Re-examine Federated Rank Learning: Analyzing Its Robustness Against Poisoning Attacks

1st Xiaofei Huang
Institute of Information Engineering, Chinese Academy of Sciences
State Key Laboratory of Cyberspace Security Defense
School of Cyber Security, University of Chinese Academy of Sciences
Beijing, China
huangxiaofei@iie.ac.cn

2nd Xiaojie Zhu*
King Abdullah University of Science and Technology
Thuwal, Kingdom of Saudi Arabia
xiaojie.zhu@kaust.edu.sa

3rd Chi Chen
Institute of Information Engineering, Chinese Academy of Sciences
State Key Laboratory of Cyberspace Security Defense
School of Cyber Security, University of Chinese Academy of Sciences
Beijing, China
chenchi@iie.ac.cn

4th Paulo Esteves-Veríssimo
King Abdullah University of Science and Technology
Thuwal, Kingdom of Saudi Arabia
paulo.verissimo@kaust.edu.sa

***Abstract*—Federated learning decentralizes the training process across clients, allowing clients that do not trust each other to collaboratively train machine learning models without sharing private local data. However, this decentralized approach makes federated learning vulnerable to *Poisoning Attacks*. Recently, some studies have proposed a new paradigm called *Federated Rank Learning (FRL)*, which uses ranking updates instead of parameter or gradient updates in federated learning. This change transforms the space of updates from continuous to discrete, making the federated learning framework more robust to poisoning attacks.**

However, we found that some simple and direct poisoning attack methods can easily cause FRL to fail to converge, contrary to the intuition that reducing the update space should limit attackers. This led us to re-examine the security of FRL. We first analyzed the robustness guarantees of FRL and confirmed its vulnerability to poisoning attacks from both theoretical and experimental perspectives. Next, we revisited the impact of changing the update space from continuous to discrete in the framework and found that the advantage of this change does not lie in directly defending against poisoning attacks, but in greatly limiting the attacker's ability to implement *stealthy poisoning attacks*. Based on this, we added appropriate defense strategies to FRL, further shrinking the discrete update space into a secure range, and limiting the effectiveness and stealthiness of attacks. Experiments show that this approach significantly improves the ability of FRL to resist poisoning attacks.

***Index Terms*—federated learning, poisoning attack, Federated Rank Learning**

I. Introduction

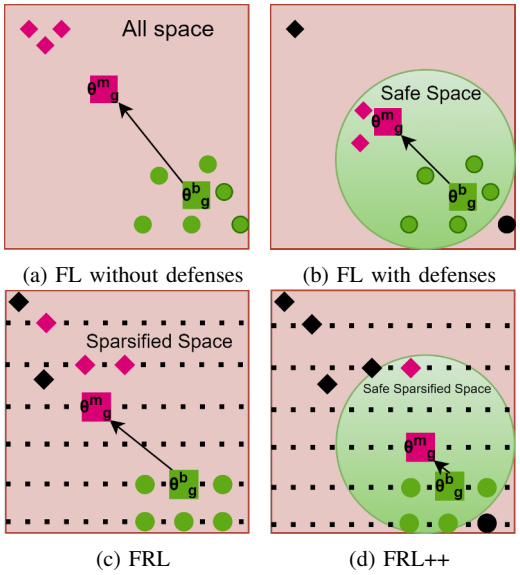
With the growing awareness of privacy protection, along with increasing legal restrictions and regulatory frameworks, federated learning (FL) has gained significant attention as a distributed learning framework that enables model training while preserving data privacy [1].

FL protects data privacy but is susceptible to poisoning attacks. FL allows decentralized training across multiple devices without the need to share sensitive data. This approach ensures that private data stays on the client side, reducing the risk of data exposure. FL has gained traction in various applications, such as industry [2], healthcare [3], [4], [5], COVID-19 detection [6], [7], drone technology [8], [9], and IoT intrusion detection [10], where privacy concerns are critical.

However, despite its advantages in protecting data privacy, FL is vulnerable to various security threats, particularly poisoning attacks [11], [12], [13]. In a poisoning attack, adversarial clients intentionally manipulate the local model updates they send to the server, aiming to degrade the performance of the global model. These attacks are particularly dangerous because they can be difficult to detect and can significantly affect the model's robustness and accuracy, all while maintaining the appearance of a legitimate learning process.

Federated rank learning (FRL) is a new FL paradigm to defend against poisoning attacks. Recently, [14] proposed a new defense method against poisoning attacks, aiming to reduce overhead and mitigate the risk of poisoning attacks during federated training. They provided both theoretical and empirical evidence that FRL effectively defends against non-targeted poisoning attacks, which aim to disrupt global model convergence. These attacks are more harmful than targeted poisoning attacks, which only try to insert a backdoor into the global model without hindering its convergence. Additionally, they demonstrated that FRL is more robust than robust aggregation methods (e.g., Multi-Krum, Trimmed Mean) when facing poisoning attacks.

*Corresponding author: Xiaojie Zhu is affiliated with Computer, Electrical and Mathematical Sciences and Engineering (CEMSE) Division, King Abdullah University of Science and Technology.



(a) FL without defenses (b) FL with defenses

(c) FRL (d) FRL++

Fig. 1: Available weight for clients' model updates. Red diamonds indicate malicious client updates retained, black diamonds show malicious updates removed by the server's defense. Small green circles represent retained benign updates, and black circles indicate benign updates removed. Green boxes denote aggregated benign models, red boxes indicate models with malicious updates. To mitigate poisoning attacks by attackers, existing robust aggregation mechanisms filter out malicious updates by employing security ranges.

[14] points out that the key to defending against poisoning attacks is to reduce the space of potentially malicious updates that an attacker could generate. When the server's aggregation is not robust, such as with dimension averaging, the attacker has no restrictions on their choices, allowing them to use any malicious update far from the benign ones to maximize their goal (Figure 1-a). As a result, even with a small number of malicious clients, the global model's accuracy can decline. When robust secure aggregation methods, such as Multi-Krum [9] or Trimmed Mean [45], are applied, the update space is constrained within a secure range (Figure 1b), limiting the attacker's choices. However, because existing FL frameworks rely on uploading and aggregating continuous neural network parameters for training, even within a limited secure range, the space for generating malicious updates remains large. Therefore, FRL creates edge rankings by adopting the novel learning paradigm of super-mask training [35, 46], further constraining the update space and reducing the continuous space to a discrete one (Figure 1c).

Research question. Although the concept of FRL for defending against poisoning attacks is both novel and reasonable, and despite the rigorous proof process and extensive experimental validation provided for its robustness, we argue that certain key oversights may lead to an incomplete analysis. In fact, we have identified that the theoretical proof guaranteeing the robustness of FRL is based on a flawed assumption. Additionally, the attack strategies employed in the experimental validation are not sufficiently potent. This prompts us to re-examine the actual robustness of FRL.

