FedTruth: Byzantine-Robust and Backdoor-Resilient Federated Learning Framework

Sheldon C. Ebron Jr.
The University of Memphis
Memphis, USA
sebron@memphis.edu

Kan Yang The University of Memphis Memphis, USA kan.yang@memphis.edu

ABSTRACT

Federated Learning (FL) enables collaborative machine learning model training across multiple parties without sharing raw data. However, FL's distributed nature allows malicious clients to impact model training through Byzantine or backdoor attacks, using erroneous model updates. Existing defenses measure the deviation of each update from a 'ground-truth model update.' They often rely on a benign root dataset on the server or use trimmed mean or median for clipping, both methods having limitations.

We introduce FedTruth, a robust defense against model poisoning in FL. FedTruth doesn't assume specific data distributions nor requires a benign root dataset. It estimates a global model update with dynamic aggregation weights, considering contributions from all benign clients. Empirical studies demonstrate FedTruth's efficacy in mitigating the impacts of poisoned updates from both Byzantine and backdoor attacks.

CCS CONCEPTS

- Computing methodologies → Machine learning approaches;
- Security and privacy → Security services.

KEYWORDS

FedTruth, Byzantine Attack, Backdoor Attack, Federated Learning, Robustness

ACM Reference Format:

Sheldon C. Ebron Jr. and Kan Yang. 2023. FedTruth: Byzantine-Robust and Backdoor-Resilient Federated Learning Framework. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 17 pages. https://doi.org/XXXXXXXXXXXXXXXXX

1 INTRODUCTION

In traditional machine learning, training data is usually hosted by a centralized server (cloud server) that runs the learning algorithm or is shared among a set of participating nodes for distributed learning [23]. However, in many applications, data cannot be shared with the cloud or other participating nodes due to privacy or legal restrictions, especially when multiple organizations are involved. Federated Learning (FL) allows multiple parties, such as clients or

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, July 2017, Washington, DC, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00
https://doi.org/XXXXXXXXXXXXXXX

devices, to collaboratively train machine learning models without sharing raw training data [15, 22]. All selected clients train the global model on their local datasets and send the local model updates to an aggregator. The aggregator then aggregates all the local model updates and sends the global model to all the clients selected for the next round of training until convergence is reached. The FL framework is suitable for many AI-driven applications where data are sensitive or legally restricted, such as smart healthcare (e.g., cancer prediction [17]), smart transportation (e.g., autonomous driving [28], road safety prediction [25]), smart finance (e.g., fraud detection [11], risk prediction [2]), and smart life (e.g., environmental quality detection [14], surveillance object detection [39]).

However, the federated nature of FL enables malicious clients to influence a trained model by injecting error model updates. For example, adversaries can control a set of clients to launch *Byzantine attacks* [5, 7, 13] (i.e., sending arbitrary model updates to make the global model converge to a sub-optimal model), or *backdoor attacks* [3, 4, 30, 33, 35] (i.e., manipulating local model updates to cause the final model to misclassify certain inputs with high confidence).

Towards model poisoning attacks in FL, existing defenses focus on designing robust aggregation rules by:

- identifying and removing malicious updates by clustering model updates (e.g., Krum [5], Bulyan [9], AFA [24], Fools-Gold [12] and Auror[29]). However, they only work under specific assumptions about the underlying data distribution of malicious clients and benign clients. For example, Krum and Auror assume that the data of benign clients are independent and identically distributed (iid), whereas FoolsGold and AFA assume the benign data are non-iid. Moreover, these defenses cannot detect stealthy attacks (e.g., constraint-and-scale attacks [3]) or adaptive attacks (e.g., Krum attack [10]).
- clipping and noising model updates using Differential Privacy (DP) techniques. This approach clips individual weights with a certain threshold and adds random noise to the weights so that the impact of poisoned model updates on the global model can be reduced [3, 30]. However, the noise can significantly reduce the accuracy of the global model. To improve the accuracy, in a recent work [26], the authors propose a hybrid approach named FLAME, which first applies clustering to filter model updates and then uses clipping and noising with adaptive clipping threshold and noise level. However, the clipping and noising also eliminates the contributions from benign clients with underrepresented datasets.
- finding the representative model updates using trimmed mean or median [37]. This approach finds the mean or median of each weight in the remaining model updates after removing some values that are bigger/smaller than some thresholds.

However, the trimmed mean or median can be easily bypassed using adaptive attacks (e.g., Trim attack [10]).

• adjusting aggregation weights using root data [1, 6]. This approach assigns different weights based on the distance between each model update and the benign model update from the root dataset trained on the aggregator. However, it requires the aggregator to access the benign root dataset.

Motivation: Based on the above discussed defenses, we have the following observations:

- Without knowing clients' local datasets or a benign root dataset, it is difficult to determine whether an outlier is a malicious update or a significant contribution from an underrepresented dataset, especially when local datasets are non-iid. It is not a good idea to remove or clip a benign outlier model update with significant contribution.
- Only one representative model update is chosen as the global model in many existing Byzantine-resilient aggregation algorithms (e.g., Krum [5], trimmed median [37]), which means the global model is trained with only a single local dataset in each round. In other words, the efforts and contributions of other clients are wasted.
- Due to various qualities of data and trained local model, it is unfair to treat all the clients equally (e.g., FLAME[26]) or evaluate client contributions based on the size of the training dataset (e.g., FedAvg [22]) during the model aggregation.

This paper aims to design a generic solution to defend against model poisoning attacks in FL with the following properties: 1) it does not have specific assumptions on benign or malicious data distribution or accessing to a benign root dataset; 2) it considers potential contributions from all the benign clients; and 3) it reduces the impacts of poisoned model updates from malicious clients. Specifically, we propose a new model aggregation algorithm, namely FedTruth, which enables the aggregator to find the truth of model update among all the received local model updates. The basic idea of FedTruth is inspired by truth discovery mechanisms [20, 21, 27, 38], which are developed to extract the truth among multiple conflicting pieces of data from different sources under the assumption that the source reliability is unknown a priori. In each round of FedTruth, the global model update (i.e., ground-truth model update) will be computed as a weighted average of all the local model updates with dynamic weights.

The contributions of this paper are summarized as follows:

- We develop FedTruth, a generic solution to defend against model poisoning attacks in FL. Compared with existing solutions, FedTruth eliminates the assumptions of benign or malicious data distribution and the need of accessing to a benign root dataset.
- We propose a new approach to estimate the *ground-truth model update* among all the model updates with dynamic aggregation weights in each round. Different from the FedAvg [22] (where the aggregation weight is determined by the size of training dataset) or FLAME [26] (where equal aggregation weight is used regardless of the size of training dataset), the aggregation weights in FedTruth are dynamically chosen based on the distances between the estimated truth and local

- model updates, following the principle that higher weights will be assigned to more reliable clients.
- We extensively evaluate the robustness of our FedTruth against both Byzantine attacks (model-boosting attack, Gaussian noise attack, and local model amplification attack) and backdoor attacks (distributed backdoor attack, edge case attack, projected gradient descent attack) under three attacking strategies (base attack, with model-boosting, and with constrain-and-scaling). The experimental results show that FedTruth can reduce the impacts of poisoned model updates against both Byzantine and backdoor attacks. Moreover, FedTruth works well on both iid and non-iid datasets.
- We further evaluate the efficiency of the FedTruth in terms of the number of iterations to reach FedTruth convergence and the time consumption for two deployments: FedTruth (with entire model updates as inputs) and FedTruth-layer (deploy FedTruth in each layer of the model).

The remainder of this paper is organized as follows: In Section 2, we describe the federated learning framework, some model poisoning attacks, related work on defense solutions, and several adaptive attacks. Section 3 presents the problem statement in terms of the system model, threat model, and design goals. Then, we describe the technical overview of our proposed FedTruth, followed by the concrete formulation. Section 5 shows the experimental results against both Byzantine attacks and Backdoor attacks. Section 6 discuss the adaptive attack, distance function and hybrid FedTruth. Section 7 concludes the paper. More experimental results are shown in the appendices.

2 BACKGROUND AND RELATED WORK

2.1 Federated Learning

A general FL system consists of an aggregator and a set of clients S. Let \mathcal{D}_k be the local dataset held by the client k ($k \in S$). The typical FL goal [22] is to learn a model collaboratively without sharing local datasets by solving

$$\min_{w} F(w) = \sum_{k \in S} p_k \cdot F_k(w), \ s.t. \ \sum_{k \in S} p_k = 1 \ (p_k \ge 0), \qquad (1)$$

where

$$F_k(w) = \frac{1}{n_k} \sum_{i_k=1}^{n_k} f_{j_k}(w; x^{(j_k)}, y^{(j_k)})$$

is the local objective function for a client k with $n_k = |\mathcal{D}_k|$ available samples. p_k is usually set as $p_k = n_k / \sum_{k \in S} n_k$ (e.g., FedAvg [22]). The FL training process usually contains multiple rounds, and a typical FL round consists of the following steps:

- (1) client selection and model update: a subset of clients S_t is selected, each of which retrieves the current global model w_t from the aggregator.
- (2) local training: each client k trains an updated model $w_t^{(k)}$ with the local dataset \mathcal{D}_k and shares the model update $\Delta_{\star}^{(k)} = w_t w_{\star}^{(k)}$ to the aggregator.
- $\Delta_t^{(k)} = w_t w_t^{(k)}$ to the aggregator. (3) *model aggregation*: the aggregator computes the global model updates as $\Delta_t = \sum_{k \in S_t} p_k \Delta_t^{(k)}$ and update the global model as $w_{t+1} = w_t - \eta \Delta_t$, where η is the server learning rate.

