FedREDefense: Defending against Model Poisoning Attacks for Federated Learning using Model Update Reconstruction Error

Yueqi Xie [1] **Minghong Fang** [2] **Neil Zhenqiang Gong** [3]

Abstract

Federated Learning (FL) faces threats from model poisoning attacks. Existing defenses, typically relying on cross-client/global information to mitigate these attacks, fall short when faced with non-IID data distributions and/or a large number of malicious clients. To address these challenges, we present FedREDefense. Unlike existing methods, it doesn't hinge on similar distributions across clients or a predominant presence of benign clients. Instead, it assesses the likelihood that a client's model update is a product of genuine training, solely based on the characteristics of the model update itself. Our key finding is that model updates stemming from genuine training can be approximately reconstructed with some distilled local knowledge, while those from deliberate handcrafted model poisoning attacks cannot. Drawing on this distinction, FedREDefense identifies and filters out malicious clients based on the discrepancies in their model update **R**econstruction **E**rrors. Empirical tests on three benchmark datasets confirm that FedREDefense successfully filters model poisoning attacks in FL—even in scenarios with high non-IID degrees and large numbers of malicious clients. The source code is available at `https://github.com/xyq7/FedREDefense`.

1. Introduction

Federated Learning (FL) is an emerging machine learning paradigm that enables collaborative model training across multiple clients utilizing their local data, without the need to centralize the data (Konečnỳ et al., 2016; McMahan et al., 2017). Its applications span a diverse range of fields, such as health care (mel; Kairouz et al., 2021; Rieke et al., 2020). Typically, FL contains multiple rounds of communications. In each round, the server dispatches the *global model* to clients. These clients then train the model with their respective *local data* and send the *model updates* to the server. Afterward, the server aggregates the model updates with a certain *aggregation rule* to obtain a new global model.

Distributed training inherently exposes FL to *model poisoning attacks* (Baruch et al., 2019; Fang et al., 2020; Bagdasaryan et al., 2020; Xie et al., 2019; Cao & Gong, 2022; Yin et al., 2024). Different from *data poisoning attacks* (Tolpegin et al., 2020; Fung et al., 2018), where the attackers alter the local data of clients, model poisoning attacks directly manipulate the model updates of clients. Prior works (Fang et al., 2020; Bagdasaryan et al., 2020) underscore the pronounced impact of these attacks on FL. Therefore, in this work, we focus on defending against these model poisoning attacks. Typically, model poisoning attacks craft the model updates with *hand-crafted rules*, such as inverting the benign gradients (Fang et al., 2020) or scaling up malicious gradients obtained from poisoned data (Bagdasaryan et al., 2020).

To defend against modeling poisoning attacks, prevalent defenses (Blanchard et al., 2017; Cao et al., 2021a; Mhamdi et al., 2018; Nguyen et al., 2022; Sun et al., 2019; Yin et al., 2018; Zhang et al., 2022; Fang et al., 2022) typically lean on *cross-client/global information* to counteract model poisoning. Leveraging *cross-client information*, the server can filter out statistical outliers in either dimension-wise (Median and Trimmed Mean (Yin et al., 2018)) or vector-wise (Krum (Blanchard et al., 2017) and FLAME (Nguyen et al., 2022)). Leveraging *global information*, the server can clip or filter the clients' updates using reference server gradient (FLTrust (Cao et al., 2021a)) or Hessian calculated with global models (FLDetector (Zhang et al., 2022)). While these techniques achieve good performance when the data is IID and the number of attackers is limited, their dependency on *similar data distribution across benign clients* becomes their Achilles' heel. Under non-IID data distribution, the disparity between the data distributions of benign clients, as well as their divergence from the global distribution, becomes pronounced. This results in substantial variations

[1]The Hong Kong University of Science and Technology [2]University of Louisville [3]Duke University. Correspondence to: Yueqi Xie <yxieay@connect.ust.hk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

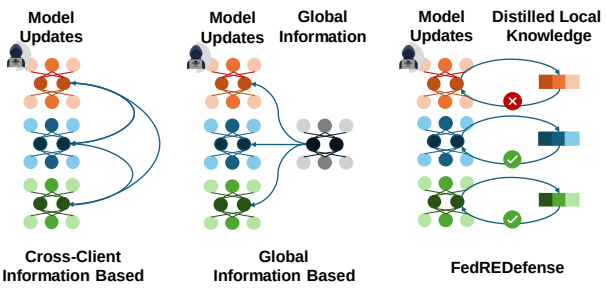


Figure 1: The comparison of FedREDefense and existing defenses. Most existing defenses rely on cross-client/global information to defend against model poisoning attacks, while FedREDefense detects malicious clients solely based on their individual model updates, thus robust to distribution shift and a large number of malicious clients.

in benign model updates, rendering it difficult to pinpoint malicious model updates as "outliers". Moreover, defenses that leverage cross-client information inherently require a predominant presence of benign clients.

Our work: We introduce FedREDefense to defend against model poisoning attacks and address the limitations of existing defenses. Distinctively, FedREDefense determines the benignity or maliciousness of a model update solely based on the individual update itself, thus eliminating the dependency on similar data distribution across benign clients. At its core, it assesses if a model update originates from genuine training or hand-crafted manipulations from model poisoning attacks. This is achieved using the *model update reconstruction error*, computed with the corresponding *distilled local knowledge*. A comparison of FedREDefense and existing defenses are illustrated in Figure 1.

Our approach is grounded in a key discovery: genuine data-trained benign model updates can be more accurately reconstructed using distilled local knowledge; conversely, model updates from model poisoning attacks, which are devised based on specific hand-crafted rules, prove difficult to reconstruct using any local knowledge. This observation draws inspiration from prior studies (Pi et al., 2023; Liu et al., 2022; Cazenavette et al., 2022) that demonstrates the potential of distilled knowledge in approximating genuine data training. Expanding on this discovery, in each round, we calculate the model update reconstruction error for every client. Clients with reconstruction errors exceeding a predefined threshold are classified as malicious and excluded from the aggregation process and future training rounds. Specifically, the process of obtaining a client's model update reconstruction error involves a bi-level optimization strategy for distilling local knowledge, followed by the use of this knowledge to reconstruct the model update and calculate the error. We improve the efficiency of the optimization process by using compact parameterization of local knowledge, maintaining the local knowledge for each client throughout training, and applying a dynamic optimization scheme that terminates upon reaching a predefined error threshold.

We evaluate FedREDefense on three benchmark datasets, including FashionMNIST (Xiao et al., 2017), CIFAR-10 (Krizhevsky et al., 2009), and CINIC (Darlow et al., 2018). We show that FedREDefense can defend against seven state-of-the-art model poisoning attacks, even in circumstances characterized by a high non-IID degree and a substantial number of malicious clients, outperform eight existing defenses against model poisoning attacks in FL. Notably, FedREDefense can achieve 100% detection accuracy of malicious clients in all scenarios, indicating that all the malicious clients are detected while no benign clients are mistakenly excluded during training. We further explore adaptive attacks to FedREDefense, demonstrating that their potency is considerably diminished in comparison to model poisoning attacks when trying to evade FedREDefense.

Our contribution is summarized as follows:

- We uncover a limitation among existing defenses: the reliance on cross-client/global information, resulting in degraded performance under non-IID data distributions and/or a large number of malicious clients.
- We draw a crucial distinction between model poisoning attacks and benign model updates, i.e., whether the model update can be approximately reconstructed using distilled local knowledge.
- We propose FedREDefense, which defends against malicious clients using update reconstruction error without reliance on similar distribution across clients.
- We conduct extensive experiments on FedREDefense in defending against existing state-of-the-art model poisoning attacks and adaptive attacks.

2. Related Work

2.1. Model Poisoning Attacks

Model poisoning attacks in FL refer to attacks where malicious clients craft their model updates with the intent to compromise the system. Distinct from data poisoning attacks, model poisoning attacks directly modify the uploaded model updates instead of local data in arbitrary handcrafted ways, resulting in potentially greater damage (Fang et al., 2020).

Untargeted attack: Untargeted attacks (Baruch et al., 2019; Fang et al., 2020; Shejwalkar & Houmansadr, 2021; Cao & Gong, 2022) aim to degrade the system's utility, resulting in a suboptimal global model that misclassifies

any given sample. These attacks typically manipulate the gradients to hinder training. For example, Fang attack (Fang et al., 2020) strategizes to move against the direction of genuine clients in every round, and MPAF (Cao & Gong, 2022) drives the global model towards a random model.

Targeted attack: The objective of targeted attacks (Bagdasaryan et al., 2020; Xie et al., 2019; Wang et al., 2020; Bhagoji et al., 2019) is to manipulate the system such that samples containing specific triggers are classified into designated classes. Generally, targeted attacks (often referred to as backdoor attacks) introduce triggers into portions of the training data during the learning phase, modify labels, and obtain poisoned model updates. To amplify their impact, these updates are often scaled up, as seen in methods like Scaling attack (Bagdasaryan et al., 2020).