Preliminary experiments show that FRL fails to defend against simpler and more direct poisoning attacks, which contradicts the intuition that restricting the update space should reduce the range of choices available to attackers. This leads us to further investigate the implications of transforming the update space from continuous to discrete in FRL and to explore the design of adaptive defense mechanisms aimed at enhancing its robustness against poisoning attacks.

This paper will re-examine FRL from three research questions: **(RQ1)** *Does FRL truly offer effective defense against poisoning attacks?* **(RQ2)** *What are the implications of reducing the update space from continuous to discrete in FRL?* **(RQ3)** *Is there potential for improvement in FRL to enhance its robustness?*

For these three research questions, we provide theoretical analysis and conduct extensive experimental validation on popular datasets such as MNIST, FEMNIST, and CIFAR-10. [1] Specifically:

For **RQ1**, we first highlight the issues in the FRL robustness proof provided by [14], as their proof starts from a flawed assumption that arithmetic mean voting is more favorable to the attacker than weighted mean voting. Subsequently, we point out that the strongest attack method claimed to test the robustness of FRL in their experiments is actually not the most effective.

Therefore, we selected two representative non-targeted poisoning attacks: label-flipping [15] and loss-taking inverse attack [16], demonstrating FRL's insufficient robustness. *These results suggest that FRL alone is insufficient to defend against more destructive poisoning attacks.*

For **RQ2**, *the main advantage of FRL lies in its ability to limit the stealthiness of attacks.* Although restricting the update space to discrete values does not effectively defend against the success rate of poisoning attacks, it limits the ability of attackers to employ both powerful and sufficiently covert attack methods. Specifically, in traditional federated learning (FL) poisoning attacks, attackers can introduce a constraint term in the training loss function to identify the most covert malicious updates, thereby making these malicious updates closer to benign updates. However, in the local training process of FRL, clients locally optimize continuous edge importance scores, but only upload the discrete ranking of the trained edge importance.

The separation between uploading and optimization makes it difficult for attackers to directly construct covert constraint terms targeting the uploaded ranking vectors, thus placing them in a dilemma where they cannot simultaneously ensure the effectiveness and stealthiness of the poisoning attack.

[1] Since FRL itself provides good defense against targeted poisoning attacks, and the impact of non-targeted attacks is more severe, we primarily focus on the latter. For the sake of completeness, we have also included corresponding experiments on targeted poisoning attacks in the experimental section.

Our experiments show that the covert constraint added by attackers during the training process does not guarantee that the final uploaded poisoned ranking vector will also maintain its stealthiness.

For **RQ3**, considering that restricting the update space to a discrete space in FRL effectively limits the stealthiness of attacks, while traditional poisoning defense methods directly reduce the update space to limit the success rate of updates, we propose combining both approaches by further narrowing the update space within the discrete domain (Figure 1d). To this end, we design an enhanced defense framework, FRL++, which employs clustering, outlier detection, and rollback mechanisms to further constrain the attacker's choices. *Experimental results demonstrate that our approach significantly enhances the robustness of FRL against poisoning attacks.*

Overall, the contribution of the paper is summarized as follows:

- We investigate the robustness issues of FRL and reveal its vulnerabilities.
- We have demonstrated both theoretically and experimentally that FRL is more advantageous than traditional defense methods in defending against poisoning attacks.
- We enhance the defense capability of FRL by proposing a three-stage defense scheme: FRL++. Comprehensive experiments are conducted to evaluate the effectiveness of this proposed defense mechanism.

II. Background

In this section, we provide a comprehensive introduction to the research background of the issues discussed in this paper. We will first give a brief overview of FL (II-A). Then, we will provide a detailed introduction to the relevant details of FRL, serving as a preliminary for the following three sections(II-B).

A. Federated Learning

FL is a distributed machine learning approach that aims to collaboratively train models among multiple participants (such as smartphones or edge devices) without the need to centrally collect data[17]. The basic process of federated learning can be summarized as follows: the server first initializes the global model parameters and distributes them to the clients. Each client uses local data to train the model and update its parameters. The local loss function is typically defined as

$$\mathcal{L}(\mathbf{w}) = \frac{1}{|\mathcal{D}_k|} \sum_{(x,y)\in\mathcal{D}_k} \mathcal{L}(M(x;\theta), y), \tag{1}$$

where k represents the k-th client, M denotes the neural network model, (x, y) represents data samples and labels, and θ denotes the neural network parameters.

After completing local training, each client sends its updated model parameters to the server. The server aggregates the models uploaded by all participants to update the global model parameters:

$$\theta_g \leftarrow \theta_g + \frac{1}{K} \sum_{k=1}^{K} \Delta\theta_k, \tag{2}$$

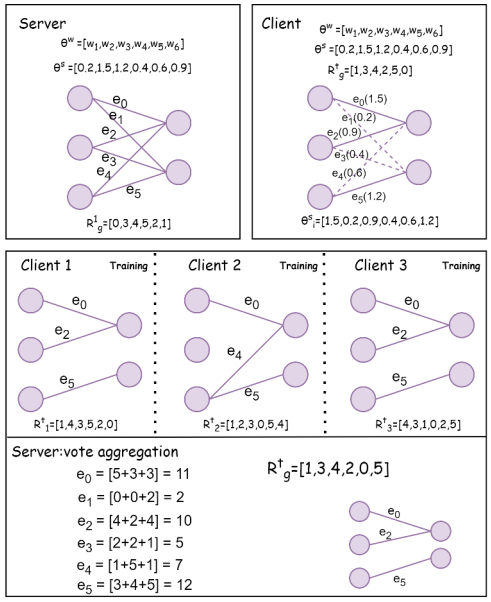


Fig. 2: Training process of FRL.

where K is the total number of clients, θ_g represents the global model parameters, and $\Delta\theta_k$ denotes the update from the k-th local model.

B. Federated Rank Learning

Federated Rank Learning (FRL) is a new defense method proposed by [14]. Unlike traditional methods, FRL allows clients to send rankings of model parameters instead of directly transmitting the parameters. This reduces the space of client updates from the model parameter updates (continuous space of floating-point numbers) in standard FL to the parameter ranking space (discrete space of integer values). As a result, malicious clients find it more difficult to choose in the larger space, making effective poisoning attacks impossible.

It is well known that a neural network (NN) is defined by its network structure and weights. In traditional NNs, the structure is determined first (i.e., the neurons and connections in the model do not change during training), and then the NN parameters are adjusted through backpropagation to minimize the loss. However, in FRL, the process is quite the opposite. FRL fixes the network parameters (**the weights of the edges in the model do not change during training.**) and learns to select the most important edges to form the optimal sub-network.