FedAvg [22] is the original aggregation rule, which generates a representative global model after receiving the local models from trustworthy (i.e., benign) participants. This algorithm averages all local model weights selected based on the number of samples the participants used. FedAvg has been shown to work well when all the participants are benign clients, but is vulnerable to model poisoning attacks.

2.2 Model Poisoning Attacks

A malicious client or an adversary who compromises a set of clients can influence the global model by changing local datasets (data poisoning attack, e.g., label-flipping attack [32]) or directly manipulating local model updates (model poisoning attack [3–5, 7, 13, 30, 35]). Specifically, the adversary can change both *direction (angle)* and *magnitude (length)* of the model updates to launch model poisoning attacks, including: *Byzantine attacks* and *backdoor attacks*.

Byzantine attacks: The goal of Byzantine attack is to make the global model converged to a sub-optimal model [5, 7, 13]. Some Byzantine attacks are:

- Amplification Attack: A trivial Byzantine attack is to simply amply the local model $w_t^{(i)}$ with some factor to degrade the global model.
- *Model-boosting Attack*: Basic aggregation algorithms like FedAvg can remove the artifact during each iteration, making it challenging to impact the final global model. Similar to the amplification attack, the model-boosting attack [4] refers to explicitly boost the local model updates $(\Delta_t^{(k)} = w_t w_t^{(k)})$ rather than the local models $(w_t^{(k)})$.
- Gaussian Noise Attack: In [5], Byzantine clients randomly draw the local model from a Gaussian Distribution, which is referred to as a Gaussian Byzantine attack. When only local model updates $(\Delta_t^{(k)} = w_t w_t^{(k)})$ are communicated, Gaussian Byzantine attack aims to add noise to the adversaries' local model. The adversarial noise used to degrade the model performance is drawn from a Gaussian distribution.
- Constraint-and-scaling Attack. Simply boosting the model can be easily detected by anomaly detection algorithms. The constraint-and-scaling attack [3] does the model boosting attack while taking the anomaly detection constraints into the crafting of the adversarial model.

Backdoor attacks: The goal of a backdoor attack is to manipulate local model updates to cause the final model to misclassify certain inputs with high confidence [3, 4, 30, 33, 35]. Some backdoor attacks are:

- Distributed Backdoor Attack (DBA) [35]: The DBA attack compromises the global model by cropping the adversarial data into multiple segments based on the number of adversaries colluding during a given FL iteration. Therefore, when the server aggregates the selected models, the backdoor artifact is inserted into the model.
- Edge Case Attack [33]: The edge-case attacks takes advantage
 of a systematic weakness that ML models face when a subset
 of labeled data is drawn from a minority subset of training
 data.

 Projected Gradient Descent Attack (PGD) [30, 33]: In PGD attacks, adversaries periodically project their local models on a small ball, centered around the global model of the previous round.

These backdoor attacks may also be combined with the *model-boosting attack* or *constrain-and-scaling attack* to increase the impact on the final global model.

2.3 Existing Defense Solutions

We first describe some Byzantine-resilient aggregation rules. Then, we show some adaptive attacks towards these Byzantine-resilient aggregation rules. Finally, we introduce a recent work, FLTrust.

Byzantine-resilient Aggregation Rules: Rather than using FedAvg, some aggregation rules are proposed to resist against the Byzantine attacks:

- Krum [5]: The goal of Krum is to choose the most representative global model from the local updates (n). Assuming that the aggregation server knows how many of the local models are Byzantine clients (f); the algorithm will compute the krum-score (si) which is the sum of the square of the normalized Euclidean distance between each local model and their closest (n − f − 2) neighbors. The client which has the smallest krum-score will be selected as the global model. This algorithm is most effective when the number of compromised clients is known and few, and the Euclidean distance is small for benign clients.
- Trimmed Mean [37]: This aggregation rule is constructed based on the principle that if an adversarial attack is launched, then the adversarial model will be an outlier. Therefore, they will first sort the models and then trim the model's outsides. Then they will average the remaining models together after the trimming process, removing the malicious models.
- Coordinate-wise Median [37]: As the name suggests, the median aggregation rule first sorts the local models that have been selected to participate in aggregating the model weights together. Then the algorithm will choose the median model. If there are an even number of models selected for aggregation, then the median aggregator will average the two local model's weights together who represent the median.

Adaptive Attacks: In [10], the authors proposed a generic framework to optimize the attacks for any given aggregation rule. The goal of this attack is to move the aggregated model in the opposite or inverse direction than it would without then if the model had only benign clients.

• Adaptive Trim Attack: This variation of the adaptive attacks is aimed at degrading the performance of both the median and trim-mean aggregation rules. This attack is constructed for one parameter of the local models (j). If the global model moves in the positive direction for the jth parameter, then the adversaries will try to move the global model in the inverse direction (negative direction). Therefore, the adversaries will minimize their models based on the other local models in the Full-Knowledge case. If the global model were to move in the negative direction, then the adversaries would return the maximum of the local weights for the jth parameter.

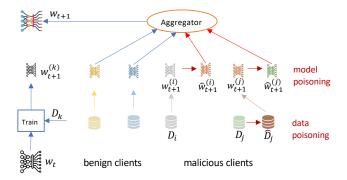


Figure 1: System Model

• Adaptive Krum Attack: Similar to the trim-attack, the goal of this attack is to craft the local adversarial models in a way that will move the global model in the inverse direction. Specifically, this attack makes Krum to select a certain crafted local model w* by setting a set of supporting local models close to the model w*.

FLTrust [6]: FLTrust assumes that there is a benign root dataset available to the aggregation server, who will also train and output a server model in each FL round. Upon receiving all the local model updates from clients, the server calculates a Trust Score using the ReLU-clipped cosine similarity between each local model update and the server model update. The global model update is computed as the average of the normalized local model updates weighted by the trust scores.

RobustFed [31]: In this independent work, the authors proposed to apply the truth discovery approach to estimate the reliability of clients in each round. Then, the estimated reliability is used to compute the next round aggregated model. In this case, an attacker can behave honestly to obtain a high reliability score and then launch the Byzantine attack in the next round. In our work, we use model updates in the current round to estimate the global model update by solving an optimization problem (i.e., dynamically allocate the aggregation weight for each client), which can easily resist the above-mentioned attack.

3 PROBLEM STATEMENT

System Model: As shown in Fig. 1, we consider a typical FL setting, which consists of two entities:

- Clients: FL clients are users who participate in the FL process with their end devices, e.g., mobile devices, computers, and vehicles. When selected in an FL round, the clients will train the model based on their local datasets and send local model updates to the aggregator. Due to personal schedules and device status, the group of clients will change dynamically in each FL round. For example, some clients may not be able to send model updates due to low battery or unstable network, and some clients may join the FL task in intermediate FL rounds.
- Aggregator: The aggregator is an entity that runs the FL algorithm with the clients, including distributing the initial model to all the selected clients, aggregating local model

updates, and sending the global model to the clients selected in a new round.

Threat Model: In this paper, we assume that the aggregator will aggregate all the local model updates honestly in each FL round. However, the clients may be compromised by adversaries and collude to launch Byzantine attacks and backdoor attacks. We assume that the adversaries cannot compromise more than half clients selected in each round. When launching an attack, the adversaries can directly modify their local models (model poisoning attack) and local datasets (data poisoning attack) while having full knowledge about the system (have direct access to any information shared through the system training). However, the adversaries cannot access the benign clients??? devices or data. During a Byzantine attack, the adversarial goal is to degrade the global model or preventing it from convergence. Alternatively, the purpose of the backdoor attack is to manipulate the global model by injecting it with a targeted backdoor.

Design Goals: We aim to design a **generic solution** to defend against model poisoning attacks in FL with the following properties: 1) it does not have specific assumptions on benign or malicious data distribution or accessing to a benign root dataset; 2) it considers the potential contributions from all the benign clients; and 3) it reduces the impacts of poisoned model updates from malicious clients.

4 FEDTRUTH

4.1 Technical Overview

In FedAvg [22], the aggregation weight is determined by the size of the training dataset, i.e., $p_k = n_k / \sum_{k=1}^m n_k$, where $n_k = |\mathcal{D}_k|$. In some other works, such as FLAME [26], equal aggregation weight is used regardless of the size of the training dataset, i.e., $p_k = 1/m$. However, neither FedAvg nor equal aggregation weights can reflect the performance of a client. Figure 2 shows an example of benign and malicious weights.

