1 Introduction

Federated Learning (FL), which typically consists of a central server and multiple clients, has gained attention for its ability to build high-precision models while avoiding the risks associated personal data leakage [1–3]. Models rather than data are transferred in FL. The client fine-tunes the global model by training local data and uploads gradient updates to a center server, where the gradients are aggregated and synchronized in the next round. However, device heterogeneity, data diversity, and restricted computing resources make FL particularly vulnerable to poisoning attacks [4–6].

In order to disrupt global patterns or leave backdoors, attackers can attack by uploading poisoned updates using contaminated local training data (i.e. data poisoning attack [7–10]), or constructing poisoned model gradient updates (i.e. model poisoning attack [11–14]). The main ideas of existing defenses include eliminating updates that differ significantly from the overall distribution [15–18], evading malicious parameters through statistical features [19–21], detecting outliers based on extracted update features [12, 22, 23], and detect anomalies relying on clean datasets [24–26].

Despite the wide variety of defenses, limitations still remain. Firstly, historical data is underutilized. Servers typically recognize client updates based on current round data or use simple distance-based discriminators on just long-term historical gradients [27, 28]. Momentum-based gradient descent methods only indirectly utilise historical data [29], which is susceptible to noise and insufficient to deal with dynamic attacks. Secondly, privacy requirements are often compromised. Especially in the case of defences that rely on clean datasets [24, 25] for anomaly detection, the central server is required to access clean validation datasets whose distribution should not be significantly different from the overall training data distribution. Thirdly, malicious clients are not well characterized [30, 31]. For example, detecting malicious clients based on historical update records requires extensive data to ensure accurate prediction and detection, whose inadequacy is underscored in thoroughly analyzing the characteristics of malicious clients due to the reliance on extensive data.

Exacerbating these limitations is the fact that the existing model poisoning attack methods [11–13] indicate that the current aggregation rule is successful in reducing the impact of an attack on any given round. However, this assumption does not hold in long-term operating FL systems, since it has been widely demonstrated to be inevitable that any aggregation rule ignoring history may ultimately lead to disagreement [28]. The inherent noise in the gradients is a mask of small perturbations that cannot be detected in a single round, so perturbations accumulate and hide in long-term historical gradients over time.

In this work, a new FL defense strategy based on historical gradients is proposed called LSH-FL to get rid of limitations above. LSH-FL integrates perturbation and defense strategies to detect malicious model updates and mitigate the impact of poisoning attacks. Specifically, in the perturbation component (P-SHG), the anomalous feature space is perturbed during local training by utilizing masks derived from differences in short-term historical gradients, thereby mitigating the potential long-term effects of attacks. Meanwhile, long-term historical gradients obtained through an exponential smoothing approach are used in the defense component (D-LHG) to extract malicious disturbance features accumulated over time, thereby achieving malicious update recognition. Numerous experiments have shown that FL defense strategies using long and short historical gradients are more effective than existing defense methods under different attacks.

Overall, we have made the following contributions:

- A unique FL defense strategy framework is proposed that fully utilizes long and short historical information to extract features and imposes perturbation to defend against poisoning attacks;
- A perturbation strategy utilizing short-term historical gradients on the client side has been developed, which perturbs the anomalous feature space based on difference masks, thereby mitigating the potential long-term impact of attacks;
- A defense strategy based on long-term historical gradients has been implemented for the detection of malicious clients. This strategy employs a new feature decomposition method LH-DnC, to detect malicious clients, with an exponential smoothing mechanism designed to enhance defense against dynamic attacks;
- Extensive experiments conducted on four benchmark datasets indicate significant improvements in defense performance against three attacks with different proportions of malicious clients, demonstrating the complementarity of perturbation and defense in LSH-FL.

2 Related Work

2.1 Poisoning Attacks in FL

Poisoning attacks are implemented by manipulating the model updates of malicious clients to mislead the global model. More specifically, model poisoning attacks directly modify model update parameters [11–14], while data poisoning attacks poison local training data to affect model updates [7–10]. In model poisoning attacks, there are two ways to modify model parameters. One is to modify the direction

or size of the parameters, such as flipping the symbol of the model parameters or modifying the norm of the gradient. And the other is random parameter substitution, where a random parameter is obtained and then put into the model parameters or uploaded as a gradient value. Data poisoning attacks are mainly divided into three types: label flipping, noise addition, and backdoor triggers. In FL, the advantage that the data is not shared and only the models are used as information carriers also happens to be an important basis for implementing data poisoning attacks without being easily detected. Here are several of common methods of poisoning attacks.

Label Flip (LF) [32] and Sign Flip (SF) [33] are two flip-based attacks where the attacker wishes to influence the direction of the gradient update as much as possible to maximise the loss. However, such direct flipping strategies produces obvious outliers that are easily detected by defence algorithms. Little Is Enough (LIE) [11] can be used to defeat existing defences and interfere with or control the training process by directly making small changes to multiple parameters at several nodes. In particular, when an attacker has access to the mean and standard deviation of benign gradients, defence algorithms are typically unable to distinguish damaged clients from benign gradients. To maximise the impact of the attack, Minimize Maximum Distance (Min-Max) [12] looks for the maximum perturbation when the distance between the malicious update and any benign update is less than the maximum distance between any two benign updates. Consistently, Minimize Sum of Distances (Min-Sum) [12] search for the maximum perturbation when the sum of distances between malicious updates and all benign updates is less than the sum of distances between any two benign updates.

2.2 Byzantine-robust Aggregation Rules

In FL, the server identifies abnormal updates from malicious attackers, and deletes or weights them based on aggregation rules as shown in Fig. S1. According to the server detection principles, existing defense schemes can be divided into three categories: distance-based [15–18], performance-based [24, 34], and statistical-based defense schemes [27–29].

Distance-based scheme: The idea is to calculate the distance between uploaded model parameters or gradients. The K points farthest from other clients updating gradients may be malicious clients, which can be selectively discarded in the gradient aggregation. To address the effect of malicious behavior on model aggregation in FL, Blanchard et al. proposed a byzantine fault-tolerant algorithm Krum [15]. At each round, the server calculates the Euclidean distance between all gradients, and selects the nearest set of model parameters as the aggregated global model parameters. Blanchard et al. also developed Multi-Krum [15] that is able to

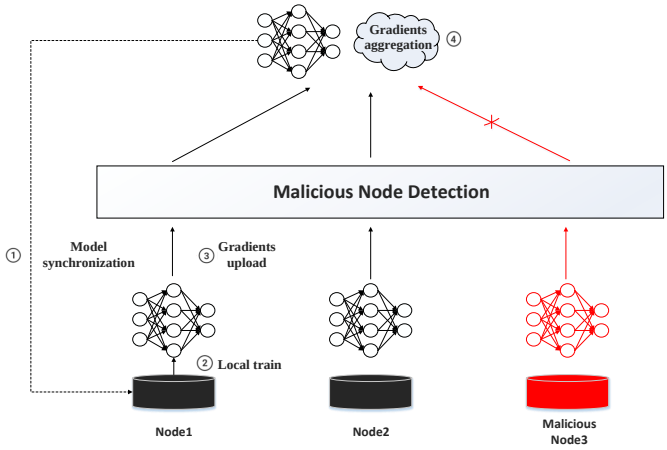


Fig. S1: The server detects malicious updates that attackers are attempting to send to prevent them from damaging the aggregation model.

select multiple sets of model parameters closest to other models for average aggregation. FABA [16] is similar to Krum, but removes the gradient that is farthest from the average of small batch gradients.

