Sine: Similarity is not enough for mitigating Local Model Poisoning Attacks in Federated Learning

Harsh Kasyap, *Student Member, IEEE,* and Somanath Tripathy*, *Senior Member, IEEE*

Abstract—Federated learning is a collaborative learning paradigm that brings the model to the edge for training over the participants' local data under the orchestration of a trusted server. Though this paradigm protects data privacy, the aggregator has no control over the local data or model at the edge. So, malicious participants could perturb their locally held data or model to post an insidious update, degrading global model accuracy. Recent Byzantine-robust aggregation rules could defend against data poisoning attacks. Also, model poisoning attacks have become more ingenious and adaptive to the existing defenses. But these attacks are crafted against specific aggregation rules. This work presents a generic model poisoning attack framework named *Sine* (Similarity is not enough), which harnesses vulnerabilities in cosine similarity to increase the impact of poisoning attacks by 20-30%. *Sine* makes convergence unachievable by maintaining the persistence of the attack. Further, we propose an effective defense technique called *FLTC* (FL Trusted Coordinates) to avoid such issues. *FLTC* selects the trusted coordinates and aggregates them based on the change in their direction and magnitude with respect to a trusted base model update. *FLTC* could successfully defend against poisoning attacks, including adaptive model poisoning attacks, by restricting the attack impact to 2-4%.

Index Terms—Federated Learning, Local Model Poisoning Attack, Hyper-Spherical Direction Cosine, Byzantine-Robust Aggregation.

1 Introduction

Federated learning (FL) has evolved as a promising solution for distributed deep learning [1]. It involves the participation of multiple client devices distributed across the demographics. The devices are diverse, from high-performing computers to IoT devices. FL makes the model training to be intelligent by learning over contemporary data. It has been used for text, emoji, and action prediction [2]. Besides that, it is used in more sensitive applications like finance [3] and healthcare [4].

FL proves to be privacy-preserving by bringing the model to the data and facilitating local model training at the edge devices. It provides autonomy and independence to the participants. On the other hand, it has no control over its local data and training process, causing poisoning attacks [5], [6], [7]. Broadly, poisoning attacks are classified as targeted and untargeted poisoning attacks. The targeted poisoning attacks [8], [9] aim to misclassify the test samples to a specific class label, while the untargeted poisoning attacks [10], [11] degrade the global model prediction indiscriminately of the test samples.

Further, poisoning attacks are classified as data poisoning and model poisoning attacks. Data poisoning attacks are generally carried out by manipulating locally held data, flipping the labels, or inserting a trojan. In local model poisoning attacks, the adversaries neither change the locally held data nor the training plan received from the aggregator. Instead, they add a perturbation or noise vector to the locally trained parameters to send erroneous updates to the aggregator. Such attacks gain comparatively more attack success. The recent Byzantine-robust defenses [12], [13], [14], [15] are proposed to mitigate the poisoning attacks. These defenses find outliers based on the statistical variance in the model updates induced by the poisoned data. However, there exists an attack against every defense, or it is said that the best defense is a good offense. This motivates us to design a more insidious poisoning attack that can evade existing Byzantine-robust aggregation schemes.

The existing local model poisoning attacks [16], [17], [18], [19] are based on different capabilities and knowledge of the adversaries. They are either tailored for any specific aggregation scheme or have less impact with impractical assumptions like knowledge of all the benign updates. These attacks are crafted as an optimization problem, making it difficult to find the solution in a large search space.

This paper presents a generic attack framework *Sine* (*Similarity is not enough*), which could be able to degrade the model performance drastically against similarity measures (Euclidean or cosine) based Byzantine-robust aggregation schemes. The adversary crafts local poisoned models without knowing the aggregation rule, with full and partial capabilities. Also, a unique and practical attack capability is introduced, where the attacker crafts malicious updates without the knowledge of other (malicious) participants, ensuring fair attack success. *Sine* exploits the vulnerabilities present in cosine similarity to obtain similar-looking vectors, which can exhibit malicious behavior.

The data used to train the local model come from various sources, including different devices, locations, and users. This heterogeneity makes detecting and preventing poisoning attacks challenging because the data may have different distributions, quality, and characteristics. On the top, existing Byzantine-robust defenses suffer from adaptive attacks, like *Sine*. Therefore, this work proposes a new Byzantine-robust aggregation scheme *FLTC (FL Trusted Coordinates)* to defend against such attacks. *FLTC* uses the concepts of Cosine similarity [12] and Trim [13] for assigning trust scores

• *Authors are associated with Indian Institute of Technology Patna, India. E-mail: {harsh_1921cs01,som}@iitp.ac.in*

**Corresponding author: Somanath Tripathy*

to each participant. It calculates the similarity using the direction cosine and trims the untrusted coordinates in the opposite direction. It is evaluated for data poisoning and local model poisoning attacks, including adaptive model poisoning attacks.

Our key contributions are summarized as follows:

- This work proposes a generalized local model poisoning attack framework named *Sine*, which perturbs the local model parameters and fools existing similarity-based Byzantine-robust aggregation rules.
- We perform extensive experiments with different datasets, learning classifiers, attack settings, attacker's capabilities, and knowledge. The results provide a shred of evidence for the proposed conjecture.
- The efficiency and efficacy of *Sine* are analyzed and compared with the state-of-the-art local model poisoning attacks against the existing baseline defenses.
- Further, a new defense mechanism named *FLTC* has been proposed. It assigns trust scores and trims the malicious coordinates, to select the best-performing coordinates. A strong adaptive attack is framed against *FLTC*. Experimental results and theoretical analysis provide guarantees for robust aggregation.

The remainder of this paper is organized as follows. Section 2 discusses the background, existing defenses, and local model poisoning attacks. Section 3 discusses the capabilities and knowledge of an adversary. Section 4 presents a generic local model poisoning attack framework. Section 5 describes the simulation setup, experiment, and results. Section 6 presents a Byzantine-robust defense. Section 7 concludes this work.

2 Background and Related Work

2.1 Federated Learning

Federated learning (FL) is a collaborative learning approach that provisions a joint model training with multiple edge devices [20]. FL aims to ensure the security and privacy of the participants by restricting the movement of local data and bringing the model to the edge. Let there be n participants owning their data D_i (for each participant $i \in n$). The central aggregator $\mathcal{A}$ broadcasts the global model w^t to the participants for local training in iteration t. Further, each participant trains locally as stated in Equation 1.

$$w_i^{t+1} \leftarrow w_i^t - \eta \nabla l(w_i^t; b \in D_i), \tag{1}$$

where η is the local learning rate of clients and b is a batch from local dataset D_i. After completing the local training at the edge, participants send their local updates ∇w_i to $\mathcal{A}$. Further, $\mathcal{A}$ collects all the updates to get the plan w^{t+1} for the next iteration. It uses different aggregation algorithms. FedAvg [1] is the most common and popular among them. It does weighted averaging of the shared updates (gradient) as stated in Equation 2.

$$w^{t+1} \leftarrow w^t + \sum_{i=1}^{n} \frac{n_i}{\sum_{j=1}^{n} n_j} \nabla w_i^{t+1}, \tag{2}$$

where n_i is the number of private data samples of client i. The whole process runs for several iterations until convergence.

2.2 Byzantine-Robust Aggregation Schemes

Federated Learning suffers due to the presence of malicious participants in adversarial settings. Therefore, different strategies have been proposed for aggregation, assuming the presence of both benign and malign participants [5], [6], [12], [13], [14]. The state-of-the-art Byzantine-robust aggregation schemes are discussed below.

Krum and Multi-Krum: Blanchard et al. [5] proposed Krum (aggregation rule), which selects one participant's update as the global model. It uses Euclidean distance as the similarity measure. If the aggregator ($\mathcal{A}$) receives n updates, and assumes the presence of m malicious participants, $\mathcal{A}$ finds the Euclidean distance of models from i^{th} participant (w_i) to $n - m - 2$ participants. Further, it computes the sum of the squared distance between w_i and the closest $n - m - 2$ updates. Then, it selects the model w_i with the minimum sum of squared distances. In the same work, they proposed Multi-Krum, a variant of Krum, where $\mathcal{A}$ selects best performing participants on an equal basis and averages them. Multi-Krum claims to achieve a theoretical guarantee of convergence, provided $m < \frac{n-2}{2}$. However, Krum has been studied for different local model poisoning attacks and found vulnerable to [7], [17], [18].

Median and Trimmed-Mean: Yin et al. [13] proposed two robust aggregation algorithms (Median and Trimmed-Mean) operating over each coordinate. Median calculates the next model parameter for each coordinate $k \in [d]$ as $g^k = med(x_i^k : i \in [n])$. In a system with n participants and a maximum of $\beta \in [0, \frac{1}{2})$ fraction of malicious participants, Trimmed-Mean calculates the model parameter for each coordinate $k \in [d]$ as stated in Equation 3.

$$g^k = \frac{1}{(1-2\beta)n} \sum_{x_i^k \in S^k} x_i^k, \tag{3}$$

where S^k is obtained by removing the largest and smallest β fraction from the sorted array of all participants $i \in [n]$. It claims to achieve an order-optimal error rate for $m \leq \beta < \frac{n}{2}$. Authors in [16], [17] crafted a local model poisoning attack against Median and Trimmed-Mean based systems. Authors in [18] also proposed an aggregation agnostic attack strategy to impact their performance.

Divide-and-Conquer (DnC): DnC [18] uses singular value decomposition (SVD) based spectral methods for identifying and removing outliers. It constructs a subsampled set $\bar{\nabla}$ of gradients based on a sorted set r of indices that are less than the dimension d of its input gradients. A centered subsampled set ∇^c of $\tilde{\nabla}$ is computed using each dimension's mean μ. Then it calculates the projection of centered gradients to their top right singular eigenvector v. Finally, a vector of outlier scores s is calculated for the removal of gradients with the highest scores. It repeats for n iterations. Finally, DnC aggregates the common gradients from the good sets of n iterations. Though DnC performs better for small datasets/models, its performance degrades for complex models. It requires very high memory and terminates in the presence of more than 40% attackers.