In a recent work, FLTrust [6], the authors proposed the use of dynamic aggregation weights to calculate the global model. The aggregation weights are estimated based on the trust values, which are calculated based on the similarity between each model update with a ground-truth model update. This ground-truth model update is trained by the aggregator using a benign root dataset. However, this benign root dataset may not be practical in many applications.

Without a benign root dataset, it is challenging to obtain the ground-truth model update among all the local model updates in an FL round. We propose a new model aggregation algorithm, namely FedTruth, which enables the aggregator to uncover the truth among all the received local model updates. The basic idea of FedTruth is inspired by truth discovery mechanisms [20, 21, 27, 38], which are developed to extract the truth among multiple conflicting pieces of data from different sources under the assumption that the source reliability is unknown a priori. Different from FLTrust, in FedTruth, we do not obtain the ground-truth model update and use it to calculate the aggregation weights. Instead, the ground-truth model update is actually the aggregated global model update, which is computed as the weighted average of all the local model updates with dynamic aggregation weights for each round. The weights in FedTruth are dynamically chosen based on the distances between

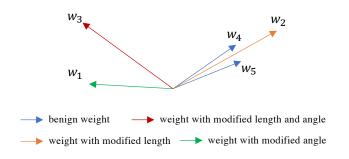


Figure 2: Example of benign and malicious weights

the estimated truth and local model updates, following the principle that higher weights will be assigned to more reliable clients.

4.2 Formulation of FedTruth

Suppose the aggregator receives n_t different model updates $\Delta_t^{(1)}, \dots, \Delta_t^{(n_t)}$ in FL round t. To find the global update Δ_t , FedTruth will solve a convex optimization problem with *linear constraint*:

$$\min_{\Delta_t^*, \mathbf{p_t}} D(\Delta_t^*, \mathbf{p_t}) = \sum_{k=1}^{n_t} g(p_t^{(k)}) \cdot d(\Delta_t^*, \Delta_t^{(k)})$$

$$s.t. \sum_{k=1}^{n_t} p_t^{(k)} = 1$$
(2)

where $d(\cdot)$ is the distance function and $g(\cdot)$ is a non-negative coefficient function. Note that for the ease of evaluating client contributions in each FL round, this formulation follows a generalized truth discovery formulation with linear constraint in a recent work [36], rather than directly applying existing truth discovery formulations with a regulation function $\delta(\mathbf{p_t})=1$ as the constraint [20, 21, 27, 38].

There are many different choices of the distance function $d(\cdot)$ and the non-negative coefficient function $g(\cdot)$. Some examples are described in [36]. Here we only show a simple example, where the distance function and coefficient function are defined as

$$d(\Delta_t^*, \Delta_t^{(k)}) = (\Delta_t^* - \Delta_t^{(k)})^2 / std(\Delta_t^{(1)}, \dots, \Delta_t^{(n_t)})$$
(3)

and

$$g(p_t^{(k)}) = -\log(p_t^{(k)}).$$
 (4)

Once the truth Δ_t^* is fixed, the aggregation weights $\{p_t^{(k)}\}(k=1,\cdots,n_t)$ can be estimated as

$$p_t^{(k)} = d(\Delta_t^*, \Delta_t^{(k)}) / \sum_{k'=1}^{n_t} d(\Delta_t^*, \Delta_t^{(k')}).$$
 (5)

Based on the new aggregation weights $\{p_t^{(1)}, \cdots, p_t^{(n_t)}\}$, the truth of global update can be estimated as

$$\Delta_t^* = \sum_{k=1}^{n_t} \frac{g(p_t^{(k)}) \cdot \Delta_t^{(k)}}{\sum_{k=1}^{n_t} g(p_t^{(k)})} = \frac{\sum_{k=1}^{n_t} \log(p_t^{(k)}) \cdot \Delta_t^{(k)}}{\sum_{k=1}^{n_t} \log(p_t^{(k)})}$$
(6)

The global model update and aggregation weights will be updated iteratively until convergence criteria are met. It is easy to see that the longer the distance between the local model update and the estimated truth, the smaller aggregation weight will be assigned in calculating the truth. This principle can eliminate the impacts of malicious model updates and keep certain contributions from a benign outlier model update.

4.3 Convergence Guarantee of FedTruth

We use the Lagrange multipliers to solve the optimization problem. Under the linear constraint $\sum_{k=1}^{n_t} p_t^{(k)} = 1$, we can define the Lagrangian function of Eq. 2 as

$$L(\{p_t^{(k)}\}_{k=1}^{n_t}, \lambda) = \sum_{k=1}^{n_t} g(p_t^{(k)}) \cdot d(\Delta_t^*, \Delta_t^{(k)}) + \lambda(\sum_{k=1}^{n_t} p_t^{(k)} - 1),$$

where λ is a Lagrange multiplier. To solve the optimization problem, we set the partial derivative with $p_t^{(k)}$ to zero:

$$g'(p_t^{(k)}) \cdot d(\Delta_t^*, \Delta_t^{(k)}) + \lambda = 0$$
 (7)

Then, the Eq. 7 can be reformulated as:

$$p_t^{(k)} = g'^{-1} \left(\frac{-\lambda}{d(\Delta_t^*, \Delta_t^{(k)})} \right) \tag{8}$$

Since the linear constraint is $\sum_{k=1}^{n_t} p_t^{(k)} = 1$, the λ and $p_t^{(k)}$ can be derived from Eq. 8.

We can see that $g(\cdot)$ must be monotonous and differentiable in the aggregation weight domain in order to guarantee the existence of $g'^{-1}(\cdot)$. In other words, we say that as long as the coefficient function $g(\cdot)$ is monotonous and differentiable in the aggregation weight domain, the convexity and convergence of FedTruth can be guaranteed. From our experiments, we find that after 5 to 40 iterations of coordinate descent, the estimated truth is close to the converged value.

4.4 Deploying FedTruth in Each Layer?

One major challenge in truth discovery is data heterogeneity, which may include non-structured data and missing values. However, this challenge is not applicable to FedTruth because all the local model updates are in the same structure. For example, in deep neural networks, the model updates can be represented as multiple-layer tensors.

FedTruth can be run for just one time by the aggregator to compute the truth of model updates by feeding all the local model updates into the FedTruth algorithm. This deployment treats the local model update as an observed value in the truth discovery algorithms.

On the other hand, we can also deploy the FedTruth on each layer to estimate the truth of that single layer, which means that the weights allocated to all the clients may vary on different layers. We denote this deployment as FedTruth-layer. Such layer-wise deployment seems reasonable, it also brings the computation overhead which is linear to the number of layers. We will compare the efficiency between FedTruth and FedTruth-layer in Section 5.5.

5 EXPERIMENTAL RESULTS

We compare the performance of FedTruth with the state-of-the-art aggregation algorithms: FedAvg [22], Krum [5], Trimmed Mean and Median [37], FLTrust [6] and FLAME [26]. The attacks (except

amplification and boosting attacks) are implemented with three attacking strategies:

- (1) base attack: During the base attack, the attacker will not boost the poisoning model or model updates.
- (2) combine with model-boosting attack: The poisoning model or model updates will be boosted with a boosting factor. Similar to [33], we set the boosting factor as $x = C_t/C_{adv,t}$, where C_t denotes the total number of clients selected in t-th round, and $C_{adv,t}$ denotes the number of adversarial clients in this round. In our experiment, we have 10 clients selected in each FL round and the default number of adversarial client is 3 in this section. So, here the default number of boosting factor is x = 10/3 for all the figures in this section.
- (3) combine with constrain-and-scaling attack: We implement this attack by letting each adversarial client train a benign local model $w_t^{j,b}$ first like a benign client. Then, the adversarial client produces a poisoning model $w_t^{j,p}$ according to the attack. A smoothed poisoning model will be calculated as $w_t^j = \alpha w_t^{j,b} + (1-\alpha) w_t^{j,p}$. In our experiment, the default value of α is 0.5. Finally, this value will be scaled or boosted by a boosting factor similar to the *model-boosting attack*.

5.1 Experimental Settings

The experimental settings are as follows:

Datasets: This research implements the experiments with the MNIST [8], FMNIST [34], and CIFAR-10 [18] datasets. FMNIST and MNIST are comprised of 60,000 black-and-white labeled images of size (28x28). MNIST contains handwritten digits, and FMNIST is images of clothing items. CIFAR-10 consists of 60000 color images of size (32x32).

During the *edge-case attack* experiment, we used the *Arkiv Digital Sweden (ARDIS)* [19] dataset as the adversarial backdoor images. The ARDIS dataset consists of handwritten digits originating from Swedish church records. This dataset is a good choice for targeted images when inserting backdoors into MNIST, as ARDIS entirely consists of naturally occurring edge cases.

Clients: When crafting the client's local datasets, we draw their local datasets randomly from a non-iid distribution. We use a non-iid distribution because it is a better representation of clients' data in practice than an iid distribution. Furthermore, each client had a different number of data points, and their distribution was uneven. The client's local data is generated a non-iid distribution, where the bias parameter default is 0.8. In addition, we further evaluate the impact of non-iid degree using the amplification attack, as seen in section 5.4. In each FL round, we randomly select 10 clients, and choose a subset of these selected clients as adversarial clients.