Performance-based scheme: Each update is detected for abnormal updates, and any poorly performing updates will be assigned low weights or deleted. There are several ways to verify performance, such as using additional clean datasets, consistency based on malicious gradient historical information, or constructing autoencoder models. Xie et al. proposed a FL aggregation scheme Zeno [24], which relies on a clean auxiliary dataset to sort each candidate gradient estimate and select the better performing gradients for aggregation updates. Chan et al. [34] trained an autoencoder model with relatively high computational complexity to detect malicious nodes and aggregated updated gradients based on error rates. FLDetector [30] detects malicious clients by measuring the consistency between current round updates and expected predicted updates based on multiple historical rounds.

Statistical-based scheme: Statistical features of uploaded gradients are applied to distinguish abnormal updates. Dong et al. [19] proposed two robust distributed gradient descent algorithms, the Median algorithm and the Trim algorithm. The Median algorithm uses the local model to update the median of each dimension as the update of the global model. And the Trim algorithm trims updates whose norm exceeds the threshold M under the assumption that updates from local model poisoning attacks are often larger.

2.3 The use of Historical Information in FL

Recent studies [28, 35, 36] have further analyzed the internal workings of existing attacks, providing new insights into the potential of attacks and the challenges of clear FL robustness solutions. It demonstrates that attackers can construct small perturbations hidden within the inherent noise of gradients,

which will accumulate over time in long-term historical gradients and affect the global model.

Momentum is currently used in many studies to capture trends in historical information and optimise the FL training process. Sai et al. developed byzantine robust optimization [28], where the historical information was used in gradient descent for momentum updating to indirectly optimise training. Kerem et al.[36] proved that malicious clients can use momentum as a reference to design attacks and escape from clipping. El Mhamdi et al. [37] demonstrated that in terms of Byzantine resilience, momentum is better utilised in local clients than in global FL. The process of parameter reduction along the loss function accumulates real gradients and errors (i.e., random noise) due to the inertia effect of momentum. When there are a large number of malicious clients, their updates may cause the accumulation direction of momentum to deviate from the global optimal direction.

Some studies have built estimators [38] or similarity matrices [39] to make the use of historical information characteristics to detect and filter out abnormal updates. Meanwhile, long-term historical gradients are now used to optimize attacks. An ensemble adversarial attack [26] is proposed to promote transferability between groups by retaining learned adversarial information stored in long-term historical gradients.

3 LSH-FL: Long Short Historical Gradients For FL

3.1 Long Short Historical Gradient

Malicious clients often generate abnormal parameters that are significantly different from normal client parameters. Therefore, in defense, differentiation and filtering can be conducted on the dispersion and deviation characteristics for the parameter distribution of malicious clients and benign clients. Using historical gradients to aggregate multiple rounds of gradients can effectively reduce the occurrence of false positives that identify normal clients as malicious clients.

During FL training process, the only information possessed by the clients responsible for aggregation is the gradient values sent by other clients in each round as the result of their local optimization. Each client obtains new model parameters $w, h_w()$ by minimizing the loss function l, which distinguishes the input data x as a function of label y. In the round t, the equation for calculating the gradient value for client i is:

$$\nabla_{t,i} = w_t - \arg\min_{w} l\left(h_{i,w}(x), y\right) h_w() \tag{S1}$$

Short term historical gradient $\nabla_{t,i}^{SHG}$ is the aggregation of the recent rounds of gradients locally on the client side, client i's short-term historical gradients in the round t can be observed by aggregating the average value of gradients in the past u rounds.

$$\nabla_{t,i}^{SHG} = \frac{1}{S} \sum_{k=max\{1,E-S\}}^{E} \nabla_{t,i}^{k} \tag{S2}$$

Long-term historical gradients $\nabla_{t,i}^{LHG}$ is the aggregation of historical gradients collected by the server for each client during the model aggregation step. The long-term historical gradient stored by the server for each client i is the sum of all gradients from the past L rounds.

$$\nabla_{t,i}^{LHG} = \sum_{j=max\{1,t-L\}}^{t} \nabla_{j,i} \tag{S3}$$

Table S1: NOTATIONS

Symbol	Description
t	Number of rounds
i	Client ID
E	Number of local updates
C	Clipping bound
$N(t)$	Subset of clients in round t
n	Total number of clients
m	Number of malicious clients
λ	Historical weights
D_i	Local training data of client i
w_t	The global model for the t-th round
$w_{t,i}$	The t-th round model of client i
$\nabla_{t,i}$	The t-th round gradient of client i
$\nabla_{t,i}^{k}$	The k-th local gradient in the t-th round of client i
$h_{i,w}(x)$	Discriminant function of client i based on model w
W^*	Weight difference matrix
$\eta_{t,i}$	The t-th learning rates of client i
σ	Standard deviation of Laplace noise
S	Window size of SHG
L	Window size of LHG
$\|\Upsilon\|$	Threshold of weight difference matrix

3.2 Perturbation Based on Short Term Historical Gradients

As mentioned earlier, using historical information is effective to detect malicious clients. However, the long-term historical information is not always sufficient to extract sufficient features for analysis, such as in the first few rounds of training. Besides, the client needs to participate in defense to ensure its robustness when the attack is very strong. Therefore, the short-term historical gradients is necessary to alleviate malicious attacks through gradient aggregation and client model perturbation.

In LSH-FL, to effectively reduce the impact of attacks on the global model and accelerate convergence, we have reformed the local model training of benign devices to achieve two objectives: (1) Minimize the loss of local benign task to maintain the benign task's performance. (2) Perturbe the kernel of hessian matrix to prevent attack impacts from being hidden in the historical gradient of benign devices.

Algorithm S1 Perturbation Algorithm Based on Short-term Historical Gradients (P-SHG)

Input: $\nabla_{t,i}$, $\nabla^1_{t,i}$, $\nabla^2_{t,i} \ldots \nabla^E_{t,i}$, $\nabla^{SHG}_{t-1,i}$, S, σ
Output: Perturbed gradients $\widetilde{\nabla}_{t,i}$
Initialization: Laplace noise matrix Υ with mean 0 and standard deviation σ
for each client i **do**
  Calculate the short-term historical gradient $\nabla^{SHG}_{t,i} = \frac{1}{S}\sum_{k=max\{1,E-S\}}^{E} \nabla^k_{t,i}$
  Reorganize gradients $\nabla^{SHG}_{t,i}$ and $\nabla^{SHG}_{t-1,i}$ into patches
  for each patch j in $\nabla^{SHG}_{t,i}$ **do**
    Calculate weight difference matrix $W^* = \nabla^{SHG}_{t,i,j} - \nabla^{SHG}_{t-1,i,j}$
    if $|W^*_j| \le |\Upsilon|$ **then**
      $\widetilde{\nabla}_{t,i,j} = \nabla^{SHG}_{t,i,j} + \eta_{t,i}\Upsilon$
    end if
  end for
end for
Return $\widetilde{\nabla}_{t,i}$

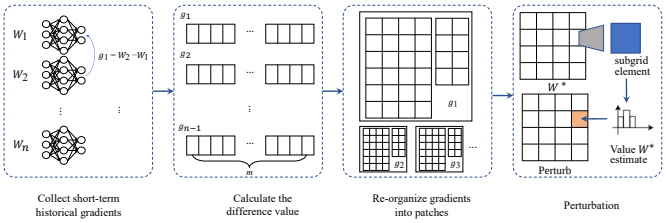


Fig. S2: The perturbation process on the client using P-SHG. Firstly, the client collects gradients to obtain SHGs, and then evaluates the differences between SHGs in blocks to establish a judgment matrix, which is used to determine whether to perturb.