Before training begins, the server first builds a super neural network (super-NN), randomly initializes this network, and keeps the parameters fixed during the training process. Each parameter (i.e., each connection in the neural network) is also assigned a random score. The server generates a random seed (SEED) to initialize these parameters, ensuring that all clients can consistently build the same network and parameters based on this seed.

Specifically, as shown in Figure 2, the process proceeds as follows: (i) Before training starts, the server first constructs a super neural network (super-NN) and randomly initializes this network θ^w, keeping the parameters fixed during training. It also assigns a random score θ^s to each parameter (i.e., each connection in the neural network). The server generates a random seed (SEED) to initialize these parameters, allowing all clients to consistently build the same network and parameters based on this seed (Figure 2, first row, left image). (ii) After training on local data, the client obtains the importance score θ_i^s for each edge, computes the importance ranking vector R_i^t, and uploads it (Figure 2, second row). (iii) The server receives the ranking vectors R_i^t uploaded by all selected clients and updates the global ranking vector R^g by ranking and voting (Figure 2, third row). (iv) After training is complete, the global model retains only the top k edges from R^g as the final sub-network.

Its global optimization objective is as follows:

$$\min_{R_g} F(\theta^w, R_g) = \min_{R_g} \sum_{i=1}^{n} \lambda_i \mathcal{L}_i(\theta^w \odot m) \tag{3}$$
$$s.t. \quad m[R_g < k] = 0, \quad m[R_g \geq k] = 1,$$

where k is the percentage threshold of the total number of edges in the supernetwork that are ultimately retained in the subnetwork. m is the global binary mask (*i.e.*, which edges are selected as components of the sub-network). Its value is determined by R_g, where for each layer, edges in the top k get a "1", and the rest get a "0". n is the number of clients, and λ_i is the importance of the i^{th} clients. $\mathcal{L}_i$ is the loss function for i^{th} clients.

III. Threat Model

In this section, we introduce the threat model for poisoning attacks against FRL, covering the attacker's objectives in Section III-A, capabilities in Section III-B, and knowledge in Section III-C.

A. Attacker's Goal

Poisoning attacks can be categorized into targeted and untargeted types. Targeted poisoning attacks aim to manipulate the global model's predictions for specific samples as desired by the attacker. In contrast, untargeted poisoning attacks primarily focus on reducing the overall accuracy of the global model. Untargeted attacks are generally more destructive in nature. Therefore, this paper primarily focuses on addressing untargeted poisoning attacks. The attacker aims to disrupt the convergence of the global model and reduce its accuracy on test data.

B. Attacker's Capability

We assume that the attacker can control a small portion of clients in FRL. In each iteration of collaborative training, the attacker has full access to the training information of the controlled clients and can manipulate their uploaded updates. In addition, we allow the attacker to flexibly choose whether the controlled clients act independently or coordinate with each other to perform collusion attacks. Following FRL [14], our experiments also evaluate the effectiveness of collusion attacks (see Section IV-C).

In addition, we assume the attacker has no control over the server.

C. Attacker's Knowledge

Since the attacker can control some clients, they can obtain knowledge from these controlled clients. However, they cannot access or learn any information from the remaining clients, including their local data and uploaded parameters.

IV. Robustness of FRL against poisoning attacks

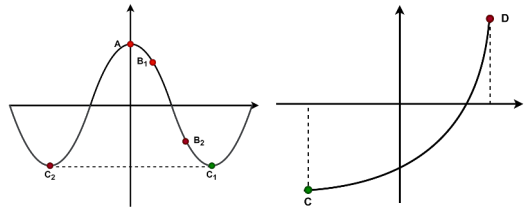
In this section, we explore the robustness of FRL against non-targeted poisoning attacks (**RQ1**). We first highlight the issues in the FRL robustness proof provided by [14], both in terms of theory (IV-A) and experimental design (IV-B). Then, by implementing two representative methods: label-flipping and loss-taking inverse attack, we validate the vulnerability of FRL to non-targeted poisoning attacks (IV-C).

A. Analysis of the Flaws in the Theoretical Analysis of FRL's Robustness

In the practical implementation of FRL, clients upload the importance ranking vectors of all the edges in the hypernetwork, and the server performs a weighted voting based on the rankings to decide the aggregated global ranking. However, in the theoretical analysis of the robustness of FRL against poisoning attacks by [14], the voting model is simplified to a simple unweighted binary vote. Specifically, they assume that clients only upload a binary vector to the server, indicating which edges should be selected in the current round to form the subnetwork; then, the server performs an unweighted vote, retaining the edges that receive a majority of the votes. [14] derived the robustness guarantees of FRL under this simplified voting model and claimed that the actual FRL voting, with finer-grained weights, therefore possesses stronger robustness.

Quite the opposite, for a malicious attacker, the ranking-weighted voting model actually favors them more than the simple binary voting model in manipulating the final aggregation result. This is because the malicious attacker can place the edges they wish to retain at the top, assigning them the highest scores, while placing the edges they want to discard at the bottom with the lowest scores. This method not only amplifies the score of the desired edges but also reduces the scores of major competitors by lowering their rankings. In contrast, under binary voting, the attacker can only very limitedly increase the number of votes for the edges they wish to retain. For a more detailed analysis of how the ranking voting model benefits attackers, refer to Appendix A.

The above conclusion indicates that the robustness guarantees derived from the simple binary model cannot be extended to the ranking-weighted voting in FRL.



(a) The loss function is not monotonic (b) The loss function is monotonic

Fig. 3: Examples of loss functions. (a) represents a non-monotonic loss function, exhibiting multiple optimal solutions such as C_1 and C_2, with the worst solution denoted as A. (b) represents a monotonic loss function, where there is only one optimal solution C and the worst solution is D.

B. Analysis of the Flaws in the Experimental Validation of FRL's Robustness

In addition to the theoretical flaws in [14], their experimental evaluation of FRL's robustness against poisoning attacks is incomplete. They only used the update reversal attack, which involves three steps: (i) The malicious client performs the normal ranking learning training process locally to obtain a benign edge ranking vector, and then sends these vectors to the attacker. (ii) The attacker acts as a pseudo-server, aggregates the edge ranking vectors, inverts the order of the aggregated vector, and then sends it back to the malicious client as a malicious update. (iii) The malicious client receives the manipulated malicious update and submits it to the server. [14] claims that since the update reversal attack inverts the benign ranking, it achieves the strongest attack, and FRL performs well under it, proving its robustness. However, we believe this assumption is flawed. Edge importance is relative, meaning the least important edge in one subnetwork could be the most important in another. Reversing the ranking essentially changes the subnetwork. In FRL, each client performs local training to minimize the loss function, aiming to find the optimal edge ranking (e.g., C_1, C_2, or C in Figure 3). The malicious client seeks to submit the worst solution (e.g., A and D in Figure 3) to disrupt the global solution. The update reversal attack assumes that the loss function is monotonic ($Lmin(x) = -Lmax(x)$), but this assumption does not hold for common loss functions like cross-entropy or MSE. Furthermore, in some cases, pairwise swaps may lead to another optimal solution.