Models: We constructed a Convolutional Neural Network (CNN) classifier for all the experiments considered in this work. In Table 1, we present how the neural network layers were constructed for the MNIST and FMNIST datasets. The ResNet-18 model was used when running experiments using the CIFAR10 dataset.

5.2 Byzantine Attacks

This section presents experimental results for two attacks: the *model-boosting attack* and the *Gaussian Noise attack*, conducted

on various FL frameworks. In these attacks, only the model updates (the difference between the newly trained local model and the global model in the previous round) are communicated. Appendix A and Appendix B contain findings from Byzantine experiments performed on the FMNIST and CIFAR10 datasets. Moreover, the results of the amplification attack on the entire local models that are sent directly to the aggregator can be found in Appendix C.

Model-boosting Attack: The model-boosting attack is a type of attack that aims to degrade the model performance by boosting the adversary's local model updates with a boosting factor set to 10. To evaluate the robustness of different aggregation algorithms under different attacking scenarios, we conducted experiments with varying percentages of compromised clients in each round.

Figure 3(a) shows the results when only 10% of clients are compromised in each round, where the subset of compromised clients is randomly selected. We observe that all aggregation algorithms perform well except FedAvg. However, when the percentage of compromised clients increases to 20%, 30%, and 40%, as shown in Figures 3(b), 3(c), and 3(d), respectively, the FedTruth methods remain unaffected by the attack, regardless of the number of adversaries present.

In contrast, as the number of adversaries increases, Krum experiences a slight impact on its performance. Meanwhile, FedAvg, Trimmed-mean, and Median algorithms experience significant effects, preventing all three from reaching convergence when four adversaries are present.

Gaussian Noise Attack: The Gaussian noise attack aims to degrade the performance of the global model by adding Gaussian noise to the model. The noise is drawn from a multivariate Gaussian distribution $N(0, \sigma^2 I)$ [5, 6] and is added directly to the adversaries' model before sending it to the aggregator. In Figure 4(a), we show the model accuracy and convergence speed for all aggregation algorithms against the Gaussian noise attack, where three adversaries launch this attack per round. Our findings are as follows: 1) FedTruth can defend against the Gaussian noise attack without significantly slowing down the convergence speed; 2) FedTruth can achieve the same model accuracy as FLTrust, which requires a benign dataset, showing that our FedTruth can actually find the ground truth of the model updates; 3) The convergence speed is impacted in FLAME and Krum, and the model accuracy of FLAME and Krum is not as high as that of FedTruth and FLTrust; and 4) FedAvg cannot converge within 100 rounds under the Gaussian noise attack.

In Figure 4(b), we combine the model-boosting attack with the Gaussian noise attack, which also does not degrade the performance

Table 1: CNN architecture trained on local devices

Layer	Size
Input	28x28x1
Convolution + ReLU	20x5x1
Max Pooling	2x2
Convolution + ReLU	4x4x50
Max Pooling	2x2
Fully Connected + ReLU	500
Fully Connected + Softmax	10

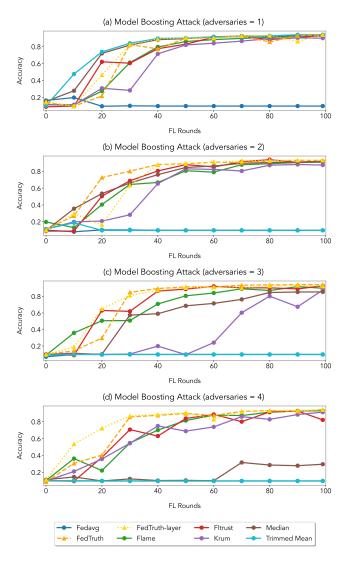


Figure 3: Model Boosting Attack (MNIST, ×10 boosting factor)

of FedTruth. However, FedAvg and Trimmed-mean do not perform well against this attack, and Krum has a slower convergence speed.

During the Gaussian noise attack, we chose not to combine it with the constrain-and-scale attack. This decision was based on the goal and implementation of the constrain-and-scale attack, which involves training two loss functions on the same model during each epoch with benign and adversarial datasets. Therefore, including the constrain-and-scale attack would not be helpful for the Gaussian noise attack, as the adversary would be directly modifying their model.

5.3 Backdoor Attacks

Distributed Backdoor Attack (*DBA***)**: The *DBA* attack distributes a portion of an adversarial background image among all malicious clients colluding each round. The shards are then added to the background of the individual adversaries' data. We combined this

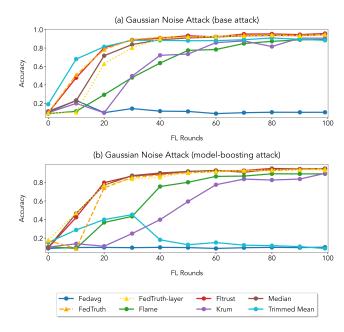


Figure 4: Gaussian Noise Attack (MNIST, 3 adversaries)

attack with the *model-boosting* and *constrain-and-scale attack*. The results shown in Figure 5 prove that regardless of the combination of attacks used on FedTruth, we can effectively remove the adversarial artifacts from the global model.

Figure 5a shows our *main task accuracy* (accuracy on a benign dataset) observed during each of the FL rounds during this attack. The results in figure 5a show that the Krum and FLAME algorithms are prevented from converging during *base* and *model boosting attacks*. We suspect that these results could be caused by the large number of adversaries colluding during this experiment or the imbalanced sampled data. It is also interesting to note that this *main task's accuracy* anomaly was not present when the DBA attack was combined with the *constrain-and-scale attack*. This does, however, validate our reasoning for why the *base* and *model-boosting attacks* results were unable to reach convergence, as the *constrain-and-scale attack* scales the amount that an adversarial model can diverge during each epoch.

Figure 5b presents our results on the *targeted task accuracy* (i.e., the backdoor accuracy) during the DBA experiments. During this attack, the adversarial goal is to add the targeted artifact to the global model without being detected and not affecting the model's performance on the main task. Therefore, we will not be considering the Krum and FLAME algorithms during the *base* and *model-boosting* attacks, as they were unable to converge when training the main task, as seen in figure 5a. In figure 5b, after the 50th FL iteration, none of the remaining algorithms exceed the 40% accuracy threshold, apart from FedTruth. We suspect that this is due to, an inverse relationship with the effects of flattening the layers when subject to stealthy attack as the number of layers in the model increases. This is further supported by FedTruth-layer (which does not flatten any of the layers) being able to consistently keep the targeted task

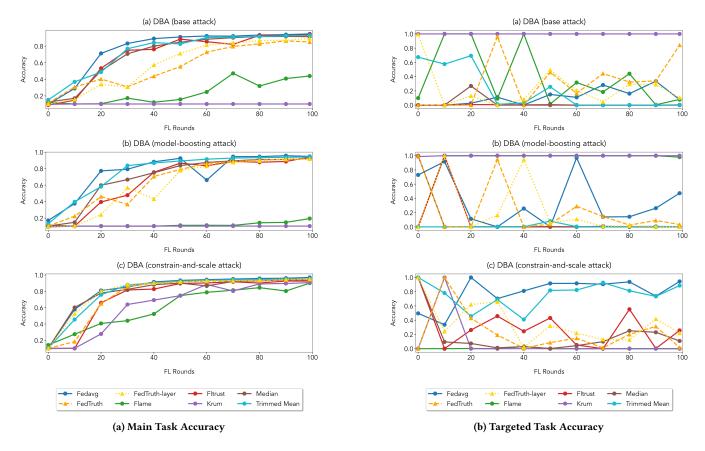


Figure 5: DBA Attack (MNIST, 3 adversaries)

accuracy below the 40% threshold after the 50th FL iterations. Furthermore, during the DBA attack with *model-boosting* combination, we observe similar results for the algorithms that we are considering, with the exception that FedAvg exceeds our threshold while FedTruth can remove the targeted task. This is a result of a tradeoff between the stealthiness of the base DBA attack being diminished when increasing the amplitude of the adversarial updates vector in hopes of replacing the global model with the adversarial model. Finally, we consider all of the aggregation algorithms (i.e., we are no longer excluding Krum and FLAME) during the constrain-and-scale attack results as seen in figure 5a results it is not prevented from converging. However, our results therefore show that this attack combination is the most effective during the DBA attack based on our threshold set; where FedAvg and Trimmed Mean are extremely defenseless to this attack combination while FLTrust is less resilient, reaching an accuracy of 50% which can be observed in the 80th FL iteration.

Edge-case Attack: Based on the attacks presented in [33], we implemented a similar attack for the MNIST dataset that used the *Arkiv Digital Sweden (ARDIS)* [19] dataset as the adversarial *edge-cases* that are being injected into the models. During each attack round, the adversaries added *edge-cases* data points, equivalent to 20% of the targeted labels in the client's dataset, from the *ARDIS* dataset.

Figure 6a shows our results for the *main task* during the *edge-case* attack. We observed that all the algorithms could reach convergence during the base attack. However, FLTrust did not reach convergence when the *edge-case* attack was combined with the *model-boosting* or *constrain-and-scale* attacks. We suspect this is because the adversarial model is being trained on edge-cases that are not present in the algorithm's root dataset.

