldots, \boldsymbol{g}_n^t[k], \bar{\boldsymbol{g}}_1^t[k], \ldots, \bar{\boldsymbol{g}}_m^t[k]\} \quad (7)$$
$$\text{s.t. } \bar{\boldsymbol{g}}_1^t[k] = \cdots = \bar{\boldsymbol{g}}_m^t[k] = \boldsymbol{g}_{i_*}^t[k].$$

Finally, the server updates the global model with $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \eta \cdot \hat{\boldsymbol{g}}^t$. It is important to note that the server does not update the global model directly using the selected local model update $\boldsymbol{g}_{i_*}^t$. Although $\boldsymbol{g}_{i_*}^t$ shows the greatest deviation from the extreme updates, it may still contain extreme values in certain dimensions. Thus, the server employs coordinate-wise robust aggregation methods like the Trimmed-mean or Median to mitigate the influence of outliers in each dimension. Algorithm 1 shows the pseudocode of FoundationFL framework.

V. Theoretical Analysis

In this section, we present a convergence analysis of our proposed FoundationFL framework. In our theoretical proof, we assume that the server generates synthetic model updates using a clean dataset D_0. Both D_0 and the collective training data $D = \bigcup_{i=1}^n D_i$ from clients are presumed to be drawn from the same distribution. This assumption is strictly for theoretical analysis purposes. As demonstrated in Section IV, the server generates synthetic model updates based exclusively on the received model updates from clients. Define Q as $Q = \max\{|D_0|, |D_1|, \cdots, |D_n|\}$. Additionally, we adopt the standard assumptions prevalent in the FL literature [16], [27], [33], [34].

Assumption 1. *The loss function $\mathcal{L}(\boldsymbol{\theta})$ is μ-strongly convex. Let Θ represent the parameter space. For any $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \Theta$, the following inequality holds:*

$$L(\boldsymbol{\theta}_1) + \langle \nabla L(\boldsymbol{\theta}_1), \boldsymbol{\theta}_2 - \boldsymbol{\theta}_1 \rangle + \frac{\mu}{2}\|\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1\|^2 \leq L(\boldsymbol{\theta}_2).$$

Assumption 2. *The loss functions are λ-smooth. For any $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \Theta$, the following inequalities are satisfied:*

$$\|\nabla L(\boldsymbol{\theta}_1) - \nabla L(\boldsymbol{\theta}_2)\| \leq \lambda \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|,$$
$$\|\nabla l(\boldsymbol{\theta}_1, D) - \nabla l(\boldsymbol{\theta}_2, D)\| \leq \lambda \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|.$$

Assumption 3. *The diameter of the parameter space is limited. Specifically, for any $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \Theta$, it can be stated that:*

$$\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\| \leq \varpi.$$

Assumption 4. *The expected squared norm of gradient is bounded by ζ, and the variance of gradient is bounded by σ^2. Specifically, for any $\boldsymbol{\theta} \in \Theta$, the following inequalities hold:*

$$\mathbb{E}[\|\nabla l(\boldsymbol{\theta}, D)\|^2] \leq \zeta,$$
$$\mathbb{E}[\|\nabla l(\boldsymbol{\theta}, D) - \mathbb{E}[\nabla l(\boldsymbol{\theta}, D)]\|^2] \leq \sigma^2.$$

Assumption 5. *For a given dimension $k \in [d]$, let $\partial_k l(\boldsymbol{\theta}, D)$ denote the partial derivative of $l(\boldsymbol{\theta}, D)$ with respect to $\boldsymbol{\theta}[k]$, where $\boldsymbol{\theta}[k]$ represents the k-th dimension of the model parameter $\boldsymbol{\theta}$. We assume that $\partial_k l(\boldsymbol{\theta}, D)$ is ρ-sub-exponential for any $k \in [d]$.*

Based on the above assumptions, we present the theoretical results of our proposed FoundationFL framework.

Theorem 1. *Assuming that Assumptions 1-3 and Assumption 5 are valid and the client's learning rate is $\alpha = \frac{1}{\lambda}$, our proposed FoundationFL framework uses the Trimmed-mean aggregation rule to combine both generated synthetic model updates and model updates from clients. Given $\upsilon > 0$, if $\beta = \frac{f}{n+m}$ and the trim parameter c meet the criteria $\beta \leq \frac{c}{n+m} \leq \frac{1}{2} - \upsilon$, where f is the number of malicious clients. Then after T rounds of global training, the probability of achieving the following result is at least $1 - \frac{4d}{(1+(n+m)\lambda\varpi Q)^d}$:*

$$\left\|\boldsymbol{\theta}^T - \boldsymbol{\theta}^*\right\| \leq (1 - \frac{\mu}{\mu+\lambda})^T \left\|\boldsymbol{\theta}^0 - \boldsymbol{\theta}^*\right\| + \frac{2B_1}{\mu},$$

where $\boldsymbol{\theta}^T$ is the global model at training round T, $\boldsymbol{\theta}^0$ is the initial global model, $\boldsymbol{\theta}^*$ is the optimal model under no attack, $B_1 = \mathcal{O}((\frac{\rho c d}{\upsilon(n+m)\sqrt{Q}} + \frac{\rho d}{\upsilon\sqrt{(n+m)Q}})\sqrt{\log((n+m)\lambda\varpi Q)})$.

Proof. The proof is relegated to Appendix A. □

Theorem 2. *Under the assumptions that Assumptions 1-4 hold true and the client's learning rate is set as $\alpha = \frac{1}{\lambda}$, our proposed framework, denoted as FoundationFL, uses the Median aggregation rule to merge synthetic updates and updates contributed by clients. Assuming $\upsilon > 0$, if $\beta = \frac{f}{n+m}$*

satisfies $\beta + \epsilon \leq \frac{1}{2} - \upsilon$*, then after* T *rounds of global training, the probability of achieving the following outcome is guaranteed to be at least* $1 - \frac{4d}{(1+(n+m)\lambda\varpi Q)^d}$*:*

$$\left\|\boldsymbol{\theta}^T - \boldsymbol{\theta}^*\right\| \leq (1 - \frac{\mu}{\mu+\lambda})^T \left\|\boldsymbol{\theta}^0 - \boldsymbol{\theta}^*\right\| + \frac{2B_2}{\mu},$$

where $\epsilon = \frac{0.4748\zeta}{\sqrt{Q}} + \sqrt{\frac{d\log(1+(n+m)\lambda\varpi Q)}{(n+m)(1-\beta)}}$, $B_2 = \frac{2\sqrt{2}}{(n+m)Q} + \frac{2\sqrt{\pi}\sigma(\beta+\epsilon)\exp(\frac{1}{2}(\Phi^{-1}(1-\upsilon))^2)}{\sqrt{Q}}$*, and* Φ *represents the cumulative distribution function of the standard Gaussian distribution.*

Proof. The proof is relegated to Appendix B. □

Remark. *In our theoretical analysis, we adopt simplifying assumptions commonly used in the FL community [16], [27], [33], [34]. However, we acknowledge that these assumptions may not fully capture real-world conditions. To assess* FoundationFL*'s sensitivity, we conduct additional experiments by relaxing certain assumptions, such as Assumption 1, which pertains to the non-convexity of deep neural networks. Specifically, we test* FoundationFL *on two CNN architectures and the more complex ResNet-18 model [35], neither of which satisfies Assumption 1. Extensive experimental results show that our proposed* FoundationFL *remains secure even when certain assumptions are partially relaxed.*

VI. Evaluation

A. Experimental Setup

1) Datasets: In our experiments, we use six distinct datasets from various domains, which encompass MNIST [36], Fashion-MNIST [37], Human Activity Recognition (HAR) [38], Purchase [39], Large-scale CelebFaces Attributes (CelebA) [40], and CIFAR-10 [41].

a) MNIST [36]: The MNIST dataset consists of 60,000 training images and 10,000 testing images, encompassing a total of 10 unique classes.

b) Fashion-MNIST [37]: The Fashion-MNIST dataset comprises a total of 70,000 fashion images. The training dataset contains 60,000 images, and the testing set contains 10,000 images. Each image in Fashion-MNIST is assigned to one of 10 classes.

c) Human Activity Recognition (HAR) [38]: The HAR dataset is a practical dataset used for predicting human activities. It includes data collected from 30 users who used smartphones in their daily routines, totaling 10,299 instances. Each instance consists of 561 features and is classified into one of six distinct categories. Following the approach in [15], in our experiments, we randomly assign 75% of the data from each user for training, while the remaining 25% is kept aside for testing.

d) Purchase [39]: The purchase classification dataset is imbalanced, containing 197,324 examples, each characterized by 600 binary attributes distributed among 100 different categories. In our experiments, we randomly choose a subset of 150,000 examples for training our models, reserving the remaining 47,324 examples for testing purposes.

e) Large-scale CelebFaces Attributes (CelebA) [40]: This dataset involves a binary classification task to determine whether the person in an image is smiling or not. The CelebA dataset includes 177,480 training examples and 22,808 testing examples.

f) CIFAR-10 [41]: CIFAR-10 is a color image classification dataset with 10 distinct classes. It includes a total of 60,000 images, with 50,000 used for training and the remaining 10,000 for testing.