The P-SHG algorithm is illustrated in Algorithm S1 and Fig. S2. Firstly, P-SHG Randomly generate a Laplace noise matrix Υ with $mean = 0$ and $std = \sigma$. In each training batch, client i first uses local data to train the model, calculates gradients, and updates model parameters. Then, based on the difference in model parameters between the previous and current rounds, a weight difference matrix W^* of short-term gradients is calculated. Next, compare $W*$ with the threshold $|\Upsilon|$ in each dimension j and then find the index of elements that are less than or equal to zero and perturb their corresponding model parameters by multiplying them. The Laplace noise matrix introduced has a certain degree of randomness, which increases the uncertainty of model parameters and hinders the effectiveness of attacks.

The server receives gradient updates uploaded by the currently selected client $N(i)$, and the Laplace mechanism perturbs the gradient updates, which not only reduces the impact of the attack but also prevents the server from obtaining personal information of participants through the uploaded gradients. The differential privacy method of reference gradient

Algorithm S2 Defense Algorithm Based on Long Term Historical Gradients (D-LHG)

Input: ω_t, n, $N(i)$, $\nabla_{1,i}$, $\nabla_{2,i} \ldots \nabla_{t,i}$, L.
Output: ω_{t+1}
Initialization: s_i: scores of client s_i
W_0: initial weight based on the amount of client data D_i
// Client side
for each client $j \in t$ and $i \in n$ **do**
  $\widetilde{\nabla}_{j,i} = P - SHG(\nabla_{j,i})$
end for
// Server side
for each client $i \in N(i)$ **do**
  $\nabla^{LHG}_{t,i} = \sum_{j=max\{1,t-L\}}^{t} \widetilde{\nabla}_{j,i}$ ⇐ Compute long-term historical gradient
  Flatten the ∇^{LHG}_{i} into $\nabla_{fl,i}$
  $\mu = \frac{1}{|N(i)|}\sum_{i \in N(i)} \nabla_{fl,i}$ ⇐ Compute the mean of flattened gradients
  Compute the centered gradients $\nabla^c = \nabla_{fl} - \mu$
  Decompose ∇^c, extract the top right singular vector matrix as v
  Calculate scores of client $s_i = (\langle \nabla_i - \boldsymbol{\mu}, v \rangle)^2$
end for
Sort s_i and get the sorted indices $[s_i]$.
Calculate aggregate weight W_t based on W_0 and $[s_i]$
$w_{t+1} = w_t + \sum_{i \in N(i)} W_t \nabla_{t,i}$
return w_{t+1}

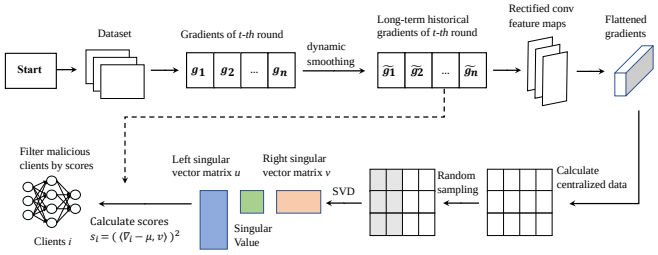


Fig. S3: The defense process on the server using D-LHG. Firstly, obtain the long-term historical gradients of each client stored on the server, then perform flattening and sampling to calculate the right singular matrix, and finally calculate the scores for each client.

desensitization or gradient depersonalization first prunes the gradient to ensure that the gradient normal form of all samples is less than the upper limit value of the gradient clipping bound C. Then, Laplace noise is added to the cropped gradient to achieve the privacy protection effect of differential privacy.

$$\nabla_{N(i)} = \nabla_{N(i)} / \max\left(1, \frac{\left\|\nabla_{N(i)}\right\|^2}{C}\right) \tag{S4}$$

Differential privacy must account for gradient distortion, controlled by two factors: noise variance σ^2 and mask M. The noise variance σ^2 can be directly controlled by the standard deviation σ of Laplacian noise, and the mask M is determined by the short-term historical gradient difference W^*. Excessive noise variance can destroy gradient information, hindering effective model updates. To prevent excessive dis-

turbances, the mask M filters perturbations. If the absolute value of a perturbation exceeds the original parameter, M is set to 1, and the perturbation is added to the model. Otherwise, M is set to 0, zeroing out the perturbation.

3.3 Defense Based on Long Term Historical Gradients

Estimators or similarity matrices have been built to make the use of historical information characteristics for malicious client detection. However, limitations emerges when the proportion of malicious clients is large. Therefore, an adapted DnC method, called LH-DnC, is used on long-term historical gradients, where an important step is singular value decomposition (SVD) to detect and remove outliers. Specially, the right singular matrix contains all own characteristics, to effectively identify malicious gradient features against attacks in FL. To address the issue of the large memory and computational costs on high dimensional gradients, LH-DnC randomly samples its input gradients and flattens gradients, achieving dimensionality reduction and convergence under long-term historical gradients.

The D-LHG algorithm is illustrated in Algorithm S2 and Fig. S3. Firstly, D-LHG calculate LHG. If current rounds less than or equal to l, we compute LHG for client over last t rounds. Otherwise, compute LHG for client i over the last l rounds. Then, apply P-SHG function to ∇_i^{LHG} to obtain perturbed gradient $\widetilde{\nabla}_i$. Next, D-LHG computes a centered subsampled set ∇^c of $\widetilde{\nabla}_i$ using mean vector μ of flattened model gradient of $\widetilde{\nabla}_i$. Then D-LHG computes projections of centered gradients along their top right singular eigenvector v, computes a vector of outlier scores s, and building a sorting index for scores. Calculate aggregate index W_t based on W_0 and $[s_i]$. Finally, D-LHG use aggregate function to compute the next round model w_{t+1} using current round gradient $\nabla_{t,i}$, current global model parameters ω_t, and aggregate index W_t.

By tracking the dynamic behaviors of malicious nodes using historical gradients, LH-DnC can make robust decisions to detect malicious local updates, even the proportion of malicious clients nears 50%, where the detection capability of DnC starts to falter, as illustrated in Fig. S4 the effectiveness on both MNIST and CIFAR-10 datasets. It is found that LH-DnC outperforms DnC even though the increased complexity of the dataset makes it more challenging to distinguish between malicious and non-malicious clients.

To avoid the issue of excessive accumulation of long-term historical gradients, direct summation is impractical. Instead, exponential smoothing is employed, which balances the short-term and long-term influence of client malice levels. It smooths time series data by assigning greater weights to recent data and lesser weights to older data, effectively capturing evolving trends in data changes.

$$\overset{\sim}{\nabla}_t^b = (1-\lambda)\nabla_t^b + \lambda \overset{\sim}{\nabla}_{t-1}^b \tag{S5}$$

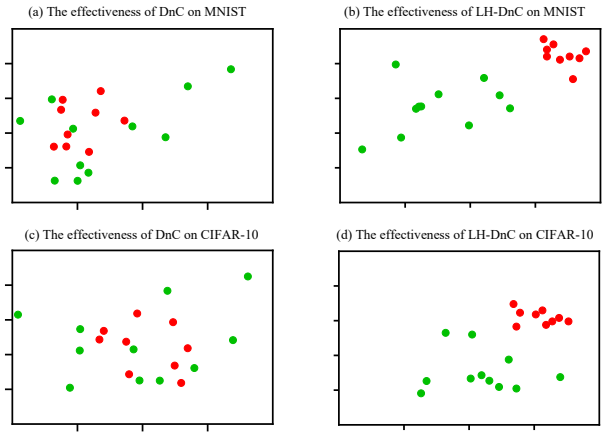


Fig. S4: Utilizing the DnC and LH-DnC apporoach to project gradients onto a two-dimensional surface. Specifically, we plot a total of 20 local updates at the 50th epoch of the training process using MNIST (top) and CIFAR-10 (bottom). Within the plotted updates, red dots represent malicious updates, while green dots represent benign ones.