Figure 6b presents our results for the targeted task accuracy during this attack. We will use the threshold presented in Section 5.3 where we denote a successful attack combination if it can reach a targeted task accuracy higher than 40% after the 50th FL iteration. FLTrust and FedTruth-layer appeared to have difficulty removing the adversaries during the base attack, as seen in Figure 6b(a). During the FedTruth-layer algorithm, we observed the algorithm reach a convergence rate of 60% at the 80th iteration. We observe that the FLTrust algorithm reaches a targeted accuracy as high as 90% during this attack. However, we suspect both algorithms are experiencing these results due to the similarity of the adversarial edge-cases to the benign dataset, coupled with the relatively minimal number of edge-cases. During the model boosting attack Figure 6b(b), we observed the Krum and Trimmed Mean algorithms exceed our threshold. We suspect these results are due to both algorithms working based on the principle of selecting a subset of the local models. Therefore, these results show that during this attack, if you are using an algorithm that selects a small proportion

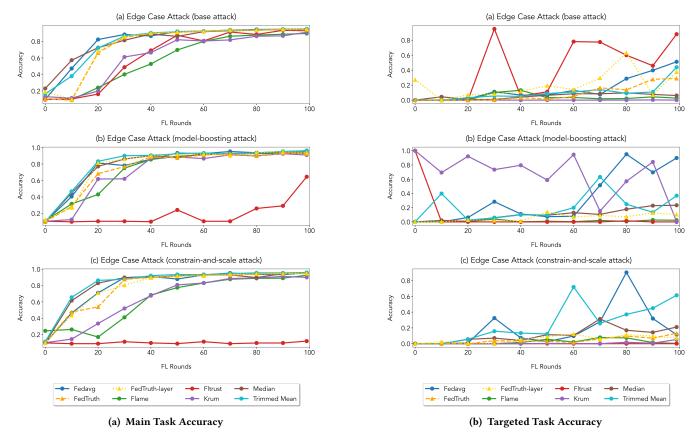


Figure 6: Edge Case Attack (MNIST, 3 adversaries)

of the local model updates, then if the adversarial model is able to inject the backdoor into the global model, it will be extremely unlikely for the model to be removed after subsequent iterations. However, algorithms like *FedTruth*, which do not use a subset of the local models, show almost no effect during this attack. The constrain-and-scale attack Figure 6b(c) shows similar results to the *model-boosting* attack, except for the fact that Krum is not affected during this version of the backdoor attack, as the accuracy never exceeds 20%. We suspect this is due to the model not diverging as drastically from the benign models, making the selection of the adversarial model less nefarious.

Projected Gradient Descent Attack (*PGD***)**: We implemented the *PGD attack* using the *Torch Attacks library* [16] which was used to create a generative model that will take as input an image and return a perturbed image. We set the max perturbation $\epsilon = 8/255$, which is how the adversarial example knows how far an image can be noised during the generation of the adversarial model. Then we set alpha (α) the step size to be $\alpha = 2/255$, and the number of steps = 4.

Figure 7a presents the *main task accuracy* for the *PGD attacks*, which we observed to exhibit a similar behavior in the results in Section 5.3. The *base attack* Figure 7a(a) and *model-boosting attack* Figure 7a(b), show a decreased in the main task accuracy when using the Krum and FLAME algorithms. Excluding the random

outlier on the FLTrust algorithm during the *constrain-and-scale* Figure 7a(c) attack, there appears to be no degradation in main task accuracy.

When evaluating the results presented in Figure 7b on the targeted task accuracy, we did not consider any algorithms that could not converge on the main task. Our results for the PGD *base attack*, Figure 7b(a) shows that FedTruth was the only algorithm to exceed the 40% threshold after the 50th iteration. During the PGD with the *model-boosting attack*, shown in Figure 7b(b), all the algorithms we are considering can remove the targeted task from the global model. However, when the PGD attack is combined with the *constrain-and-scale attack*, presented in Figure 7b(c), we observe that both the Trimmed Mean and FedAvg algorithms are not resilient to the attack combination.

5.4 The Impact of non-iid on FedTruth

Figure 8 shows how various non-iid bias parameters affect the experiments when adversaries apply the *model-boosting* attack. During these experiments, three adversaries were selected in each round. The non-iid data were sampled from various degrees based on the primary label bias. For example, if the primary label bias were 1.0, all the client's local data would consist of whichever label

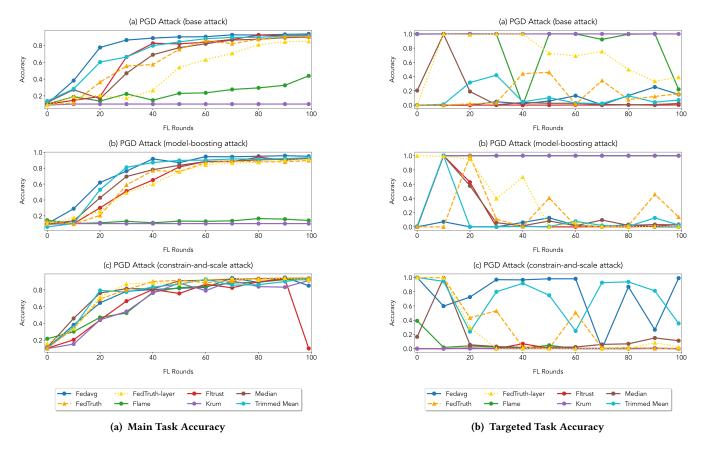


Figure 7: PGD Attack (MNIST, 3 adversaries)

was determined to be their primary label. For this experiment, we set the bias parameters to be 0.1, 0.3, 0.5, 0.8, and 0.95.

The results of the experiments suggest that FedTruth can mitigate the impacts of the boosted model regardless of the non-iid degree of the datasets. Therefore, though the FedTruth system does experience some performance degradation, it is resilient to non-iid biases in the training data. In addition, Appendix C, Figure 12 also demonstrates that FedTruth works well on different non-iid data under the amplification attack.

According to the data presented in this paper, the *FLTrust* algorithm was found to be less effective compared to *FedTruth* and *FedTruth-Layer* algorithms in most cases, including the results shown in Figure 8. However, it is worth noting that we originally assumed that *FLTrust* would manage to achieve accuracy at or above *FedTruth* or *FedTruth-Layer* in most instances after 100 iterations. It was expected that *FLTrust* algorithms would be able to remove adversarial models more efficiently and with higher accuracy due to the separate dataset that is assumed to be non-adversarial.

We hypothesize that this is due to the NonIID data sampling used in the experiments. Essentially, *FLTrust* may not be producing an optimal server model with each FL iteration because the data used to train the model isn't necessarily drawn from a similar data distribution as the data used to train clients' models. This becomes more of an issue as more adversaries are added during each

iteration and the degree of Non-iid increases, as shown in Figure 8. The weights used to generate client updates may not contain a representative subset of the data points, leading to the models' suboptimal convergence. Conversely, *FedTruth* and *FedTruth-Layer* are adaptive and not limited by representative datasets; instead, they use client observations as the truth for generating weights.

5.5 Efficiency Evaluation of FedTruth and FedTruth-layer

From our experimental results, we observe that both FedTruth and FedTruth-layer perform similarly in terms of model accuracy and robustness. To evaluate their efficiency, we present the average time consumption for each aggregation algorithm in Table 2. We measure the average aggregation time and average training time on each client, based on training the CNN model on MNIST for 100 rounds with three adversaries in each round.

We find that the FedAvg algorithm is the most efficient, while Trimmed Mean, Median, and FLTrust take less than 0.05 seconds. FLAME takes about 0.19 seconds to filter and clip, and Krum is the slowest among all the algorithms. FedTruth and FedTruth-layer are slower than some other aggregation algorithms but faster than Krum. Considering that the aggregation algorithm is run by the aggregator, it is still practical and efficient to run FedTruth and FedTruth-layer within 3 seconds.

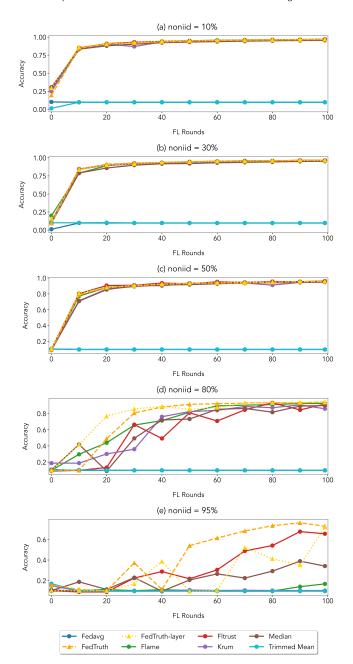


Figure 8: Model Boosting Attack non-iid Degree Impact (MNIST, 3 adversaries, ×10 boosting factor)

Interestingly, we observe that FedTruth and FedTruth-layer have similar total time consumption when using CNN with a small number of layers. We count the number of iterations required for each algorithm to reach convergence and present the results in Table 3. For CNN (8 layers) on MNIST dataset, we find that FedTruth requires an average of 39 iterations to estimate the ground-truth model update, while FedTruth-layer requires an average of 91.4 iterations (three times more than FedTruth) to reach convergence,

despite having eight layers in the CNN model. Moreover, FedTruth on each layer has a smaller input size and requires less computation time on the distance compared to FedTruth with the entire model update as input. In conclusion, we find that both FedTruth and FedTruth-layer have similar performance on the CNN (8 layers) model in terms of accuracy and robustness. However, FedTruth is much more efficient than FedTruth-layer when there are a large number of layers in the model (e.g., ResNet-18).