2.2. Defense against Poisoning Attacks

Cross-client information based defenses: To defend against poisoning attacks in FL, several defenses (Yin et al., 2018; Mhamdi et al., 2018; Bernstein et al., 2019; Mhamdi et al., 2018; Li et al., 2019; Shejwalkar & Houmansadr, 2021; Pillutla et al., 2019) utilize cross-client information to filter out or weaken the influence of statistical outliers. For instance, aggregation techniques, such as Median and Trimmed Mean (Yin et al., 2018), employ dimension-wise cross-client information to filter out outlying parameters, while Krum (Blanchard et al., 2017) and Flame (Nguyen et al., 2022) resort to vector wise distance (or cos distance) to filter out some outlying model updates. Norm Bound (Sun et al., 2019) and some clipping-based methods (Nguyen et al., 2022) apply clipping to each individual client, where the threshold can be set using cross-client information. However, such methods fall short when the degree of non-IID is high or when there is a large number of malicious clients, in which cases the cross-client information becomes unstable and unreliable.

Global information based defenses: Some defenses (Cao et al., 2021a; Park et al., 2021; Zhang et al., 2022) leverage global information as a trust anchor to identify/mitigate malicious model updates. For instance, FLTrust (Cao et al., 2021a) operates under the assumption that the server possesses data conforming to a global distribution. It then uses updates on this data to clip and filter client updates. FLDetector (Zhang et al., 2022) utilizes the Hessian estimated from the global models to assess the consistency of a client's model, thereby identifying malicious clients. However, they underperform when there's a significant disparity between the global distribution and individual local distributions, particularly when the non-IID degree is elevated.

Ensemble methods: Additionally, some defenses (Cao et al., 2021b; 2022) such as FLCert (Cao et al., 2022) adopt an ensemble approach, training multiple global models to achieve provable robustness. Note that ensemble methods are based on a specific base defense technique when training one model, and non-trivial provable robustness can only be achieved when a small number of clients is malicious. Therefore, they inherit the limitations from the base defense relying on cross-client/global information.

3. Problem Formulation

Federated learning (FL): We consider a FL system consisting of p clients, each possessing a local training dataset $\mathcal{D}_i$, $i \in [p]$, where $[p] \coloneqq \{1, 2, \ldots, p\}$. The objective of the FL system is to obtain an optimal global model $\boldsymbol{w}^*$ by solving the optimization problem:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{p} f\left(\mathcal{D}_i, \boldsymbol{w}\right), \tag{1}$$

where $f\left(\mathcal{D}_i, \boldsymbol{w}\right)$ represents the local training loss. To approach this objective without centralizing the data, the clients iteratively train the global model with total T communication rounds. In round t, participating client i receives the current global model $\boldsymbol{w}^{t-1}$, trains it with its local dataset $\mathcal{D}_i$ for an updated local model $\boldsymbol{w}_i^t$ by minimizing the local training loss $f\left(\mathcal{D}_i, \boldsymbol{w}\right)$ using optimizers such as Stochastic Gradient Descent (SGD). Then the clients send the local update $\boldsymbol{g}_i^t = \boldsymbol{w}_i^t - \boldsymbol{w}^{t-1}$ to the server. The server then aggregates these updates, generating a new global model $\boldsymbol{w}^t$, and commences the succeeding training round:

$$\boldsymbol{w}^t = \boldsymbol{w}^{t-1} + AR(\{\boldsymbol{g}_i^t\}_{i\in[p]}), \tag{2}$$

where AR corresponds to the server aggregation rule.

Attack model: For the attack, we assume that q out of the p clients are malicious clients controlled by the attacker by compromising genuine clients. We follow the original attack objective and knowledge settings of different model poisoning attacks (Fang et al., 2020; Shejwalkar & Houmansadr, 2021; Xie et al., 2019; Bagdasaryan et al., 2020; Cao & Gong, 2022) respectively. The attacker's goal for targeted attacks (Xie et al., 2019; Bagdasaryan et al., 2020) is to misclassify data with the trigger to a specific class, while the goal for untargeted attacks (Fang et al., 2020; Shejwalkar & Houmansadr, 2021; Cao & Gong, 2022) is degrading the utility of the global model. For the knowledge of the attacker, we assume the attacker has the knowledge of the genuine local data and genuine model update on the compromised clients as well as the aggregation rule. In each round, these malicious clients send crafted model updates to compromise the FL system.

Defense objective: Our primary objective is to discern and exclude malicious clients' model updates from aggregation in every round, subsequently removing them from

Algorithm 1 FedREDefense

Input: Total communication rounds T, initial global model $\boldsymbol{w}^0$, threshold for model update reconstruction error γ, and random initialized distilled local knowledge $\mathcal{K}_i^0, i \in [p]$.
Output: Final global model $\boldsymbol{w}^T$ after T training rounds.

for $t = 1 \to T$ **do**
  $\mathcal{M}^t \leftarrow$ Selected subset of the unmarked clients
  for each client $i \in \mathcal{M}^t$ **in parallel do**
    ▷ Client local training:
    $\boldsymbol{g}_i^t \leftarrow$ Local Model Update$(i, \boldsymbol{w}^{t-1})$
  end for
  for each client $i \in \mathcal{M}^t$ **in parallel do**
    ▷ Compute the model update reconstruction error e_i^t and update distilled local knowledge $\mathcal{K}_i^t$:
    $e_i^t, \mathcal{K}_i^t \leftarrow$ Reconstruction$(\boldsymbol{g}_i^t, \boldsymbol{w}^{t-1}, \mathcal{K}_i^{t-1})$
  end for
  ▷ Filter out and mark the clients with the model reconstruction error e_i^t exceeding the threshold γ.
  ▷ Aggregate the new global model $\boldsymbol{w}^t$ with the remaining client updates.
end for
return $\boldsymbol{w}^T$

Algorithm 2 Reconstruction

Input: Global model $\boldsymbol{w}^{t-1}$, local model update $\boldsymbol{g}_i^t$, and distilled local knowledge $\mathcal{K}_i^{t-1}$.
Output: Reconstruction error e_i^t and updated distilled local knowledge $\mathcal{K}_i^t$.

▷ Initialize the distilled local knowledge with $\mathcal{K}_i^{t-1}$
$\mathcal{K}_i \coloneqq \mathcal{K}_i^{t-1}$
for $m = 1 \to M$ **do**
  ▷ Initialize virtual network with global model $\boldsymbol{w}^{t-1}$:
  $\hat{\boldsymbol{w}}_0 \coloneqq \boldsymbol{w}^{t-1}$
  for $n = 0 \to N-1$ **do**
    ▷ Update virtual network w.r.t. classification loss:
    $\hat{\boldsymbol{w}}_{n+1} = \hat{\boldsymbol{w}}_n - \alpha_i \nabla f(\mathcal{K}_i, \hat{\boldsymbol{w}}_n)$
  end for
  ▷ Compute reconstruction loss between ending virtual model and the client's updated model:
  $\mathcal{L}_m = \|\hat{\boldsymbol{w}}_N - \hat{\boldsymbol{w}}_0 - \boldsymbol{g}_i^t\|_2^2 \ / \ \|\boldsymbol{g}_i^t\|_2^2$
  ▷ Update $\mathcal{K}_i$ and α_i with respect to $\mathcal{L}_m$ with SGD
end for
$\mathcal{K}_i^t = \mathcal{K}_i$
▷ Calculate the model update reconstruction error e_i^t with the distilled local knowledge $\mathcal{K}_i^t$:
$\hat{\boldsymbol{w}}_0 \coloneqq \boldsymbol{w}^{t-1}$
for $n = 0 \to N-1$ **do**
  $\hat{\boldsymbol{w}}_{n+1} = \hat{\boldsymbol{w}}_n - \alpha_i \nabla f(\mathcal{K}_i^t, \hat{\boldsymbol{w}}_n)$
end for
$e_i^t = \|\hat{\boldsymbol{w}}_N - \hat{\boldsymbol{w}}_0 - \boldsymbol{g}_i^t\|_2^2 \ / \ \|\boldsymbol{g}_i^t\|_2^2$
return $e_i^t, \mathcal{K}_i^t$

the FL system. Specifically, in each round, FedREDefense calculates the model update reconstruction errors for the participating clients using their individual model updates and marks those with reconstruction errors larger than the threshold as malicious. The model updates from remaining unmarked clients in the round will be aggregated for a new global model. When a malicious client is identified in round t, its update for that round is discarded, and the client is permanently expelled from the FL system.

4. FedREDefense

4.1. Overview

Our aim is to identify and exclude malicious clients even under situations of highly non-IID distribution and/or a large number of attackers. To achieve this, instead of relying on cross-client or global information, we use the individual model update itself to assess the likelihood of a model update stemming from genuine training. We note that, for genuine clients, certain distilled local knowledge can be employed to approximately reconstruct their model updates. In contrast, model poisoning attacks that do not stem from authentic training cannot be reconstructed in the same manner. Drawing on this distinction, we introduce FedREDefense, a defense against model poisoning attacks using model update **R**econstruction **E**rror.