Romoa: Romoa [14] is a **Ro**bust **Mo**del **A**ggregation strategy based hybrid similarity measurements and a lookahead strategy. Romoa uses element-wise cosines to measure similarities between the participant's parameters and the

parameters from the last synced state. Then, it examines divergences in unselected parameters based on cosine similarity and Pearson correlation. A sanitizing factor with momentum is calculated, which punishes the adversaries and preserves the voting rights of honest participants. The sanitizing factor is determined by participants' previous behaviors and temporal similarity measurements.

Romoa alters in traditional federated learning settings and imposes overheads on participants. According to the results provided, the performance of Romoa degrades for large datasets and complex models. Against targeted poisoning attacks, its performance degrades drastically.

FLTrust: Cao et al. [12] proposed a Byzantine-robust aggregation algorithm named FLTrust. Using a clean and small root dataset, it bootstraps trust at the aggregator ($\mathcal{A}$). $\mathcal{A}$ trains over root dataset and has its own root update g_0. Then, it calculates a cosine score for each update g_i as $c_i = \frac{g_i.g_0}{||g_i||.||g_0||}$, where $g_i.g_0$ is the dot product between them and $||.||$ is the norm of the corresponding update. Next, the trust score is calculated as ReLU-clipped cosine similarity $i.e.$, $TS_i = ReLU(c_i)$. Considering the scaling attack, $\mathcal{A}$ normalizes every update as $\bar{g_i} = \frac{||g_0||}{||g_i||}.g_i$. Finally, $\mathcal{A}$ computes the weighted average as stated in Equation 4.

$$g = \frac{1}{\sum_{i \in n} TS_i} \sum_{i \in n} TS_i.\bar{g_i}. \tag{4}$$

FoolsGold: Fung et al. [6] proposed a defense to detect sybil-based poisoning attacks, computing the pairwise similarity between the aggregated historical updates. The intuition is that there is a higher likelihood of the directions of aggregated historical updates aligning for malicious clients to approach the poisoning objective. However, it is only good at detecting Sybil and does not achieve state-of-the-art performance (accuracy), even in normal settings. Authors also discussed adding intelligent noises (orthogonal noise vectors), to frame adaptive attacks (make the updates look different), against FoolsGold, and showed it to be resistant. Also, for another cosine similarity based defense like FLTrust, adding any piece of random noise from orthogonal decomposition will not maintain the cosine similarity with respect to the trusted update; thus, it will be easily detected.

2.3 Local Model Poisoning Attacks

The above-discussed Byzantine-robust aggregation schemes claim to converge even in the presence of malicious participants. However, local model poisoning attacks [16], [17], [18] have been framed against FL, to evade such schemes. The state-of-the-art attacks are discussed below.

LIE: Little is Enough: Baruch et al. [16] proposed an attack paradigm to backdoor in the model, using the inherent statistical vulnerabilities. The attacker exploits the empirical variance between participants' updates and the identical distribution of different parameters of the participants. The model parameters of the honest participants are centered around the mean. Some updates at the extremes act as supporters, while some act as opposers in the opposite direction. This attack computes the malicious model parameters between mean and supporters such that they will be closer to the mean than opposers. It shifts the mean and chooses the Byzantine updates. The unit of change is the standard deviation. They also defined a range to deviate, $i.e.$, $(\mu - z^{max}\sigma, \mu + z^{max}\sigma)$. The adversary calculates z^{max} using Cumulative Standard Normal Function $\phi(z)$, such that s supporters remain far from the mean and the Byzantine updates get selected.

Fang attack: Fang et al. [17] formulated the local model poisoning attack as an optimization problem. The intuition behind the attack is that the asymptotically bounded Byzantine-robust aggregation schemes do not precisely tell about the model's performance. The error rates ignore constants, which significantly influence the model's behavior. They framed the attack with different capabilities of the attacker, with and without the knowledge of aggregation rules and benign updates. However, they assume that all the m compromised participants collude to solve the attack optimization problem. The optimization problem is defined as a directed deviation goal, to deviate the global model parameters, as maximum in the opposite direction, $i.e.$, $\max_{\mathbf{w}'_1,\cdots,\mathbf{w}'_m} \mathbf{s}^T \left(\mathbf{w} - \mathbf{w}'\right)$, where s^T is the transpose of the vector with changing directions for all global model parameters, w is the before-attack global model and w' is an after-attack global model. Further, they restrict the deviation of w'_i, for each attacker $i \in m$, as $w_{re} - \lambda s$ where w_{re} is the global model of the previous iteration. Finally, the upper bound of λ is calculated using a binary search to get a malicious model.

Authors in [21] proposed another model poisoning attack based on fake clients called **MPAF**. The final fake local model update is calculated as $\lambda\left(\boldsymbol{w'} - \boldsymbol{w^t}\right)$, where $\boldsymbol{w'}$ is a base model selected by the attacker, with low test accuracy. Then, the attacker chooses a large λ to increase the attack impact. However, Fang and MPAF attacks are based on a similar principle, and the assumption of attacker capabilities of MAPF is different from baseline attacks. Thus, we consider Fang as a baseline attack for comparison.

Min-Max and Min-Sum attacks: Shejwalkar et al. [18] presented a general model poisoning framework. They solved the optimization problem for maximal perturbation from the benign updates. They crafted the malign updates as $(\nabla w_b + \lambda \nabla p)$ where ∇w_b is the average of benign updates. They fixed the type of adversarial perturbation (∇p) and searched for an optimal coefficient (λ). They introduced different perturbation vectors as unit vector $(\nabla p^{uv} = -\frac{\nabla b}{||\nabla b||_2})$, inverse standard deviation $(\nabla p^{std} = -std(\nabla b))$ and inverse sign $(\nabla p^{sgn} = -sign(\nabla b))$. Further, they proposed two aggregation-agnostic attack strategies. First, Minimize-Maximum (Min-Max) distance attack strategy computes the malicious update such that its maximum distance from any other update is upper bounded by the maximum distance between any two benign updates. Second, Minimize-Sum (Min-Sum) distance attack strategy computes the malicious update such that the sum of squared distances of the malicious update from all the benign updates is upper bounded by the sum of squared distances of any benign update from the other benign updates. Finally, they solve λ by first scaling down from a large value, followed by scaling up with a reduced step size.

Table 1 compares the proposed attack with the state-of-the-art local model poisoning attacks. The existing works

TABLE 1: Comparison between model poisoning attacks.

Attack	Det.	non-IID	Agg. Agnostic	Adversary Capabilities	Computation
LIE	✗	✗	✗	Full	Low
Fang	✗	✓	✗	Full and Partial	Low
Min-Max & Min-Sum	✗	✓	✓	Full and Partial	High
Sine	✓	✓	✓	Full, Partial and Individual	Medium

Det. → Deterministic, Agg. → Aggregation rule

solve an optimization problem to find the malicious model updates, while *Sine* aims to design a deterministic aggregation agnostic attack. This ensures the effectiveness in both the attack impact and computational cost. Further, this is the first attack to introduce attackers with individual capabilities, where the attacker could find a malicious update without knowing any other update in the system.

3 Threat Model

We consider an FL system with n participants, out of which m are compromised. The participants train locally using data $D(x_k, y_k)_{k \in n_i^K}$, for each participant i holding n_i^K samples. The compromised participants aim to degrade the global model performance by framing a local model poisoning attack. Now, we discuss the attacker's goal, knowledge, and capabilities.

Attacker's goal: The target of the attackers is to impact the global model performance, indiscriminately of the test samples. Such attacks are called untargeted attacks, which can be achieved by framing local model poisoning attacks.

Attacker's knowledge: Like other participants, the attackers also know the training plan sent by the central trusted server. There can be two cases: (1) AK: Aggregation rule is known and (2) AN: Aggregation rule is not known, where aggregation rule can be any Byzantine-robust defense. This work considers a pragmatic scenario AN, where the attacker does not know the aggregation rule. We also consider a case tailored to FLTrust, $i.e.$, AK-BSU, where the attacker knows the base server update, $i.e.$, a trusted root update used by the FLTrust aggregator.

Attacker's capabilities: The attackers train locally without disturbing the training cycle and manipulating the local data. They can only make changes to the locally trained model parameters, based on their knowledge and capabilities. The capabilities of an attacker are categorized below.

- *Full capability (FC):* The attacker has the complete knowledge of local model updates of n participants.
- *Partial capability (PC):* The attacker has the knowledge of local model updates of m malicious participants.
- *Individual capability (IC):* The attacker has no knowledge of the local model update of other participants.

To frame the attack with full and partial capabilities, the benign updates of the participants need to be shared in order to find the malicious update [17]. To reduce the communication overhead, a compromised (participants) leader (CL) receives local model updates from the (compromised) participants and frames the local model poisoning attack with full and partial capabilities, while with individual attack capability, the adversary acts independently and calculates the malicious update only using its own benign update. To avoid confusion, we must understand that CL is like any other compromised participant regarding ability and resources and does not need out-of-band communication and extra working phases; rather, it reduces $m \times m$ communications to $m \times 1$. It does not enhance the adversary's capabilities compared to baseline attacks.

4 *Sine:* Our Local Model Poisoning Attack

The proposed local model poisoning attack *Sine* (Similarity is not enough) focuses primarily on an aggregation-agnostic attack strategy. This section discusses the attacker's objective and assumptions, intuitions and correctness, and the procedure to be carried out for a successful attack. Table 2 summarizes the main notations.

TABLE 2: Notations (N).