Table 2: Aggregation and training time for different FL algorithms (Model Boosting Attack, 3 Adversaries)

A1 . 201	Avg. Ag	gregation Time (s)	Avg. Training Time (s)	
Algorithm	MNIST	CIFAR10	MNIST	CIFAR10
FedAvg	0.0062	0.0258	0.0767	0.8528
FedTruth	2.9511	3.2127	0.0773	0.8617
FedTruth-layer	2.8591	5.5734	0.0778	0.8528
Flame	0.1936	2.7209	0.0800	0.8508
FLTrust	0.0326	0.1566	0.0808	0.8555
Krum	4.0604	0.7023	0.0782	0.8579
Median	0.0413	0.0030	0.0804	0.8578
Trimmed Mean	0.0266	0.0077	0.0792	0.8580

Table 3: Number of Iterations for FedTruth and FedTruthlayer (Model Boosting Attack, 3 Adversaries)

Algorithm	Avg. # of Iterations to FedTruth Convergence		
Aigorithin	MNIST	CIFAR10	
FedTruth	39	39	
FedTruth-layer Total	91.40	860.70	
FedTruth-layer L1	8.45	10.20	
FedTruth-layer L2	4.81	6.85	
FedTruth-layer L3	11.01	6.49	
FedTruth-layer L4	5.29	10.24	
FedTruth-layer L5	39.00	6.10	
FedTruth-layer L6	6.64	6.21	
FedTruth-layer L7	10.55	14.79	
FedTruth-layer L8	5.65	5.71	

6 DISCUSSION

Adaptive Attacks on FedTruth: FedTruth estimates the ground truth of global model updates in each round by minimizing the distance between each local model update and the estimated ground-truth model update. However, an adaptive attack on FedTruth is possible, where an adversary can compromise a set of clients and produce the same or similar model updates, such as a Sybil attack. The effectiveness of FedTruth relies on the assumption that the majority of clients are reliable and diverse. If an adversary compromises more than 50% of the clients, they can dominate the results of FedTruth.

From our experimental results, we find that when an adversary compromises 40% (4 out of 10) clients in each round, FedTruth can still prevent Byzantine and Backdoor attacks. Moreover, even when the training data has a high non-iid degree of 80%, FedTruth outperforms existing methods with up to 30% adversaries, as shown

in Figure 8. However, when the non-iid degree is further increased to 95%, the accuracy drops because some uncompromised clients may perform poorly with highly non-iid training data, leading to a bad estimation of the ground truth by FedTruth. This scenario is similar to when an adversary compromises more than half of the clients.

Distance Function in FedTruth: In our experiments, we calculate the Euclidean Distance between each local model update and the estimated truth of the global model update. While cosine similarity is another important metric that has been used to evaluate the distance between the local model and the global model, it may not work well when the adversary amplifies the entire local model update with the same factor, as the cosine similarity remains the same as before. However, if the adversary combines some base attacks with model-boosting or constraint-and-scale attacks, we may integrate both Euclidean distance and cosine similarity into the distance function in FedTruth. Therefore, the choice of the distance metric depends on the type and complexity of the attack. We should carefully consider the attack scenarios and choose the appropriate metric for distance calculation in FedTruth.

Hybrid FedTruth: Although FedTruth aims to consider the contributions of all clients, it may be necessary to exclude inputs from clients who have a bad reputation or low reliability in previous FL tasks. To evaluate the reputation or reliability of clients, we need to assess the contributions of each client in an FL task. This motivated us to formulate FedTruth with linear constraints. In practice, we can trim the inputs of clients who have been identified as untrustworthy or unreliable based on their past contributions to FL tasks. By doing so, we can improve the accuracy and robustness of the global model by preventing the contributions of bad actors from affecting the overall performance. However, we should also be aware that trimming inputs from clients may have unintended consequences, such as reducing the diversity of the training data and reducing the number of participating clients, potentially leading to overfitting and decreased overall performance. Therefore, we need to carefully evaluate the trade-offs between trimming inputs and maintaining the diversity of the training data, and choose appropriate strategies to ensure the effectiveness of the FL system.

7 CONCLUSION

In this paper, we developed FedTruth, a generic solution to defend against model poisoning attacks in FL. Compared with existing solutions, FedTruth eliminates the assumptions of benign or malicious data distribution and the need of accessing to a benign root dataset. Specifically, a new approach was proposed to estimate the ground-truth model update (i.e., the global model update) among all the model updates with dynamic aggregation weights in each round, following the principle that higher weights will be assigned to more reliable clients. The experimental results show that FedTruth can reduce poisoned model updates' impacts against Byzantine and backdoor attacks. Moreover, FedTruth works well on both iid and non-iid datasets.

REFERENCES

 Sebastien Andreina, Giorgia Azzurra Marson, Helen Möllering, and Ghassan Karame. 2021. Baffle: Backdoor detection via feedback-based federated learning.

- In 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS). IEEE, 852–863.
- [2] Dmitrii Babaev, Maxim Savchenko, Alexander Tuzhilin, and Dmitrii Umerenkov. 2019. Et-rnn: Applying deep learning to credit loan applications. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2183–2190.
- [3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International Conference* on Artificial Intelligence and Statistics. PMLR, 2938–2948.
- [4] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. 2019. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*. PMLR, 634–643.
- [5] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In Proceedings of the 31st International Conference on Neural Information Processing Systems. 118–128.
- [6] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Gong. 2021. FLTrust: Byzantinerobust Federated Learning via Trust Bootstrapping. In Proceedings of NDSS.
- [7] Yudong Chen, Lili Su, and Jiaming Xu. 2017. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. Proceedings of the ACM on Measurement and Analysis of Computing Systems 1, 2 (2017), 1–25.
- [8] Li Deng. 2012. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. IEEE Signal Processing Magazine 29, 6 (2012), 141–142. https://doi.org/10.1109/MSP.2012.2211477
- [9] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Louis Alexandre Rouault. 2018. The Hidden Vulnerability of Distributed Learning in Byzantium. In International Conference on Machine Learning.
- [10] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local model poisoning attacks to Byzantine-Robust federated learning. In 29th USENIX Security Symposium. 1605–1622.
- [11] Úgo Fiore, Alfredo De Santis, Francesca Perla, Paolo Zanetti, and Francesco Palmieri. 2019. Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. *Information Sciences* 479 (2019), 448– 455.
- [12] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. 2020. The limitations of federated learning in sybil settings. In 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020). 301–316.
- [13] Rachid Guerraoui, Sébastien Rouault, et al. 2018. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*. PMLR. 3521–3530.
- [14] Miyuki Hino, Elinor Benami, and Nina Brooks. 2018. Machine learning for environmental monitoring. Nature Sustainability 1, 10 (2018), 583–588.
- [15] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977 (2019).
- [16] Hoki Kim. 2020. Torchattacks: A pytorch repository for adversarial attacks. arXiv preprint arXiv:2010.01950 (2020).
- [17] Konstantina Kourou, Themis P Exarchos, Konstantinos P Exarchos, Michalis V Karamouzis, and Dimitrios I Fotiadis. 2015. Machine learning applications in cancer prognosis and prediction. Computational and structural biotechnology journal 13 (2015), 8–17.
- [18] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
- [19] Huseyin Kusetogullari, Amir Yavariabdi, Abbas Cheddad, Håkan Grahn, and Johan Hall. 2020. ARDIS: a Swedish historical handwritten digit dataset. Neural Computing and Applications 32, 21 (2020), 16505–16518.
- [20] Yaliang Li, Qi Li, Jing Gao, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2015. On the discovery of evolving truth. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 675–684.
- [21] Yaliang Li, Qi Li, Jing Gao, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2016. Conflicts to harmony: A framework for resolving conflicts in heterogeneous data by truth discovery. IEEE Transactions on Knowledge and Data Engineering 28, 8 (2016), 1986–1999.
- [22] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In Artificial Intelligence and Statistics. PMLR, 1273–1282.
- [23] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. 2016. Mllib: Machine learning in apache spark. The Journal of Machine Learning Research 17, 1 (2016), 1235–1241.
- [24] Luis Muñoz-González, Kenneth T Co, and Emil C Lupu. 2019. Byzantine-robust federated machine learning through adaptive model averaging. arXiv preprint arXiv:1909.05125 (2019).
- [25] Alameen Najjar, Shun'ichi Kaneko, and Yoshikazu Miyanaga. 2017. Combining satellite imagery and open data to map road safety. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 31.