Specifically, in every round t, we compute the reconstruction error e_i^t for each client i in the selected clients set $\mathcal{M}^t$ and subsequently filter out and mark any client with an error exceeding the predetermined threshold γ. The remaining model updates are aggregated with some aggregation rule (such as FedAVG) for the new global model $\boldsymbol{w}^t$. Clients marked as malicious in this manner are not considered for participation in subsequent rounds. The overall algorithm for FedREDefense is demonstrated in Algorithm 1.

We note that the core of the algorithm is the computation of reconstruction error, which encompasses two primary steps. First, we obtain the distilled local knowledge using a bi-level optimization technique (Section 4.2). Second, we measure the reconstruction error by comparing the model update obtained from training with this distilled knowledge and the actual model update (Section 4.3). Moreover, we discuss how we accelerate the process for enhanced efficiency (Section 4.4). The complete algorithm for the computation of reconstruction error is demonstrated in Algorithm 2. We conclude all the notations in Table 4 in Appendix.

4.2. Obtaining Distilled Local Knowledge

Our goal is to obtain the distilled local knowledge $\mathcal{K}_i^t$ for each client i in each round t so that training with this knowledge can produce a model update closely resembling the actual local model update $\boldsymbol{g}_i^t$. The distilled local knowledge $\mathcal{K}_i$ is parameterized as a small amount (e.g., 10) of data-label pairs. They are initialized as noise and zero, respectively, and maintained during training for acceleration.

In round t, the problem is as follows: starting from global model $\boldsymbol{w}^{t-1}$, we want to optimize local knowledge $\mathcal{K}_i^t$ which minimizes the discrepancy between its trained model

update $\hat{\boldsymbol{w}}_N - \boldsymbol{w}^{t-1}$ and real model update $\boldsymbol{g}_i^t$. Formally, we formulate the optimization problem for obtaining distilled local knowledge as follows:

$$\begin{aligned} &\min_{\mathcal{K}_i^t} \|\hat{\boldsymbol{w}}_N - \boldsymbol{w}^{t-1} - \boldsymbol{g}_i^t\|_2^2 \,, \\ &s.t.\ \hat{\boldsymbol{w}}_N = \mathcal{A}(\mathcal{K}_i^t, \boldsymbol{w}^{t-1}, N), \end{aligned} \tag{3}$$

where $\mathcal{A}(\mathcal{K}_i^t, \boldsymbol{w}^{t-1}, N)$ denotes training the network starting from $\boldsymbol{w}^{t-1}$ using $\mathcal{K}_i^t$ for N steps.

To solve this problem, the whole process can be articulated as a bi-level optimization procedure (Algorithm 2 line 1-15). The inner loop (Algorithm 2 line 5-10) mimics training with distilled local knowledge $\mathcal{K}_i$ with the local training loss $f(\mathcal{K}_i, \hat{\boldsymbol{w}})$. Specifically, a virtual network $\hat{\boldsymbol{w}}$ is initialized with the global model $\boldsymbol{w}^{t-1}$, and subsequently trained using the distilled local knowledge $\mathcal{K}_i$ for N steps with gradient descent and learning rate α_i, eventually arriving at $\hat{\boldsymbol{w}}_N$. In the outer loop, the objective is to minimize the reconstruction loss to refine the distilled local knowledge. This is achieved by minimizing the distance between the model update from actual model update of client i, noted as $\boldsymbol{g}_i^t$, and the model update obtained using local knowledge $\mathcal{K}_i$ within the inner loop, noted as $\hat{\boldsymbol{w}}_N - \hat{\boldsymbol{w}}_0$. Note that for the outer loop, the optimization is over the trainable local knowledge $\mathcal{K}_i$ and inner loop learning rate α_i using stochastic gradient descent (SGD) for M iterations.

4.3. Calculating Reconstruction Error

With the obtained distilled local knowledge $\mathcal{K}_i^t$, we can perform a virtual training using the distilled local knowledge on the model initialized with $\boldsymbol{w}^{t-1}$ (Algorithm 2 line 16-21), similar to the inner loop of optimization. This learning process can be viewed as a reconstruction of the model update using the distilled local knowledge $\mathcal{K}_i^t$. Then, we can calculate the model update reconstruction error e_i^t by contrasting the reconstructed model update $\hat{\boldsymbol{w}}_N - \hat{\boldsymbol{w}}_0$ with the real model update $\boldsymbol{g}_i^t$, normalized with the ℓ_2 norm of real model update $\boldsymbol{g}_i^t$.

We observe that for benign clients, the model update reconstruction error e_i^t, typically falls within the range of 0 to 0.3. Conversely, for instances of model poisoning attacks, this error spans from 0.7 to 1.0. These distinct distributions enable us to establish a predefined threshold, γ, to facilitate the differentiation between benign and malicious updates.

4.4. Acceleration

To accelerate the calculation of construction errors for clients in each round, we consider the following aspects:

Compact local knowledge: We employ highly compact distilled local knowledge, parameterized as a very limited set of parameters, e.g., 10 data-label pairs, around 2% of the client's local data, facilitating rapid optimization.

Maintaining local knowledge: Instead of optimizing the distilled local knowledge from noise every round, we maintain and keep refining the local knowledge of every client. An insight underpinning this is that the local distribution of a specific benign client remains relatively static. Therefore, the distilled local knowledge of a certain client should exhibit consistency across multiple rounds. Empirically, it's only when a client participates for the first time that we use a larger value of optimization iterations M. For subsequent refinements, a much smaller value of M suffices.

Dynamic optimization: We propose a dynamic optimization scheme to further accelerate the optimization process. The key insight is that, for our algorithm, it is merely necessary to ascertain if the reconstruction error can attain a value below a specified threshold, indicating that the model update can be approximated effectively. Hence, full convergence is not obligatory. Consequently, the number of optimization iterations M is not predetermined but is dynamically adjusted. The optimization ceases once the reconstruction error falls below the threshold γ.

5. Evaluation

5.1. Experimental Setup

5.1.1. Datasets

We utilize three commonly used FL datasets to evaluate our method: FashionMNIST (Xiao et al., 2017), CIFAR-10 (Krizhevsky et al., 2009), and CINIC (Darlow et al., 2018). Following prior studies (Pi et al., 2023; Lin et al., 2020; Gu et al., 2022), we employ the Dirichlet distribution to emulate non-IID data distribution among clients. The hyperparameter β dictates the extent of non-IID nature, where a smaller β suggests higher non-IID characteristics. The training samples are distributed to the clients for training according to the Dirichlet distribution, while we test the final global model on the official testing samples. Detailed information of the dataset is deferred to Appendix A.1.

5.1.2. Attacks and Compared Defenses

We consider seven state-of-the-art model poisoning attacks, including five untargeted attacks (Fang (Fang et al., 2020), LIE (Baruch et al., 2019), Min-Max (Shejwalkar & Houmansadr, 2021), Min-Sum (Shejwalkar & Houmansadr, 2021), and MPAF (Cao & Gong, 2022)) and two targeted attacks (Scaling (Bagdasaryan et al., 2020) and DBA (Xie et al., 2019)). Details for these attacks are illustrated in Appendix A.2. We compare FedREDefense with eight state-of-the-art defenses, including four defenses leveraging cross-client information (Krum (Blanchard et al., 2017), Median (Yin et al., 2018), Trimmed Mean (Yin et al., 2018), Norm Bound (Sun et al., 2019), and FLAME (Nguyen

Table 1: DACC (%) ↑, FPR (%) ↓, FNR (%) ↓, and AAR ↓ of FedREDefense and FLDetector against different attacks. ↑ (↓) indicates the larger (smaller) the better metric.

Attack	Detector	FashionMNIST				CIFAR-10				CINIC			
		DACC	FPR	FNR	AAR	DACC	FPR	FNR	AAR	DACC	FPR	FNR	AAR
Fang	FLDetector	93.00	9.72	0.00	60.00	94.00	8.33	0.00	0.00	86.00	19.44	0.00	60.00
	FedREDefense	100.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00
LIE	FLDetector	72.00	0.00	100.00	500.00	0.00	100.00	100.00	500.00	5.00	93.06	100.00	500.00
	FedREDefense	100.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00
Min-Max	FLDetector	72.00	0.00	100.00	500.00	0.00	100.00	100.00	500.00	72.00	0.00	100.00	500.00
	FedREDefense	100.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00
Min-Sum	FLDetector	72.00	0.00	100.00	500.00	0.00	100.00	100.00	500.00	8.00	88.89	100.00	500.00
	FedREDefense	100.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00
MPAF	FLDetector	0.00	100.00	100.00	500.00	0.00	100.00	100.00	500.00	0.00	100.00	100.00	500.00
	FedREDefense	100.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00
DBA	FLDetector	100.00	0.00	0.00	60.00	99.00	0.00	3.57	75.71	100.00	0.00	0.00	60.00
	FedREDefense	100.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00
Scaling	FLDetector	100.00	0.00	0.00	60.00	100.00	0.00	0.00	60.00	100.00	0.00	0.00	60.00
	FedREDefense	100.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00

et al., 2022)), two defenses leveraging global information (FLTrust (Cao et al., 2021a) and FLDetector (Zhang et al., 2022)), and one ensemble method (FLCert (Cao et al., 2022)), detailed in Appendix A.3.