3.4 The Design of LSH-FL Method

In this section the overview of LSH-FL is introduced, which is mainly composed of four steps like the classic FL method and runs in a loop during the training process.

Model synchronization: In first round, the server initializing the global model w and determines hyperparameters. in round t, the server randomly selects a subset of clients N(t) as participants and synchronizes the global model with clients.

Local model train: In LSH-FL, each client firstly performs several rounds of local training on own data D to obtain stable short-term historical gradients (SHG).

Local model upload: Before uploading, the short-term historical gradients are perturbed through the P-SHG algorithm, determining whether to filter the perturbation parameters and scale the perturbation with differential privacy requirements.

Model aggregation: The central server performs D-LHG algorithm to verify the gradients of clients $N(t)$. The clients with abnormal scores will be removed from the model aggregation. The server updates the global model by taking a weighted average gradient of benign clients $N(b)$, as shown in Equation S6:

$$w_{t+1} = w_t + \sum_{i=1}^{N(b)} \frac{|D_i|}{|D_{N(b)}|} \nabla_{i,t} \tag{S6}$$

4 Experiments

4.1 Experimental Setup

Datasets. We consider four widely-used benchmark datasets CIFAR-10, CIFAR-100 [40], MNIST and Fashion MNIST [41] to evaluate LSH-FL. We simulate the Non-IID FL by sampling $p_i \sim Dir_N(\alpha)$, where α is the parameter of the Dirichlet distribution. The level of heterogeneity among local

Table S2: Attack impacts (%) of 5 poisoning attacks defended by our LSH-FL and existing defenses on CIFAR-10 and MNIST datasets.

Dataset	Defenses	Attack Type				
		LF	BF	LIE	Min-Max	Min-Sum
MNIST	LSH-FL	**0.47**	**0.51**	**0.64**	**0.73**	**1.06**
	FLDetector	3.32	2.45	3.71	2.21	3.31
	Gas	1.61	1.85	2.13	3.23	4.11
	Muti-Krum	1.41	2.45	2.23	3.89	4.32
	Trim	4.83	3.45	3.85	4.23	5.11
Fashion MNIST	LSH-FL	**1.27**	**1.21**	**1.94**	**3.63**	**1.86**
	FLDetector	3.42	2.32	10.21	12.38	10.04
	Gas	11.05	9.89	20.58	24.69	37.88
	Muti-Krum	10.21	18.44	23.43	48.21	51.11
	Trim	17.78	20.28	42.21	40.26	48.24
CIFAR-10	LSH-FL	**3.27**	**2.21**	**3.94**	**5.63**	**6.76**
	FLDetector	6.51	5.13	14.27	8.28	20.32
	Gas	9.64	7.4	13.21	19.68	15.28
	Muti-Krum	16.21	23.13	32.35	38.49	35.21
	Trim	10.12	14.01	37.14	44.12	40.13
CIFAR-100	LSH-FL	**1.89**	**1.23**	**1.04**	**3.41**	**4.81**
	FLDetector	3.24	6.57	13.14	8.28	12.24
	Gas	9.91	9.77	13.88	24.51	21.14
	Muti-Krum	12.32	23.19	32.33	36.74	48.21
	Trim	13.27	24.08	41.01	46.21	51.22

datasets across different clients decreases when α increases. We changed it in different Non-IID scenarios, where $\alpha = 0.1$ indicates a higher level Non-IID effect that can better distinguish the attack impact, and $\alpha = 1$ is more in line with the actual data distribution among distributed clients, which can better account for the global model performance [42].

Machine learning models. We consider four presentative DNN models: CNN (which consists of two convolutional layers with 32 and 64 filters), AlexNet, ResNet18 and LR. In particular, CNN, AlexNet, ResNet18 and LR serve as the global model architecture for CIFAR-10, MNIST, CIFAR-100 and Fashion MNIST, respectively.

Parameter settings. Experiments are implemented in PyTorch [43] on Python 3.7 with RTX 3060 GPU. Each experiment is conducted 10 times independently to report the average result, omitting variances with small observations. By default, a total number of $N = 50$ clients are included. In each round, server randomly selects $n = 20$ clients to participate, with $m = 4$ of them being malicious clients. The effect of the percentage of malicious clients on global model accuracy of MNIST and CIFAR-10 is illustrated in Fig. S5. Each client applies $E = 5$ rounds of the stochastic gradient descent to update its local model. Different training rounds and learning rate η settings are adopted as follows: 100 rounds and $\eta = 0.1$ for CNN on MNIST datasets, 500 rounds and $\eta = 0.01$ for Alexnet on CIFAR-10 datasets. The batch size is set to be 64. The importance of historical information is a tunable parameter. We set $\lambda = 0.5$ in all of our experiments where the LSH-FL method is demonstrated to be the best defense through the following sensitivity analysis.

Defenses and attacks. We consider four defense methods: Muti-Krum [15], Trim [19], FABA [16], Gas [44] and FLDetector [30]. The methods ZENO [24] and FLTrust [25] are not considered due to its additional requirement of a clean validation dataset. The following five representative poisoning attacks are cosidered: Lable Flip [32], Sign Flip [33], LIE [11], Min-Sum and Min-Max [12].

4.2 LSH-FL Most Effective in Mitigating the Impact of Attack

The accuracy of benchmark models varies widely across datasets. Therefore, we use attack impact as an evaluation metric, which means the decrease in model accuracy under attack compared to the baseline (i.e., no attack). Table S2 shows the attack impact on LSH-FL compared with other state-of-the-art defenses, against different poisoning attacks on the four datasets. It can be seen that LSH-FL significantly reduces the impact of state-of-the-art poisoning attacks in different cases.

The experimental results over the MNIST dataset show that LSH-FL outperforms state-of-the-art solutions against all the five types of attacks. Specifically, under different attacks, LSH-FL achieves an attack impact of less than 1%, implying that LSH-FL performs almost identically to the baseline. Followed LSH-FL are FLDetector and Muti-Krum, with the average attack impact of about 3%. There is also no significant difference in the attack impact for Trim, approximately 3% to 5%. FABA performs the worst, especially on Min-Max and Min-Sum, with attack impacts of 14.82% and 11.32%, respectively.

On the Fashion MNIST dataset, the decrease in global model accuracy of LSH-FL is still small under different attacks. However, the other defense strategies are greatly affected by attacks. Overall, the attack impact for different defense methods on the Fashion MNIST dataset is higher than that on the MNIST dataset. However, LSH-FL and FLDetector using historical information show smaller increases, while FABA, Trim, and Muti-Krum using simple statistical or distance discrimination show larger increases.