N	Description	N	Description
∇w_b	benign model update	γ_c	cosine scaling factor
$\nabla_b(B)$	avg benign model update	γ_n	norm scaling factor
$\nabla_m(M)$	malign model update	ϵ	random perturbation noise
$\nabla_{CL}(C)$	compromised leader update	λ	degree of heterogeneity
∇_{BS}	base server (trusted) update	i	denotes participant
$\mathcal{F}$	attack objective function	k	represents coordinate
cs_{mb}	cosine similarity between ∇_m and ∇_b	φ_i^k	trusted coordinate score for k^{th} dimension of i^{th} client

Generic attack objective in FL: The attacker aims to generate a malicious model update (∇_m) and fool the aggregator for selecting it. ∇_m can be expressed as a function of benign model update (∇w_b), $i.e.$, $\nabla_m = \mathcal{F}(\nabla w_b)$. The existing attacks model this function as an optimization problem $\nabla_m = \nabla w_b + \gamma \nabla p$ and solve it for γ with a known perturbation vector ∇p. It is computationally infeasible to search for a solution of the arbitrary variable γ in a large search space. On the other hand, *Sine* uses a deterministic strategy to search for ∇_m. The compromised leader (CL) compromises m participants. Each compromised participant trains on their locally held benign data and shares their benign updates (∇w_b) to CL. Then, CL averages the received benign updates (∇w_b) for each participant ($i \in m$) to get the averaged benign update, $i.e.$, ∇_b. Further, CL trains locally over a small and clean dataset to produce a compromised leader update, $i.e.$, ∇_{CL}, treated as a trusted benign update.

With this information, CL updates the attack objective function as $\mathcal{F}(\nabla_b, \nabla_{CL})$. Then, CL computes the malicious model update (∇_m) by solving $\mathcal{F}$ and broadcasts it to m participants. The compromised participants submit the malicious update ∇_m to the trusted server for aggregation, while the benign participants submit their benign local model updates. The compromised participants expect the aggregator to consider their update as a benign update. CL performs these steps with different attack capabilities, as discussed in the threat model. Then *Sine* is executed to find the malicious update (∇_m) with the knowledge of averaged benign update (∇_b) and the compromised leader update (∇_{CL}). CL executes *Sine* in common for all the attackers

with full and partial capabilities, while each attacker executes *Sine* locally with the individual capability.

Correctness of the attack: Given two vectors $\vec{A}$ and $\vec{B}$, we can find another vector $\vec{C}$, such that $\vec{C}$ is cosine similar with $\vec{A}$ as much $\vec{B}$ is to $\vec{A}$, while $\vec{C}$ is dissimilar to $\vec{B}$. Cosine similarity between $\vec{A}$ and $\vec{B}$ is defined as $\frac{\vec{A}.\vec{B}}{||A||.||B||}$. Thus, for given vectors $\vec{A}$ and $\vec{B}$, there exist many vectors $\vec{C}$, which would behave maliciously. Though $\vec{C}$ is a cosine similar to $\vec{A}$, while highly dissimilar to $\vec{B}$, as it is pointing in the direction of the mirror image of $\vec{B}$ in reference to $\vec{A}$ as shown in Figure 1. So, there exist malicious updates (∇_m) closer to ∇_{CL} and far from ∇_b, which would help to collect supporters for the malicious updates and opposers for the other benign updates.

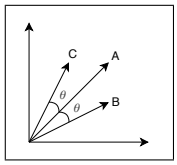


Fig. 1: Motivation.

Formally, Theorem 1 suffices to show the correctness of the proposed attack *Sine*.

Lemma 1. Given the averaged benign model update (∇_b) and the compromised leader update (∇_{CL}), there exists multiple vectors ∇_m such that, $cos(\nabla_{CL}, \nabla_b) = cos(\nabla_{CL}, \nabla_m)$.

Proof. See Appendix A. □

Theorem 1. There exists multiple vectors ∇_m (malicious model updates), such that ∇_m can have different cosine relations with ∇_b (averaged benign model update) while maintaining the same cosine relation with ∇_{CL} (compromised leader update).

Proof. For brevity, it is shown for three dimensions and can be generalized for n dimensions.

From Lemma 1, it is clear that there exist multiple vectors ∇_m, each of which makes an angle θ with ∇_{CL}. Let ∇_m rotates on the conic path of the cone, with apex as zero coordinates, and ∇_{CL} be the axis of the cone. Now ∇_m can make a minimum of zero and a maximum angle of 2θ to ∇_b (as illustrated in Figure 10 of Appendix A).

Let ∇_m is initialized as ∇_b. Now, it can be solved as a quadratic equation to get another ∇_m, for every single coordinate k. ∇_m moves from zero to 2θ for every solution. To ensure the movement in the same direction, there is an imposed condition, such that $cos(\nabla_b, \nabla_m^k) < cos(\nabla_b, \nabla_m^{k-1})$, where ∇_m^k is the vector after flipping the k^{th} coordinate, and ∇m^{k-1} is the last known ∇_m.

Thus, ∇_m has no fixed cosine relation with ∇_b, and it has increased distance to ∇_b. □

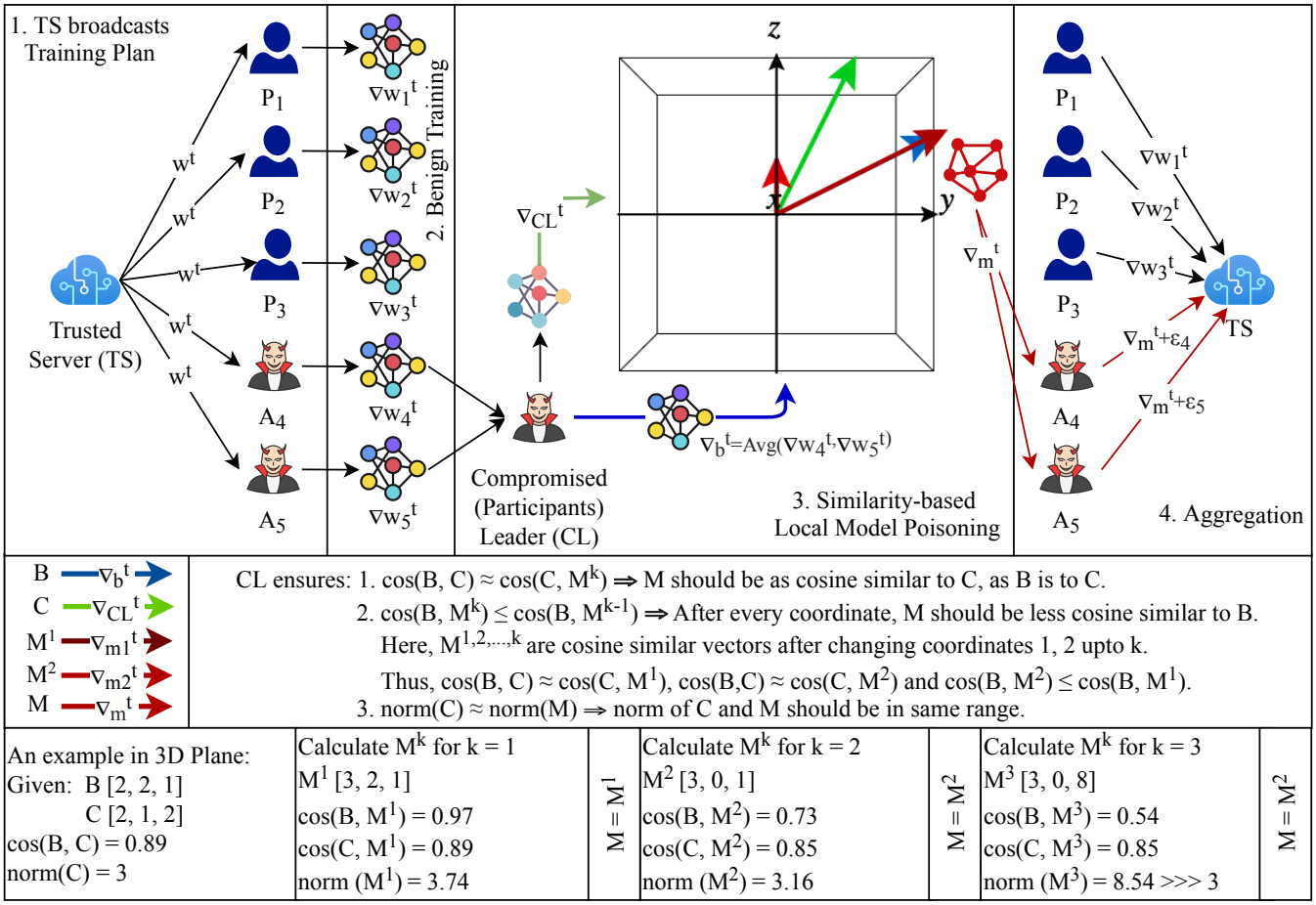
Attack Procedure: *Sine* finds the malicious model update (∇_m^t), referenced as M for a given round t. The attack is explained through an example in Figure 2. In step 1, the trusted server broadcasts w^t to each participant. The participants (including compromised peers) perform benign model training in step 2. Step 3 illustrates the attack process in detail. Let CL compromises participants A_4 and A_5 and receives their benign updates ∇w_4^t and ∇w_5^t for round t. It averages them to get ∇_b^t, referenced as B. It trains locally with the global model w^t and gets its own local update ∇_{CL}^t, referenced as C. Given B and C, CL has to find a malicious vector M.

Based on Lemma 1, we can find multiple malicious vectors (M) in 3D or high-dimensional space. However, to make this problem deterministic, we follow the approach discussed in Algorithm 1. *Sine* initializes M as B in line 10. It also introduces scaling factors γ_c (cosine scaling factor) and γ_n (norm scaling factor), which are discussed below. Then, it finds the coordinates of B deviating most from C and reverses sorts to find the interesting coordinates. These coordinates are important because the adversary can find a more malicious update only after a few coordinates. For each coordinate k, it calculates a similar cosine vector M, as shown in lines 15-21. First, it calculates the cosine similarity between the previous M and C, multiplies it with the cosine scaling factor γ_c, and stores it as lhs. Then, it calculates the rhs expression with an unknown variable for coordinate k of previous M and solves for (lhs==rhs). It results in two values, one is the k^{th} coordinate of previous M, and the other is selected as the updated k^{th} coordinate of M iff, $cos(B, M^k) < cos(B, M^{k-1})$. As a result, after flipping every k^{th} coordinate, M becomes more dissimilar to B, than the previously calculated M. It should also satisfy $||C|| \approx ||M||$, *i.e.*, norm of M is in the range of $[\frac{1}{\gamma_n} \times ||p||, \gamma_n \times ||C||]$.