Therefore, the update reversal attack is not the greatest threat to FRL, and defending against it does not prove FRL's robustness.

C. Label-flipping and Loss-taking Inverse Attack on FRL

To enhance the evaluation of FRL's robustness, we adapt two untargeted poisoning attack schemes.

There are usually two methods for launching an untargeted poisoning attack: data poisoning and model poisoning. Data poisoning involves altering benign data by assigning malicious labels, which can result in malicious behavior when the model is trained on this compromised dataset. In model poisoning, the attacker can manipulate the training process arbitrarily, e.g., they can alter the model parameters or modify the loss function during training.

We propose a stronger attack method for each type of poisoning attack: the label-flipping attack for data poisoning and the loss-taking inverse attack for model poisoning. We design the label-flipping attack based on the approach proposed by Tolpegin *et al.*[15]. Tolpegin *et al.* focus on backdoor attacks [15], while we modify their approach to implement untargeted attacks. Specifically, we modify the labels of all samples to a uniform designated label. The objective is to induce misclassification and thereby affect the accuracy of the global model. For the loss-taking inverse attack, we adapt the approach of Xie *et al.* [16] by altering the malicious client's loss function to reverse its objective from minimization to maximization. To clearly evaluate the impact of stronger collusion attacks, we divide the loss-taking inverse attack into two types: non-collusive loss-taking inverse attack (LTI-NCA) and collusive loss-taking inverse attack (LTI-CA). In LTI-NCA, each malicious client simply uploads its locally trained ranking vector to the server. In contrast, LTI-CA assumes that malicious clients first send their locally generated ranking vectors to the attacker (here we assume the attacker is aware of the server's aggregation rule [18]), who then aggregates these vectors via a voting mechanism and returns the aggregated ranking vector to each malicious client. All malicious clients then upload the same aggregated ranking vector to the server. Although LTI-CA is more destructive, it comes at the cost of reduced stealthiness.

In summary, we implemented four attack methods on FRL to evaluate its robustness against non-target poisoning, which are: LTI-CA, LTI-NCA, update reversal attack (URA), and the label-flipping attack (LFA).

As shown in Figure 7, the loss-taking inverse attack is the most destructive, and can completely compromise the defenses of FRL. For example, when attacking with LTI-CA on the CIFAR-10 dataset, the accuracy of the global model drops to around 10%. For detailed experimental setup and results, refer to Section VII-B.

V. Internal Mechanism Analysis of FRL

In this section, we will explore the advantages of FRL in defending against poisoning attacks (RQ2) compared to traditional defense methods. We first introduce the methods used by attackers to balance the effectiveness and stealthiness of their attacks (V-A), and then highlight why these methods are difficult to implement in the FRL setting (V-B).

A. Common Methods for Implementing Stealthy Poisoning Attacks

To minimize the difference between the malicious and benign models, attackers often rely on a fixed parameter between the training objective and the additional loss function introduced to balance effectiveness and stealthiness. Attackers

typically introduce one or more constraints in the training loss function of the malicious clients, forcing the final malicious model to resemble the benign model [19]. These constraints are typically defined based on the Euclidean distance or cosine similarity between the parameters of the benign and malicious models. Thus, the training loss function of the malicious client is as follows.

$$\begin{aligned}\mathcal{L}(\theta_{\mathbf{m}}) =& \frac{1}{|\mathcal{D}_k|} \sum_{(x,y)\in\mathcal{D}_k} (1-\alpha)\cdot \mathcal{L}(M(x;\theta_{\mathbf{m}}), y) \\ &+ \alpha \cdot \mathcal{L}_{distance}(\theta_m, \theta_b),\end{aligned} \tag{4}$$

where θ_m represents the model parameters of the malicious clients, and θ_b represents the model parameters of the benign client. α is a weighting factor.

Although models with constraints in the loss function can evade the potential detection metrics of the server, they depend on manually specified parameters. These parameters balance the main training objective and the supplementary loss function introduced to enhance stealthiness against detection metrics. Finding suitable parameters requires significant human effort, and even then, it cannot guarantee a satisfactory solution.

As a result, adaptive attacks have been proposed [20]. These attacks dynamically adjust parameters during the training process based on the server's detection status, increasing the proportion of stealthy terms as the detection rate rises. This approach allows attackers to maintain a high success rate while adapting to existing defense measures [21].

From the above analysis, we can conclude that both stealthy and adaptive poisoning attacks use constraints in the loss function to build a malicious model that closely resembles the benign model. By fine-tuning the weight coefficients of these constraints, attackers can effectively balance the attack's effectiveness and the model's stealthiness.

B. Analysis of the Effectiveness of Stealthy Poisoning Attacks in the FRL Framework

Unlike traditional FL (which transmits model weights or gradients as updates), in FRL, clients optimize the edge importance scores through ranking learning and upload them as discrete ranking vectors. It is important to note that this discretization process is non-differentiable, so it occurs after training. Therefore, attackers cannot add constraints in the loss function based on the discretized ranking vectors to ensure the stealthiness of the attack.

Another option for attackers is to construct constraints for the edge importance scores during the ranking learning process (before discretization), as follows:

$$\begin{aligned}\mathcal{L}(\theta^{\mathbf{s}}_{\mathbf{m}}) =& \frac{1}{|\mathcal{D}_k|} \sum_{(x,y)\in\mathcal{D}_k} (1-\alpha)\cdot \mathcal{L}(M(x;\theta^{\mathbf{s}}_{\mathbf{m}}), y) \\ &+ \alpha \cdot \mathcal{L}_{distance}(\theta^s_m, \theta^s_b),\end{aligned} \tag{5}$$

where, θ^s_m represents the edge score of the malicious client, and θ^s_b represents the edge score of the benign client.

However, as analyzed in Appendix B, the similarity between continuous real-valued edge importance score vectors and discrete importance ranking vectors does not exhibit a strong consistency relationship. In other words, forcing the edge importance scores in the malicious model to be similar to those in the benign model does not guarantee that the edge importance rankings in the malicious model will be similar to those in the benign model. Therefore, it is difficult for attackers to ensure the stealthiness of the final uploaded malicious ranking updates by adding constraints, which makes poisoning attacks potentially detectable by simple statistical defense mechanisms. This difficulty in balancing attack effectiveness and stealthiness is an inherent advantage of the FRL mechanism.