In addition, under the same conditions, the attack has a greater impact on the CIFAR-10 dataset, because the CIFAR-10 dataset is more heterogeneous than the MNIST dataset, making the attack less likely to be detected and having a greater impact. Using historical data can process Non-IID data and reduce differences in updates, making the gradients

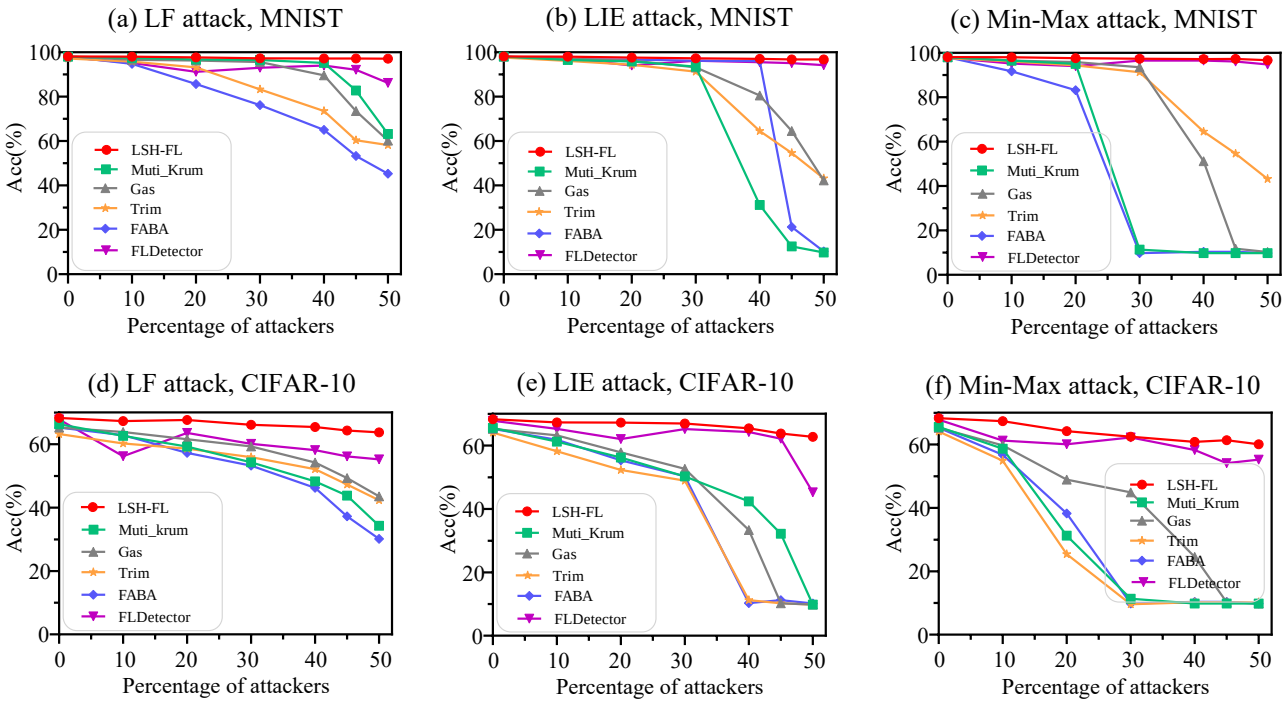


Fig. S5: The global model accuracy of our LSH-FL and existing defenses on MNIST (top) and CIFAR-10 (bottom) under poisoning attacks. (a) LF attack. (b) LIE attack. (c) Min-Max attack.

in training similar to IID type settings. LSH-FL performs far better on Non-IID data such as CIFAR-10 and CIFAR-100 than other defense methods.

4.3 Effect of the Proportion of Malicious Clients

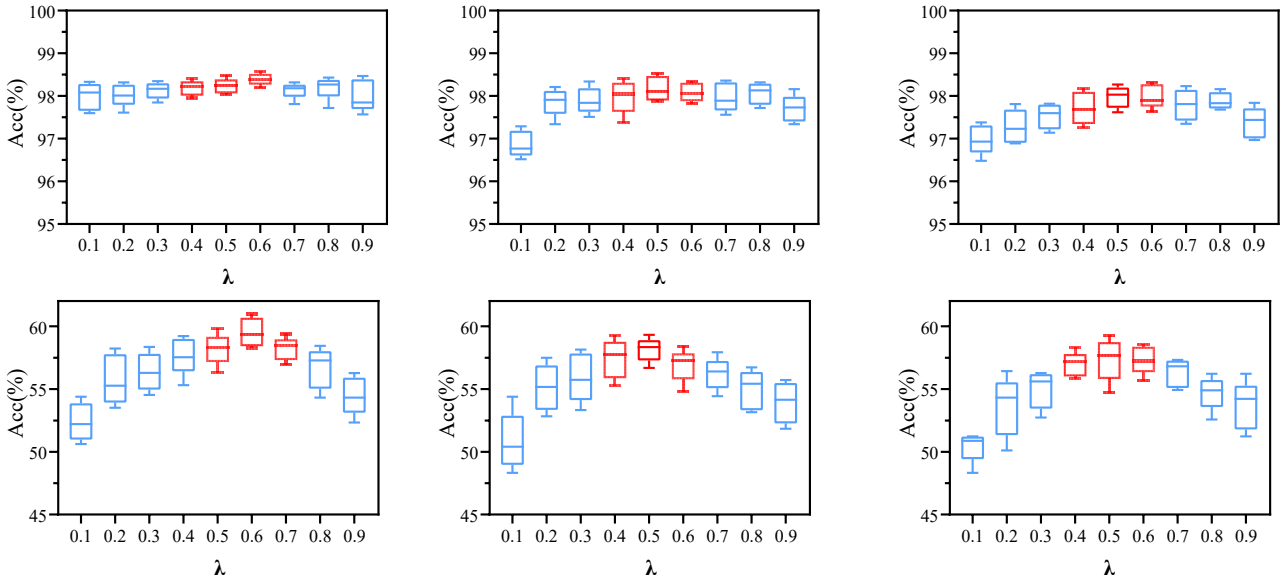
Fig.S5 shows that LSH-FL maintains a top level of accuracy under attacks with different proportions of malicious clients. In addition, it consistently outperforms state-of-the-art defense methods.

For LF attack on the MNIST dataset, the accuracy of LSH-FL is pretty high about 98% on the MNIST dataset, almost unaffected by the proportion of attackers. Besides, it is still good for FLDetector, Gas, and Muti-Krum to defense against situations where the proportion of attackers does not exceed 40% on the MNIST dataset. Except for FLDetector, reference defense methods are almost ineffective when half clients are attackers. On the CIFAR-10 dataset, LSH-FL performs best among defenses, with the accuracy drops slightly to 65% when the proportion of attackers reaches 50%. The accuracy of all other defense methods has also declined as the proportion of attackers increases. A special case is that FLDetector performs not that good in the attacker ratio of 10% to 20%, as it requires a large number of malicious clients to establish historical information to ensure the accuracy of prediction and detection.

When against LIE attack or Min-Max attack, LSH-FL is still the one with the best defense, followed by FLDetector. Eventually, FLDetector's accuracy in the CIFAR-10 dataset drops significantly as the attacker percentage approaches 50%.