2) Poisoning Attacks: By default, our experiments examine six untargeted attacks (label flipping attack [11], Gaussian attack [8], Trim attack [9], Krum attack [9], Min-Max attack [10], Min-Sum attack [10]) and one targeted attack (Scaling attack [5], [15]). Note that we also consider two additional targeted attacks and three more sophisticated attacks in Section VII. By evaluating our method with a total of 12 representative poisoning attacks, covering both untargeted and targeted strategies, we ensure a comprehensive evaluation that reflects real-world scenarios and challenges, rigorously testing our method's robustness across diverse attack models. These attacks are selected because they are widely studied in the literature [9], [10], [11], [15], [19], [25] and represent a range of commonly observed methods posing substantial threats to FL systems.

a) Label flipping (LF) attack [11]: In the LF attack, the attacker modifies the labels of training data on malicious clients. Specifically, for a training example originally labeled as y, the attacker changes it to $M-y-1$, where M represents the total number of labels.

b) Gaussian attack [8]: In this specific attack, each malicious client sends a randomly generated vector to the server. These vectors are sampled from a Gaussian distribution with a mean of 0 and a variance of 200.

c) Trim attack [9]: The Trim attack is a strategy targeting aggregation Trimmed-mean and Median aggregation rules. In this attack, the attacker carefully designs the model updates on malicious clients to ensure that the post-attack model update diverges significantly from its pre-attack one.

d) Krum attack [9]: The Krum attack is an advanced strategy aimed at exploiting the Krum aggregation rule. The attacker strategically crafts the model updates on malicious clients to influence the Krum rule into choosing the malicious update as the final aggregated outcome.

e) Min-Max attack [10]: In the Min-Max attack, the attacker tailors local model updates on malicious clients to achieve their objectives, ensuring that these updates closely resemble benign updates. Specifically, the maximum distance between a malicious local model update and any benign local model update is smaller than the maximum distance between any two benign local model updates.

f) Min-Sum attack [10]: The Min-Sum attack is another attack model that is agnostic to aggregation rules. Unlike the Min-Max attack, in the Min-Sum attack, the attacker ensures that the sum of distances between a malicious local model

update and all benign local model updates does not exceed the maximum sum of distances between any two benign updates.

g) Scaling attack [5], [15]: The Scaling attack is a type of targeted attack where the attacker initially augments the local training data of malicious clients by introducing backdoor triggers into duplicated data copies. Subsequently, these malicious clients train their local models using the augmented training data and further amplify their local model updates before transmitting them to the server.

3) Compared Aggregation Rules: By default, we compare our proposed FoundationFL with the following seven aggregation rules.

a) FedAvg [1]: FedAvg is a non-robust aggregation method where the server aggregates received local model updates by computing the average of all updates.

b) Trimmed-mean (Trim-mean) [16]: Trimmed-mean is an aggregation method applied per coordinate. For each dimension, the server removes the largest c and smallest c elements, then computes the average of the remaining values. Here, c is referred to as the trim parameter.

c) GAS + Trim-mean [42]: Upon receiving the local model updates from all clients, the server splits each update into multiple parts. The server then applies the Trimmed-mean [16] aggregation rule to compute the aggregated result for each part. Subsequently, the server calculates an identification score for each client and selects the $n - f$ local model updates with the lowest identification scores for aggregation by taking the average of these $n - f$ updates, where f represents the total number of malicious clients.

d) Gaussian + Trim-mean: Upon receiving n local model updates from clients, the server creates m synthetic updates. Each k-th dimension of these updates follows a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, where μ and σ are the mean and standard deviation of $\{\boldsymbol{g}_i^t[k] : i \in [n]\}$ with $\boldsymbol{g}_i^t[k]$ representing the k-th dimension of $\boldsymbol{g}_i^t$, and $k \in [d]$. Subsequently, the server uses the Trimmed-mean [16] aggregation rule to merge these m synthetic updates with the n received updates.

e) Median [16]: The Median is another aggregation rule that is applied on a per-coordinate basis. For each dimension, the server calculates the median value from all the received model updates.

f) GAS + Median [42]: Similar to the GAS + Trim-mean aggregation rule, the server in the GAS + Median rule also divides each local model update into multiple parts. However, in this method, the server applies the Median [16] aggregation rule to each part. Then, the server calculates the final aggregated update by averaging the $n - f$ local model updates with the lowest identification scores.

g) Gaussian + Median: For this method, the server follows the same procedure as in the Gaussian + Trim-mean method to produce the m synthetic updates. The only distinction lies in the aggregation technique; here, the server employs the Median [16] aggregation rule to aggregate the m synthetic updates with the n local model updates from clients.

4) Evaluation Metrics: Two evaluation metrics are explored in this paper.

a) Testing error rate: The testing error rate reflects the percentage of test instances incorrectly classified by the global model. A lower testing error rate signifies a stronger defense.

b) Attack success rate: The attack success rate is determined by the proportion of targeted examples predicted as the labels chosen by the attacker. A lower attack success rate indicates more effective defense.

5) Non-IID Setting: In FL, a distinctive aspect is that clients' local training data are not independently and identically distributed (Non-IID). In our study, we adopt the following method to simulate the Non-IID setting as described in [9]. For a dataset with M classes, clients are initially randomly grouped into M clusters. A training example labeled y is then assigned to clients in cluster y with probability h, and to other clusters with probability $\frac{1-h}{M-1}$. A higher value of h indicates greater Non-IID characteristics in the clients' training data. For the MNIST and Fashion-MNIST datasets, we set $h = 0.5$. It is important to note that we do not simulate the Non-IID setting for the HAR, Purchase, and CelebA datasets, as these datasets inherently exhibit heterogeneity.

6) Parameter Settings: We consider 100 clients each for the MNIST, Fashion-MNIST, and CIFAR-10 datasets ($n = 100$), 40 clients for the Purchase dataset, 20 clients for the CelebA dataset, and 30 clients in total for the HAR dataset, where each real-world user is treated as a client. By default, we assume 20% of the clients are malicious. For the MNIST, Fashion-MNIST, and CelebA datasets, we train a convolutional neural network (CNN) whose architecture is detailed in Table IXa in Appendix. The HAR dataset is trained using a logistic regression classifier. The Purchase dataset employs a fully connected neural network as the global model architecture with one hidden layer consisting of 1,024 neurons and Tanh activation function. We train a ResNet-18 [35] model for the CIFAR-10 dataset. We conduct training for 2,000 rounds on the MNIST dataset, 3,000 rounds on Fashion-MNIST, 1,000 rounds each on the HAR and Purchase datasets, 1,000 rounds on CelebA, and 1,000 rounds on CIFAR-10. The batch sizes are set to 32 for MNIST and Fashion-MNIST, 32 for HAR, 128 for Purchase, 20 for CelebA, and 40 for CIFAR-10. The corresponding learning rates are 1/3,200 for MNIST, Fashion-MNIST, and HAR, 1/1,280 for Purchase, 1/20,000 for CelebA, and 0.005 for CIFAR-10 dataset. Following [8], [16], [42], we set $c = f$ by default. In the GAS approach, we use the parameter as recommended in [42]. Unless stated otherwise, we assume that all clients participate in the training process in every round. In the FoundationFL framework we propose, the server generates $\frac{n}{2}$ synthetic updates in each training round by default for MNIST, Fashion-MNIST, HAR, Purchase, and CelebA. The results are primarily reported using the MNIST dataset by default.

B. Experimental Results

Our FoundationFL is effective: Table I displays the performance of various FL methods under different poisoning

TABLE I: Results of different FL methods on MNIST, Fashion-MNIST, HAR, and Purchase datasets. The results of Scaling attack are shown as "testing error rate / attack success rate".

(a) MNIST dataset.

Aggregation rule	No attack	LF attack	Gaussian attack	Trim attack	Krum attack	Min-Max attack	Min-Sum attack	Scaling attack
FedAvg	0.05	0.07	0.90	0.32	0.10	0.90	0.90	0.64 / 0.70
Trim-mean	0.06	0.06	0.06	0.27	0.08	0.19	0.13	0.13 / 0.02
GAS + Trim-mean	0.05	0.05	0.11	0.29	0.07	0.10	0.11	0.43 / 0.47
Gaussian + Trim-mean	0.05	0.11	0.91	0.91	0.05	0.08	0.06	0.91 / 1.00
FoundationFL + Trim-mean	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05 / 0.02
Median	0.05	0.09	0.16	0.23	0.17	0.19	0.23	0.05 / 0.02
GAS + Median	0.05	0.05	0.12	0.26	0.06	0.10	0.10	0.59 / 0.65
Gaussian + Median	0.05	0.90	0.90	0.90	0.05	0.14	0.14	0.91 / 1.00
FoundationFL + Median	0.05	0.05	0.05	0.05	0.07	0.05	0.05	0.06 / 0.02

(b) Fashion-MNIST dataset.

Aggregation rule	No attack	LF attack	Gaussian attack	Trim attack	Krum attack	Min-Max attack	Min-Sum attack	Scaling attack
FedAvg	0.18	0.21	0.21	0.38	0.21	0.30	0.32	0.63 / 0.68
Trim-mean	0.21	0.29	0.26	0.47	0.32	0.24	0.23	0.26 / 0.01
GAS + Trim-mean	0.21	0.21	0.24	0.50	0.26	0.26	0.23	0.67 / 0.62
Gaussian + Trim-mean	0.18	0.90	0.90	0.90	0.20	0.32	0.90	0.90 / 1.00
FoundationFL + Trim-mean	0.18	0.18	0.18	0.19	0.20	0.19	0.18	0.22 / 0.03
Median	0.23	0.27	0.27	0.35	0.29	0.31	0.29	0.29 / 0.03
GAS + Median	0.20	0.20	0.24	0.44	0.25	0.26	0.23	0.60 / 0.64
Gaussian + Median	0.23	0.90	0.35	0.90	0.90	0.90	0.90	0.90 / 1.00
FoundationFL + Median	0.18	0.20	0.18	0.18	0.21	0.19	0.20	0.23 / 0.03

(c) HAR dataset.