In the given example of Figure 2, CL first flips the first coordinate of B and gets M^1. It can be seen that M^1 is as cosine similar to C as B to C. In addition, CL checks for norm similarity of M with C and finds it to be closer. Again, it checks for the dissimilarity of M^1 with B and accepts it as M. Then, CL flips the second coordinate to find M^2, satisfying all the conditions, and accepts M^2 as M. Finally,

Algorithm 1 Local Model Poisoning Attack $Sine(\nabla_b, \nabla_{CL})$

1: **function** SIMILARCOSINEVECTOR($\nabla_m, \nabla_{CL}, \gamma_c, k$)
2: $lhs \leftarrow \gamma_c \times \cos(\nabla_m, \nabla_{CL})$
3: $org_k \leftarrow \nabla_m^k$
4: $\nabla_m^k \leftarrow$ Unknown variable
5: $rhs \leftarrow \cos(\nabla_m, \nabla_{CL})$
6: Solve for $(lhs == rhs) \rightarrow [org_k, org_k']$ ▷ `It comes to solving a quadratic equation.`
7: $\nabla_m^k \leftarrow org_k'$
8: return ∇_m
9: $\nabla_m \leftarrow \nabla_b$ ▷ `Set Adversary update as a benign update.`
10: $cs_{mb} \leftarrow 1$ ▷ `Set maximal cosine similarity between adversarial and benign update as 1.`
11: $\gamma_c \leftarrow$ Initialize cosine scaling factor
12: $\gamma_n \leftarrow$ Initialize norm scaling factor
13: $imp_k \leftarrow IndicesOf(ReverseSort(|\nabla_b - \nabla_{CL}|))$ ▷ `Absolute difference between the benign and compromised leader update for finding the most interesting coordinates.`
14: **for** each dimension $k \in imp_k$ **do**
15: $\nabla_m^k \leftarrow$ SimilarCosineVector($\nabla_m, \nabla_{CL}, \gamma_c, k$) ▷ k^{th} `coordinate is recently modified to find a similar cosine vector.`
16: $cs_{mb}^k \leftarrow \cos(\nabla_m^k, \nabla_b)$
17: **if** $cs_{mb}^k < cs_{mb}$ **then**
18: **if** $||\nabla_{CL}|| \times \frac{1}{\gamma_n} < ||\nabla_m^k|| < ||\nabla_{CL}|| \times \gamma_n$ **then** ▷ `Malicious update to be cosine dissimilar to benign update and magnitude in the range of compromised leader update.`
19: $cs_{mb} \leftarrow cs_{mb}^k$
20: $\nabla_m \leftarrow \nabla_m^k$
21: return ∇_m



进行聚合，带有恶意客户端的梯度更新。

Fig. 2: Illustration of proposed local model poisoning (*Sine*) attack.

CL flips the third coordinate and finds it violating condition 3 for norm similarity with C. Hence, it discards M^3 and settles with the previous M. CL follows the same steps in n dimensions and finds a malicious vector (M) that looks very similar to C, but highly dissimilar to B. In step 4, all the benign participants submit their original local updates, while the adversaries submit ($M + \epsilon_i$) to the trusted server for aggregation, where ϵ_i is random perturbation added to M by adversary i, to not get detected by Sybil-based poisoning defense. Only in full and partial capabilities, adversaries have to add random perturbations to M, while in individual attack capability, each attacker prepares a unique malicious update, $w.r.t.$, its own benign model.

The computational complexity is $\mathcal{O}(kd)$, as *Sine* calculates a similar cosine vector till k important dimensions, where $k << d$, for total d dimensions in a local model update. The complexity for finding cosine similarity is reduced from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$ by storing (and reusing) the calculations of the dot product and norms from the previous coordinate.

Scaling factors: In Algorithm 1, *Sine* introduces two scaling factors, γ_c as the cosine scaling factor, and γ_n as the norm scaling factor. γ_c is increased for getting high cosine similarity with the compromised leader update while maintaining dissimilarity from the benign updates. Increasing γ_c from 1 allows generating an update closer to the base update, but a much higher value of γ_c does not allow deviating more from a benign update. γ_n is used to restrict the norm of the malicious update and bind it in the range of the compromised leader update. We suggest keeping γ_n as a multiple of $10\times$-$100\times$ such that simple defenses can not detect the malicious updates. With full knowledge, the attack is successful with smaller γ_n, while we set a larger γ_n with partial and individual knowledge.

Impact of *Sine* on Euclidean-based defenses: *Sine* generates updates closer to a few benign updates, such that the benign updates act as its supporters. At the same time, it increases opposers for other benign updates. It helps to move the mean towards the malicious update, fooling the defenses to get selected. It can be noted that M does not go beyond an upper bound of the Euclidean distance from C, because the range of C bounds the norm of M. M also maintains a fixed cosine relation with C. For given B and C, the attacker calculates M. If the angle between B and C is θ, then M also makes an angle θ with C. Solving the Pythagoras's theorem with sides C and M, we get the distance between them as $\sqrt{||M||^2 + ||C||^2 - 2.||M||.||C||.cos(\theta)}$. Please refer to Appendix B for a detailed explanation.

Based on the condition, $||C|| \approx ||M||$, and the norm scaling factor (γ_n), we know that, $||M|| = \alpha.||C||$, where $\frac{1}{\gamma_n} \leq \alpha \leq \gamma_n$. Therefore, the distance between C and M can be deduced to $||C||.\sqrt{\alpha^2 + 1 - 2.\alpha.cos(\theta)}$. The benign updates also make similar relations for distance from C, which makes M look similar and get selected for aggregation. This attack works for both the point-wise and coordinate-wise

aggregation rules, as *Sine* restricts any single coordinate from deviating much to maintain the norm similarity with C. *Sine* solves M, to maintain cosine relation and norm to C, like any other benign update. However, there remains a scope to minimize α for more insidious updates.

Impact of *Sine* on cosine-based defenses: *Sine* makes a substantial attack impact on FLTrust, as it is a cosine similarity-based defense. By creating a cosine update that is similar to the trusted root update, but opposite of all other benign updates, it attacks FLTrust. It is known that FLTrust trims the updates with negative cosine similarity. However, it does not consider the case for the mirror image updates, with nearly equal and positive cosine similarity. Empirical results show that one can successfully generate malicious updates that are highly cosine similar to the trusted root update but highly dissimilar to the benign updates.

5 Attack Evaluation

Sine is evaluated against state-of-the-art (SOTA) defenses including FedAvg, Multi-Krum [5], Trimmed-Mean [13], DnC [18], Romoa [14] and FLTrust [12]. Also, *Sine* is compared with the existing state-of-the-art (SOTA) attacks like Fang [17], Lie [16], Min-Max and Min-Sum attacks [18].

5.1 Experimental Setup

Datasets: We considered five datasets MNIST, Fashion-MNIST (F-MNIST), CIFAR-10, CINIC-10 and CIFAR-100 for evaluation. MNIST is a dataset of handwritten grayscale digits ranging from 0-9, with a training set of 60,000 training images and 10,000 test samples of size 28x28 distributed across all ten categories. Fashion-MNIST is a dataset of grayscale fashion products, which is equally distributed among ten classes with 7000 images per category. To test with complex datasets, we considered CIFAR-10 and CIFAR-100, which consist of 60000 32x32 color images with 10 and 100 classes, respectively. CINIC-10 is a complex image classification dataset constructed from ImageNet and CIFAR-10, with 270,000 images.

Learning classifiers: For MNIST and Fashion-MNIST, we used a Convolutional Neural Network (CNN). CNN has two convolution layers with 32 and 64 kernels of size 3×3, followed by a max-pooling layer and two fully connected layers with 9216 and 128 neurons. ReLU activation is used in each layer with a dropout of 0.25. No-attack accuracies over MNIST and F-MNIST are 97.74% and 87.09%, respectively. We used a complex model architecture (ResNet-101) for training over CIFAR-10, CINIC-10, and CIFAR-100. No-attack accuracies over CIFAR-10, CINIC-10, and CIFAR-100 are 75.60, 74.28, and 72.86, respectively. However, our goal is not to have the best-fit classifier. Instead, we wanted to increase the attack impact.

Our attack: We evaluated *Sine* and compared it with the SOTA attacks, assuming that the adversary does not know the aggregation rule (AN). We evaluated different adversary capabilities, $i.e.$, Full Capability (FC), Partial Capability (PC), and Individual Capability (IC).

Sine-FC: The attacker knows all the benign updates and finds a malicious update, which is much cosine similar to the compromised leader update but highly dissimilar to the averaged benign update.

Sine-PC: The attacker knows the benign updates of all compromised participants and finds a malicious update, which is much cosine similar to the compromised leader update but highly dissimilar to the averaged benign updates of the compromised participants.

Sine-IC: The attacker calculates the malicious update, which is much cosine similar to the compromised leader update but highly dissimilar to its own update.

Compared attacks: We considered SOTA local model poisoning attacks, Fang [17], LIE [16], Min-Max (MM) and Min-Sum (MS) [18] for comparison against *Sine* in both full and partial capabilities.

Baseline defenses: We evaluated SOTA attacks against SOTA defenses, Multi-Krum (MKrum) [5], Trimmed-Mean (TMean) [13], DnC [18], Romoa [14] and FLTrust [12]. We also evaluated against FedAvg with and without the attack (NA: no-attack) scenarios for baseline comparisons.

FL settings: We considered 100 participants. Each participant trains locally for 1 round with a batch size of 32 before submitting an update to the server. The server runs for a total of 50 epochs. For carrying out poisoning attacks, we assume 5%, 10%, and 20% of the participants are malicious.