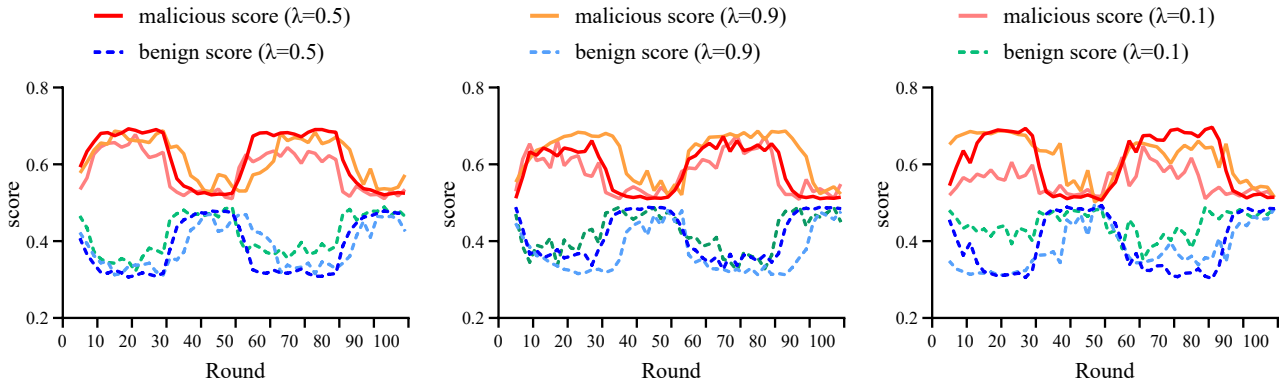
4.4 Sensitivity of Historical Weights λ

We leverage long-term historical gradients to detect outlier-scores via a exponential smoothing rule in Equation S5. Therefore, the sensitivity of the Historical Weight λ should be evaluated, which is considered as a candidate values from 0 to 1. The historical weight λ is proportional to the use of historical information in malicious client detection, where a low λ means that the current round of information is mainly used for malicious client detection, and a large λ means that historical information is mainly used for malicious client detection. Fig. S6 shows the global model accuracy under multiple experiments with different λ used in LSH-FL to defense against state-of-the-art attacks on MNIST and CIFAR-10. Given the three attacks, as the historical weight λ increases, the accuracy of the global model shows an inverted V-shaped trend of increasing and then decreasing. When the global model accuracy increases to the highest λ is in the middle of the interval (0,1), where λ is 0.6 on LF, 0.5 on LIE and MIN-MAX. When λ is less than the critical value, as λ increases, LSH-FL begins to consider both historical information and current gradients on malicious client identification, and the accuracy of the model gradually increases. Especially, when the weight λ is 0 or 0.1, the global model accuracy under different malicious attacks are significantly lower than other situations. This reveals the fact that contributes to the reduced accuracy



(a) LF attack (b) LIE attack (c) Min-Max attack

Fig. S6: The impact of Historical Weight λ on global model accuracy when three poisoning attacks are defended by our LSH-FL on MNIST (top) and CIFAR-10 (bottom), where the proportion of attackers is 40%. Setting $\lambda \in [0.4, 0.6]$ in different situations produces sound performances.



(a) LF attack (b) LIE attack (c) Min-Max attack

Fig. S7: Dynamics of the clients' suspicious scores when malicious clients perform attacks periodically on CIFAR-10, where the proportion of attackers is 40%.

is the current gradient information of the very large weights for identifying malicious clients.

Since the model relies too much on historical information and can not generalize to new inputs, when λ exceeds the critical value, the global model accuracy under different malicious attacks decreases, but not significantly. Meanwhile, although the global model accuracy at high λ ($\lambda \geq 0.9$) is lower than the peak value, it is relatively higher than the global model accuracy at low λ ($\lambda \leq 0.1$). Too high a historical weight λ would make our method lose sensitivity to client changes in the current round, resulting in LSH-FL not being well adapted to changes in the upload gradient (e.g., periodic attacks) and reacting quickly. In addition to the average value of accuracy, we can also see that the global model accuracy fluctuates relatively more when $\lambda \leq 0.2$. Considering that setting $\lambda \in [0.4, 0.6]$ in different situations produces sound performances, we set $\lambda = 0.5$ in all of our experiments for real combination of short and long history gradients.

4.5 Detection of Periodic Attacks through Suspicious Scores

Fig. S7 shows the average suspicious scores of benign clients and malicious clients as a function of the training round t. To clearly illustrate the dynamics of the suspicious scores, we assume that malicious clients execute attacks in the first 30 rounds of every 50 rounds, starting from the 5th round. Note that LSH-FL is ignorant of the specific times when the attack starts and ends. We observe that the periodic pattern

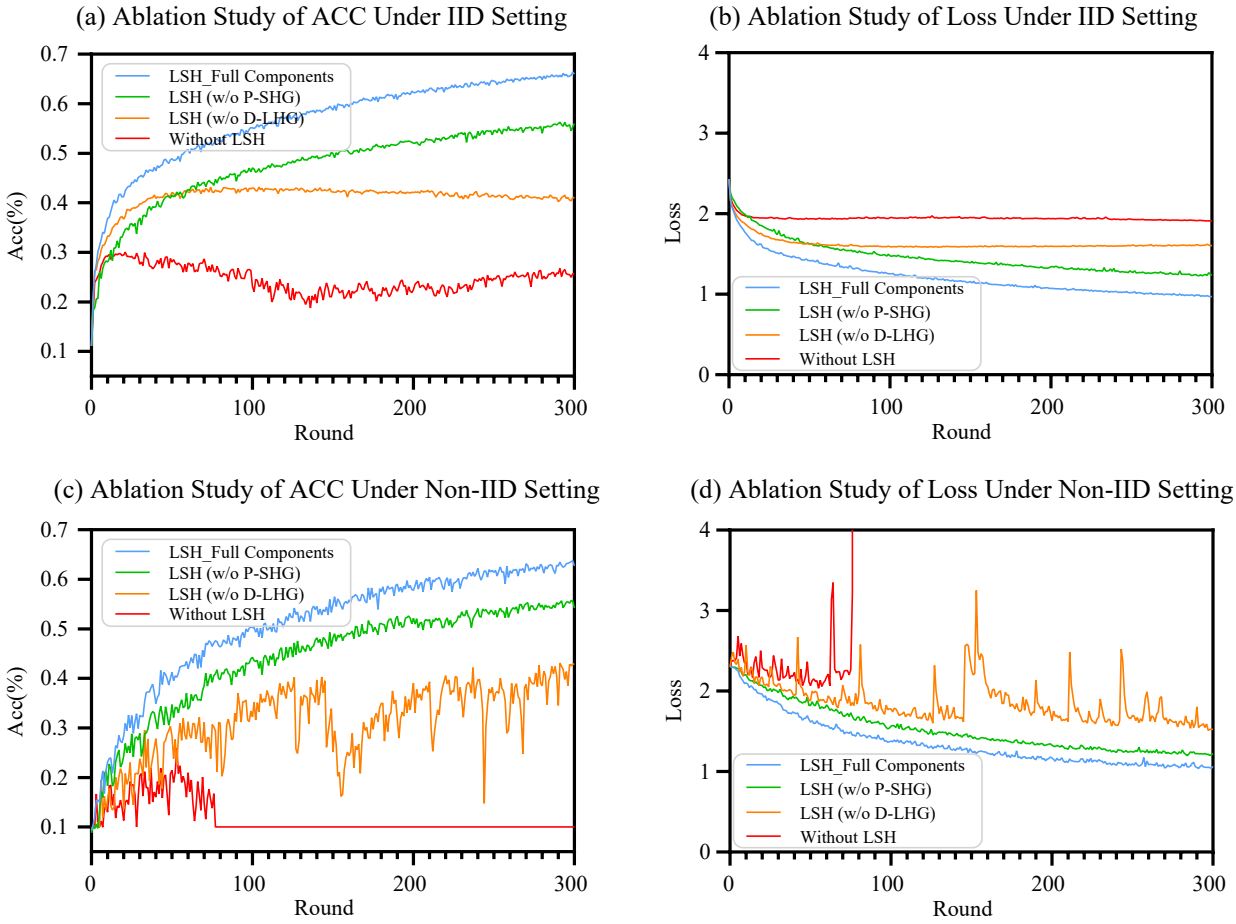


Fig. S8: Performance comparison of ablations across communication rounds under LIE attack on MNIST (top) and CIFAR-10 (bottom) where the proportion of attackers is 40%. Removing any component results in a performance drop.

of suspicious scores conforms to the attack pattern. In particular, the average suspicious score of malicious clients increases rapidly at the beginning of the attack, and decreases to roughly the same level as the average suspicious score of benign clients when the attack stops. In the rounds with attacks, malicious and benign clients can be well separated based on the suspicious scores. In these experiments, LSH-FL can detect malicious clients at about 35th round. Similarly, suspicious values can return the normal range within a short period of time after stopping the attack at about 55th round.