Aggregation rule	No attack	LF attack	Gaussian attack	Trim attack	Krum attack	Min-Max attack	Min-Sum attack	Scaling attack
FedAvg	0.05	0.12	0.13	0.38	0.06	0.14	0.17	0.05 / 0.91
Trim-mean	0.07	0.07	0.07	0.26	0.09	0.16	0.16	0.07 / 0.02
GAS + Trim-mean	0.06	0.09	0.12	0.35	0.06	0.12	0.15	0.07 / 0.92
Gaussian + Trim-mean	0.05	0.12	0.41	0.24	0.06	0.17	0.17	0.62 / 0.01
FoundationFL + Trim-mean	0.05	0.08	0.05	0.08	0.06	0.09	0.09	0.05 / 0.01
Median	0.07	0.08	0.09	0.16	0.08	0.17	0.15	0.07 / 0.02
GAS + Median	0.07	0.10	0.11	0.41	0.06	0.16	0.17	0.07 / 0.88
Gaussian + Median	0.06	0.14	0.11	0.30	0.06	0.15	0.15	0.08 / 0.10
FoundationFL + Median	0.05	0.09	0.06	0.09	0.06	0.09	0.09	0.05 / 0.02

(d) Purchase Dataset.

Aggregation rule	No attack	LF attack	Gaussian attack	Trim attack	Krum attack	Min-Max attack	Min-Sum attack	Scaling attack
FedAvg	0.17	0.32	0.72	0.60	0.20	0.34	0.34	0.99 / 0.41
Trim-mean	0.21	0.28	0.21	0.47	0.26	0.40	0.40	0.25 / 0.02
GAS + Trim-mean	0.18	0.30	0.79	0.70	0.21	0.37	0.35	0.23 / 0.01
Gaussian + Trim-mean	0.17	0.20	0.96	0.58	0.17	0.38	0.38	0.99 / 1.00
FoundationFL + Trim-mean	0.17	0.20	0.17	0.21	0.19	0.21	0.22	0.18 / 0.02
Median	0.23	0.32	0.25	0.48	0.30	0.49	0.49	0.26 / 0.03
GAS + Median	0.17	0.30	0.70	0.72	0.21	0.37	0.37	0.20 / 0.02
Gaussian + Median	0.18	0.22	0.58	0.58	0.18	0.29	0.29	0.99 / 1.00
FoundationFL + Median	0.18	0.23	0.19	0.24	0.24	0.24	0.24	0.21 / 0.03

attacks on the MNIST, Fashion-MNIST, HAR, and Purchase datasets. The results for the CelebA and CIFAR-10 datasets are presented in Table X in Appendix. The term "No attack" indicates that all client in the FL system are benign without any malicious clients. The results of the Scaling attack are shown in the form of "testing error rate / attack success rate". The terms "FoundationFL + Trim-mean" and "FoundationFL + Median" describe our method where the server combines clients' local model updates and synthetic updates using Trimmed-mean and Median aggregation rules, respectively. Notably, under benign conditions, our FoundationFL framework mirrors the performance of FedAvg. For instance, both "FoundationFL + Trim-mean" and "FoundationFL + Median" achieve similar testing error rates as FedAvg on the MNIST, Fashion-MNIST, and HAR datasets when no malicious clients are present. However, in the presence of malicious clients attempting to compromise the system, our FoundationFL framework is uniquely capable of defending against such attacks. For example, on the MNIST dataset, our approach under attack performs comparably to FedAvg in a non-attack scenario. Nonetheless, existing Byzantine-robust foundational aggregation rules like Trim-mean and Median show inherent weaknesses to poisoning attacks; for example, the testing error rate for Trim-mean on the Fashion-MNIST dataset escalates from 0.21 under no attack to 0.47 under the Trim attack. Similarly, more complex aggregation schemes such as "GAS + Trim-mean" and "GAS + Median" are also susceptible. Even on a complex dataset and with a complicated architecture like the ResNet-18 model for the CIFAR-10 image classification task, our proposed FoundationFL can protect

FL against poisoning attacks. In contrast, other aggregation rules, such as Trim-mean and Median, show higher testing error rates of 0.79 and 0.84, respectively, under the Trim attack (see Table Xb in Appendix). Our framework not only combats untargeted attacks but also protects effectively against targeted threats like the Scaling attack, as evidenced in Table I and Table X. Note that in our proposed FoundationFL, the server produces synthetic updates after collecting local model updates from clients. Using established Byzantine-robust aggregation rules, the server then combines these augmented model updates to reduce update variance. This is further supported by our experimental findings. For instance, under the Trim attack on the MNIST dataset, the mean variances for the Median and "FoundationFL + Median" approaches are 3.39 and 0.41, respectively. This indicates that FoundationFL effectively reduces the variance in updates.

It is important to highlight that in our proposed framework, the server employs robust, coordinate-wise aggregation methods like Trimmed-mean or Median, aiming to minimize outlier effects in each dimension. That is to say, the server does not directly use the selected local model update (one synthetic update) to update the global model. "Synthetic only" in Table II illustrates the performance when the global model is updated exclusively with the selected local model update. This approach, as shown in Table II, is particularly susceptible to Krum and Scaling attacks, with testing error rates soaring from 0.08 under normal conditions to 0.91 during these attacks. "FoundationFL + FedAvg" in Table II refers to the method in which the server applies the FedAvg rule to merge synthetic updates with clients' model updates. We observe that this method is susceptible to Gaussian attack, as the synthetic updates can include extreme values in some dimensions. Therefore, we require robust aggregation rules to filter out these extreme values.

In recent years, a variety of new and sophisticated robust aggregation rules have been introduced. Table III displays the testing error rate and attack success rate for three complex and representative Byzantine-robust aggregation rules on the MNIST dataset: Krum [8], FoolsGold [43], and FLAME [19]. As observed in Table III, these advanced robust aggregation rules remain susceptible to poisoning attacks. For example, the Krum aggregation rule is fundamentally vulnerable to the Krum attack, and the FLAME method is susceptible to the Trim attack. Comparing Table I with Table III, it is evident that our proposed FoundationFL framework demonstrates greater robustness compared to these sophisticated aggregation rules.

Impact of fraction of malicious clients: Fig. 1 illustrates the impact of poisoning attacks on the performance of various FL aggregation methods within the MNIST dataset, where the proportion of malicious clients increases from 0% to 50%, with a total client base of 100. The fraction of malicious clients is computed as f/n, with f representing the number of malicious clients and n the total client count. Note that for the Trim-mean and "GAS + Trim-mean" aggregation rules, the fraction of malicious clients ranges only from 0 to 45%, as these two methods require the number of malicious clients to be less than half of the total clients. From Fig. 1, it's evident that our proposed methods, "FoundationFL + Trim-mean" and "FoundationFL + Median", remain robust against poisoning attacks, even when up to 45% of the clients are malicious. For example, the testing error rate for "FoundationFL + Trim-mean" only slightly increases from 0.05 with no attack to 0.06 under the strong Trim attack when 30% of clients are malicious. With 45% malicious clients, "FoundationFL + Trim-mean" and "FoundationFL + Median" maintain error rates no higher than 0.12 across different attacks. Conversely, with just 10% malicious clients, the testing error rate for "Gaussian + Median" reaches 0.90 under the Trim attack.

Impact of degree of Non-IID: In FL, a distinct characteristic is the Non-IID nature of clients' local training data. When this data is notably diverse, the attacker find it easier to craft malicious model updates that appear benign yet can significantly disrupt the targeted FL system. This is particularly problematic when the system employs Byzantine-robust foundational aggregation rules like Trimmed-mean or Median to merge these updates. Our findings, depicted in Fig. 2, investigates the influence of Non-IID data heterogeneity on the efficacy of various FL methods. The degree of Non-IIDness explored ranges from 0.1 to 0.7, with other parameters set to default values. The findings illustrated in Fig. 2 demonstrate that despite the high heterogeneity in clients' training data, our proposed FoundationFL framework effectively shields against diverse poisoning attacks.

Impact of fraction of synthetic updates: In our framework, once the server collects n local model updates from clients during each global training round, it proceeds to generate m synthetic updates. These $n + m$ updates are then aggregated using either the Trim-mean or Median method. This section explores how the proportion of synthetic updates, calculated as m/n, influences our proposed FoundationFL. The results are displayed in Fig. 3. From these findings, it is evident that our methods, "FoundationFL + Trim-mean" and "FoundationFL + Median", exhibit robustness against variations in the proportion of synthetic updates. It is important to note that when the fraction of synthetic updates is 0%, our "FoundationFL + Trim-mean" and "FoundationFL + Median" methods are equivalent to the "Trim-mean" and "Median" rules, respectively. From Fig. 3, we observe that FoundationFL requires between 10% and 60% synthetic updates to effectively defend against various poisoning attacks. This is because too few synthetic updates fail to mitigate the influence of malicious clients, while too many could overshadow the benign updates within the system.

Impact of total number of clients: Fig. 4 displays results across a varying total number of clients from 50 to 300, with a consistent fraction of 20% malicious clients and other parameters at default settings, using the MNIST dataset. The figure demonstrates that our methods, "FoundationFL + Trim-mean" and "FoundationFL + Median", consistently defend against poisoning attacks effectively across all client scales.

TABLE II: Results when the server uses either the synthetic update alone or FedAvg to merge the augmented model updates.

Aggregation rule	No attack	LF attack	Gaussian attack	Trim attack	Krum attack	Min-Max attack	Min-Sum attack	Scaling attack
Synthetic only	0.08	0.12	0.08	0.12	0.91	0.12	0.08	0.91 / 1.00
FoundationFL + FedAvg	0.05	0.14	0.87	0.12	0.06	0.07	0.08	0.46 / 0.59

TABLE III: Results of complex Byzantine-robust aggregation rules.