As shown in Table II, after adding constraints with different weight coefficients α, the Spearman distance difference between malicious and benign updates is not significant. Therefore, attackers find it hard to ensure the stealthiness of the final uploaded malicious ranking updates through the addition of constraint terms. For detailed experimental setup and results, please refer to Section VII-C. This demonstrates the difficulty of conducting stealthy attacks under FRL and suggests that adaptive attack strategies are ineffective in this setting. Although such strategies can balance attack success and stealthiness in standard FL, their ability to remain covert is significantly constrained in FRL.

VI. FRL++: An Enhanced Framework for More Robust Defense Against Poisoning Attacks

In this section, we propose an enhanced scheme for adversarial poisoning attacks designed for FRL: FRL++ **(RQ3)**. Specifically, we aim to further restrict the update space to a smaller, safer range based on the discrete update space. To achieve this, we incorporate clustering and outlier detection methods as pre-aggregation detection techniques, along with a rollback strategy post-aggregation on the server side. Our enhancement method removes clearly outlying vectors from the update space, addressing the vulnerability of FRL to simple and direct poisoning attacks without introducing significant costs. An overview of our method is shown in Figure 4 , due to space limitations, we have placed the algorithmic flow in the Appendix C.

A. Clustering

Before aggregation, we apply clustering to filter malicious ranking vectors. Specifically, clustering algorithms are applied to all ranking vectors collected by the server, retaining only the largest cluster while removing others and outliers. We avoid using k-means and HDBSCAN due to their limitations: k-means requires a predefined number of clusters, which can be problematic in collusion attacks, and HDBSCAN, while effective in some cases, performs inconsistently across layers in FRL due to varying vector dimensions. We chose BIRCH, which combines hierarchical clustering with k-means, as it adapts to data scale, does not require the number of clusters

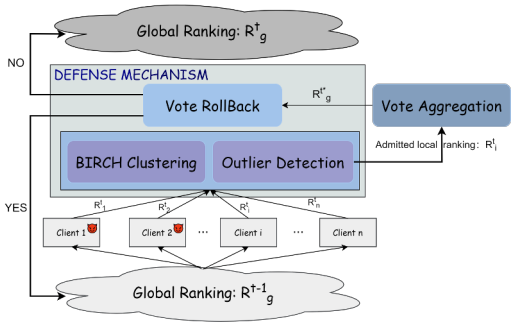


Fig. 4: In round t+1, the server sends global rankings from round t-1 to clients, who upload local rankings. The server clusters rankings using BIRCH, detects and excludes malicious clients, and performs voting aggregation on the remaining rankings to create the new model. After testing, clients vote to retain or replace the model, and the server determines whether to adopt the new aggregated ranking or revert to round t-1's ranking.

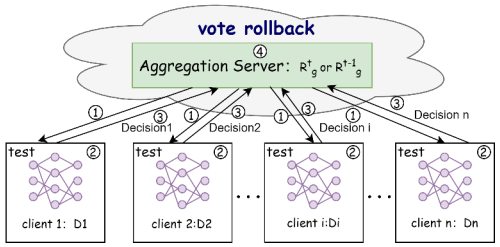


Fig. 5: Workflow illustration of the proposed vote rollback mechanism. Each client receives the global model and first tests the global model with a small batch of local data and votes on whether it is a malicious aggregation or not, and the server aggregates the votes to decide whether to roll back to the previous round of the global model or not.

to be specified, and can automatically detect outliers. The algorithm utilizes a tree structure similar to a B+ tree, specifically known as a clustering feature tree (CF-tree), to facilitate efficient clustering operations. The main process of the algorithm begins with scanning the uploaded data to create a CF-tree, which is then stored in memory. This CF-tree serves as a multi-level compression of the input data, preserving its inherent clustering structure. Subsequently, clustering algorithms like K-means are applied to cluster the leaf nodes of the CF-tree. For clustering distance, we use Euclidean distance to balance computational cost and effectiveness.

B. Outlier Detection

In the clustering component for detecting malicious vectors, due to computational cost considerations, we use the Euclidean distance metric, which is more suitable for measuring continuous-valued vectors. This may allow some malicious ranking vectors to evade clustering detection. Therefore, we designed another independent Outlier Detection component with lower computational cost, using Spearman's rank correlation distance [22], which is more suitable for measuring discrete vectors, to eliminate clearly anomalous ranking vectors.

To calculate the Spearman rank correlation distance, we first need to compute the Spearman rank correlation coefficient ρ:

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)}, \tag{6}$$

where d_i represents the difference between the bit values of the ith data pair of two vectors.

Since the Spearman correlation coefficient ($\rho \in [-1, 1]$) does not meet the definition of distance, it needs to be converted into the Spearman correlation distance $d_\rho = 1 - \rho$.

In the outlier detection component, we first defined a standard permutation vector $[0, 1, ..., n]$, where n is the number of edges in the clustering layer. In each round, we compute the Spearman correlation distance between all ranking vectors and the standard vector, requiring only m calculations (where m is the number of clients). We prune vectors with rankings below the 10th percentile or above the 90th percentile to remove malicious ranking vectors that Euclidean clustering cannot detect.

C. Vote Rollback

The clustering and outlier detection components applied before aggregation can identify most malicious clients, but some malicious updates may still be missed. For example, in Fig.6, even with clustering and outlier detection employed, there are still rounds with steep drops in accuracy due to a poisoned global model. To address this, inspired by the work of Andreina *et al.* [23], we designed a rollback component for post-aggregation to prevent these malicious updates from causing severe impact.

As shown in Fig.5, at round t: 1) The server sends the global model from the previous round to each client (Fig.5-①). 2) Each client tests the global model using its local dataset (Fig.5-②). 3) Based on the test results, each client sets a rollback poll value and sends it back to the server (Fig.5-③). If the total number of rollback votes exceeds a predefined threshold, the server distributes the previous global model as the new global model. Otherwise, no action is taken(Fig.5-④).

The client uses the following formula to calculate its rollback vote value:

$$rv_m = \begin{cases} \text{False}, & \text{if } acc(R_i^{t-1}) - acc(R_g^t) \le \tau \\ \text{True}, & \text{if } acc(R_i^{t-1}) - acc(R_g^t) > \tau \end{cases}, \tag{7}$$

where $acc(\cdot)$ represents the validation accuracy and τ represents the pre-set rollback threshold.