Because LSH-FL utilizes rounds of historical information, akin to a reputation mechanism where the influence of past malicious (or normal) behaviors on current assessments is regulated by a historical weight λ. A higher λ results in the judgment of the current round being greatly influenced by historical information, making periodic attack identification challenging. With $\lambda = 0.9$ as an example, detection responses are delayed by approximately five rounds compared to normal conditions, and misjudgments persist post-attack cessation as observed. Conversely, a lower λ such as $\lambda = 0.1$ facilitates faster response to attack dynamics, yet may obtain a lower malicious value, making it harder to detect malicious clients.

4.6 Complementarity of perturbation and defense in LSH-FL

To investigate the contribution of different components, we have conducted an ablation study by removing and altering the key components. We compare the following variations:

- **LSH_Full Components:** Use all components, including P-SHG (Section 2) and D-LHG (Section 3).
- **LSH_Step1:** Only using the perturbation part P-SHG, with D-LHG removed.
- **LSH_Step2:** Only using the defense part D-LHG, with P-SHG removed.
- **Without LSH:** Do not use any LSH-FL components.

Fig. S8 shows the changes in the accuracy and loss of models with different components as the number of rounds increases. Overall, except for a few fluctuation rounds, Full Components (blue line) has a smallest loss and highest accuracy, followed by LSH_Step2 (green line), LSH_Step1 (yellow line), and Without LSH (red line). As illustrated in Fig.

S8, Without LSH is severely affected and then completely destroyed by malicious attacks, with loss exceeding the limit and accuracy decreasing to around 10%. LSH_Step1 outperforms the Without LSH, suggesting that perturbation in Step1 can alleviate the attack. Using only Step1 shows more robust performance than the Without LSH and is not compromised by attackers after a certain number of rounds. Moreover, the impact on attacks can not be completely eliminated by perturbation, just as LSH_Step1 dose not work as well as LSH_Step2, indicating the availability of D-LHG to resist attacks of malicious clients. The difference in performance between using only Step2 and using all components is due to the possibility of non-detection of malicious clients, whereas the perturbation in Step1 helps to mitigate the effects of the attack and speeds up convergence, justifying the need for the perturbation in Step1.

Under IID conditions, removing certain components does not lead to model divergence but instead results in a reduced level of accuracy. Moreover, the fluctuations in accuracy and loss during training are notably smaller compared to the non-IID scenario. In IID settings, the model exhibits greater resilience to missing components, as evidenced by the significantly diminished fluctuations in performance metrics when components are omitted. Even in the absence of perturbation and defense strategies, the model does not experience catastrophic failure. However, its performance still lags behind the baseline where all components are present, highlighting the collective contribution of the components to the model's robustness and convergence rate, particularly when addressing more heterogeneous, non-IID data

5 Discussion

Historical gradients mitigate poisoning attacks in two primary ways. First, the perturbation of short-term historical gradients introduces controlled noise to disrupt malicious gradients hidden within the natural gradient noise. Attackers often embed small, subtle perturbations into the gradients, which can accumulate over time. The short-term perturbation prevents these malicious accumulations from significantly affecting the model. Second, through the analysis of long-term historical gradients, it becomes possible to identify and detect malicious gradients that may remain undetected in individual training rounds. These detected malicious gradients can be excluded from the aggregation process or assigned smaller weights during the federated learning global model aggregation. Together, these two strategies provide a comprehensive defense, utilizing the information in historical gradients to detect and mitigate the impact of long-term poisoning attacks.

The inclusion of historical gradients takes into account resource consumption and computational complexity. Short-term historical gradients do not require additional storage, because their dimensions or structure do not change when

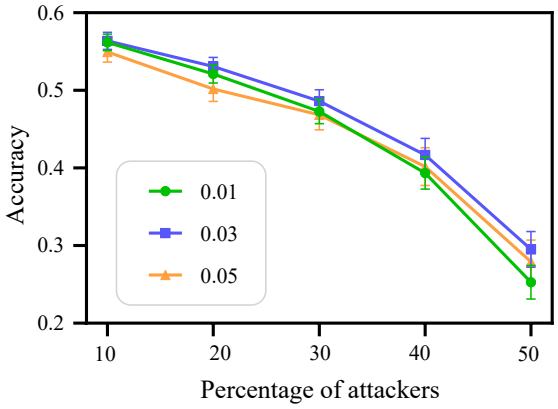


Fig. S9: Impact of noise magnitude on model accuracy with varying malicious proportions under LIE attack on CIFAR-10.

the client is perturbed. For long-term historical gradients, exponential smoothing is used to keep only one set of gradients on the server, which is updated once per round, minimizing memory requirements and ensuring efficient use of resources. Besides, our approach does not alter the standard Federated Learning process and therefore does not significantly affect communication costs. Specifically, the global model synchronization remains unchanged, and gradient uploads are consistent with typical practice. Regarding computational complexity, the primary additional computation arises from applying SVD to historical gradients for anomaly detection. Although the SVD is computationally intensive, it is only used on flattened and centred sets of gradients, thus ensuring that the overall computational effort remains manageable. In summary, while integrating historical gradients has a modest memory cost and slightly increases computational complexity, the impact on communication costs is negligible, and the overall system remains efficient and feasible in practice.

When facing with the Non-IID data, the gradients among clients could also vary a lot. It is indeed challenging to distinguish vary gradients caused by Non-IID data and the malicious clients. Targeted attacks typically produce gradient patterns with direction-specific features for certain outcomes, and these gradients are easily distinguishable from those produced by Non-IID data. Moreover, non-targeted malicious clients typically hope to achieve a stronger attack effect with fewer malicious participants, in which case the gradients from malicious clients tend to be more similar than the natural variance caused by Non-IID data between honest clients. However, when faced with extremely high data heterogeneity, especially in the absence of explicit regularization or handling of Non-IID data, gradients from different clients may resemble malicious patterns, thus complicating the distinction between Non-IID effects and malicious behaviour. In our work, long-term historical gradients are leveraged to

identify stable patterns, thereby reducing randomness in identification and allowing us to more consistently distinguish the effects of Non-IID data from malicious gradients.

To clarify the noise effects introduced in P-SHG, we presents the training results of our proposed LSH-FL method in Fig. S9, evaluated under varying noise magnitudes and proportions of malicious clients on the CIFAR-10 dataset. The values indicate the mean accuracy and standard deviation over 200 training rounds. As the proportion of malicious clients increases, model accuracy tends to decrease due to poisoning attacks. However, introducing controlled noise perturbations helps mitigate this accuracy drop, especially when the proportion of malicious clients is high, suggesting that noise can act as a defensive mechanism to reduce the impact of malicious clients on the model's performance. Notably, while noise perturbation enhances robustness against adversarial clients, it introduces a slight trade-off, as higher noise levels lead to greater variability in accuracy, which is reflected in the increasing standard deviation. This highlights the balance between defense effectiveness and model stability. In our work, noise is used not only for privacy protection but primarily to strengthen the robustness of FL, aiming to counteract hidden perturbations from attackers without hindering model convergence. The optimal noise level is determined through iterative tuning, thus striking a balance between security and accuracy.