Parameter Settings: The cosine scaling factor (γ_c) and the norm scaling factor (γ_n) are used in *Sine*. We use $\gamma_c = 2$ and $\gamma_n = 100$ unless specified otherwise. We evaluated the attack impact in a non-IID environment by varying λ (degree of heterogeneity) as 0.1, 0.5, and 0.9. Further, CL chooses 100 samples uniformly, from the clean training dataset to train its own model, in both IID or non-IID settings. We set parameter ϵ as 0.01 unless specified otherwise. β is set as the number of compromised worker devices for TMean.

Measurement metrics: For given attack settings, we measured attack impact as defined in [18], which is the reduction in accuracy of the global model due to the attack.

5.2 Attack results

Comparing with state-of-the-art attacks: We analyzed the attack impact of the SOTA attacks against the baseline defenses. Figure 3 illustrates the result considering 5% malicious participants over the Fashion-MNIST dataset. It is noticeable that FLTrust is attack-resistant against existing SOTA attacks but fails to defend against *Sine*. Though FLTrust considers both Euclidean and cosine similarity, *Sine* adaptively finds a malicious update harnessing the vulnerabilities in cosine similarity. *Sine* degrades the accuracy of FLTrust by 20% with full and 18% with partial capabilities. *Sine* also performs comparatively better than the existing attacks against other baseline defenses. Min-Max and Min-Sum achieve similar attack impacts only against Multi-Krum and Trimmed-Mean. But, they fail to find an optimal coefficient, especially against FLTrust. Figure 4 illustrates the result considering 10% malicious participants over the Fashion-MNIST dataset. FLTrust has a maximum attack impact of 27.5%. Romoa has an attack impact of around 13% with 5% malicious participants and 18% with 10% malicious participants. This is because, Romoa also measures similarity based on cosine similarity and Pearson correlation. DnC proves robust to other attacks, while failing to defend against *Sine* with an attack impact around 8% with 5% malicious participants and 10% with 10% malicious

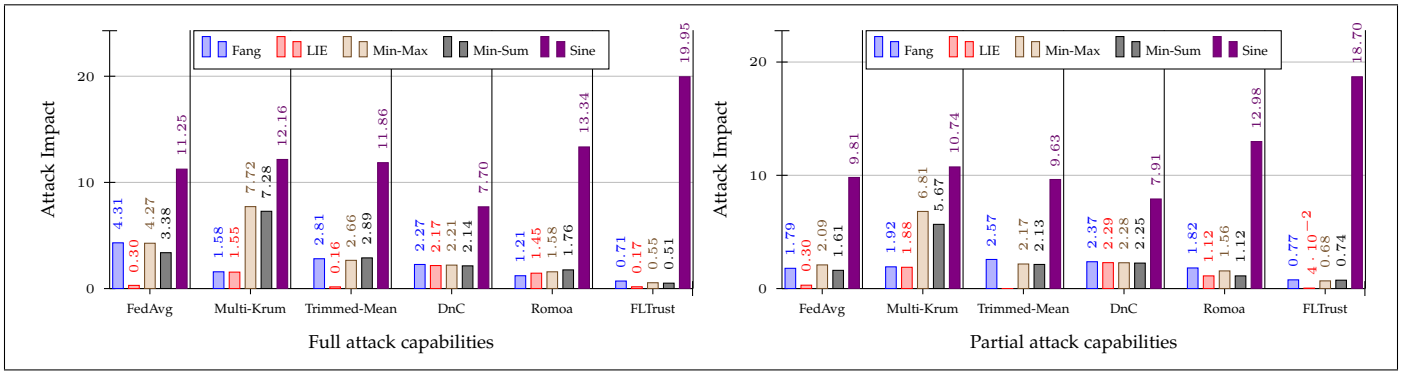


Fig. 3: Attack Impact for SOTA local model poisoning attacks against baseline defenses with 5% attackers (F-MNIST).

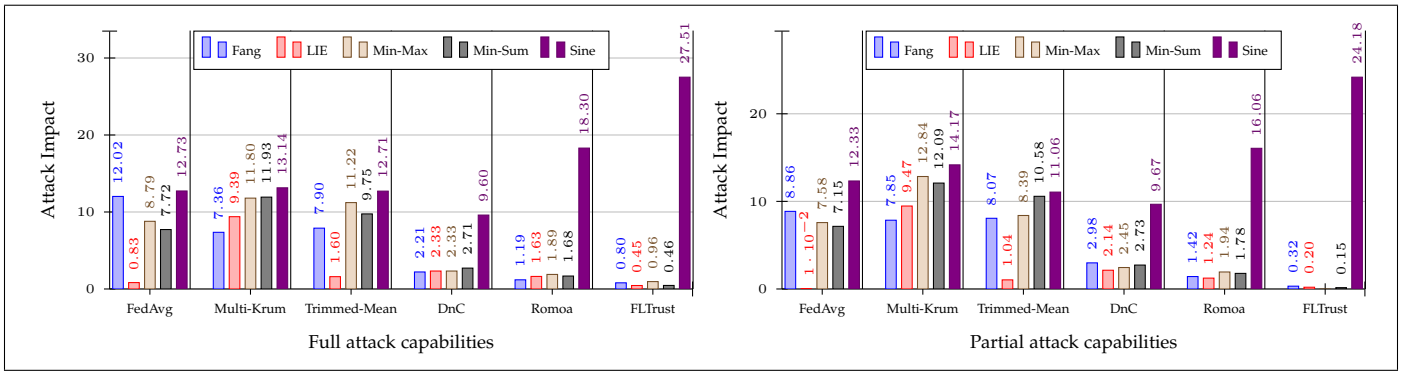


Fig. 4: Attack Impact for SOTA local model poisoning attacks against baseline defenses with 10% attackers (F-MNIST).

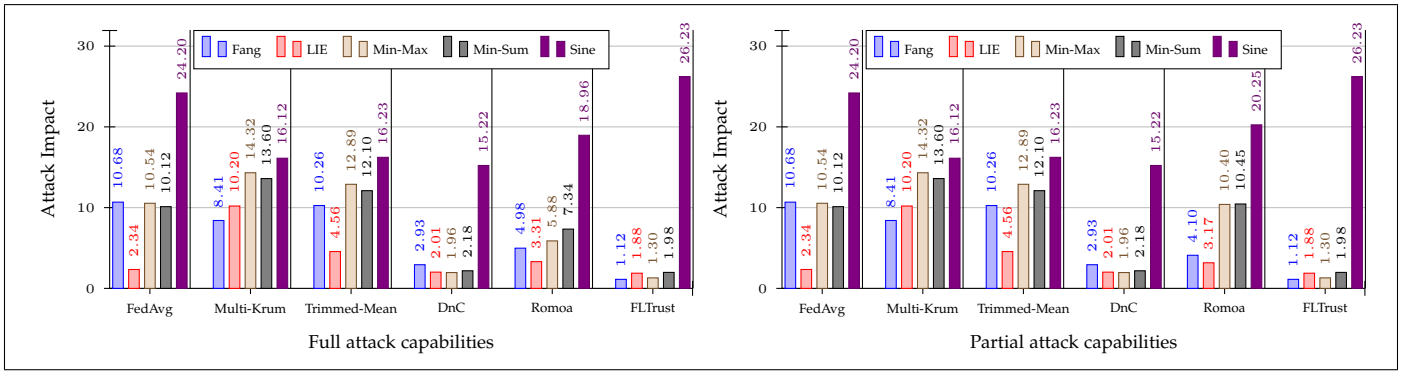


Fig. 5: Attack Impact for SOTA local model poisoning attacks against baseline defenses with 5% attackers (CIFAR-100).

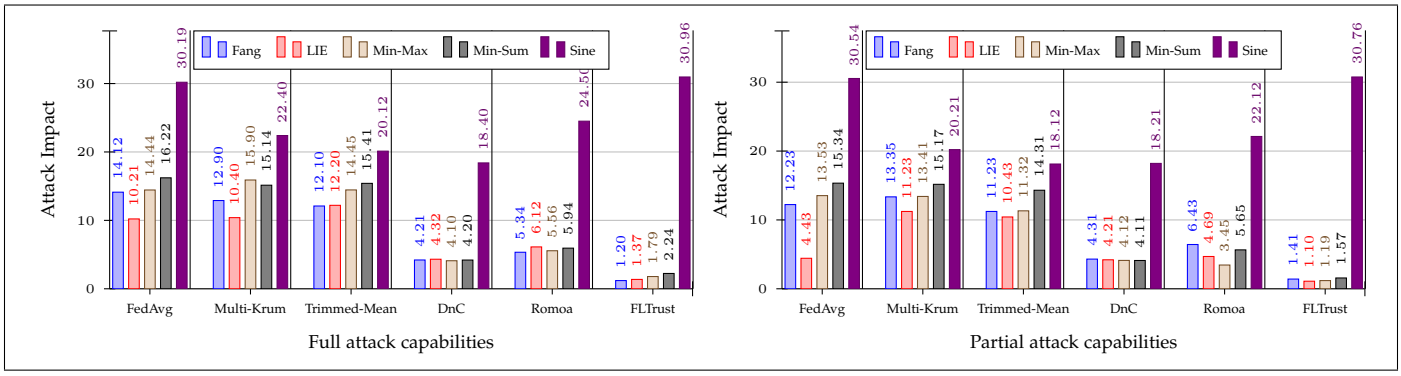


Fig. 6: Attack Impact for SOTA local model poisoning attacks against baseline defenses with 10% attackers (CIFAR-100).

participants. Trimmed-Mean is robust to existing attacks with 5% attackers, but has an attack impact of around 12% against *Sine*. Still, the marginal gain in attack impact is only due to the coordinate-wise aggregation in Trimmed-Mean.