5.1.3. Evaluation Metrics

We consider a set of metrics for evaluating detection and defense effectiveness, including Detection Accuracy (DACC), False Positive Rate (FPR), False Negative Rate (FNR), Averaged Aggregated Rounds for Malicious Clients (AAR), and Testing Accuracy (TACC). Detailed descriptions of the metrics are deferred to Appendix A.4.

5.1.4. FL Settings

We illustrate the default configurations for our experiments in this section. Note that we have investigated the effects of various FL settings as elaborated in Section 5.3. By default, the system involves 100 clients, out of which 28 are malicious. The local epoch is set to 1, spanning a total of 500 training rounds. The non-IID degree β is set to 0.1. For FedREDefense, the basic aggregation rule is set as FedAVG for the remaining clients after detection in each round. The optimizers for local knowledge and learning rate are Stochastic Gradient Descent (SGD). The distilled local knowledge $\mathcal{K}_i^t, i \in [p]$ is parameterized as 10 data-label pairs and initiated with uniform noise. The threshold γ and the step for inner loop N are set to 0.6 and 5, respectively. The selection of the threshold is further discussed in Appendix B.4. M is dynamically set according to the algorithm specified in Section 4.4 with upper bounds in Appendix Table 7. More detailed configurations are shown in Appendix A.5.

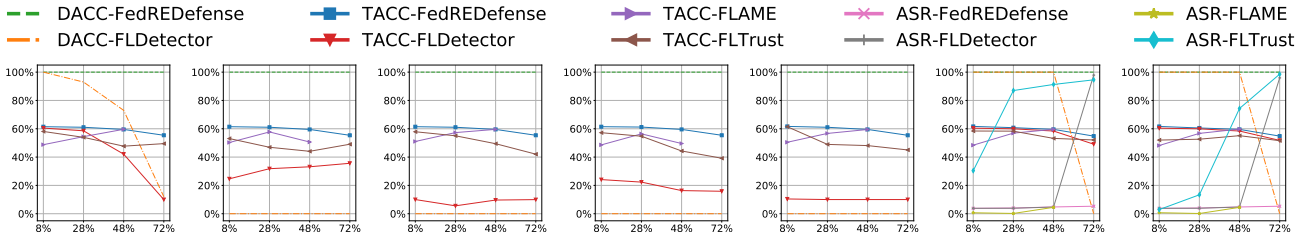
5.2. Main Results

FedREDefense can detect and filter out malicious clients once they participate: We first evaluate the detection performance for identifying malicious clients, in comparison with existing defenses which can perform detection of malicious clients, i.e., FLDetector. At the end of the training process, alongside obtaining a global model, these methods also produce labels for all clients, categorizing them as either benign or malicious. We demonstrate the detection-related metrics and Averaged Aggregated Rounds for Malicious Clients (AAR) for three datasets in Table 1. We make the following observations.

First, FedREDefense can detect the malicious clients once they participate in training with AAR of 0. This means no need to recover from poisoning attacks after detection (Cao et al., 2023). Furthermore, the DACC of FedREDefense consistently stands at 100% across all scenarios. This denotes that during the entire training span, there is not a single instance where a benign client is erroneously flagged as malicious, ensuring optimal integration of model updates exclusively from genuine clients. Meanwhile, FLDetector exhibits mixed results in its detection capabilities. It successfully identifies attacks such as Fang, DBA, and Scaling, aligning with observations from the original study. However, it falls short in detecting specially designed malicious model updates like LIE, Min-Sum, Min-Max, and MPAF. This limitation stems from FLDetector's methodology, which assumes clients presenting consistent updates across training rounds to be benign. This assumption fails to account for the fact that some malicious strategies can produce updates that are even more consistent than those from benign clients, especially in high non-IID scenarios. Moreover, to better investigate some defenses which do not output labels of

Table 2: TACC (%) ↑ (/ASR (%) ↓) of the global model after training under different attacks and defenses on FashionMNIST, CIFAR-10, and CINIC datasets. ↑ (↓) indicates the larger (smaller) the better metric. For untargeted attacks or no attack, we highlight in **bold** the highest TACC. For targeted attacks, we highlight in **bold** the highest TACC with ASR ≤ 0.05.

Dataset	Attack	FedAVG	Krum	Median	Trimmed Mean	Norm Bound	FLAME	FLTrust	FLDetector	FLCert	FedREDefense
FashionMNIST	No att	85.94	60.45	82.20	85.98	85.37	81.73	**86.97**	86.03	81.79	86.04
	Fang	28.18	39.63	35.67	68.41	70.86	82.43	84.06	84.62	75.87	**85.54**
	Min-Max	76.85	49.09	72.10	72.44	77.96	82.05	82.10	76.69	78.89	**85.61**
	Min-Sum	84.12	53.47	76.16	78.70	83.76	79.08	80.66	84.08	79.75	**85.68**
	MPAF	20.46	10.22	76.73	75.39	20.26	82.44	79.95	10.00	79.15	**85.61**
	DBA	85.86/98.33	65.44/7.93	82.48/5.69	83.37/9.24	84.11/69.11	82.30/4.57	76.27/98.11	**85.96/2.54**	82.49/25.42	85.54/2.43
	Scaling	87.45/98.42	65.41/6.56	82.54/78.04	84.39/87.46	85.11/88.86	82.03/4.97	82.16/97.66	86.07/8.53	82.86/82.00	**85.66/2.50**
CIFAR-10	No att	61.51	26.86	51.68	61.44	61.36	51.17	57.28	61.59	50.89	**61.69**
	Fang	21.77	3.04	20.10	23.64	37.50	54.44	53.94	58.56	36.04	**61.06**
	Min-Max	45.25	9.88	38.62	38.20	47.69	57.30	55.04	5.57	39.39	**61.06**
	Min-Sum	55.14	16.37	39.58	43.23	54.60	56.40	54.73	22.32	44.15	**61.20**
	MPAF	11.77	10.42	33.26	26.39	11.65	56.71	48.96	10.00	39.45	**61.05**
	DBA	59.32/79.90	28.43/4.90	51.85/18.98	51.47/38.48	59.78/17.82	56.74/0.15	52.63/13.36	59.97/3.86	50.21/41.61	**60.56/3.81**
	Scaling	63.85/92.08	27.69/4.62	51.04/75.59	52.72/82.31	60.16/73.50	56.92/0.19	58.31/87.00	59.97/3.86	50.73/80.51	**60.78/4.03**
CINIC	No att	**52.25**	25.40	43.89	52.10	52.00	46.24	39.66	52.15	42.96	52.20
	Fang	16.10	4.47	18.84	22.27	32.87	42.17	46.02	46.01	32.97	**50.81**
	Min-Max	35.26	9.96	31.12	30.33	38.32	46.12	38.54	13.23	31.70	**50.96**
	Min-Sum	45.26	14.51	31.28	34.40	45.79	45.67	40.12	16.33	35.80	**50.81**
	MPAF	11.91	10.91	29.19	20.10	11.86	49.65	33.80	10.91	34.24	**51.04**
	DBA	44.7/85.49	22.51/4.48	43.33/27.59	44.53/35.84	49.99/23.51	45.95/5.65	40.35/43.14	52.17/5.12	42.65/39.19	**50.87/4.48**
	Scaling	53.43/91.27	22.82/4.72	43.48/81.06	45.52/79.80	51.46/75.61	45.43/6.41	36.78/98.60	**50.90/4.24**	42.78/83.14	**50.90/4.02**



(a) Fang (b) LIE (c) Min-Max (d) Min-Sum (e) MPAF (f) Scaling (g) DBA

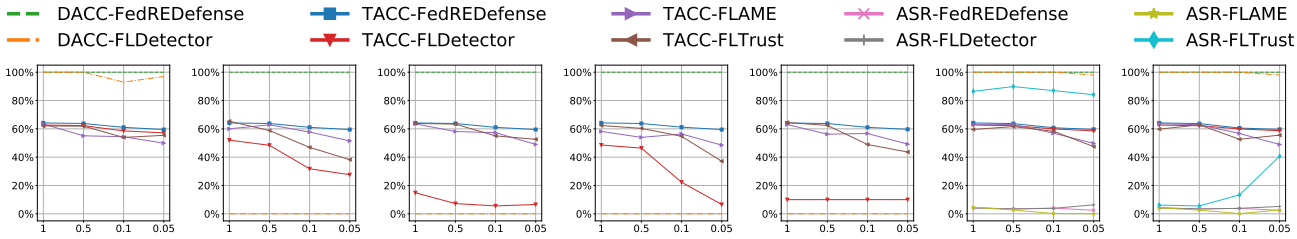
Figure 2: DACC of FedREDefense and FLDetector, and TACC(/ASR) of the global model for all defenses as a function of the fraction of malicious clients under different defenses and attacks.

clients after training but also perform filtering in each round (i.e., FLAME and FLTrust), we discuss averaged detection performance during all training rounds in Appendix B.1.