Aggregation rule	No attack	LF attack	Gaussian attack	Trim attack	Krum attack	Min-Max attack	Min-Sum attack	Scaling attack
Krum	0.10	0.10	0.10	0.10	0.90	0.11	0.11	0.10 / 0.01
FoolsGold	0.09	0.12	0.09	0.37	0.12	0.25	0.22	0.13 / 0.05
FLAME	0.07	0.08	0.08	0.17	0.08	0.07	0.07	0.08 / 0.02

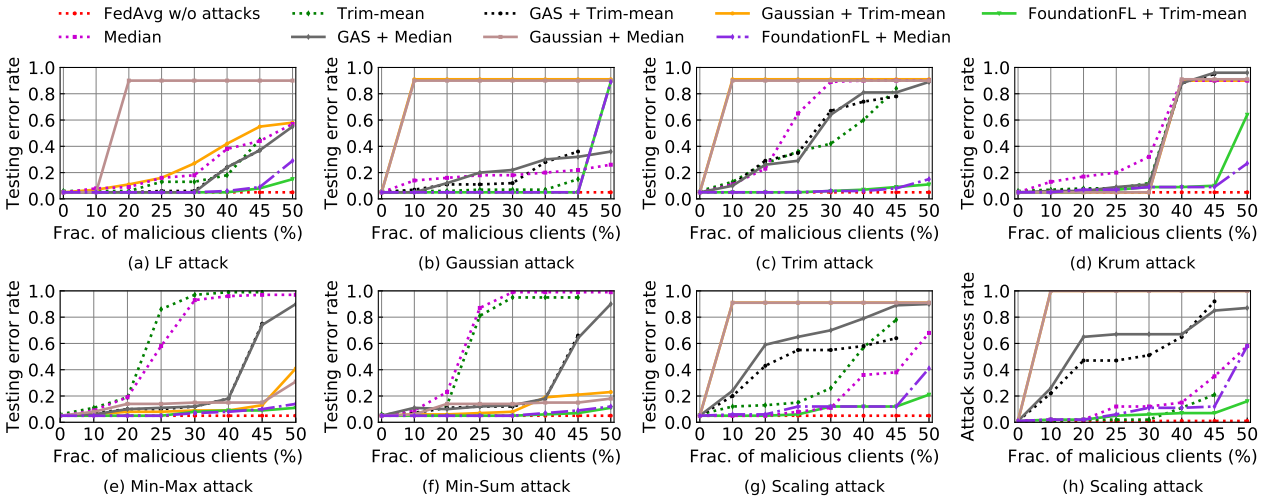

Fig. 1: Impact of fraction of malicious clients.

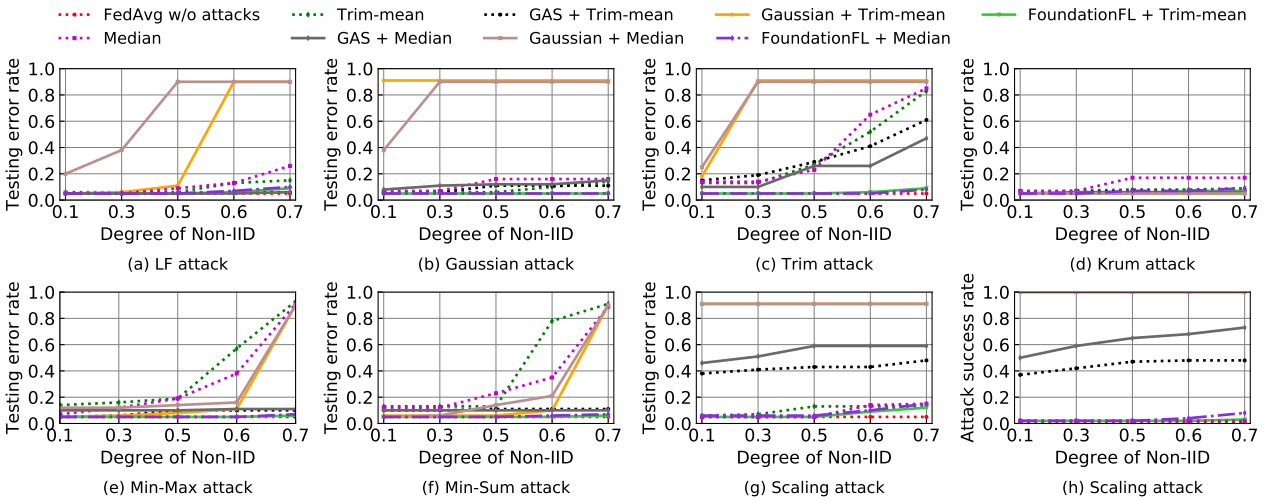

Fig. 2: Impact of degree of Non-IID.

Conversely, traditional robust aggregation methods like "GAS + Median" fail to adequately counteract the effects of these attacks. Notably, as the total number of clients ranges from 50 to 300, the testing error rates for "GAS + Median" remain significantly high under a Trim attack.

Results of various defense methods on an alternative CNN architecture: In this part, we demonstrate the robustness of various aggregation methods on an alternative CNN architecture, with details of this architecture provided in Table IXb in Appendix. Results for the different defense methods are presented in Table XI in Appendix. From Table XI, we observe that our proposed FoundationFL can effectively defend against various poisoning attacks, even with this CNN architecture. For instance, the test error rates of "FoundationFL + Trim-mean" and "FoundationFL + Median" under different attacks match those of FedAvg in the absence of any attacks. In contrast, existing FL methods show vulnerability; for example, the testing error rate of "GAS + Median" reaches 0.28 under

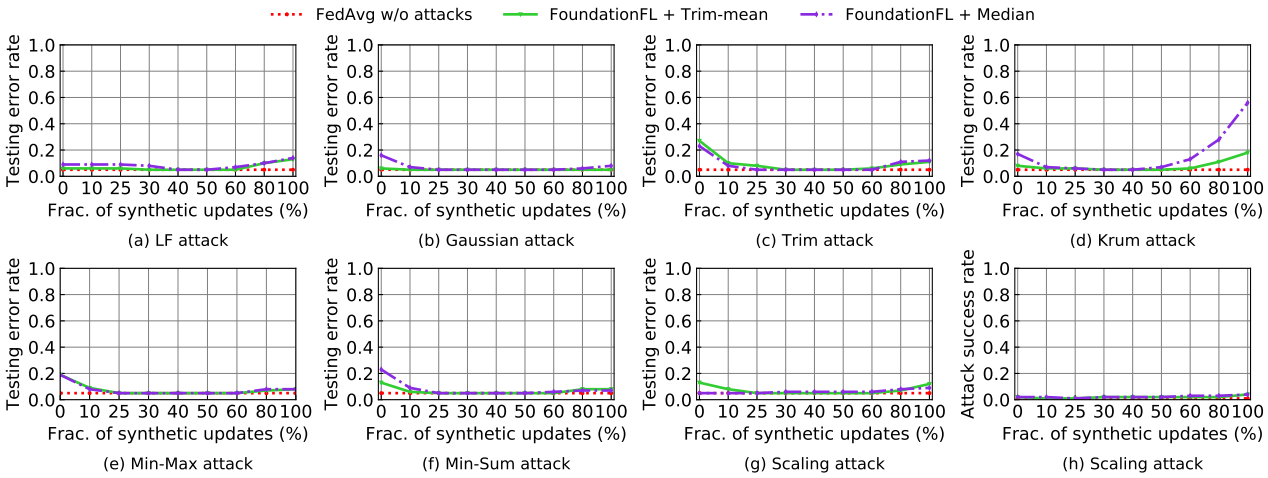

Fig. 3: Impact of fraction of synthetic updates.

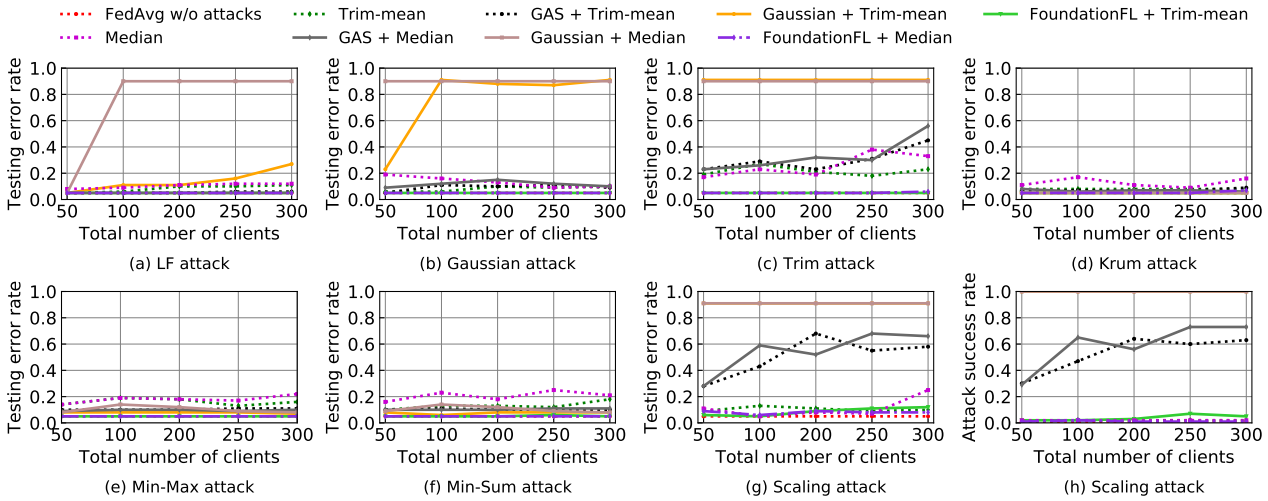

Fig. 4: Impact of total number of clients.

the strong Trim attack.