At the end of the local training, clients submit the updates of the current round along with rv to the server, which counts the rollback vote values of each client. If more than half of the clients believe that the global model under the round is affected by malicious updates, the server drops R_g^t and performs the

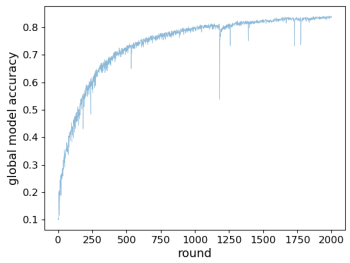


Fig. 6: Performance of the defense mechanism without vote rollback. When no vote rollback mechanism is implemented, there are periods of significant drops in global model accuracy.

rollback, enables R_g^{t-1} and re-selects the clients to perform the local training. Following the setup of most federated learning scenarios, we assume that the share of malicious clients does not exceed 50% of the total number of clients. Therefore, even if malicious clients intentionally vote the opposite way, it will not lead to the invalidation of the rollback vote.

D. Comparison of FL, FRL and FRL++.

In Table I, we provide a comprehensive comparison between FL, FRL, and FRL++, highlighting our improvements. FRL transforms the training paradigm of FL from continuous parameter space to a discrete ranking space, while our FRL++ further introduces a three-stage defense mechanism that narrows the ranking space even more. FRL(++) significantly improves communication efficiency compared to FL. However, the introduction of defense mechanisms inevitably incurs additional server-side computational overhead, whether applied to FL or to FRL (as in our FRL++).

VII. Experimental Results

In this section, we first introduce the experimental setup in Section VII-A, including the experimental environment and datasets. Next, in Section VII-B, we present the experimental results of the vulnerabilities in FRL. In Section VII-C, we demonstrate that adding stealth terms to the loss function is not feasible. Finally, in Section VII-D, We present the experimental results of the defense scheme along with the ablation experiments.

A. Experimental Setting

In this subsection, we will introduce the general setup used for all the experimental analyses in the following.

1) Datasets:

- CIFAR10 dataset is a 10-class classification task consisting of 60,000 RGB images, with 50,000 images in the training set and 10,000 images in the test set. Each image has a size of 32 × 32. Consistent with FRL[14], we distribute CIFAR10 datasets among 1,000 clients in non-iid fashion using Dirichlet distribution with parameter β = 1.
- MNIST dataset is a 10-class classification task containing 70,000 grayscale images, each with a size of 28 × 28. Following FRL[14], we distribute MNIST among 1,000 clients in non-iid fashion using Dirichlet distribution with parameter β = 1.
- Federated Extended MNIST (FEMNIST) [24] is a 62-class character recognition classification task consisting of 52 upper- and lower-case letters and 10 numbers. The dataset comprises 805,263 grayscale images of handwritten alphanumeric characters distributed across 3400 clients.

2) Models:

- We use a VGG-like model architecture for CIFAR10 dataset. Each client undergoes 5 training rounds with a batch size of 8, and a test batch size of 128. The SGD optimizer is used with a learning rate of 0.4, a momentum of 0.9, and a weight decay of 1e-4.
- We use the LeNet architecture, which consists of two convolution layers and two fully connected layers, for MNIST and FEMNIST datasets. Each client undergoes 2 training rounds with a batch size of 8, and a test batch size of 128. The SGD optimizer is used with a learning rate of 0.4, a momentum of 0.9, and a weight decay of 1e-4.

3) Implementation Details: All of our experiments are based on the open-source framework provided by [14], which implements the full functionality of heterogeneous data distribution and FRL. We run the experiments on two machines: one machine runs on Windows with an NVIDIA 4060ti GPU and 8GB of graphics memory and the other machine running on Windows is equipped with an NVIDIA 4090 GPU and has 24GB of graphics memory.

B. Experimental Results of Vulnerability of FRL

Figure 7 shows the performance of FRL against loss-taking inverse attack (both collusive and non-collusive) and label-flipping attacks. As seen in the figure, both types of poisoning attacks can disrupt the convergence of FRL, with the loss-taking inverse attack having a more significant impact.

Figure 7 (a)-(c) show the impact of different attack methods on FRL's performance on the CIFAR10 dataset, specifically their effect on the global model's accuracy, with malicious client proportions of 5%, 10%, and 20%, respectively. Similarly, Figure 7 (d)-(f) display the performance of the attacks on the MNIST dataset, with malicious client proportions of 5%, 10%, and 20%. Finally, Figure 7 (g)-(i) illustrate the performance of the attacks on the FEMNIST dataset, with malicious client proportions of 5%, 10%, and 20%.

The green line in the figure represents FRL without any attacks, the blue line represents attacks using LTI-CA, the orange line represents attacks using LTI-NCA, the purple line represents attacks using URA, and the red line represents attacks using LFA.

As shown in Figure 7, the most destructive loss-taking inverse attack significantly affects the convergence of FRL's global model. For example, on the CIFAR10 dataset, the loss-taking inverse attack causes FRL to fail to converge, with the model's accuracy remaining around 10%. When the proportion of malicious clients reaches 20%, the accuracy of the model

TABLE I: Comparison of FL, FRL, and FRL++. Here, L is the number of layers, N_ℓ is the number of parameters in layer ℓ, k_ℓ is the number of top-ranked parameters uploaded in FRL++, K is the number of participating clients, and b is the number of bits per parameter.

Comparison Item	**FL**	**FRL**	**FRL++**
Network architecture	Traditional neural network	Ranking-based network	Ranking-based network
Update space	Continuous parameter space	Discrete ranking space	Discrete ranking space
Upload content	Full model parameters	Per-layer parameter ranking vectors	Per-layer top-k ranking vectors
Communication cost	$\sum_{\ell=1}^{L} N_\ell \cdot b$	$\sum_{\ell=1}^{L} N_\ell \cdot \log_2 N_\ell$	$\sum_{\ell=1}^{L} k_\ell \cdot \log_2 N_\ell$
Aggregation mechanism	FedAvg / Trimmed-mean / Multi-Krum	Majority voting over rankings	Majority voting with pre/post filtering
Defense mechanism	Trimmed-mean, Multi-Krum, etc.	None	Clustering + Outlier Detection + Vote Rollback
Server aggregation cost	FedAvg: $O(KN_\ell)$ Trimmed-mean: $O(KN_\ell \log K)$ Multi-Krum: $O(K^2 N_\ell)$	$O(KN_\ell \log K)$	$O(KN_\ell \log K + K \log K + K^2)$

TABLE II: A comparison of the Spearman distance between malicious and benign updates after adding constraint terms with different weights α. Case 1 represents with constraint term VS. without constraint term, case 2 represents with constraint term VS. benign, and case 3 represents without constraint term VS. benign.