Attack impact on larger datasets with complex models: We analyzed the attack impact for *Sine* on complex datasets CIFAR-10 and CIFAR-100 with complex model architecture ResNet-101. Figure 5 and 6 illustrate the result, considering

TABLE 3: Attack Impact for *Sine* with Individual Capability.

DS	%m	FedAvg	MKrum	TMean	DnC	Romoa	FLTrust
F-MNIST	5	12.41	10.77	9.54	7.67	19.65	21.44
	10	15.7	14.56	12.47	10.45	23.45	25.51
	20	17.91	15.23	14.78	12.43	27.68	30.43
CIFAR-100	5	26.76	18.80	17.36	16.20	20.74	22.90
	10	30.82	20.20	19.45	18.78	27.18	30.56
	20	38.20	27.18	26.46	24.12	32.6	35.78

5% and 10% malicious participants over the CIFAR-100 dataset. It can be observed that FLTrust is most vulnerable with an attack impact of around 26% with 5% malicious participants and 30% with 10% malicious participants. Multi-Krum, Trimmed-Mean, and DnC have an attack impact of around 16% with 5% malicious participants and 20% with 10% malicious participants. It can also be observed that DnC is resistant against the existing SOTA attacks, with negligible attack impact around 2% and 4% in the presence of 5% and 10% malicious participants, respectively. We also observed that DnC consumed the highest memory, and also has a higher attack impact to complex datasets. Romoa has an increased attack impact of around 20% with 5% malicious participants and 24% with 10% malicious participants.

Attack Impact with Individual Capability: This is the most practical attack capability, where each adversary frames the malicious update, only based on its own benign update. Table 3 plots the result for attack impact of *Sine* considering (5%, 10%, and 20%) malicious participants with individual capabilities. FLTrust has an attack impact of around 21%, 25%, and 30% in the presence of 5%, 10%, and 20% malicious participants, respectively, over the Fashion-MNIST dataset. Further, it has an attack impact of around 23%, 31%, and 36% in the presence of 5%, 10%, and 20% malicious participants, respectively, over the CIFAR-100 dataset. It shows that FLTrust performance continues to degrade across all the datasets. Similarly, Romoa also has a high attack impact over both datasets. However, other defenses suffer comparatively less but still have a significant attack impact than full and partial attack capabilities. DnC has the minimum attack impact of around 8%, 11%, and 13% in the presence of 5%, 10%, and 20% malicious participants, respectively, over the Fashion-MNIST dataset. Further, it has an attack impact of around 16%, 18%, and 24% in the presence of 5%, 10%, and 20% malicious participants, respectively, over the CIFAR-100 dataset. This shows, that DnC is not suitable for complex datasets. Multi-Krum and Trimmed-Mean have an attack impact of around 10%, 12%, and 15% in the presence of 5%, 10%, and 20% malicious participants, respectively, over the Fashion-MNIST dataset. Further, they have an attack impact of around 18%, 20%, and 27% in the presence of 5%, 10%, and 20% malicious participants, respectively, over the CIFAR-100 dataset.

Impact of different parameter settings: We studied the impact of different parameters used in *Sine* including degree of heterogeneity (λ), epsilon (ϵ) and scaling (norm (γ_n) and cosine (γ_c)) factors.

Impact of heterogeneity: We evaluated *Sine*, by simulating federated learning with non-IID training data distributions.

TABLE 4: Attack Impact for *Sine* by varying λ (non-IID).

DS	λ	FedAvg	MKrum	TMean	DnC	Romoa	FLTrust
F-MNIST	0.1	12.81	10.74	11.01	8.26	18.45	22.10
	0.5	14.15	12.64	12.76	9.20	20.05	24.20
	0.9	17.18	13.17	15.76	10.18	24.36	28.36
CIFAR-100	0.1	26.98	20.88	19.26	17.12	20.56	24.24
	0.5	27.82	22.46	21.44	18.48	23.52	27.20
	0.9	28.30	22.90	22.68	20.18	26.85	30.68

TABLE 5: Attack Impact for *Sine* by varying ϵ (noise).

DS	ϵ	FedAvg	MKrum	TMean	DnC	Romoa	FLTrust
F-MNIST	0.01	12.28	10.18	11.41	10.56	18.48	21.57
	0.1	8.38	7.12	7.02	6.98	15.28	18.24
	1	6.98	5.84	6.24	5.86	12.94	14.60
CIFAR-100	0.01	26.48	20.46	18.88	18.40	22.88	24.60
	0.1	20.42	16.40	15.34	11.26	18.10	21.78
	1	12.34	11.65	11.12	9.80	12.68	13.98

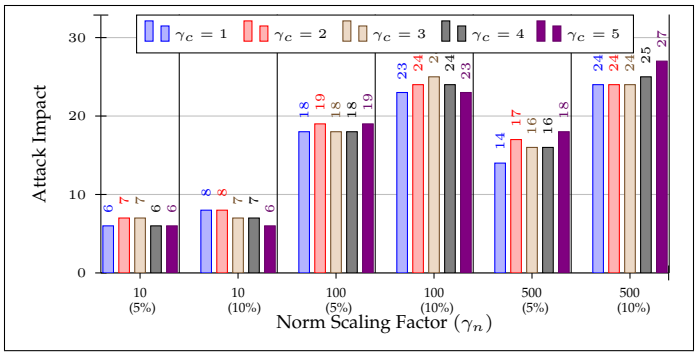


Fig. 7: Attack impact with varying scaling factors.

Dirichlet's distribution is used to simulate such label imbalance. We sampled $p_l \sim \text{Dir}_N(\lambda)$ and allocate $p_{l,i}$ proportion of class l to participant i, where λ is the concentration parameter ($\lambda > 0$). *Sine* is evaluated with varying degree of heterogeneity (λ = 0.1, 0.5 and 0.9). Table 4 logs the attack impact for *Sine* with individual capability in the presence of 5% attackers. The attack impact against FLTrust increases by more than 6%, when λ is increased from 0.1 to 0.9, over both the Fashion-MNIST and CIFAR-100 datasets. Similarly, against other defenses, it can be observed that the attack impact increases with increasing λ.

Impact of parameter ϵ: We evaluated *Sine* with varying ϵ. Adversaries add ϵ and post $(M + \epsilon_i)$ to the trusted server for aggregation, where ϵ_i is random perturbation added to M by adversary i. Table 5 evaluates the impact of ϵ on the effectiveness of *Sine* with individual capabilities, considering 5% attackers. We evaluated by varying ϵ as 0.01, 0.1, and 1. It can be observed that keeping ϵ smaller, *Sine* achieves higher attack impact, as smaller ϵ maintains high Euclidean and cosine similarity. For example, the attack impact against FLTrust decreases by more than 6%, when ϵ is increased from 0.01 to 1, over both Fashion-MNIST and CIFAR-100 datasets.

Impact of scaling factors: We analyzed the impact of scaling

factors γ_c and γ_n against FLTrust, considering 5% and 10% attackers over MNIST dataset. As shown in Figure 7, it can be observed that while keeping the norm scaling factor (γ_n) less and increasing the cosine scaling factor (γ_c) degrades the attack performance. However, increasing γ_c with larger values of γ_n improves the performance. We investigated this behavior and observed that keeping γ_n less with high γ_c does not allow changing multiple coordinates for getting the malicious vector, resulting in poor performance.

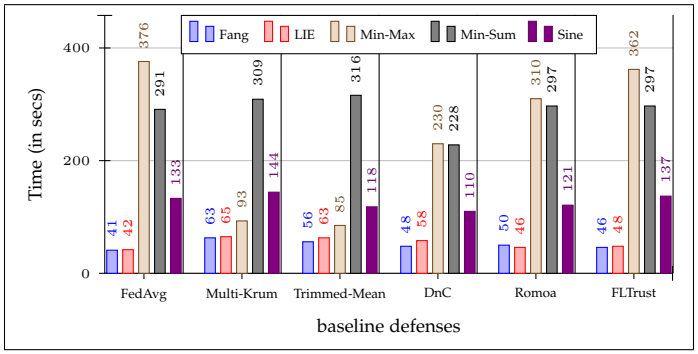


Fig. 8: Time taken by SOTA local model poisoning attacks against baseline defenses.

Attack Time: We analyzed the attack performance in terms of time taken and the computation involved. We used a machine with a processor Intel® Core™ i7-7700 CPU @ 3.60GHz × 8, and a Memory of 8 GB. Figure 8 shows the time taken to run these attacks. It can be observed that Min-Max, Min-Sum and *Sine* consume more time than Fang and LIE. Fang uses binary search to solve the optimization problem and stops at a threshold if no solution is found. LIE involves deterministic statistical operations (mean and variance) to find the perturbation noise. Min-Max and Min-Sum keep oscillating between a range to find a scaling factor until a malicious gradient is selected. They also involve normalization operation on gradients, which is a costlier operation. Thus, Min-Max and Min-Sum consume more time than Fang and LIE. *Sine* also involves a normalization operation, but it has a deterministic search space. Therefore, *Sine* consumes less time compared to Min-Max and Min-Sum. Fang and LIE consumed around 600 seconds over the CIFAR-100 dataset. *Sine* consumed around 800 seconds, while Min-Max and Min-Sum consumed around 1080 seconds over the CIFAR-100 dataset. We refer to Appendix C for more results on *Sine*. Based on the results, it is claimed that the impact of *Sine* is significant against Byzantine-robust aggregation rules.

6 *FLTC:* The Proposed Defense

A strong adaptive attack using local model poisoning raises severe concerns, which motivated us to design a new aggregation rule. We learn the following lessons from the above attacks, that *Similarity (Euclidean or cosine) is not enough*.

Lesson 1: Euclidean distance-based aggregation rules can be manipulated by a single Byzantine participant. They introduce sybils supporting each other, who help themselves to get selected by moving the center [5].

Lesson 2: A Byzantine participant can maintain high as well as low cosine similarity with two different benign updates. It helps to increase the supporters for malicious updates and opposers for the benign updates [16].