Results of various defenses with subset client selection per training round: By default, we assume full client participation in each round of FL training. Here, we examine a setup where only a subset of clients joins each round. Specifically, the server randomly selects 30% of the clients each round to receive the current global model. In this configuration, FoundationFL still generates synthetic updates amounting to half of the received local updates. For instance, if the server receives 30 model updates, it generates an additional 15 synthetic ones. Table XII in Appendix shows defense results for this subset selection scenario. We observe that FoundationFL remains robust against various poisoning attacks, whereas existing FL methods show vulnerabilities. For instance, under the Scaling attack, the testing error rate and attack success rate for "GAS + Trim-mean" reach 0.36 and 0.41, respectively.

Transferability of FoundationFL: In this part, we demonstrate that our proposed FoundationFL is transferable to other aggregation rules. Specifically, after receiving clients' model updates, the server generates synthetic updates as outlined in Section IV. Rather than applying the Trimmed-mean or Median aggregation rules, the server instead uses the aggregation methods listed in Table III, such as Krum, FoolsGold, or FLAME, to merge these updates. The results, presented in Table XIII in Appendix, show that FoundationFL effectively transfers to different aggregation protocols. For example, with FLAME method, "FoundationFL + FLAME" achieves a testing error rate of 0.10 under the Trim attack, whereas FLAME alone results in a 0.17 error rate (refer to Table III).

Scalability of FoundationFL: To illustrate the scalability of our proposed FoundationFL method, we conducted experiments on a production FL system [25], [44], [45]. Following the setup in [25], we assume a total of 1,000 clients, with 20% being malicious. In each training round, the server randomly selects 30% of clients to participate in the training process. The results, displayed in Table IV, show that both "FoundationFL + Trim-mean" and "FoundationFL + Median" under the Trim

TABLE IV: Results of different defenses on a production FL system.

Aggregation rule	No attack	LF attack	Gaussian attack	Trim attack	Krum attack	Min-Max attack	Min-Sum attack	Scaling attack
FedAvg	0.05	0.08	0.06	0.30	0.08	0.75	0.75	0.71 / 0.69
Trim-mean	0.06	0.09	0.06	0.27	0.09	0.28	0.13	0.13 / 0.01
GAS + Trim-mean	0.06	0.07	0.07	0.32	0.07	0.10	0.10	0.32 / 0.28
Gaussian + Trim-mean	0.05	0.19	0.91	0.91	0.06	0.06	0.08	0.89 / 1.00
FoundationFL + Trim-mean	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05 / 0.01
Median	0.06	0.08	0.06	0.38	0.11	0.21	0.16	0.06 / 0.02
GAS + Median	0.05	0.05	0.07	0.47	0.17	0.08	0.08	0.55 / 0.53
Gaussian + Median	0.05	0.91	0.91	0.91	0.05	0.08	0.07	0.91 / 1.00
FoundationFL + Median	0.05	0.06	0.05	0.05	0.07	0.05	0.05	0.05 / 0.02

TABLE V: Results of "FoundationFL + Trim-mean" in scenarios where the server lacks knowledge of the number of malicious clients f, the trim parameter is set to $c = 30$ for $f = 10$ and $f = 20$.

f	No attack	LF attack	Gaussian attack	Trim attack	Krum attack	Min-Max attack	Min-Sum attack	Scaling attack
$f = 10$	0.05	0.07	0.06	0.07	0.06	0.06	0.06	0.05 / 0.02
$f = 20$	0.05	0.06	0.05	0.07	0.06	0.05	0.05	0.05 / 0.02
Estimate	0.05	0.08	0.06	0.08	0.06	0.06	0.05	0.06 / 0.02

attack achieve testing error rates that align with FedAvg's performance in the absence of an attack, confirming the scalability of FoundationFL.

Performance of "FoundationFL + Trim-mean" when the number of malicious clients is unknown: Note that in the Trim-mean aggregation method, the server initially excludes the largest c and smallest c values for each dimension before calculating the average of the remaining values. Following previous studies [8], [16], [42], we assume that the trim parameter c equals the total number of malicious clients f. However, in practical scenarios, the server may not have exact knowledge of the number of malicious clients, and the attacker only knows that f is bounded by c, i.e., $c > f$. Table V presents the results of our proposed "FoundationFL + Trim-mean" approach when the server lacks precise information about the number of malicious clients. Specifically, the trim parameter is set to $c = 30$, meaning the server excludes the largest 30 and smallest 30 values per dimension and averages the remaining 40 values (out of 100 clients in total). "$f = 10$" indicates that there are actually 10 malicious clients. Note that in both cases where "$f = 10$" and "$f = 20$", the server still generates $\frac{n}{2} = 50$ synthetic updates. "Estimate" refers to the approach in which the server approximates the number of malicious clients in the system. Specifically, in each round, the server calculates the pairwise cosine distances between each pair of client model updates, then applies the K-means [46] clustering algorithm to divide all client model updates into two clusters. Based on the assumption that the majority of clients are benign, the cluster with fewer clients is considered malicious, and the trim parameter, representing the estimated number of malicious clients, is set to the size of this cluster. The results in Table V demonstrate that our proposed "FoundationFL + Trim-mean" remains effective even when the actual number of malicious clients is within the bounds defined by the trim parameter or the server approximates the count of malicious clients.

Computation cost of different FL methods: Fig. 5 illustrates the computational costs of various FL methods during 2,000 rounds of training on the MNIST dataset. The computation cost represents the time required by the server to aggregate model updates over these rounds. It is important to note that for our proposed method, the computation cost also includes the time taken to generate synthetic updates. In Fig. 5, "Trim", "GAS+Trim", "Gaussian+Trim", "Found+Trim", and "Found+Median" refer to the methods "Trim-mean", "GAS + Trim-mean", "Gaussian + Trim-mean", "FoundationFL + Trim-mean", and "FoundationFL + Median", respectively. The additional computational overhead introduced by FoundationFL primarily stems from the generation of synthetic updates, which are designed to enhance robustness. Although this process requires extra computations compared to standard FL approaches, the overhead remains manageable and does not compromise system robustness or accuracy. The synthetic update generation process has been fine-tuned to operate efficiently within each round, ensuring that the overall latency does not diverge significantly from baseline methods. Additionally, as shown in Fig. 5, the method incorporating FoundationFL with a robust aggregator (Trim-mean or Median) exhibits slightly higher computational overhead compared to FedAvg, underscoring the effectiveness of our approach in balancing robustness and efficiency.

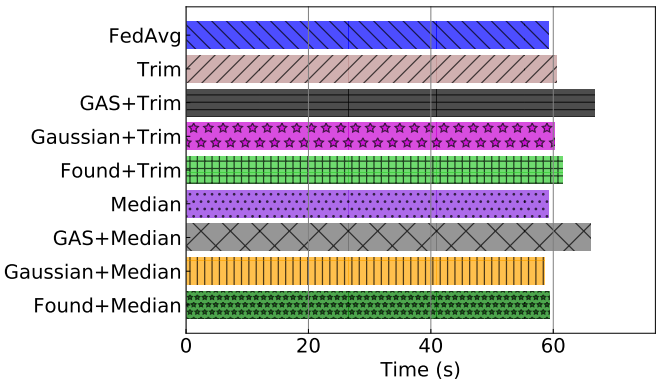

Fig. 5: Computation cost of different FL methods.

TABLE VI: Results of different FL methods, with each client possessing only four labels.

Aggregation rule	No attack	LF attack	Gaussian attack	Trim attack	Krum attack	Min-Max attack	Min-Sum attack	Scaling attack
FedAvg	0.06	0.27	0.09	0.13	0.06	0.11	0.09	0.07 / 0.15
Trim-mean	0.08	0.31	0.08	0.42	0.19	0.12	0.12	0.08 / 0.05
GAS + Trim-mean	0.06	0.06	0.09	0.31	0.12	0.11	0.11	0.09 / 0.29
Gaussian + Trim-mean	0.06	0.26	0.91	0.91	0.06	0.08	0.08	0.91 / 1.00
FoundationFL + Trim-mean	0.06	0.08	0.06	0.09	0.07	0.07	0.06	0.06 / 0.03
Median	0.07	0.32	0.33	0.58	0.25	0.19	0.20	0.09 / 0.05
GAS + Median	0.07	0.07	0.11	0.35	0.12	0.11	0.11	0.15 / 0.23
Gaussian + Median	0.06	0.90	0.89	0.89	0.07	0.17	0.18	0.91 / 1.00
FoundationFL + Median	0.06	0.10	0.06	0.11	0.11	0.07	0.08	0.06 / 0.02

TABLE VII: Results of various defenses against DBA and Neurotoxin attacks.

(a) MNIST dataset.

Aggregation rule	DBA	Neurotoxin
FedAvg	0.48 / 0.65	0.76 / 0.73
Trim-mean	0.06 / 0.02	0.09 / 0.02
GAS + Trim-mean	0.27 / 0.19	0.43 / 0.54
Gaussian + Trim-mean	0.91 / 1.00	0.91 / 1.00
FoundationFL + Trim-mean	0.05 / 0.02	0.05 / 0.01
Median	0.05 / 0.02	0.05 / 0.02
GAS + Median	0.43 / 0.51	0.76 / 0.82
Gaussian + Median	0.91 / 1.00	0.91 / 1.00
FoundationFL + Median	0.05 / 0.02	0.06 / 0.02

(b) Fashion-MNIST dataset.

Aggregation rule	DBA	Neurotoxin
FedAvg	0.76 / 0.72	0.71 / 0.79
Trim-mean	0.29 / 0.06	0.30 / 0.05
GAS + Trim-mean	0.53 / 0.62	0.60 / 0.71
Gaussian + Trim-mean	0.90 / 1.00	0.90 / 1.00
FoundationFL + Trim-mean	0.22 / 0.02	0.22 / 0.02
Median	0.27 / 0.03	0.31 / 0.03
GAS + Median	0.60 / 0.61	0.78 / 0.83
Gaussian + Median	0.90 / 1.00	0.90 / 1.00
FoundationFL + Median	0.23 / 0.03	0.22 / 0.02

(c) HAR dataset.