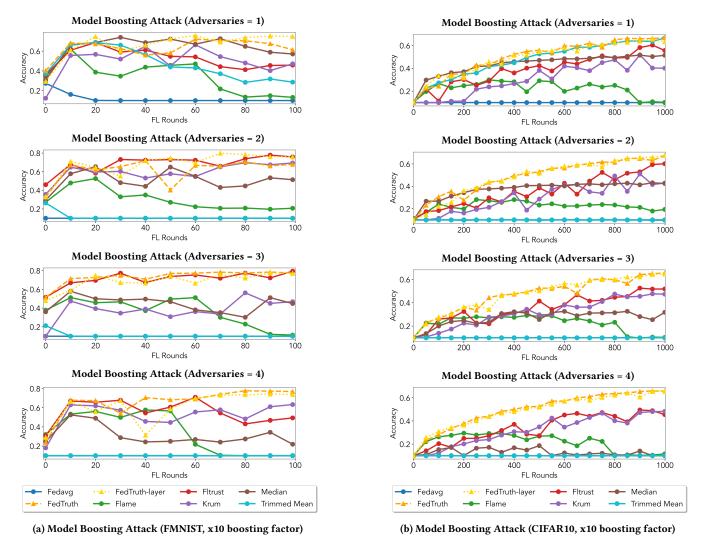


Figure 9: Model Boosting Attack (FMNIST & CIFAR10)

- [26] Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, et al. 2022. FLAME: Taming Backdoors in Federated Learning. In Proceedings of 31st USENIX Security Symposium. to appear.
- [27] Robin Wentao Ouyang, Lance M Kaplan, Alice Toniolo, Mani Srivastava, and Timothy J Norman. 2016. Aggregating crowdsourced quantitative claims: Additive and multiplicative models. IEEE Transactions on Knowledge and Data Engineering 28, 7 (2016), 1621–1634.
- [28] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. 2017. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging* 2017, 19 (2017), 70–76.
- [29] Shiqi Shen, Shruti Tople, and Prateek Saxena. 2016. Auror: Defending against poisoning attacks in collaborative deep learning systems. In Proceedings of the 32nd Annual Conference on Computer Security Applications. 508–519.
- [30] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. 2019. Can you really backdoor federated learning? arXiv preprint arXiv:1911.07963 (2019).
- [31] Farnaz Tahmasebian, Jian Lou, and Li Xiong. 2022. Robustfed: a truth inference approach for robust federated learning. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management. 1868–1877.
- [32] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data poisoning attacks against federated learning systems. In European Symposium on Research in Computer Security. Springer, 480–501.

- [33] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. 2020. Attack of the tails: Yes, you really can backdoor federated learning. Advances in Neural Information Processing Systems 33 (2020), 16070–16084.
- [34] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017).
- [35] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. 2019. Dba: Distributed back-door attacks against federated learning. In International Conference on Learning Representations.
- [36] Lei Yan, Kan Yang, and Shouyi Yang. 2022. Reputation-Based Truth Discovery With Long-Term Quality of Source in Internet of Things. IEEE Internet of Things Journal 9, 7 (2022), 5410–5421. https://doi.org/10.1109/JIOT.2021.3110511
- [37] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*. PMLR, 5650–5659.
- [38] Xiaoxin Yin, Jiawei Han, and S Yu Philip. 2008. Truth discovery with multiple conflicting information providers on the web. IEEE Transactions on Knowledge and Data Engineering 20, 6 (2008), 796–808.
- [39] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. 2019. Object detection with deep learning: A review. IEEE transactions on neural networks and learning systems 30, 11 (2019), 3212–3232.

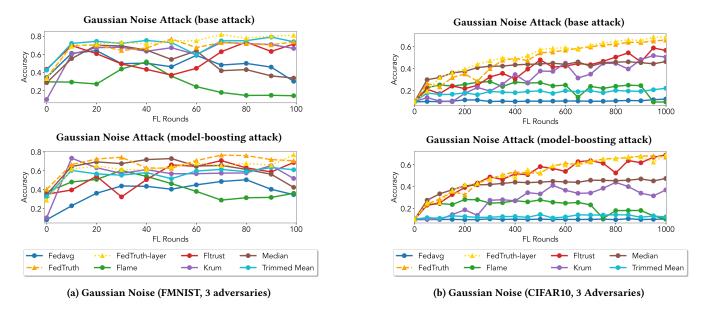


Figure 10: Gaussian Attack (FMNIST & CIFAR10, 3 Adversaries)

A MORE RESULTS ON MODEL-BOOSTING ATTACK ON FMNIST AND CIFAR-10

This section presents the additional figures for the model-boosting attack on the previous aggregation algorithms when using the FMNIST dataset and CIFAR-10 dataset.

Model-boosting Attack on FMNIST: Figure 9a shows the results of the model-boosting attack on the FMNIST dataset. During this attack, we present the base attack with various numbers of adversaries, ranging from 1 to 4. It has been observed that selecting only one adversarial client per round hinders the convergence rate of the FedAvg algorithm. This is because? FedAvg? distributes the adversarial clients evenly based on the number of data points the model was trained on. Within a few rounds, the convergence rate drops to accuracy, and after 20 iterations, the algorithm reaches an average accuracy of around. During our experiments, we noticed a group of algorithms that gradually lost their effectiveness over time. These algorithms exhibited some fluctuations, but particularly those that ended the last 10 iterations below a target threshold of 50% caught our attention. Specifically,? FLAME,? Trimmed Mean,? FLTrust, and? Krum were the affected algorithms. We suspect that the combination of NonIID (80%), attack selected, and one adversary (10%) among the clients could gradually decrease the performance of the aggregation algorithm over time.

Suppose we only consider the algorithms that are able to converge at a rate at or higher after 100th iterations during the one adversary per round attack. In that case, we are left with? FedTruth,? FedTruth-layer,? FLTrust,? Median, and? Krum? algorithms. We observed that? Krum? and? Median? consistently have a global model accuracy below the? FedTruth,? FedTruth-layer, and? FLTrust? algorithms after 100 iterations when considering the attacks that used 2, 3, or 4 adversaries per round. However, we did see? Krum? outperform? FLTrust? when there were 4 adversaries, which could be

based on? FLTrust? selecting an inadequate representative dataset during that experiment. However, since? FedTruth? and? FedTruth-layer? use the adaptive approach, we have presented in this work that does not use a representative dataset, the results for these aggregation algorithms outperform all of the other aggregation algorithms regardless of the number of adversaries present during the experiment.

In addition, based on our observations, we noticed a similar pattern in the behavior of the? FedTruth? and? FedTruth-layer? algorithms when there were 2 to 4 adversaries per iteration. However, our results indicated that the final accuracy of the global model was better for? FedTruth? when there were more adversaries present (specifically, 3 or 4) compared to? FedTruth-layer. This aligns with our hypothesis on the two versions of the algorithm, where the? FedTruth-layer? was designed to be more sensitive to minor (i.e., stealthy) model perturbations and therefore performed better when there were fewer adversaries (1 or 2).

Model-boosting Attack on CIFAR-10: Figure 9b presents our results for the *Model Boosting attack* when there are 3 adversaries boosting their update by a multiple of 10 using the CIFAR10 dataset. We significantly increased the number of iterations for the CIFAR10 dataset experiment, conducting 1000 iterations compared to the 100 iterations used for the MNIST and FMNIST experiments. Furthermore, we leveraged the power of the ResNet Model to train the model with confidence.

After running the experiments, we noticed a slight variation in accuracy among one group of the aggregation algorithms, which include *FedTruth*, *FedTruth-Layer*, *FLTrust*, *Median*, and *Krum*. The other group consisted of *Trimmed Mean* and *FedAvg*, which exhibited poor performance with a consistent accuracy range of 0% to 1% throughout the training process. This was not, however, unexpected and consistent with the findings for similar attacks like Figure 9b;

yet for these reasons, we will not be discussing the *Trimmed Mean* and *FedAvg* algorithms in this section.

During our experiment, we found that both FedTruth and FedTruth-Layer consistently yielded the highest accuracy throughout the entire experiment. Although there was a slight increase in accuracy for our proposed algorithms when compared to FLTrust, Median, and Krum, this difference was only noticeable after the 225th iteration. Upon closer examination, after the 600th iteration, we observed a significant variance in accuracy between the Median and other algorithms. This is due to the NonIID sampling rate, causing overfitting on primary features during initial rounds; the Krum algorithm mitigates this but still affects its performance. Furthermore, we suspect that the high NonIID may also affect the FLTrust algorithm, as the weight assignment may be biased towards primary features in the server model.

However, since the *FedTruth* and *FedTruth-Layer* algorithms use weighted averages, new features can still be added to the model. Since the weights are computed based on an adaptive truth, the results of the primary features' selected yield minimal effects model's global model accuracy.