Lesson 3: Coordinate-level aggregation rules are better resistant against the state-of-the-art attacks among the existing baseline defenses. Empirical results justify this claim.

We devise a new aggregation algorithm, *FLTC* (FL Trusted Coordinates), based on the lessons learned and ideas drawn from the existing similarity-based defenses. It performs coordinate-wise aggregation operations to find the similarity of the participant's local model updates with the trusted base update. The aggregator trains over a small dataset to get the trusted base update and considers it as a benign reference. The aggregator only needs a small and clean dataset, so it can manually collect and label the samples. In particular, we sample the root dataset uniformly from the clean training dataset and exclude those from the participant's training data. Unless specified otherwise, we consider 100 samples in the root dataset. *FLTC* relies on direction cosine for finding similarity, which measures the change in the direction and magnitude of the participant's local update against the trusted base update. However, coordinates can achieve high similarities on both sides of the reference update. One way to address this is to consider a majority-based approach. Summarizing this, *FLTC* relies on direction cosine for measuring the similarity of updates and a majority-based system for trimming the updates with high similarity, but in the reverse direction.

Algorithm 2 Federated Learning - Defense

$S \leftarrow$ Set of participants
$w^t \leftarrow$ Initialize global model for round $t = 0$
$\nabla w_i^t \leftarrow$ Refers to local model update of client i in round t.
for each round $t = 1, 2, \ldots, R$ **do**
  $S^t \leftarrow$ Set of participants selected for round t
  $\nabla_{BS}^t \leftarrow$ Base Server Update.
  ∇w_i^t = UpdateClients(w^t)
  $w^{t+1} \leftarrow$ FLTC($w^t, \nabla w_i^t, \nabla_{BS}^t$)
function FLTC($w^t, \nabla w_i, \nabla_{BS}$)
  $\nabla w \leftarrow 0$
  $Euclidean\_distances \leftarrow dist(\nabla w_i - \nabla_{BS})$
  for for each dimension $k \in 1, 2, \ldots, d$ **do**
    $trust\_vector \leftarrow \frac{\nabla_{BS}^k - \nabla w_i^k}{Euclidean\_distances}$ ▷ It is an $S^t \times 1$ vector with direction cosines of all the model updates $w.r.t.$ the base update.
    $sign_p \leftarrow$ Indices of coordinates with positive direction cosine.
    $sign_n \leftarrow$ Indices of coordinates with negative direction cosine.
    $trusted\_coords \leftarrow sign_p$ if $len(sign_p) > len(sign_n)$ else $sign_n$ ▷ Remaining coordinates are considered as $untrusted\_coords$.
    $trust\_vector[untrusted\_coords] \leftarrow 0$
    $min\_max\_norm(trust\_vector[trusted\_coords])$
    $\nabla w^k \leftarrow \frac{\nabla w_i^k \times trust\_vector^T}{sum(trust\_vector)}$
  $w_i^{t+1} \leftarrow w_i^t + \nabla w$
  return w_i^{t+1}

The aggregator broadcasts the model w^t, for any iteration t. Each participant ($i \in n$) performs local training and pushes its local model update ∇w_i^t to the aggregator. The aggregator performs local training over a small and clean dataset, resulting in a trusted base update (∇_{BS}^t). Next, the

aggregator executes *FLTC* with parameters as the current global model (w^t), local model updates (∇w_i) and the base model update (∇_{BS}), as shown in Algorithm 2. *FLTC* uses hyper-spherical direction cosine to analyze the change in direction and magnitude, as stated in Equation 5.

$$cos(\theta^k) = \frac{\nabla^k_{BS} - \nabla w^k}{\sqrt{\sum_{k=1}^{d}(\nabla w^k - \nabla^k_{BS})^2}}, \tag{5}$$

for each $k \in [d]$ dimension.

FLTC first computes the Euclidean distance for all the updates, $w.r.t.$ the base update. With S^t participants in round t, it gets a vector of size ($S^t \times 1$) for all the Euclidean distances. For each dimension, $k \in [d]$, *FLTC* calculates a trust vector using hyper-spherical direction cosine (lines 12-19 in Algorithm 2). The trust vector is calculated for each coordinate of all the participants in iteration t, which is of size ($S^t \times 1$). It finds the number of coordinates with positive and negative direction cosine and selects the ones in the majority. Then, it performs min-max normalization over the selected trusted coordinates and assigns zero scores to the remaining untrusted coordinates. The updated k^{th} coordinate is calculated by multiplying the coordinates for k^{th} the dimension for all the participants in S^t with the transpose of the trust vector, divided by the sum of the trust vector. Similarly, *FLTC* updates each coordinate $k \in [d]$ for the training plan of the next iteration. The computational complexity of this approach is $\mathcal{O}(d)$, because it requires calculating total d direction cosines, *i.e.*, for each dimension in the local model update.

Theoretical analysis: Theoretical analysis of *FLTC* provides guarantees for removing malicious gradients on each coordinate. So the difference between the global model learned by *FLTC* and the optimum global model w^* is bounded. We consider similar assumptions as in [12], (1) The expected loss function $F(w)$ is μ-strongly convex and differentiable over the space Θ with L-$Lipschitz$ continuous gradient; (2) The gradient of the empirical loss function $\nabla f(D, w^*)$ at w^* is bounded; and (3) the local and root datasets are sampled independently of the distribution $\mathcal{X}$.

Lemma 2. For an arbitrary number of malicious clients, the distance between ∇w^k and $F(w^k)$ is bounded, as stated in Equation 6:

$$\left\|\nabla w^k - \nabla F\left(w^k\right)\right\| \leq 3\left\|\nabla^k_{BS} - \nabla F\left(w^k\right)\right\| + 2\left\|\nabla F\left(w^k\right)\right\|, \tag{6}$$

for every coordinate $k \in [d]$.

Proof. See Appendix D. □

Theorem 2. For an arbitrary number of malicious clients, the distance between the global model (w^t, in t^{th} iteration) learned by *FLTC* and optimum global model w^* is bounded with probability at least $1 - \delta$, as stated in Equation 7:

$$\left\|\boldsymbol{w}^t - \boldsymbol{w}^*\right\| \leq (1-\rho)^t\left\|\boldsymbol{w}^0 - \boldsymbol{w}^*\right\| + 12\alpha\Delta_1/\rho. \tag{7}$$

Proof. See Appendix E. □

6.1 Adaptive attack against *FLTC*

If an adversary is aware of the defense used for aggregation, it can adapt the attack accordingly. To evaluate the robustness of the proposed defense, we consider that the adversary has full knowledge of the aggregation rule, the trusted root update (∇_{BS}), and the other client updates (∇w_i).

FLTC can be rewritten as Equation 8:

$$\nabla w^k = \sum_{i=1}^{n} \frac{\varphi_i^k}{\sum_{j=1}^{n}\varphi_j^k} \nabla w_i^k, \tag{8}$$

where φ_i^k is the trusted coordinate score for the k^{th} dimension of i^{th} client. φ_i^k is calculated as $\frac{\nabla^k_{BS} - \nabla w_i^k}{\|\nabla_{BS} - \nabla w_i\|}$, which is further trimmed based on the majority of the sign of φ_i^k, for $i \in n$, followed by min-max normalization.

Suppose there are first m malicious clients out of n clients, then the poisoned global model under *FLTC* can be rewritten as Equation 9:

$$\begin{aligned}\nabla w^{k'} = &\sum_{i=1}^{m} \frac{\varphi_i^k}{\sum_{j=1}^{m}\varphi_j^k + \sum_{j=m+1}^{n}\varphi_j^k} \nabla w_i^{k'} \\ &+ \sum_{i=m+1}^{n} \frac{\varphi_i^k}{\sum_{j=1}^{m}\varphi_j^k + \sum_{j=m+1}^{n}\varphi_j^k} \nabla w_i^k.\end{aligned} \tag{9}$$

Local model poisoning attack framework: The local model poisoning framework proposed in [17] is formulated as the optimization problem, as stated in Equation 10:

$$\begin{aligned}&\max_{\boldsymbol{\nabla w}'_1, \boldsymbol{\nabla w}'_2, \cdots, \boldsymbol{\nabla w}'_m} s^T\left(\boldsymbol{\nabla w} - \boldsymbol{\nabla w}'\right) \\ &\text{subject to } \boldsymbol{\nabla w} = \mathcal{A}\left(\boldsymbol{\nabla w}_1, \cdots, \boldsymbol{\nabla w}_m, \boldsymbol{\nabla w}_{m+1}, \boldsymbol{\nabla w}_n\right) \\ &\boldsymbol{\nabla w}' = \mathcal{A}\left(\boldsymbol{\nabla w}'_1, \cdots, \boldsymbol{\nabla w}'_m, \boldsymbol{\nabla w}_{m+1}, \boldsymbol{\nabla w}_n\right),\end{aligned} \tag{10}$$

where ∇w is the before attack global model and s is the column vector of the sign of ∇w. The attacker crafts the compromised model updates in the inverse of the direction along which ∇w would change, based on the maximum or minimum benign local model updates for every coordinate. More precisely, it reduces to solve the optimization function as $\nabla w - \lambda s$ for evaluating the value of λ.

When this attack framework is adapted for FLTrust, λ is upper-bounded in terms of the Euclidean distance between the benign and malign local model updates. It deviates from the maximum or minimum of the before-attack global model parameters, maintaining Euclidean distance. However, it does not consider the cosine similarity principle, which is the backbone of FLTrust. As a result, the malicious updates do not maintain cosine relations, and FLTrust is shown to resist this adaptive attack framework. However, *Sine* harnesses the vulnerabilities based on cosine similarity and successfully impacts the performance of FLTrust.

Adapting attack framework for *FLTC*: As *FLTC* is a coordinate-wise aggregation rule, it assigns the trust score for the coordinates of local model updates of all the participants. If φ is a trust score matrix with rows as participants and columns as dimensions, s can be computed as $\left[\text{sgn}\left(\varphi_i^1\right), \text{sgn}\left(\varphi_i^2\right), \ldots, \text{sgn}\left(\varphi_i^k\right)\right]_{i \in n}$, where $k \in [d]$ and n is the number of participants. sgn is based on the majority of the sign of φ_i^k for the local model updates before attack.