Attack Method	Dataset	$\alpha = 0.9$			$\alpha = 0.5$			$\alpha = 0.1$		
		Case 1	Case 2	Case 3	Case 1	Case 2	Case 3	Case 1	Case 2	Case 3
LFA	CIFAR10	0.966	0.975	0.974	0.990	0.985	0.989	0.995	0.998	0.997
	MNIST	0.988	0.996	0.9898	0.995	0.997	0.979	0.998	0.995	0.997
	FEMNIST	0.991	0.988	0.997	0.995	0.994	0.993	0.994	0.997	0.996
LTI-CA	CIFAR10	0.976	0.982	0.996	0.981	0.985	0.972	0.990	0.995	0.995
	MNIST	0.984	0.995	0.993	0.962	0.981	0.988	0.990	0.995	0.992
	FEMNIST	0.992	1.003	0.998	0.986	1.003	1.001	0.993	1.001	1.006

also drops to around 10%. On the MNIST dataset, the loss-taking inverse attack causes noticeable oscillations in the FRL model's accuracy curve, preventing convergence. On the FEMNIST dataset, when the proportion of malicious clients reaches 5%, the loss-taking inverse attack causes severe oscillations in the FRL model's accuracy curve, preventing convergence. When the proportion of malicious clients increases to 10%, the model's accuracy first fluctuates significantly, then drops to 12.3%, without convergence. When the proportion of malicious clients reaches 20%, FRL completely fails to converge.

In summary, FRL fails to defend effectively against more destructive poisoning attacks.

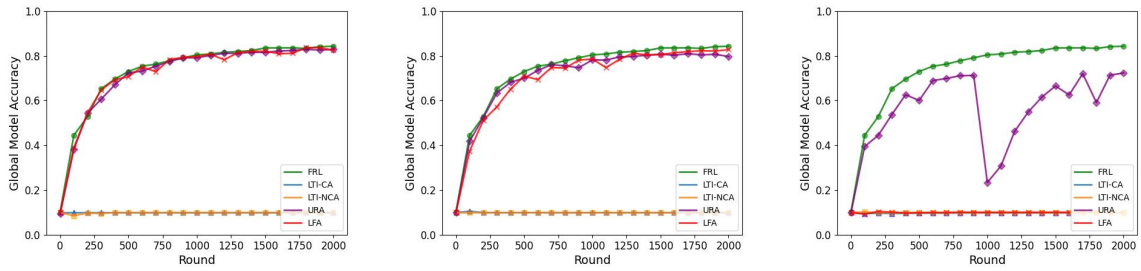
C. Experimental Results of Incorporating a Stealth Term in the Loss Function

1) Experimental setup: To illustrate that adding a stealth term to the attacker's loss function does not achieve the expected effect, we first train a benign model using benign data, and then use cosine similarity as a measure of similarity to construct constraints related to the edge importance scores in the model's loss function, as shown in Equation 5. These constraints are used to train a malicious model with edge importance scores that deviate from the benign model. Finally, we train another malicious model without any constraints.

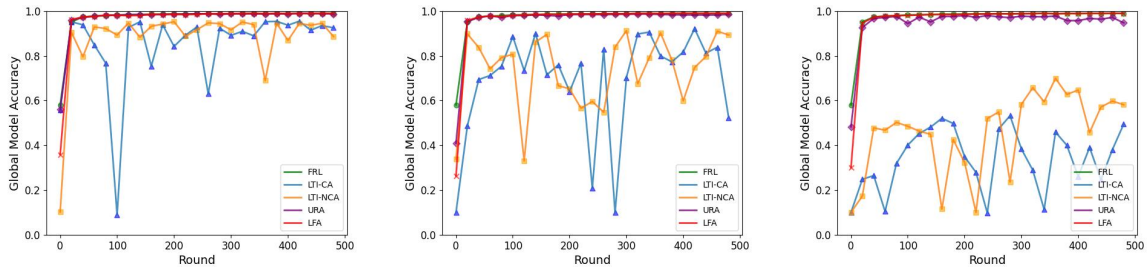
We selected two malicious attack methods: LTI-CA and LFA. After 500 rounds of local training, we computed the Spearman correlation distance between the three ranking vectors (sorted by the trained edge importance scores). The valid range of Spearman correlation distance is [0-2], where a distance of 0 indicates perfect correlation, 1 indicates no correlation, and 2 indicates perfect negative correlation.

We manually adjusted the stealth coefficient α in Equation 5. For clarity and simplicity, we selected α = 0.1, 0.5, and 0.9 as representative values.

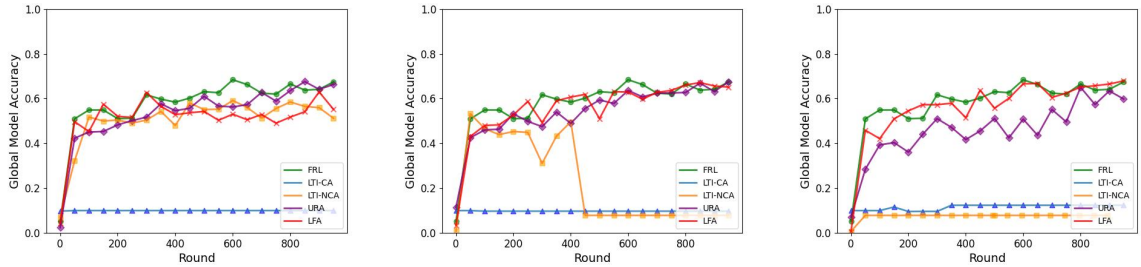
2) Experimental results: As shown in Table II, regardless of the weight coefficient of the constraint term, the correlation distance between the malicious ranking and the benign ranking (whether or not constraints are used) is quite large, indicating that the influence of the constraint term is minimal. Increasing the weight of the constraint term slightly reduces the distance between the malicious and benign rankings, but even when the weight coefficient is increased to 0.9 (close to 1, meaning no correlation), the distance between the malicious and benign rankings remains large and easily detectable by the server. For example, on the CIFAR10 dataset, even with a stealth coefficient α set to 0.9, the spearman distance between the constrained malicious ranking vector and the unconstrained malicious ranking vector is 0.966. On the FEMNIST dataset, even with a stealth coefficient α set to 0.9, the spearman distance between the constrained malicious ranking vector and the unconstrained malicious ranking vector is 0.991. This phenomenon is observed across different attack schemes and



(a) CIFAR10: 5% of the total number of clients are malicious (b) CIFAR10: 10% of the total number of clients are malicious (c) CIFAR10: 20% of the total number of clients are malicious



(d) MNIST: 5% of the total number of clients are malicious (e) MNIST: 10% of the total number of clients are malicious (f) MNIST: 20% of the total number of clients are malicious



(g) FEMNIST: 5% of the total number of clients are malicious (h) FEMNIST: 10% of the total number of clients are malicious (i) FEMNIST: 20% of the total number of clients are malicious

Fig. 7: Impact of different attacks on the global model accuracy of FRL across CIFAR, MNIST, and FEMNIST datasets.