B MORE RESULTS ON GAUSSIAN NOISE ATTACK ON FMNIST AND CIFAR-10

Gaussian Noise Attack on FMNIST:

Figure 10a shows our results for the *Gaussian-noise attack on the FMNIST dataset*. We have grouped the aggregation models into three categories based on their accuracy after the 100th iteration. The top performers among them were *FedTruth*, *FedTruth-Layer*, *FLTrust*, *Trimmed Mean*, and *Krum*, and the final accuracy was above 65%. The *Median* and *FedAvg* algorithms comprise the second grouping, with a final accuracy of around 35% to 40%. The final grouping of the lowest-performing algorithms under this attack configuration is *FLAME*, with a test accuracy of 15% after the 100th iteration.

Taking a closer look at the lowest-performing group (*FLAME*), we assume the subpar performance is most likely a result of the clipping effects as this is the stage in the algorithm responsible for removing models whose weights seem to be scaled or boosted. However, with the number of adversaries (30% each round) and large NonIID sampling rate (80%), it seems to cause the clipping stage of the algorithm to lose its effectiveness over time.

Interestingly, we observed that the *Median* and *FedAvg* algorithms in the second group showed some peculiar behaviors, having initially performed similarly to the group with the highest-performing algorithms after 100 iterations; however, despite this finding, with a final accuracy between 35% to 40%, they demonstrated a high global model accuracy of 60 to 70% before gradually declining in performance from the 70th iteration. This decline persisted until the end of the experiment, indicating a significant decrease in performance for these two algorithms. It has come to our attention that the *FedAvg* algorithm is experiencing a negative impact from the noised models of the three adversaries. This effect has become apparent after several rounds of basic averaging without any assistance from boosting or scaling techniques. In contrast, the *Median* algorithm selects a model from a group of sorted models. As more adversarial models are added to the group, the probability

of selecting one as the final model increases. Furthermore, due to the random distribution of the data, the likelihood of selecting an adversarial model also increases.

The group of high-performing aggregation algorithms after 100 iterations, which included FedTruth, FedTruth-Layer, FLTrust, Trimmed Mean, and Krum, displayed some unexpected behavior during our experiment. It's interesting to note that the accuracy of the Trimmed Mean and Median algorithms differed after 100 iterations, despite having similar structures. Upon closer examination, our team discovered that the Trimmed Mean algorithm outperformed the Median algorithm due to its unique ability to eliminate outliers and average the remaining models in the middle, which increased its robustness. A way to better understand this is by thinking of Trimmed Mean as a combination of Median and FedAvg. This combination makes it more resilient against attacks from adversaries, as the averaging process gives them less control over the system.

It is possible that *Krum* can easily detect adversarial models because there is a noticeable difference in the Euclidean distance between benign and adversarial models. After the 100th iteration, *FedTruth*, *FedTruth-Layer*, and *FLTrust* produce the best results by using **truth discovery** to calculate weights for each adversarial model. However, we noticed that *FLTrust* experiences a decrease in accuracy around the 50th iteration, which may be due to the NonIID degree of the data sampling, causing the server model not accurately to represent the client's model. As a result, the weight assignment may overweight the adversarial models too high. Thankfully, the **adaptive truth discovery** algorithm used by *FedTruth* and *FedTruth-Layer* solves this problem and avoids any dips in accuracy.

Figure 10a also presents our results of the Gaussian Noise Attack with Model Boosting when there are 3 adversaries per round and performing model boosting. Our results in Figure 10a can be divided into two groups based on their final accuracy: those algorithms that achieve an accuracy above 50% and those that fall below this threshold. The former group includes FedTruth, FedTruth-Layer, FLTruth, and Trimmed Mean, which were the top performers in the FMNIST version of this attack. However, Krum did not meet the threshold this time. The latter group, excluding Krum, which consists of Median, FedAvg, and FLAME, experienced poor performance similar to what was observed in Figure 10a's base attack results. Interestingly, the addition of Model-boosting allowed the FLAME algorithm to achieve an overall accuracy of 30%, which is higher than before. This could indicate that the algorithm was better at distinguishing between adversarial and benign models after boosting.

Gaussian Noise Attack on CIFAR-10: Figure 10b offers our results for the Gaussian Noise attack combined with the model-boosting attack on the CIFAR10 dataset when three adversaries were present during each round. During this attack, it is apparent that the *FedTruth-layer* is among the few algorithms that can reach convergence. Additionally, this attack configuration significantly hinders FedAvg, FLAME, and Trimmed Mean. At the same time, Krum and Trimmed Mean suffer a much slower convergence speed during this attack.

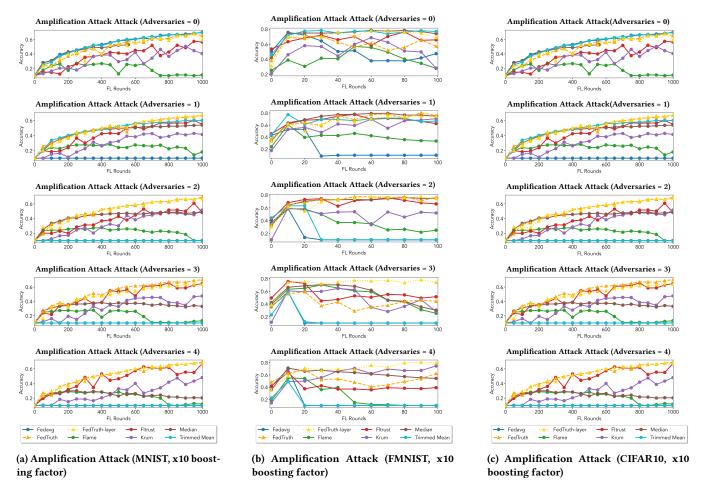


Figure 11: Amplification Attack (MNIST, FMNIST, & CIFAR10)

C RESULTS ON AMPLIFICATION ATTACKS

This section presents the experimental results of the local model amplification attack. We utilized the MNIST, FMNIST, and CIFAR-10 datasets during this experiment and fixed the amplification factor ×10. Additionally, we have incorporated an attack parameter for instances with zero adversaries for each dataset. This will illustrate how each of the models performs under the ideal scenario.

Amplification Attack on MNIST: Figure 11a illustrates the average accuracy of each aggregation algorithm as the number of attackers per round increases. It is evident that without any adversaries present, all of the aggregation algorithms can achieve an accuracy of over 80% after 100 iterations. However, with the addition of just one adversary per round, FedAvg is unable to reach convergence and maintains an accuracy of less than 1%. All of the other algorithms perform as if there are no adversaries present. As we increase the number of adversaries to 2, Trimmed Mean also fails to maintain an accuracy above 1%. Krum and FLAME exhibit slight fluctuations in accuracy around the 30th to 40th iteration, with a more pronounced degradation in performance when there are 3

adversaries. Nevertheless, the overall results are consistent, with Trimmed Mean and FedAvg being the most affected algorithms.

Amplification Attack on FMNIST: Figure 11b shows the results of the amplification attacks on the FMNIST dataset when there are various adversarial attackers. As seen in the figure, the performance of all methods decreases as the number of attackers increases.

Amplification Attack on CIFAR-10: Our analysis begins by focusing on the amplification attack on the CIFAR10 dataset, with one adversary selected per round as seen in Figure 11c. After 1000 iterations, we observed that FedTruth and FedTruth-Layer had the highest accuracy, reaching a total accuracy of 60%. We divided the results into two categories based on whether they achieved final accuracy over or under 40%. Among the group with results over 40%, we include FLTrust, Krum Trimmed-Mean, and Median, along with FedTruth and FedTruth-Layer. Krum performed significantly worse than the other algorithms in its group, with a final accuracy of about 41%, just meeting the accuracy threshold. This leaves only FLAME and FedAvg as adversaries that fall below our accuracy threshold when only one adversary is present per round.

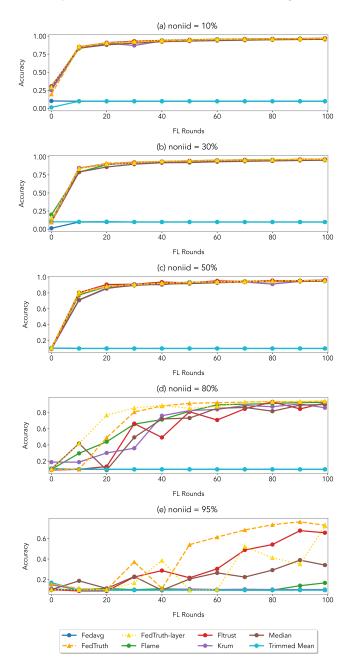


Figure 12: Amplification Attack non-iid Degree Impact (MNIST, 3 adversaries, ×10 amplification factor)

As the number of adversaries in each round increases, ranging from one to four, the performance of FedTruth and FedTruth-Layer remains constant. FLTrust and Krum also maintain accuracy above 40% but experience some fluctuations as the number of adversaries increases. However, the Trimmed-Mean and Median algorithms drop to the 40% group when there are three or four adversaries. This may be due to the increased number of adversarial models skewing the algorithms, causing the median to not be a representative value during the attack.

Amplification Attack with Non-iid Degrees: Figure 12 shows the results of the amplification attack with different non-iid degrees on the training data. Consistent with the results presented in section 5.4, the adverse effects of the different non-iid degrees has a minimum impact on FedTruth.