Aggregation rule	DBA	Neurotoxin
FedAvg	0.05 / 0.91	0.05 / 0.89
Trim-mean	0.07 / 0.02	0.07 / 0.02
GAS + Trim-mean	0.14 / 0.68	0.07 / 0.84
Gaussian + Trim-mean	0.59 / 0.02	0.75 / 0.16
FoundationFL + Trim-mean	0.05 / 0.01	0.05 / 0.01
Median	0.07 / 0.02	0.07 / 0.02
GAS + Median	0.07 / 0.75	0.11 / 0.82
Gaussian + Median	0.07 / 0.03	0.08 / 0.10
FoundationFL + Median	0.05 / 0.01	0.05 / 0.01

(d) Purchase dataset.

Aggregation rule	DBA	Neurotoxin
FedAvg	0.99 / 0.31	0.99 / 0.35
Trim-mean	0.21 / 0.04	0.24 / 0.02
GAS + Trim-mean	0.25 / 0.09	0.33 / 0.06
Gaussian + Trim-mean	0.99 / 1.00	0.99 / 1.00
FoundationFL + Trim-mean	0.20 / 0.02	0.18 / 0.01
Median	0.23 / 0.01	0.26 / 0.03
GAS + Median	0.22 / 0.02	0.29 / 0.18
Gaussian + Median	0.99 / 1.00	0.99 / 1.00
FoundationFL + Median	0.21 / 0.03	0.23 / 0.03

(e) CelebA dataset.

Aggregation rule	DBA	Neurotoxin
FedAvg	0.41 / 0.05	0.46 / 0.05
Trim-mean	0.49 / 0.11	0.35 / 0.03
GAS + Trim-mean	0.36 / 0.04	0.36 / 0.03
Gaussian + Trim-mean	0.52 / 0.17	0.64 / 0.20
FoundationFL + Trim-mean	0.24 / 0.02	0.24 / 0.02
Median	0.28 / 0.08	0.39 / 0.12
GAS + Median	0.32 / 0.05	0.31 / 0.09
Gaussian + Median	0.43 / 0.02	0.49 / 0.04
FoundationFL + Median	0.24 / 0.01	0.25 / 0.02

(f) CIFAR-10 dataset.

Aggregation rule	DBA	Neurotoxin
FedAvg	0.90 / 1.00	0.90 / 1.00
Trim-mean	0.28 / 0.96	0.26 / 0.99
GAS + Trim-mean	0.25 / 0.80	0.31 / 0.92
Gaussian + Trim-mean	0.99 / 1.00	0.98 / 1.00
FoundationFL + Trim-mean	0.22 / 0.01	0.23 / 0.02
Median	0.31 / 0.84	0.37 / 0.91
GAS + Median	0.45 / 0.96	0.32 / 0.99
Gaussian + Median	0.99 / 1.00	0.99 / 1.00
FoundationFL + Median	0.24 / 0.02	0.26 / 0.03

VII. Discussion and Limitations

More extreme Non-IID distribution: In the previous section, we demonstrated the effectiveness of our proposed method in safeguarding FL systems against highly heterogeneous local training data among clients. Here, we explore a more extreme scenario where each client possesses only a few distinct labels. Specifically, we consider a situation where each client has only four different labels in their training data. For instance, one client may have data labeled from one to four, while another client may have data labeled from five to eight. The results of various FL methods under different attack conditions in this extreme setting are presented in Table VI. From Table VI, it is evident that FoundationFL can effectively mitigate poisoning attacks in FL systems even under such challenging conditions.

More targeted attacks: Here, we demonstrate the robustness of various defense mechanisms against two more targeted attacks: the DBA attack [12] and the Neurotoxin attack [47]. Table VII presents the results across six datasets, where "DBA" and "Neurotoxin" represent the respective attacks. Our FoundationFL method shows low testing error rates and attack success rates under these attacks. In contrast, existing aggregation rules remain vulnerable to poisoning attacks; for example, the attack success rate of Trim-mean reaches 0.96 on the CIFAR-10 dataset under the DBA attack.

More sophisticated and adaptive attacks: In Section VI, we show the capability of our proposed FoundationFL framework to effectively mitigate seven distinct poisoning attacks. In this section, we further examine three additional sophisticated attacks, which include two adaptive attack strategies.

a) MPAF attack [48]: MPAF is an untargeted attack where the attacker steers the current global model towards an attacker-chosen model during each training round.

b) Adaptive attack I [6]: In this untargeted attack, the attacker introduces small perturbations to benign local model updates to hinder the convergence of the global model. As shown in [6], this method allows the attacker to effectively compromise the final learnt global model while evading detection.

c) Adaptive attack II [6]: This attack is both targeted and adaptive, involving the insertion of backdoor triggers into local training data by the attacker on a malicious client. However, unlike scaling malicious local model updates with a fixed factor, the attacker dynamically determines the scaling factor through an optimization process.

Table VIII presents the results of various FL methods under sophisticated attack scenarios. "MPAF", "Adaptive I", and "Adaptive II" represent the "MPAF attack", "Adaptive attack I", and "Adaptive attack II" respectively. From the table, it is evident that despite the attacker's efforts to evade

TABLE VIII: Results of different FL methods under more sophisticated and adaptive attacks.

Aggregation rule	MPAF	Adaptive I	Adaptive II
FedAvg	0.90	0.90	0.90 / 1.00
Trim-mean	0.23	0.21	0.19 / 0.06
GAS + Trim-mean	0.13	0.28	0.60 / 0.62
Gaussian + Trim-mean	0.90	0.90	0.91 / 1.00
FoundationFL + Trim-mean	0.05	0.06	0.06 / 0.02
Median	0.29	0.17	0.13 / 0.02
GAS + Median	0.06	0.19	0.71 / 0.54
Gaussian + Median	0.91	0.91	0.91 / 1.00
FoundationFL + Median	0.06	0.06	0.06 / 0.03

detection, our proposed FoundationFL method effectively mitigates these sophisticated attacks.

Further discussion on the threat model for the ratio of malicious clients: From Fig. 1, we observe that our proposed FoundationFL framework can tolerate up to 45% of malicious clients. However, when 50% of clients are malicious, the testing error rates of FoundationFL become significantly higher; for example, under the Krum attack, error rates rise to 0.64 for "FoundationFL + Trim-mean" and 0.27 for "FoundationFL + Median". Nonetheless, compromising such a large fraction of malicious clients is impractical, as demonstrated in [25]. This is because achieving such high numbers of malicious clients in a real-world FL setup is unlikely due to the distributed and decentralized nature of the system, making it challenging to mobilize or control such a large fraction of participants for malicious purposes.

Potential challenges introduced by FoundationFL: Our proposed methodology incorporates synthetic updates to mitigate the influence of potentially malicious updates within the system. By employing techniques such as Trimmed-Mean or Median aggregation, we effectively reduce the weight of outlier contributions, thereby minimizing their impact and preventing the model from overfitting. Through extensive experiments, we demonstrate that this approach does not degrade performance metrics, such as testing accuracy, serving as evidence that our method is not prone to overfitting. Although bias is not the focus of this paper, we can pair our FoundationFL with bias reduction techniques, such as regularization and fairness-constrained optimization methods [49], [50], [51]. By combining our synthetic update strategy with these bias reduction techniques, we can create a more robust and fair model that not only performs well but also addresses potential biases.

Limitations of FoundationFL: In this study, we show that creating entirely new robust aggregation protocols may not be necessary to secure FL systems effectively. Rather, by strengthening the robustness of existing Byzantine-resistant foundational aggregation methods, such as Trimmed-mean or Median, we can achieve substantial resilience against poisoning attacks. Nevertheless, our proposed FoundationFL framework has certain limitations. Firstly, it is restricted to coordinate-wise aggregation rules, limiting its compatibility with other aggregation approaches. Secondly, FoundationFL introduces a slightly higher computational overhead compared to the commonly used FedAvg approach.

VIII. Conclusion

In this work, we introduced a new approach, referred to as FoundationFL, aimed at countering poisoning attacks in FL systems. Rather than designing intricate new Byzantine-robust aggregation protocols, our goal is to bolster the resilience of FL systems using established Byzantine-robust foundational aggregation protocols. In our proposed framework, the server takes a proactive stance by generating synthetic updates upon receiving local model updates from clients. Subsequently, the server employs existing Byzantine-robust foundational aggregation protocols like Trimmed-mean or Median to merge the local model updates from clients with the generated synthetic updates. We demonstrated the convergence performance of our framework under poisoning attacks and conducted extensive experiments across diverse scenarios to validate the effectiveness of our proposed techniques. In the future, we intend to expand our approach to incorporate other non-coordinate wise aggregation protocols such as Krum.

Acknowledgement

We thank the anonymous reviewers for their constructive comments.

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[2] *Utilization of FATE in Risk Management of Credit in Small and Micro Enterprises*. [Online]. Available: https://www.fedai.org/cases/utilization-of-fate-in-risk-management-of-credit-in-small-and-micro-\enterprises/

[3] *Federated Learning: Collaborative Machine Learning without Centralized Training Data*. [Online]. Available: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

[4] M. Paulik, M. Seigel, H. Mason, D. Telaar, J. Kluivers, R. van Dalen, C. W. Lau, L. Carlson, F. Granqvist, C. Vandevelde *et al.*, "Federated evaluation and tuning for on-device personalization: System design & applications," *arXiv preprint arXiv:2102.08503*, 2021.

[5] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *AISTATS*, 2020.

[6] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," in *NeurIPS*, 2019.

[7] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *ICML*, 2019.

[8] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *NeurIPS*, 2017.

[9] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *USENIX Security Symposium*, 2020.

[10] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *NDSS*, 2021.

[11] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *ESORICS*, 2020.