6 Conclusion

FL has been enormously successful in building high-precision models and ensuring privacy protection in many fields. However, defending against malicious attacks and identifying diverse malicious clients remain significant challenges. In this work, a new FL defense mechanism LSH-FL is designed to combine short and long historical gradients.. Spotlight on the interplay between long and short historical gradients, the P-SHG and D-LHG algorithms are offered to identify malicious model updates while mitigating the effects of poisoning attacks. Experiments carried out on four benchmark datasets suggest that LSH-FL significantly mitigates the impact of state-of-the-art poisoning attacks in different cases, maintaining a top level of accuracy under attacks with different proportions of malicious clients. Note that LSH-FL outperforms the best referenced defense method FLDetector in all experimental conditions. Specifically, under different attacks over the MNIST dataset with IID distribution, LSH-FL achieves an attack impact of less than 1%, while the average attack impact is about three times more under FLDetector. Also, LSH-FL performs far better on Non-IID data such as CIFAR-10 and CIFAR-100 than other state-of-the-art solutions, especially when the proportion of attackers exceeds 40%, the state-of-the-art defenses referenced are almost ineffective. Further ablation study gives an insight into the complementarity of defense and perturbation in LSH-FL, as removing any component results in performance degradation. In the era dominated by artificial intelligence and data elements, we anticipate a broad application of the LSH-FL framework in a variety of federated learning enabled applications.

7 Acknowledgments

This work was supported by The National Science and Technology Major Project (Grant: 2021ZD0201302) and the Fundamental Research Funds for the Central Universities (Grant: YWF-23-Q-1092) and Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing.

References

1. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

2. Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

3. Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

4. Christos Xenofontos, Ioannis Zografopoulos, Charalambos Konstantinou, Alireza Jolfaei, Muhammad Khurram Khan, and Kim-Kwang Raymond Choo. Consumer, commercial, and industrial iot (in) security: Attack taxonomy and case studies. *IEEE Internet of Things Journal*, 9(1):199–221, 2021.

5. Mohamed Amine Ferrag, Burak Kantarci, Lucas C Cordeiro, Merouane Debbah, and Kim-Kwang Raymond Choo. Poisoning attacks in federated edge learning for digital twin 6g-enabled iots: An anticipatory study. In *2023 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1253–1258. IEEE, 2023.

6. Wei Wan, Jianrong Lu, Shengshan Hu, Leo Yu Zhang, and Xiaobing Pei. Shielding federated learning: A new attack approach and its defense. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–7. IEEE, 2021.

7. Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor

federated learning. In *International conference on artificial intelligence and statistics*, pages 2938–2948. PMLR, 2020.

8. Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*, pages 480–501. Springer, 2020.

9. Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daume III, and Tudor Dumitras. When does machine learning fail? generalized transferability for evasion and poisoning attacks. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1299–1316, 2018.

10. Mohammad Al-Rubaie and J Morris Chang. Privacy-preserving machine learning: Threats and solutions. *IEEE Security & Privacy*, 17(2):49–58, 2019.

11. Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. *Advances in Neural Information Processing Systems*, 32, 2019.

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

13. Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to byzantine-robust federated learning. In *29th USENIX security symposium (USENIX Security 20)*, pages 1605–1622, 2020.

14. Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643. PMLR, 2019.

15. Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30, 2017.

16. Qi Xia, Zeyi Tao, Zijiang Hao, and Qun Li. Faba: an algorithm for fast aggregation against byzantine attacks in distributed neural networks. In *IJCAI*, 2019.

17. Rachid Guerraoui, Sébastien Rouault, et al. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, pages 3521–3530. PMLR, 2018.

18. Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659. Pmlr, 2018.

19. Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659. Pmlr, 2018.

20. Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Slsgd: Secure and efficient distributed on-device machine learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 213–228. Springer, 2019.

21. Mengyao Ma, Yanjun Zhang, Pathum Chamikara Mahawaga Arachchige, Leo Yu Zhang, Mohan Baruwal Chhetri, and Guangdong Bai. Loden: Making every client in federated learning a defender against the poisoning membership inference attacks. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, pages 122–135, 2023.

22. Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*, pages 480–501. Springer, 2020.

23. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

24. Cong Xie, Sanmi Koyejo, and Indranil Gupta. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *International Conference on Machine Learning*, pages 6893–6901. PMLR, 2019.

25. Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. *arXiv preprint arXiv:2012.13995*, 2020.

26. Zhaohui Che, Ali Borji, Guangtao Zhai, Suiyi Ling, Jing Li, and Patrick Le Callet. A new ensemble adversarial attack powered by long-term gradient memories. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3405–3413, 2020.

27. Minghui Li, Wei Wan, Jianrong Lu, Shengshan Hu, Junyu Shi, Leo Yu Zhang, Man Zhou, and Yifeng Zheng. Shielding federated learning: Mitigating byzantine attacks with less constraints. In *2022 18th International Conference on Mobility, Sensing and Networking (MSN)*, pages 178–185. IEEE, 2022.

28. Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. Learning from history for byzantine robust optimization. In *International Conference on Machine Learning*, pages 5311–5319. PMLR, 2021.

29. Zeyuan Allen-Zhu, Faeze Ebrahimian, Jerry Li, and Dan Alistarh. Byzantine-resilient non-convex stochastic gradient descent. *arXiv preprint arXiv:2012.14368*, 2020.

30. Zaixi Zhang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2545–2555, 2022.

31. Jingwei Sun, Ang Li, Louis DiValentin, Amin Hassanzadeh, Yiran Chen, and Hai Li. Fl-wbc: Enhancing robustness against model poisoning attacks in federated learning from a client perspective. *Advances in neural information processing systems*, 34:12613–12624, 2021.

32. Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

33. Liping Li, Wei Xu, Tianyi Chen, Georgios B Giannakis, and Qing Ling. Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1544–1551, 2019.

34. Zhangming Chan, Juntao Li, Xiaopeng Yang, Xiuying Chen, Wenpeng Hu, Dongyan Zhao, and Rui Yan. Modeling personalization in continuous space for response generation via augmented wasserstein autoencoders. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)*, pages 1931–1940, 2019.

35. Hidde Lycklama, Lukas Burkhalter, Alexander Viand, Nicolas Küchler, and Anwar Hithnawi. Rofl: Robustness of secure federated learning. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 453–476. IEEE, 2023.

36. Kerem Özfatura, Emre Özfatura, Alptekin Küpçü, and Deniz Gündüz. Byzantines can also learn from history: Fall of centered clipping in federated learning. *IEEE Transactions on Information Forensics and Security*, 2023.

37. El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Louis Alexandre Rouault. Distributed momentum for byzantine-resilient stochastic gradient descent. In *9th International Conference on Learning Representations (ICLR)*, 2021.

38. Minghui Li, Wei Wan, Jianrong Lu, Shengshan Hu, Junyu Shi, Leo Yu Zhang, Man Zhou, and Yifeng Zheng. Shielding federated learning: Mitigating byzantine attacks with less constraints. In *2022 18th International Conference on Mobility, Sensing and Networking (MSN)*, pages 178–185. IEEE, 2022.

39. Zhaosen Shi, Xuyang Ding, Fagen Li, Yingni Chen, and Canran Li. Mitigation of a poisoning attack in federated learning by using historical distance detection. *Annals of Telecommunications*, 78(3):135–147, 2023.

40. Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

41. Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

42. Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1354–1371. IEEE, 2022.

43. Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

44. Yuchen Liu, Chen Chen, Lingjuan Lyu, Fangzhao Wu, Sai Wu, and Gang Chen. Byzantine-robust learning on heterogeneous data via gradient splitting. In *International Conference on Machine Learning*, pages 21404–21425. PMLR, 2023.