FedREDefense outperforms existing defenses: Table 2 presents the end-to-end testing performance of FedREDefense in comparison with the state-of-the-art defenses when subjected to various attacks. "No att" corresponds to no attack. Notably, FedREDefense is resilient to model poisoning attacks since it excludes them from aggregation once the attacker performs attacks and does not misclassify benign clients as malicious, ensuring that its performance remains unaffected by either targeted or untargeted attacks. This indicates that these state-of-the-art model poisoning attacks, when faced with the protection provided by FedREDefense, no longer pose a threat to the FL system, even under a high non-IID degree. Meanwhile, existing defenses, which predominantly depend on cross-client/global information, do not achieve satisfactory performance for certain attacks in the experimental setting. This is largely due to the substantial impact of non-IIDness on the effectiveness of these defenses. For instance, some filtering-based approaches, such as FLAME, FLTrust, and FLDetector, incorrectly classify benign and malicious clients as discussed. Moreover, statistical analysis used in algorithms such as Median and Trimmed Mean become less reliable because of distribution shifts. FLCert, which relies on the performance of its basic aggregation rule, Median in the experiment, also falls short as Median does not work well.

5.3. Impact of FL Settings

Next, we study the influence of different FL settings on our defense, including the fraction of malicious clients, the degree of non-IIDness, local training epochs, and participation rate. Due to limited space, the results on local training epochs and participation rate are demonstrated in Appendix B.2. By default, we conduct experiments on the CIFAR-10 dataset, with all other parameters kept consistent with the default settings elaborated in Section 5.1.4. We compare FedREDefense with the baselines showing competitive performance in Table 2, i.e., FLDetector, FLAME, and FLTrust, in their applicable scenarios.



(a) Fang (b) LIE (c) Min-Max (d) Min-Sum (e) MPAF (f) Scaling (g) DBA

Figure 3: DACC of FedREDefense and FLDetector, and TACC(/ASR) of the global model for all defenses as a function of the non-IID degrees under different defenses and attacks.

Impact of the fraction of malicious clients: We first examine the impact of varying fractions of malicious clients, as illustrated in Figure 2. We test scenarios where 8, 28, 48, and 72 out of 100 clients were malicious. Remarkably, FedREDefense maintains a 100% DACC, achieving complete exclusion of malicious clients across all fractions. Since the acquisition of reconstruction errors pertains solely to each individual client and is independent of cross-client or global information, an increase in the number of malicious clients does not affect our detection efficacy. TACC experiences a minor decrease as the number of malicious clients increases, primarily due to the reduced number of benign clients, leading to less data available for training. Conversely, many existing methods (e.g., FLAME), which rely on cross-client information, intrinsically cannot handle more than 50% malicious clients since they determine the malicious clients based on the assumption that benign clients constitute the majority. Thus, we do not show the results of FLAME for 72% malicious clients. Meanwhile, both existing methods based on cross-client and global information exhibit a notable performance decrease with a larger fraction of malicious clients. For example, FLDetector also faces difficulty in detecting malicious clients when the fraction grows large since the global information (i.e., global model) is dominated by malicious clients.

Impact of non-IID degree: We evaluate the impact of varying non-IID degrees on defense effectiveness, as demonstrated in Figure 3. Specifically, we consider non-IID degrees stemming from the Dirichlet distribution, with β ranging from 1 to 0.05. Smaller β indicates a higher non-IID degree. Our results indicate that even in scenarios with extreme non-IID conditions (e.g., $\beta = 0.05$), our approach still ensures a 100% DACC, effectively filtering out malicious clients. It is because FedREDefense only determines based on whether a model update is likely to be the product of genuine training, without any dependence on the distribution of data across clients. TACC slightly decreases as the non-IID degree increases (i.e., as β decreases). This decline can be attributed to the fact that an increased non-IID degree, even in the absence of attacks, affects the convergence and performance of FedAVG. Meanwhile, we note that existing state-of-the-art defenses, due to the reliance on the consistency of distribution across clients (also known as the consistency of local and global distribution), generally exhibit degraded effectiveness in high non-IIDness.

6. Discussion and Limitations

6.1. Data Poisoning and Adaptive Attacks

Data poisoning attacks: Given that our defense hinges on examining whether clients' model updates are based on genuine training, data poisoning attacks, which train the model update with genuine but poisoned data, cannot be detected by our defense. Consequently, we investigate the effects of untargeted and targeted label flip attacks:

- **Label Flip attack** (Fung et al., 2018): Label Flip attack is an untargeted data poisoning attack, which poisons the label for each data sample as the next class of the ground truth label. The malicious client then trains with the local poisoned data and sends the model update to the server.
- **Targeted Label Flip attack** (Tolpegin et al., 2020): Targeted Label Flip attack aims to mistake the samples from one specific source class to another target class. It poisons the data by setting the labels for samples from the source class as the target class. In practice we set the source class as label '0' and the target class as label '2'. The ASR is calculated with the fraction of samples from the source class misclassified as the target class divided by the total number of samples in the source class.

Adaptive attacks: We further derive an adaptive attack to enhance the impact of the data poisoning attack while preventing it from being filtered out. We contemplate an adaptive attack setting wherein the attacker has access to all settings of FedREDefense, such as the methodology for obtaining the model update reconstruction error and all its associated hyperparameters, like the threshold γ. The objective of the adaptive attack is to identify the maximal scaling factor λ_i^t that can upscale the data poisoning effect

Algorithm 3 Adaptive scaling attacks

Input: Global model $\boldsymbol{w}^{t-1}$, local model update of the basic data poisoning attack $\boldsymbol{g}_i^t$, initial step for scaling factor δ_{init}, threshold for the step of scaling factor γ_{att}, threshold for model update reconstruction error γ, and distilled local knowledge $\mathcal{K}_i^{t-1}$.
Output: Maximized scaling factor λ_i^t and updated distilled local knowledge $\mathcal{K}_i^t$.

$\lambda \leftarrow 1$
$\delta \leftarrow \delta_{\text{init}}$
while $\delta \geq \gamma_{att}$ **do**
  $\lambda = \lambda + \delta$
  ▷ Compute the model update reconstruction error e_i^t for scaled model update $\lambda \boldsymbol{g}_i^t$:
  $e_i^t, \mathcal{K}_i^t \leftarrow \text{Reconstruction}(\lambda \boldsymbol{g}_i^t, \boldsymbol{w}^{t-1}, \mathcal{K}_i^{t-1})$
  if $e_i^t > \gamma$ **then**
    $\lambda = \lambda - \delta$
  end if
  $\delta = \delta / 2$
end while
$\lambda_i^t = \lambda$
return $\lambda_i^t, \mathcal{K}_i^t$

on the model update without being detected for malicious client i in round t. We employ a binary search method to pinpoint this scaling factor. The overall algorithm for finding λ_i^t is illustrated in Algorithm 3. This algorithm is applied to Label Flip attack and Targeted Label Flip attack as **Adaptive Label Flip attack** and **Adaptive Targeted Label Flip attack**.

Evaluation: For evaluation, we set the hyperparameters for adaptive scaling attacks as follows: γ_{att} and δ_{init} are set to 0.125 and 2, respectively. Note that FedREDefense, as a detection method, can apply any aggregation rule after filtering out detected malicious clients. In Table 3, we demonstrate the TACC(/ASR) of FedREDefense using different aggregation rules (i.e., FedAVG and Norm Bound) when faced with various adaptive attacks. We note that although these adaptive attacks can not be detected by FedREDefense, they do not pose large threats to the FL system with simple aggregation rules compared with the impactful model poisoning attacks. Furthermore, while adaptive scaling attacks showcase a heightened attack effectiveness relative to their data poisoning counterparts, they still don't inflict substantial damage on the system. This is attributable to a trade-off: in their bid to evade detection by our method, attackers inadvertently compromise the potency of their attack. We notice that the scaling factor determined through binary search during the training phase remains relatively modest, falling in the range from 1 to 2. In conclusion, FedREDefense demonstrates strong efficacy against prominent model poisoning attacks. When it comes to the more subdued adaptive attacks, the system can relatively afford to their bypassing the detection and directly integrate them using FedAVG or alternative aggregation rules.

Table 3: TACC(%)(/ASR(%)) of FedREDefense using different aggregation rules under adaptive attacks.