[12] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *ICLR*, 2019.

[13] M. Yin, Y. Xu, M. Fang, and N. Z. Gong, "Poisoning federated recommender systems with fake users," in *The Web Conference*, 2024.

[14] Z. Zhang, M. Fang, J. Huang, and Y. Liu, "Poisoning attacks on federated learning-based wireless traffic prediction," in *IFIP/IEEE Networking Conference*, 2024.

[15] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *NDSS*, 2021.
[16] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *ICML*, 2018.
[17] H. Fereidooni, A. Pegoraro, P. Rieger, A. Dmitrienko, and A.-R. Sadeghi, "Freqfed: A frequency analysis-based approach for mitigating poisoning attacks in federated learning," in *NDSS*, 2024.
[18] L. Muñoz-González, K. T. Co, and E. C. Lupu, "Byzantine-robust federated machine learning through adaptive model averaging," *arXiv preprint arXiv:1909.05125*, 2019.
[19] T. D. Nguyen, P. Rieger, R. De Viti, H. Chen, B. B. Brandenburg, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen *et al.*, "Flame: Taming backdoors in federated learning," in *USENIX Security Symposium*, 2022.
[20] X. Pan, M. Zhang, D. Wu, Q. Xiao, S. Ji, and M. Yang, "Justinian's gaavernor: Robust distributed learning with gradient aggregation agent," in *USENIX Security Symposium*, 2020.
[21] J. Park, D.-J. Han, M. Choi, and J. Moon, "Sageflow: Robust federated learning against both stragglers and adversaries," in *NeurIPS*, 2021.
[22] P. Rieger, T. D. Nguyen, M. Miettinen, and A.-R. Sadeghi, "Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection," in *NDSS*, 2022.
[23] N. Wang, Y. Xiao, Y. Chen, Y. Hu, W. Lou, and Y. T. Hou, "Flare: defending federated learning against model poisoning attacks via latent space representations," in *ASIACCS*, 2022.
[24] C. Xie, S. Koyejo, and I. Gupta, "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance," in *ICML*, 2019.
[25] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning," in *IEEE Symposium on Security and Privacy*, 2022.
[26] Y. Xie, M. Fang, and N. Z. Gong, "Model poisoning attacks to federated learning via multi-round consistency," *arXiv preprint arXiv:2404.15611*, 2024.
[27] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," in *POMACS*, 2017.
[28] M. Fang, J. Liu, N. Z. Gong, and E. S. Bentley, "Afguard: Byzantine-robust asynchronous federated learning," in *ACSAC*, 2022.
[29] M. Fang, Z. Zhang, P. Khanduri, J. Liu, S. Lu, Y. Liu, N. Gong *et al.*, "Byzantine-robust decentralized federated learning," in *CCS*, 2024.
[30] K. Kumari, P. Rieger, H. Fereidooni, M. Jadliwala, and A.-R. Sadeghi, "Baybfed: Bayesian backdoor defense for federated learning," in *IEEE Symposium on Security and Privacy*, 2023.
[31] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," in *ICML*, 2018.
[32] Y. Xu, M. Yin, M. Fang, and N. Z. Gong, "Robust federated learning mitigates client-side training data distribution inference attacks," in *The Web Conference*, 2024.
[33] T. Chu, A. Garcia-Recuero, C. Iordanou, G. Smaragdakis, and N. Laoutaris, "Securing federated sensitive topic classification against poisoning attacks," in *NDSS*, 2023.
[34] S. P. Karimireddy, L. He, and M. Jaggi, "Byzantine-robust learning on heterogeneous datasets via bucketing," in *ICLR*, 2022.
[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
[36] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *Available: http://yann. lecun. com/exdb/mnist*, 1998.
[37] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.
[38] D. Anguita, A. Ghio, L. Oneto, X. Parra Perez, and J. L. Reyes Ortiz, "A public domain dataset for human activity recognition using smartphones," in *ESANN*, 2013.
[39] *Acquire Valued Shoppers Challenge*. [Online]. Available: https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data
[40] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *ICCV*, 2015.
[41] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.
[42] Y. Liu, C. Chen, L. Lyu, F. Wu, S. Wu, and G. Chen, "Byzantine-robust learning on heterogeneous data via gradient splitting," in *ICML*, 2023.
[43] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *arXiv preprint arXiv:1808.04866*, 2018.
[44] K. Bonawitz, "Towards federated learning at scale: System design," in *MLSys*, 2019.
[45] F. Lai, Y. Dai, S. Singapuram, J. Liu, X. Zhu, H. Madhyastha, and M. Chowdhury, "Fedscale: Benchmarking model and system performance of federated learning at scale," in *ICML*, 2022.
[46] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," in *Journal of the royal statistical society. series c (applied statistics)*, 1979.
[47] Z. Zhang, A. Panda, L. Song, Y. Yang, M. Mahoney, P. Mittal, R. Kannan, and J. Gonzalez, "Neurotoxin: Durable backdoors in federated learning," in *ICML*, 2022.
[48] X. Cao and N. Z. Gong, "Mpaf: Model poisoning attacks to federated learning based on fake clients," in *CVPR Workshops*, 2022.
[49] A. Abay, Y. Zhou, N. Baracaldo, S. Rajamoni, E. Chuba, and H. Ludwig, "Mitigating bias in federated learning," *arXiv preprint arXiv:2012.02447*, 2020.
[50] S. Cui, W. Pan, J. Liang, C. Zhang, and F. Wang, "Addressing algorithmic disparity and performance inconsistency in federated learning," in *NeurIPS*, 2021.
[51] Y. Guo, X. Tang, and T. Lin, "Fedbr: Improving federated learning on heterogeneous data via local learning bias reduction," in *ICML*, 2023.
[52] S. Bubeck *et al.*, "Convex optimization: Algorithms and complexity," in *Foundations and Trends in Machine Learning*, 2015.

APPENDIX

A. Proof of Theorem 1

Following [15], we adopt a slight notation abuse in our proof, using $\boldsymbol{g}_i^t$ to denote the gradient of client i in training round t. Prior to proving our main theoretical results, we first present several helpful lemmas.

Lemma 1. *Suppose that Assumptions 1-3 and Assumption 5 are satisfied, and the server employs the Trimmed-mean aggregation rule to merge the synthetic model updates and model updates from clients. At training round t, there exists a probability of at least $1 - \frac{4d}{(1+(n+m)\lambda\varpi Q)^d}$ such that the following holds:*

$$\left\|\boldsymbol{g}(\boldsymbol{\theta}^t) - \nabla L(\boldsymbol{\theta}^t)\right\| \leq B_1, \tag{8}$$

where $\boldsymbol{g}(\boldsymbol{\theta}^t)$ is the global model update, $B_1 = \mathcal{O}((\frac{\rho c d}{\upsilon(n+m)\sqrt{Q}} + \frac{\rho d}{\upsilon\sqrt{(n+m)Q}})\sqrt{\log((n+m)\lambda\varpi Q)})$.

Proof. The proof proceeds similarly to Theorem 11 in [16], and we omit it here for conciseness. □

Lemma 2. *Suppose Assumptions 1-2 hold. If the learning rate α used by the clients satisfies $\alpha = \frac{1}{\lambda}$, then in training round t, we have:*

$$\left\|\boldsymbol{\theta}^t - \alpha\nabla L(\boldsymbol{\theta}^t) - \boldsymbol{\theta}^*\right\| \leq \left(1 - \frac{\mu}{\mu+\lambda}\right)\left\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^*\right\|,$$

where $\boldsymbol{\theta}^*$ is the optimal model under no attack.

Proof. We start by analyzing the squared norm:

$$\begin{aligned}\left\|\boldsymbol{\theta}^t - \alpha\nabla L(\boldsymbol{\theta}^t) - \boldsymbol{\theta}^*\right\|^2 = \left\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^*\right\|^2 + \alpha^2\left\|\nabla L(\boldsymbol{\theta}^t)\right\|^2 \\ - 2\alpha\left\langle\boldsymbol{\theta}^t - \boldsymbol{\theta}^*, \nabla L(\boldsymbol{\theta}^t)\right\rangle.\end{aligned} \tag{9}$$

According to [52], for any $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \Theta$, we have:

$$\begin{aligned}\frac{\mu\lambda}{\mu+\lambda}\left\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\right\|^2 + \frac{1}{\mu+\lambda}\left\|\nabla L(\boldsymbol{\theta}_1) - \nabla L(\boldsymbol{\theta}_2)\right\|^2 \\ \leq \left\langle\nabla L(\boldsymbol{\theta}_1) - \nabla L(\boldsymbol{\theta}_2), \boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\right\rangle.\end{aligned} \tag{10}$$

Setting $\boldsymbol{\theta}_1 = \boldsymbol{\theta}^t$ and $\boldsymbol{\theta}_2 = \boldsymbol{\theta}^*$, and noting $\nabla L(\boldsymbol{\theta}^*) = 0$, we obtain:

$$\frac{\mu\lambda}{\mu+\lambda}\left\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^*\right\|^2 + \frac{1}{\mu+\lambda}\left\|\nabla L(\boldsymbol{\theta}^t)\right\|^2 \leq \left\langle \boldsymbol{\theta}^t - \boldsymbol{\theta}^*, \nabla L(\boldsymbol{\theta}^t)\right\rangle. \quad (11)$$

Furthermore, with $\alpha = \frac{1}{\lambda}$, we derive:

$$\left\|\boldsymbol{\theta}^t - \alpha\nabla L(\boldsymbol{\theta}^t) - \boldsymbol{\theta}^*\right\|^2 \leq \left(1 - \frac{2\mu}{\mu+\lambda}\right)\left\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^*\right\|^2. \quad (12)$$