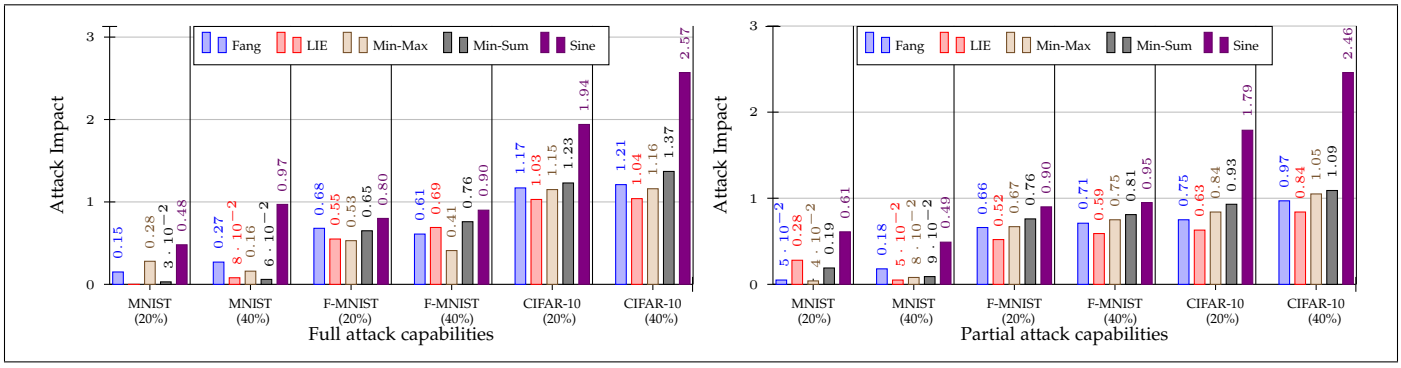


Fig. 9: Attack Impact for SOTA local model poisoning attacks against *FLTC*.

Further, we solve the optimization function for crafting a poisonous model update against *FLTC* as $\nabla_{BS} - \lambda s$ for λ, by assigning the malicious parameters similar to the trim attack proposed in [17]. If s_j=-1, the malicious j^{th} coordinate is sampled in the interval $[\nabla^j_{BS}, b.\nabla^j_{BS}]$ (when $\nabla^j_{BS} > 0$) or $[\nabla^j_{BS}, \nabla^j_{BS}/b]$ (when $\nabla^j_{BS} \leq 0$), otherwise, it is sampled in the interval $[\nabla^j_{BS}/b, \nabla^j_{BS}]$ (when $\nabla^j_{BS} > 0$) or $[b.\nabla^j_{BS}, \nabla^j_{BS}]$ (when $\nabla^j_{BS} \leq 0$). As in trim attack, the attack does not depend on b, and we set it as 2.

TABLE 6: Attack Impact for Adaptive Attack against *FLTC*.

λ	MNIST			F-MNIST			CIFAR-10		
%m	10	20	40	10	20	40	10	20	40
0.1	0.95	1.13	1.67	1.17	1.39	1.82	2.09	2.31	2.94
0.5	1.14	1.43	1.87	1.47	1.56	1.97	2.79	2.81	3.08
0.9	1.87	1.83	2.13	1.71	1.91	2.31	3.27	3.56	3.89

λ : Degree of heterogeneity (non-IID)

6.2 Defense results

Figure 9 plots the attack impact with all the attack settings in the threat model, considering 20% and 40% malicious participants. It can be observed that the attack impact is bounded by 0.5% for MNIST and Fashion-MNIST, 2% for CIFAR-10, considering 20% malicious participants. The attack impact is bounded by 1% for MNIST and Fashion-MNIST, 2.5% for CIFAR-10, considering 40% malicious participants. Table 6 logs the attack impact for an adaptive (more potent) attack against *FLTC*, considering 10%, 20% and 40% malicious participants, with varying degree of heterogeneity (λ denotes Dirichlet distribution hyperparameter). The attack impact is bounded by 2% for MNIST and Fashion-MNIST, 3% for CIFAR-10, considering 40% attackers, for IID (homogeneous) distribution. The attack impact is not impacted by large for non-IID (heterogeneous) distribution and increases by a maximum of 1%. This is because, both the server and honest participants with heterogeneous data use the same global model in the last iteration to train trusted update and local model updates, respectively. *FLTC* also proves resistant to data poisoning attacks; results can be referred to in Appendix F.

Aggregation Time: Table 7 logs the aggregation time taken by SOTA defenses. FLTrust and Romoa consume less time to converge, as they compute the trust score in reference to a single trusted update. DnC also consumes less time, equivalent to FedAvg. Multi-Krum suffers due to nested Euclidean operations, while Trimmed-Mean and *FLTC* suffer due to coordinate-wise operations. However, *FLTC* can be parallelized to reduce aggregation time.

TABLE 7: Aggregation time (secs) for baseline defenses.

Rule	FedAvg	MKrum	TMean	DnC	Romoa	FLTrust	***FLTC***
Time	39	63	56	46	45	42	54

Discussion: We discuss the robustness of *FLTC* in comparison with the existing state-of-the-art defenses [5], [12], [13], [18]. It is known that a single model parameter can substantially influence the Euclidean distance between two local model updates. A large change to a single parameter would impact the model performance without impacting the Euclidean distance in case of high dimensionality. To mitigate this, Median and Trimmed-Mean [13] perform coordinate-wise aggregation. However, to circumvent such defenses, an attack can be crafted by sampling Byzantine values between the mean and the supporters, such that they become closer to the mean than the opposers [6], [16], [18]. FLTrust [12] proposed a cosine similarity based defense, which bootstraps trust with its own clean dataset. It proved efficient and resistant to data and local model poisoning attacks until we harnessed its vulnerability. DnC [18] is the new state-of-art defense, which needs to memorize a good set of participants for multiple iterations. It is also vulnerable to larger datasets and complex model architectures. *FLTC* is the first defense, considering Euclidean and cosine similarity along with coordinate-wise aggregation. On the top, it proves to be equally efficient in terms of time and memory allocation. It achieves the best accuracy on larger datasets, even with more complex model architectures. It is resistant to the existing data and local model poisoning attacks, even considering strong adaptive attacks and heterogeneous data distribution.

7 Conclusion

This paper analyzed the vulnerabilities existing in the state-of-the-art Byzantine-robust aggregation schemes based on similarity-based measures. A generalized local model poisoning attack framework (*Sine*) is presented, which degrades the global model performance indiscriminately of the

test samples. *Sine* is efficient in terms of time and computation due to deterministic search space. *Sine* is aggregation agnostic, non-omniscient, and works with different attack capabilities. The proposed attack is limited to untargeted poisoning attacks, which can be extended to targeted poisoning attacks. The results raised the alarm over the existing defenses and motivated the design of a new defense to mitigate such ingenious attacks. Further, we proposed a defense algorithm *FLTC*, which operates over each coordinate and aggregates based on the change in direction and magnitude in reference to the base model update. *FLTC* restricts the attack impact to 2-4%, against state-of-the-art poisoning attacks, including strong adaptive attacks.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[2] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[3] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 374–388, 2019.

[4] B. Han, R. Jhaveri, H. Wang, D. Qiao, and J. Du, "Application of robust zero-watermarking scheme based on federated learning for securing the healthcare data," *IEEE Journal of Biomedical and Health Informatics*, 2021.

[5] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 118–128.

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

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

[8] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948.

[9] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *NDSS*, 2018.

[10] B. Biggio, L. Didaci, G. Fumera, and F. Roli, "Poisoning attacks to compromise face templates," in *2013 International Conference on Biometrics (ICB)*. IEEE, 2013, pp. 1–7.

[11] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 19–35.

[12] X. Cao, M. Fang, J. Liu, and N. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *NDSS*, 2021.

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

[14] Y. Mao, X. Yuan, X. Zhao, and S. Zhong, "Romoa: Robust model aggregation for the resistance of federated learning to model poisoning attacks," in *European Symposium on Research in Computer Security*. Springer, 2021, pp. 476–496.

[15] X. Li, Z. Qu, S. Zhao, B. Tang, Z. Lu, and Y. Liu, "Lomar: A local defense against poisoning attack on federated learning," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2021.

[16] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[17] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 1605–1622.

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

[19] H. Kasyap and S. Tripathy, "Hidden vulnerabilities in cosine similarity based poisoning defense," in *2022 56th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2022, pp. 263–268.

[20] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016. [Online]. Available: http://arxiv.org/abs/1602.05629

[21] X. Cao and N. Z. Gong, "Mpaf: Model poisoning attacks to federated learning based on fake clients," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3396–3404.

Harsh Kasyap is pursuing his Ph.D. at the Department of Computer Science and Engineering, Indian Institute of Technology Patna, India. From Aug 2016 to Jan 2019, he worked at Diebold Nixdorf, Mumbai, India as a Software Engineer. Prior to this, he completed a Bachelor of Engineering (Computer Science) from Vishwakarma Institute of Information Technology, Savitribai Phule Pune University, Pune, India, in 2016. His research interests include machine learning, federated learning, and computer security.

Somanath Tripathy is a Professor in the Department of Computer Science and Engineering at the Indian Institute of Technology, Patna. He joined the faculty in December 2008. He was the Associate Dean, Academics from January 2016 to March 2017 and the Associate Dean, Administration from July 2021 to Nov 2023 at IIT Patna. His research interests include lightweight cryptography, security issues in resource-constrained devices, blockchain, malware detection and secure machine learning. He has published more than 100 research papers in various reputed journals and conferences. He has been the Principal Investigator of several projects related to security. Dr. Tripathy is currently an editor of the IETE Technical Review and an associate editor of the Multimedia Tools and Applications, an International Journal.