		FashionMNIST	CIFAR-10	CINIC
Label Flip	FedAVG	72.99	52.95	44.58
	Norm Bound	83.30	54.34	45.26
Targeted Label Flip	FedAVG	85.62/7.00	60.95/18.40	51.33/17.96
	Norm Bound	85.30/6.20	60.40/18.60	50.86/18.58
Adaptive Label Flip	FedAVG	71.57	47.04	38.66
	Norm Bound	77.30	47.67	39.48
Adaptive Targeted Label Flip	FedAVG	81.24/6.10	59.56/10.40	45.14/30.87
	Norm Bound	81.99/5.10	56.62/16.80	44.27/17.37

6.2. Additional Computational and Storage Overhead

The server is required to maintain local knowledge for each client. Therefore, its storage overhead is $O(pK)$, where K denotes the number of parameters in the local knowledge and p is the number of clients. In practical settings, we've set K to be quite small, corresponding to 10 data samples (around 2% of the client's local data), thereby economizing on overhead. In each round, the computational overhead is linear to the number of clients p and the local knowledge K. The optimization iteration is dynamically set as discussed in Section 4.1 to efficiently obtain the distilled local knowledge. An empirical comparison for the efficiency is presented in Appendix B.3. It is noteworthy that the computation overhead is on the server, which we anticipate to possess high computational and storage capacities, such as the data center. Regarding the clients' local training and communication, we haven't added any additional burdens. Moreover, our aggregation does not require cross-client information as the prior works (Nguyen et al., 2022; Yin et al., 2018; Blanchard et al., 2017), enabling *asynchronous and parallel computation* of clients' reconstruction errors. This further reduces the computation latency for FedREDefense, making it sufficiently practical for real-world FL systems.

7. Conclusion

In this work, we present FedREDefense to defend against model poisoning attacks for FL. Instead of relying on cross-client/global information, FedREDefense detects malicious clients only by assessing whether the model updates are likely to be a product of genuine training using the model update reconstruction errors. Therefore, FedREDefense can defend against model poisoning attacks when the data distribution is highly non-IID and when there is a large number of malicious clients. We show that FedREDefense can successfully detect seven state-of-the-art model poisoning attacks in various scenarios, without mistaking benign clients as malicious during the whole FL training process. Overall, FedREDefense provides a novel effective defense technique for FL, bridging the gap of existing defenses in real-world FL systems where the data distribution may be non-IID and the number of malicious clients may be large.

Impact Statement

This paper aims to further advance the security of Federated Learning (FL). We introduce a novel defense mechanism FedREDefense against model poisoning attacks. Overall, we believe that our approach will foster risk mitigation in real-world FL systems.

Acknowledgements

We thank the anonymous reviewers for their comments. This work was supported by NSF under grant No. 2112562, 1937786, 2131859, 2125977, and 1937787, as well as ARO under grant No. W911NF2110182.

References

Machine learning ledger orchestration for drug discovery (melloddy). https://www.melloddy.eu/.

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. How to backdoor federated learning. In *AISTATS*, 2020.

Baruch, G., Baruch, M., and Goldberg, Y. A little is enough: Circumventing defenses for distributed learning. In *NeurIPS*, 2019.

Bernstein, J., Zhao, J., Azizzadenesheli, K., and Anandkumar, A. signsgd with majority vote is communication efficient and fault tolerant. In *ICLR*, 2019.

Bhagoji, A. N., Chakraborty, S., Mittal, P., and Calo, S. Analyzing federated learning through an adversarial lens. In *ICML*, 2019.

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

Cao, X. and Gong, N. Z. Mpaf: Model poisoning attacks to federated learning based on fake clients. In *CVPR Workshops*, 2022.

Cao, X., Fang, M., Liu, J., and Gong, N. Z. Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *NDSS*, 2021a.

Cao, X., Jia, J., and Gong, N. Z. Provably secure federated learning against malicious clients. In *AAAI*, 2021b.

Cao, X., Zhang, Z., Jia, J., and Gong, N. Z. Flcert: Provably secure federated learning against poisoning attacks. In *IEEE Transactions on Information Forensics and Security*, 2022.

Cao, X., Jia, J., Zhang, Z., and Gong, N. Z. Fedrecover: Recovering from poisoning attacks in federated learning using historical information. In *IEEE Symposium on Security and Privacy*, 2023.

Cazenavette, G., Wang, T., Torralba, A., Efros, A. A., and Zhu, J.-Y. Dataset distillation by matching training trajectories. In *CVPR*, 2022.

Darlow, L. N., Crowley, E. J., Antoniou, A., and Storkey, A. J. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

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

Fang, M., Liu, J., Gong, N. Z., and Bentley, E. S. Aflguard: Byzantine-robust asynchronous federated learning. In *ACSAC*, 2022.

Fung, C., Yoon, C. J., and Beschastnikh, I. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

Gu, H., Guo, B., Wang, J., Sun, W., Liu, J., Liu, S., and Yu, Z. Fedaux: An efficient framework for hybrid federated learning. In *ICC*, 2022.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 2021.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Li, L., Xu, W., Chen, T., Giannakis, G. B., and Ling, Q. Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *AAAI*, 2019.

Lin, T., Kong, L., Stich, S. U., and Jaggi, M. Ensemble distillation for robust model fusion in federated learning. In *NeurIPS*, 2020.

Liu, P., Yu, X., and Zhou, J. T. Meta knowledge condensation for federated learning. *arXiv preprint arXiv:2209.14851*, 2022.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

Mhamdi, E. M. E., Guerraoui, R., and Rouault, S. The hidden vulnerability of distributed learning in byzantium. In *ICML*, 2018.

Nguyen, T. D., Rieger, P., De Viti, R., Chen, H., Brandenburg, B. B., Yalame, H., Möllering, H., Fereidooni, H., Marchal, S., Miettinen, M., et al. Flame: Taming backdoors in federated learning. In *USENIX Security Symposium*, 2022.

Park, J., Han, D., Choi, M., and Moon, J. Sageflow: Robust federated learning against both stragglers and adversaries. In *NeurIPS*, 2021.

Pi, R., Zhang, W., Xie, Y., Gao, J., Wang, X., Kim, S., and Chen, Q. Dynafed: Tackling client data heterogeneity with global dynamics. In *CVPR*, 2023.

Pillutla, K., Kakade, S. M., and Harchaoui, Z. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

Rieke, N., Hancox, J., Li, W., Milletari, F., Roth, H. R., Albarqouni, S., Bakas, S., Galtier, M. N., Landman, B. A., Maier-Hein, K., et al. The future of digital health with federated learning. In *NPJ digital medicine*, 2020.

Shejwalkar, V. and Houmansadr, A. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*, 2021.

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

Tolpegin, V., Truex, S., Gursoy, M. E., and Liu, L. Data poisoning attacks against federated learning systems. In *ESORICS*, 2020.

Wang, H., Sreenivasan, K., Rajput, S., Vishwakarma, H., Agarwal, S., Sohn, J.-y., Lee, K., and Papailiopoulos, D. Attack of the tails: Yes, you really can backdoor federated learning. In *NeurIPS*, 2020.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

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

Yin, D., Chen, Y., Kannan, R., and Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*, 2018.

Yin, M., Xu, Y., Fang, M., and Gong, N. Z. Poisoning federated recommender systems with fake users. In *The Web Conference*, 2024.

Zhang, Z., Cao, X., Jia, J., and Gong, N. Z. Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In *KDD*, 2022.

A. Additional Experimental Setups

A.1. Datasets

FashionMNIST is a grayscale dataset consisting of 10 clothing classes. It comprises 60,000 training images and 10,000 test images. CIFAR-10 is a color image classification dataset comprising 10 daily object classes. It consists of 50,000 training samples and 10,000 testing samples. CINIC extends CIFAR-10 to a total of 270,000 images with the downsampled image from ImageNet, equally split as training, validation, and testing samples.

A.2. Attacks

Fang (Fang et al., 2020): Fang attack is an untargeted attack. In each round, the attacker's optimization objective is to make the attacked model update sufficiently divergent from the benign aggregate without attacks. Particularly, they propose attack strategies specifically tailored for the Krum and Trimmed Mean/Median aggregation rules. Notably, the latter is observed to possess significant transfer capabilities. Hence, we employ Krum attack for the Krum aggregation rule and the Trimmed Mean attack for other defenses.

A little is enough (LIE) (Baruch et al., 2019): LIE adds small noise to the benign aggregate to prevent convergence. Specifically, it sets the model updates for all malicious clients as $\mu_j - z_{\max} \cdot \sigma_j$, where $z_{\max}$ is set as the maximal z with $\phi(z) < (p-q-s)/(p-q)$, where p represents the total number of participating clients and q denotes the number of malicious clients, supporter $s = \lfloor p/2 + 1 \rfloor - q$, and ϕ corresponds to Cumulative Standard Normal Function.

Min-Max (Shejwalkar & Houmansadr, 2021): Min-Max shares the same attack objective as Fang, but it is aggregation rule agnostic. Specifically, Min-Max sets the attack deviation to oppose the direction of the benign aggregate. We employ the "standard deviation" variant of this approach. Moreover, its scale is adjusted to the maximal extent possible, ensuring that its distance from any benign client never surpasses the distance between the two most distant benign clients.