Given $\mu \leq \lambda$, it follows that:

$$\left\|\boldsymbol{\theta}^t - \alpha\nabla L(\boldsymbol{\theta}^t) - \boldsymbol{\theta}^*\right\| \leq \left(1 - \frac{\mu}{\mu+\lambda}\right)\left\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^*\right\|, \quad (13)$$

completing the proof. □

Proof of Theorem 1: Given the lemmas presented earlier, we proceed to establish Theorem 1. In the t-th global training round, the following holds:

$$\|\boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^*\| = \|\boldsymbol{\theta}^t - \alpha\nabla \boldsymbol{g}(\boldsymbol{\theta}^t) - \boldsymbol{\theta}^*\|. \quad (14)$$

Applying the triangle inequality to the gradient terms, the right-hand side of Eq. (14) satisfies:

$$\leq \|\boldsymbol{\theta}^t - \alpha\nabla L(\boldsymbol{\theta}^t) - \boldsymbol{\theta}^*\| + \alpha\|\nabla \boldsymbol{g}(\boldsymbol{\theta}^t) - \nabla L(\boldsymbol{\theta}^t)\|. \quad (15)$$

Based on the above Lemmas 1-2, this can be simplified to:

$$\leq (1 - \frac{\mu}{\mu+\lambda})\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^*\| + \frac{B_1}{\lambda}, \quad (16)$$

where B_1 is defined as:

$$B_1 = \mathcal{O}((\frac{\rho c d}{\upsilon(n+m)\sqrt{Q}} + \frac{\rho d}{\upsilon\sqrt{(n+m)Q}})\sqrt{\log((n+m)\lambda\varpi Q)}). \quad (17)$$

Applying the condition $\alpha = \frac{1}{\lambda}$, and since $\mu \leq \lambda$, then after T global training rounds, one can further have the following:

$$\left\|\boldsymbol{\theta}^T - \boldsymbol{\theta}^*\right\| \leq (1 - \frac{\mu}{\mu+\lambda})^T\left\|\boldsymbol{\theta}^0 - \boldsymbol{\theta}^*\right\| + \frac{2B_1}{\mu}. \quad (18)$$

This concludes the convergence proof of the global model under attack to the optimal point under the given conditions and assumptions.

Lemma 3. *Assuming Assumptions 1 to 4 hold, our proposed framework, FoundationFL, employs the Median aggregation rule to integrate synthetic updates and client-contributed updates. Given $\upsilon > 0$, if $\beta = \frac{f}{n+m}$ satisfies $\beta + \epsilon \leq \frac{1}{2} - \upsilon$, then after T rounds of global training, the probability of achieving the following outcome is guaranteed to be at least $1 - \frac{4d}{(1+(n+m)\lambda\varpi Q)^d}$:*

$$\left\|\boldsymbol{g}(\boldsymbol{\theta}^t) - \nabla L(\boldsymbol{\theta}^t)\right\| \leq B_2, \quad (19)$$

where $\boldsymbol{g}(\boldsymbol{\theta}^t)$ is the global model update, ϵ is defined as $\epsilon = \frac{0.4748\zeta}{\sqrt{Q}} + \sqrt{\frac{d\log(1+(n+m)\lambda\varpi Q)}{(n+m)(1-\beta)}}$, $B_2 = \frac{2\sqrt{2}}{(n+m)Q} + \frac{2\sqrt{\pi}\sigma(\beta+\epsilon)\exp(\frac{1}{2}(\Phi^{-1}(1-\upsilon))^2)}{\sqrt{Q}}$.

Proof. The proof follows a similar approach to Theorem 8 in [16], and we omit it here for brevity. □

B. Proof of Theorem 2

The proof of Theorem 2 follow the same procedure as that of Theorem 1. The only difference is that we simplify Eq. (15) using Lemma 2 and Lemma 3 rather than Lemmas 1-2. For brevity, we omit the full proof.

TABLE IX: CNN architectures.

(a) The default CNN architecture.

Layer	Size
Input	$28 \times 28 \times 1$
Convolution + ReLU	$3 \times 3 \times 30$
Max Pooling	2×2
Convolution + ReLU	$3 \times 3 \times 50$
Max Pooling	2×2
Fully Connected + ReLU	100
Softmax	10

(b) An alternative CNN architecture.

Layer	Size
Input	$28 \times 28 \times 1$
Convolution + ReLU	$3 \times 3 \times 30$
Fully Connected + ReLU	100
Softmax	10

TABLE X: Results of different FL methods on CelebA and CIFAR-10 datasets. The results of Scaling attack are shown as "testing error rate / attack success rate".

(a) CelebA dataset.

Aggregation rule	No attack	LF attack	Gaussian attack	Trim attack	Krum attack	Min-Max attack	Min-Sum attack	Scaling attack
FedAvg	0.23	0.33	0.37	0.44	0.56	0.56	0.57	0.45 / 0.02
Trim-mean	0.31	0.33	0.30	0.48	0.36	0.56	0.52	0.34 / 0.07
GAS + Trim-mean	0.33	0.34	0.34	0.48	0.53	0.48	0.51	0.36 / 0.03
Gaussian + Trim-mean	0.23	0.30	0.53	0.32	0.42	0.54	0.52	0.48 / 0.05
FoundationFL + Trim-mean	0.23	0.23	0.23	0.23	0.25	0.23	0.23	0.24 / 0.02
Median	0.31	0.32	0.31	0.46	0.35	0.47	0.47	0.33 / 0.05
GAS + Median	0.32	0.35	0.35	0.48	0.53	0.48	0.51	0.35 / 0.03
Gaussian + Median	0.25	0.25	0.53	0.41	0.44	0.56	0.57	0.49 / 0.04
FoundationFL + Median	0.24	0.25	0.25	0.26	0.24	0.25	0.24	0.26 / 0.02

(b) CIFAR-10 dataset.

Aggregation rule	No attack	LF attack	Gaussian attack	Trim attack	Krum attack	Min-Max attack	Min-Sum attack	Scaling attack
FedAvg	0.22	0.30	0.72	0.83	0.23	0.23	0.23	0.90 / 1.00
Trim-mean	0.25	0.32	0.26	0.79	0.26	0.25	0.25	0.28 / 0.96
GAS + Trim-mean	0.25	0.29	0.80	0.90	0.25	0.25	0.26	0.37 / 0.93
Gaussian + Trim-mean	0.28	0.38	0.90	0.71	0.33	0.28	0.32	0.99 / 1.00
FoundationFL + Trim-mean	0.22	0.23	0.22	0.25	0.22	0.23	0.23	0.22 / 0.02
Median	0.25	0.30	0.25	0.84	0.30	0.27	0.26	0.28 / 0.96
GAS + Median	0.25	0.26	0.78	0.90	0.28	0.25	0.25	0.27 / 0.89
Gaussian + Median	0.32	0.72	0.85	0.76	0.80	0.67	0.75	0.90 / 1.00
FoundationFL + Median	0.23	0.25	0.23	0.24	0.23	0.23	0.23	0.24 / 0.02

TABLE XI: Results of different defense approaches on a distinct CNN architecture.

Aggregation rule	No attack	LF attack	Gaussian attack	Trim attack	Krum attack	Min-Max attack	Min-Sum attack	Scaling attack
FedAvg	0.08	0.08	0.12	0.23	0.09	0.17	0.18	0.47 / 0.66
Trim-mean	0.10	0.11	0.10	0.24	0.10	0.20	0.25	0.12 / 0.02
GAS + Trim-mean	0.08	0.08	0.11	0.22	0.09	0.13	0.13	0.39 / 0.42
Gaussian + Trim-mean	0.08	0.91	0.91	0.91	0.08	0.13	0.12	0.91 / 1.00
FoundationFL + Trim-mean	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08 / 0.01
Median	0.09	0.19	0.20	0.25	0.09	0.15	0.14	0.15 / 0.02
GAS + Median	0.08	0.08	0.13	0.28	0.08	0.13	0.13	0.55 / 0.61
Gaussian + Median	0.08	0.16	0.31	0.75	0.08	0.11	0.10	0.89 / 1.00
FoundationFL + Median	0.08	0.09	0.08	0.08	0.08	0.08	0.08	0.08 / 0.01

TABLE XII: Results of different defenses when only a subset of clients are selected in each training round.

Aggregation rule	No attack	LF attack	Gaussian attack	Trim attack	Krum attack	Min-Max attack	Min-Sum attack	Scaling attack
FedAvg	0.06	0.08	0.12	0.13	0.06	0.75	0.75	0.68 / 0.81
Trim-mean	0.06	0.07	0.07	0.26	0.08	0.11	0.15	0.13 / 0.02
GAS + Trim-mean	0.06	0.06	0.12	0.22	0.06	0.09	0.09	0.36 / 0.41
Gaussian + Trim-mean	0.06	0.12	0.91	0.91	0.06	0.07	0.06	0.89 / 0.92
FoundationFL + Trim-mean	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06 / 0.02
Median	0.06	0.08	0.07	0.28	0.29	0.11	0.13	0.06 / 0.02
GAS + Median	0.06	0.06	0.12	0.25	0.07	0.09	0.10	0.55 / 0.65
Gaussian + Median	0.06	0.90	0.90	0.90	0.06	0.09	0.09	0.91 / 1.00
FoundationFL + Median	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06 / 0.02

TABLE XIII: Transferability of FoundationFL.

Aggregation rule	No attack	LF attack	Gaussian attack	Trim attack	Krum attack	Min-Max attack	Min-Sum attack	Scaling attack
FoundationFL + Krum	0.07	0.07	0.07	0.07	0.85	0.08	0.08	0.08 / 0.01
FoundationFL + FoolsGold	0.06	0.09	0.08	0.13	0.06	0.08	0.08	0.09 / 0.02
FoundationFL + FLAME	0.07	0.08	0.08	0.10	0.07	0.07	0.07	0.07 / 0.02