Min-Sum (Shejwalkar & Houmansadr, 2021): Min-Sum differs from Min-Max primarily in its scale configuration. For Min-Sum, the scale ensures that the cumulative distance to all benign clients does not surpass the cumulative distance of any benign client to other clients.

MPAF (Cao & Gong, 2022): MPAF is an untargeted attack

in which the attacker drives the current global model to a random model in every round. We follow the original paper to set the random model as the initial model. The scaling factor is set to 3.

Scaling (Bagdasaryan et al., 2020): Scaling attack is a targeted attack. In Scaling attack, malicious clients create replicas of their local training data, embed triggers, and assign new labels to these images. Subsequently, these clients train using both the original and poisoned datasets. The resulting model update is further scaled to intensify its impact. The scaling factor is set at 3, and the target label is set to '0'.

DBA (Xie et al., 2019): DBA is a targeted attack. DBA differs from Scaling attack in that the backdoor trigger is decomposed into four local patterns and embedded into the local data of different malicious clients. The scaling factor is set at 3, and the target label is set to '0'.

A.3. Compared Defenses

Krum (Blanchard et al., 2017): Krum employs a selection criterion for model updates based on Euclidean distance. Specifically, it chooses the model update that exhibits the minimal cumulative Euclidean distance to its $p - q - 2$ nearest neighbors, where p represents the total number of participating clients and q denotes the number of malicious clients.

Median (Yin et al., 2018): Median calculates the aggregated update for each dimension by selecting the median value from all updates in that specific dimension.

Trimmed Mean (Yin et al., 2018): Trimmed Mean removes the largest and smallest m values for each dimension. The final aggregated update for that specific dimension is then computed by averaging the remaining $p - 2q$ values.

Norm Bound (Sun et al., 2019): Norm Bound clips the magnitude of the model updates within a threshold to weaken the influence of attack. The threshold is set as the average of the norm of the benign model updates.

FLAME (Nguyen et al., 2022): FLAME clusters the clients based on the cosine distance of their local models. The cluster with the fewer members is subsequently discarded. Following this, the remaining model updates are clipped, subjected to noise addition, and subsequently aggregated.

FLTrust (Cao et al., 2021a): FLTrust assumes the availability of server data and computes a server model update. Then the cosine similarity between the clients' model updates and the server model update as well as the scale is applied to filter and clip the client model updates. The number of server data samples is set to 500 to match the clients' local data and the server's local epoch and local optimizer are set the same as the clients.

Table 4: Notations.

Notation	
p	total client number
q	malicious client number
$\mathcal{D}_i$	local dataset for client i
AR	aggregation rule
T	total communication round
$\boldsymbol{w}^t$	global model after round t
$\boldsymbol{g}_i^t$	local model update of client i in round t
γ	threshold for model update reconstruction error
$\mathcal{M}^t$	selected clients in round t
$\mathcal{K}_i^t$	distilled local knowledge of client i in round t
e_i^t	model update reconstruction error of client i in round t
N	inner loop iterations for FedREDefense
M	outer loop iterations for FedREDefense
α_i	learning rate of virtual network for FedREDefense
λ_i^t	maximized scaling factor for adaptive scaling attacks
γ_{att}	threshold for step of scaling factor for adaptive scaling attacks
β	hyperparamter for the Dirichlet distribution

FLDetector (Zhang et al., 2022): FLDetector detects malicious clients by estimating the gradient of clients using the global models, and the cluster of clients showing less consistency with the estimation is classified as malicious. Note that FLDetector requires training trajectory in multiple rounds (set as 10 following the original paper) after cold start rounds (set as 50 following the original paper) to detect malicious clients. Therefore, FLDetector requires reloading the initial model if malicious clients are detected since the model is already poisoned. We keep the total training rounds the same as other defenses for fairness. The basic aggregation rule is set as FedAVG.

FLCert (Cao et al., 2022): FLCert uses model ensembling to provide provable robustness against attacks when the percentage of malicious clients is limited. We divide the clients into 10 groups and use Median to train a global mode in each group. The prediction is made based on majority voting among all the final global models.

A.4. Evaluation Metrics

- **Detection Accuracy (DACC)** is the fraction of correctly classified clients to the total number of clients.
- **False Positive Rate (FPR)** is the fraction of the benign clients that are wrongly classified as malicious to the total number of benign clients.

Table 5: Model architecture for FashionMNIST.

Layer Type	Size
Convolution + ReLU	3x3x30
Max Pooling	2x2
Convolution + ReLU	3x3x50
Max Pooling	2x2
Fully Connected + ReLU	100
Softmax	10

Table 6: Model architecture for CIFAR-10 and CINIC.

Layer Type	Size
Convolution + ReLU	3x3x32
Max Pooling	2x2
Convolution + ReLU	3x3x64
Max Pooling	2x2
Fully Connected + ReLU	512
Softmax	10

- **False Negative Rate (FNR)** is the fraction of the malicious clients that are wrongly classified as benign to the total number of malicious clients.

- **Averaged Aggregated Rounds for Malicious Clients (AAR)** represents the average number of total rounds in which a malicious client remains undetected and thus aggregated.

- **Testing Accuracy (TACC)** is the fraction of correctly classified cases in the testing set to the total testing cases of the final global model.

- **Attack Success Rate (ASR)** is the ratio of instances that, upon addition of the trigger, are successfully classified into the specified category (excluding those instances in the original images already belonging to that category), to the total number of test cases of the final global model.

A.5. Configurations

Every client participates in each training round, and malicious clients conduct attacks in each training round. The model architecture is 2-layer Convolutional Neural Network (CNN) with architecture specified in Table 5 and Table 6 for FashionMNIST and CIFAR-10/CINIC, respectively. For local training, we adopt a batch size of 32. Stochastic Gradient Descent (SGD) is our optimization technique, using a learning rate of $1e-3$. The specific configuration for learning rates and initialization is detailed in Table 7.

Table 7: Setting of FedREDefense for different datasets. Learning Rate is denoted as LR.

	Fashion MNIST	CIFAR	CINIC
LR of Images in $\mathcal{K}_i$	5e-1	1e-1	1e-1
LR of Labels in $\mathcal{K}_i$	2e-1	5e-2	5e-2
Virtual Model LR	1e-1	5e-2	5e-2
LR of Virtual Model LR	5e-6	5e-5	5e-5
Initial Round of M	800	500	2000
Upper Bound of M	2000	1500	4000

B. Addtional Experimental Results

B.1. Detection Performance of FLAME and FLTrust

To better investigate some defenses that do not output labels of clients after training but also perform filtering in each round (i.e., FLAME and FLTrust), we show their averaged detection metrics of DACC, FPR, and FNR along all the training rounds, denoted as ADACC, AFPR, and AFNR, in Table 8. The results demonstrate that both FLTrust and FLAME typically produce a relatively large AFPR across training rounds suggesting that some genuine clients are misclassified as malicious, which inevitably hampers the global model's performance. Also, this inaccurate detection makes them not applicable in marking and removing the clients identified as malicious, but have to let them continue participating in the training.

B.2. Impact of FL Settings

Impact of local training epochs: We evaluate the impact of local training epochs on defensive performance, as demonstrated in Figure 4. We consider 1, 2, and 5 local epochs, with the corresponding communication rounds set to 500, 250, and 100, respectively. For our method, the upper bound of optimization iteration is proportionally adjusted, being two and five times the default value. For our approach, we observe that an increase in local epochs does not impact the discriminative ability. Notwithstanding the increased number of local epochs, we can still achieve a 100% DACC. This observation is consistent with previous research presented in (Pi et al., 2023), which indicates that larger local epochs do not impede the distilled knowledge's capacity to emulate genuine data training. On the contrary, the model updates from malicious clients remain difficult to reconstruct, regardless of the local epoch number. Concurrently, the task becomes more challenging for FLDetector with an increase in local epochs. The underlying reason is that FLDetector relies on hessian-based predictions, and its accuracy diminishes as local epochs increase. Similarly, for

Table 8: ADACC (%) ↑, AFPR (%) ↓, AFNR (%) ↓, and ARR ↓ of FLAME and FLDetector for different attacks on FashionMNIST, CIFAR-10, and CINIC datasets. ↑ (↓) indicates the larger (smaller) the better metric.

Attack	Detector	FashionMNIST				CIFAR-10				CINIC			
		ADACC	AFPR	AFNR	AAR	ADACC	AFPR	AFNR	AAR	ADACC	AFPR	AFNR	AAR
Fang	FLAME	79.34	28.69	0.00	0.00	81.09	26.27	0.00	0.00	79.46	28.53	0.00	0.00
	FLTrust	87.11	17.90	0.00	0.00	90.36	13.39	0.00	0.00	91.32	12.06	0.00	0.00
LIE	FLAME	27.58	63.57	95.20	476.00	73.09	28.98	21.60	108.00	78.59	27.87	4.80	24.00
	FLTrust	57.70	20.79	97.60	488.00	56.52	22.98	96.20	482.00	57.75	21.73	95.00	475.00
Min-Max	FLAME	80.33	27.23	0.00	0.00	79.50	28.48	0.00	0.00	79.92	27.88	0.00	0.00
	FLTrust	87.03	18.01	67.40	337.00	66.15	20.03	69.40	347.00	63.68	19.88	78.60	393.00
Min-Sum	FLAME	29.04	62.08	93.80	469.00	73.81	29.06	18.80	94.00	79.99	27.80	0.00	0.00
	FLTrust	57.93	20.71	97.00	485.00	58.24	22.93	90.20	451.00	58.58	21.83	91.80	459.00
MPAF	FLAME	79.87	27.96	0.00	0.00	80.14	27.58	0.00	0.00	79.87	27.96	0.00	0.00
	FLTrust	87.03	18.01	0.00	0.00	90.14	13.70	0.00	0.00	90.74	12.86	0.00	0.00
DBA	FLAME	79.87	27.95	0.00	0.00	80.72	26.78	0.00	0.00	80.02	27.75	0.00	0.00
	FLTrust	65.54	15.30	83.73	418.64	66.75	12.28	87.15	435.75	67.17	11.54	87.59	437.93
Scaling	FLAME	79.84	28.00	0.00	0.00	80.74	26.75	0.00	0.00	79.84	28.00	0.00	0.00
	FLTrust	64.92	15.23	86.12	430.61	67.06	10.54	90.52	452.61	67.41	10.48	89.44	447.21

FLTrust, more local epochs introduce greater uncertainty. Given the inherent non-IIDness, more extensive local training amplifies the variability in local model updates.

Impact of participation rate: We investigate the impact of the participation rate, as shown in Figure 5. We considered participation rates of 0.1, 0.5, and 1. For our method, the upper bound of optimization iteration is also proportionally adjusted due to a larger interval between subsequent participation of an individual client. A participation rate of less than 1 implies that not all malicious clients would be involved in the initial round. Note that FLDectector is not compared because its method initially assumes the clients participate every round so that the estimation can be done upon the model update of the last round. Thus it is not directly applicable in partial participation. Since our technique does not rely on cross-client information, it is still effective irrespective of the participation rates. We also observe that FLAME, which utilizes cross-client information, demonstrates a decline in performance in some scenarios when the participation rate decreases. This can be attributed to the reduced number of clients available for clustering, which increases uncertainty.

B.3. Impact of FedREDefense Variants

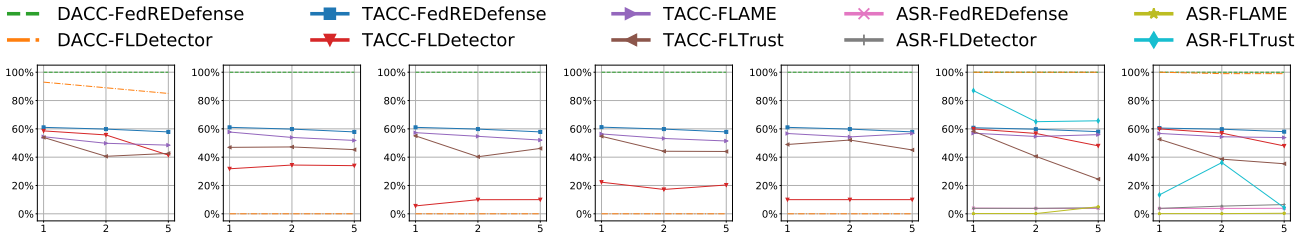
In the whole training process, FedREDefense maintains and refines the distilled local knowledge for each client. In this section, we delve deeper into how the chosen optimization schemes influence the number of necessary optimization iterations required, which reflects the overall efficiency of different FedREDefense variants. We experiment with several variants of optimization as follows:

- **Continuous optimization:** For a client's initial participation, we conduct M distillation iterations. Subsequent participations involve a quicker M'-iteration refinement. In practice, we set $M = 500$ and $M' = 60$.
- **Intermittent optimization:** For a client's initial participation, M rounds of distillation iterations are performed. Thereafter, every T_I communication rounds, a M'-iteration refinement is executed. In the other rounds, no optimization is performed; only inference based on existing local knowledge is used to calculate reconstruction errors. Practically, we set $M = 500$, $T_I = 10$, and $M' = 500$.
- **Dynamic optimization:** As specified in Section 4.1, in every round and for each client i the optimization stops once the reconstruction error e_i^t gets below the threshold γ. In practice, we set the upper bound of M for the optimization as 1500, if the reconstruction error does not reach $\gamma = 0.6$.

Note that empirically, we tune hyperparameters for the three variants to ensure that all of them can achieve 100% DACC for all attacks, thus we focus on measuring the efficiency of the three variants with the following metric:

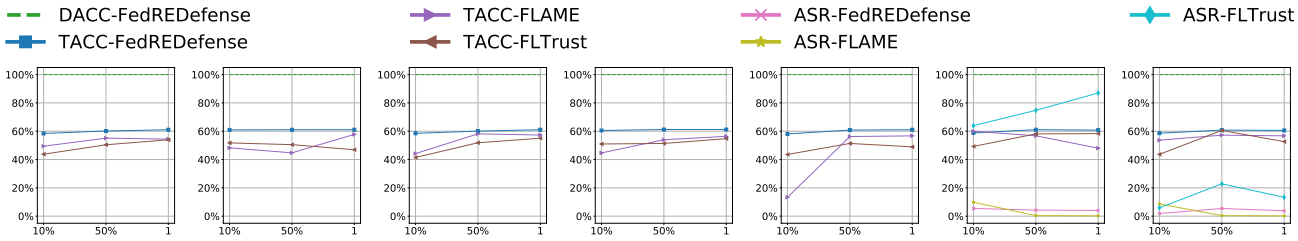
Average Optimization Iterations (AOI): This metric represents the average optimization iterations in the outer loop, denoted as M, required for an individual client per round across the entire training process. It provides insights into the average number of iterations mandated per round to adeptly discern model poisoning attacks using FedREDefense.

Table 9 shows the comparison of AOI for the three vari-



(a) Fang (b) LIE (c) Min-Max (d) Min-Sum (e) MPAF (f) Scaling (g) DBA

Figure 4: DACC of FedREDefense and FLDetector, and TACC(/ASR) of the global model for all defenses as a function of the local training epochs under different defenses and attacks.



(a) Fang (b) LIE (c) Min-Max (d) Min-Sum (e) MPAF (f) Scaling (g) DBA

Figure 5: DACC of FedREDefense and TACC(/ASR) of the global model for all defenses as a function of participation rate under different defenses and attacks.

Table 9: Averaged Optimization Iterations (AOI) for a client per round of FedREDefense under Fang attack on CIFAR-10 dataset.

	Continuous	Intermittent	Dynamic
AOI	60.88	50.00	2.40

ants of FedREDefense. Note that the experiment follows the default setting elaborated in Section 5.1.4. As shown in Section 5.2, for all the studied model poisoning attacks, FedREDefense can detect the malicious clients once they participate. Thus, we only experiment on Fang attack for comparing the AOI. The result reveals that dynamic optimization markedly curtails the number of optimization iterations for each client, thereby bolstering efficiency substantially. In fact, after the initial participation of a benign client, the refinement of distilled local knowledge in dynamic optimization generally requires only one or two iterations. Overall, attaining 100% DACC in dynamic optimization necessitates only a small fraction, on the order of a few percent, of the optimization iterations typically required for fixed rounds. It is because optimizing with fixed iteration requires a large number of iterations to adapt to *the most latency-prone client* in all communication rounds to prevent mistaking benign clients as malicious. However, such redundant optimization is not indispensable for the detection process as we only need to know whether a model update can be reconstructed with an error smaller than the threshold. Therefore, the crux lies in optimizing with minimal computational overhead to guarantee precise detection adaptively for every single client. The proposed dynamic optimization achieves this by early stopping once the threshold is reached.

Table 10: Distribution of reconstruction errors (mean $\pm$ standard deviation) in different scenarios.

	FashionMNIST	CIFAR-10	CINIC
Benign	0.36±0.03	0.18±0.02	0.31±0.03
Fang	0.97	0.97	0.99
MinMax	0.99	0.99	1.00
MinSum	0.98	0.97	0.97
DBA	0.74±0.02	0.82±0.02	0.81±0.02
Scaling	0.74±0.03	0.83±0.02	0.82±0.02

B.4. Distribution of Reconstruction Error

In Table 10, we show the mean $\pm$ standard deviation of the reconstruction errors with different datasets and attacks, where for Fang, MinMax, and MinSum, only mean is reported because of negligible standard deviation $< 1e-3$. We note that the reconstruction error is clearly split for benign and malicious clients. Therefore, we can pick an intermediate value as a threshold.