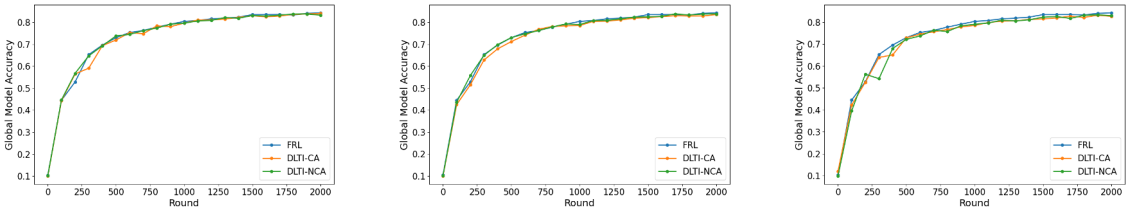
datasets.

Therefore, this experiment confirms our conclusion: attackers cannot force the final malicious ranking to closely resemble the benign ranking by adding constraints to the edge importance score's loss function, making the attack difficult to evade detection by the server.

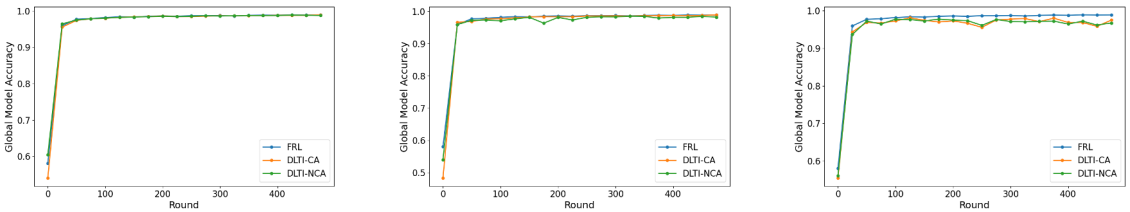
D. Experimental Results of FRL++ Against Poisoning Attacks

1) Experimental setup: To evaluate the effectiveness of FRL++ in defending against poisoning attacks, we performed three types of attacks: LTI-CA, LTI-NCA, and LFA. The experiments were conducted on three datasets (CIFAR10, MNIST, and FEMNIST), assessing the effectiveness of our defense mechanism against these three attack methods at three different proportions of malicious clients (5%, 10%, and 20%). In [14], FRL has been shown to outperform robust aggregation methods such as Multi-Krum and Trimmed-Mean in defending against poisoning attacks. Since FRL++ is an improved and enhanced version of FRL, in this work we primarily compare FRL++ with FRL and its variants, rather than repeating comparisons with mainstream defense methods. For such comparisons, we refer readers to the original FRL paper [14].

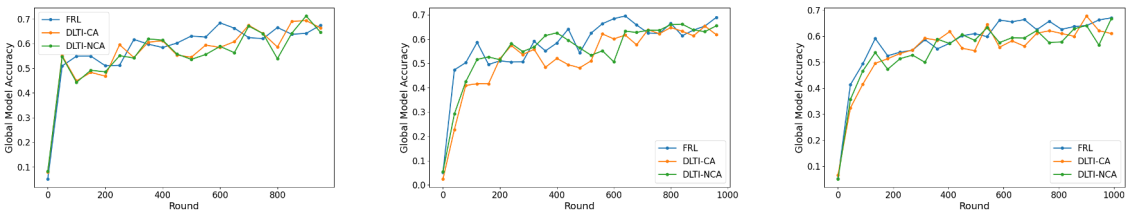
2) Robustness of FRL++ against non-targeted poisoning attack: The results are shown in Figure 8, where the blue line represents the baseline FRL with no attacks, the orange line represents the global model's accuracy when the attacker implements the collusion loss-taking inverse attack, and the green line represents the global model's accuracy when the non-collusion loss-taking inverse attack is implemented. FRL++ shows strong defense against the loss-taking inverse attack on the CIFAR10 and MNIST datasets, with the global model's accuracy convergence curve almost overlapping with the baseline FRL when there are no attacks. Although FRL++



(a) CIFAR10: 5% of the total number of clients are malicious (b) CIFAR10: 10% of the total number of clients are malicious (c) CIFAR10: 20% of the total number of clients are malicious



(d) MNIST: 5% of the total number of clients are malicious (e) MNIST: 10% of the total number of clients are malicious (f) MNIST: 20% of the total number of clients are malicious



(g) FEMNIST: 5% of the total number of clients are malicious (h) FEMNIST: 10% of the total number of clients are malicious (i) FEMNIST: 20% of the total number of clients are malicious

Fig. 8: Performance of our defense mechanism against loss-taking inverse attack on CIFAR10, MNIST, and FEMNIST datasets.

still shows some oscillations on the FEMNIST dataset, it is worth noting that the baseline FRL's accuracy curve also exhibits noticeable oscillations. In comparison, the oscillations caused by FRL++ under attack do not significantly increase compared to the baseline FRL. Due to large oscillations during the convergence process, we choose to raise the voting rollback threshold τ to 0.15 to avoid infinite rollback. All of the above results demonstrate that our defense mechanism provides strong protection against the loss-taking inverse attack.

To ensure the completeness of the experiments, we implemented update reversal attack, label-flipping attack, and the proposed loss-taking inverse attack on FRL++, with results shown in Fig.9. In the figure, the blue line represents the baseline FRL with no malicious attackers involved. The orange, green, and red lines represent the global model's accuracy curves when the attacker implements the collusive loss-taking inverse attack, the non-collusive loss-taking inverse attack, and the label-flipping attack, respectively. We only conducted experiments with 20% malicious clients, as poisoning attacks like update reversal attack and label-flipping do not significantly affect the accuracy of the FRL global model when the malicious client percentage is below this threshold.

As shown in Fig.9, FRL++ is effective against both the more destructive update reversal attack and label-flipping attack.

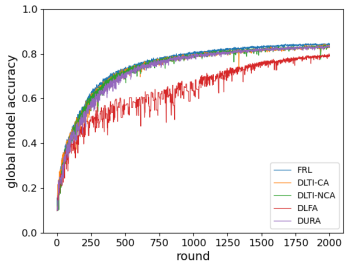
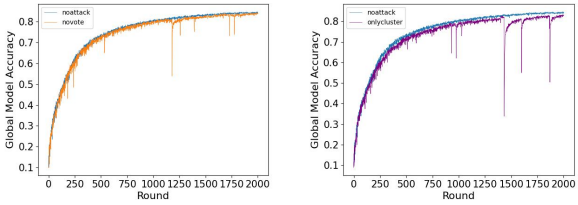


Fig. 9: Performance of the defense mechanism against the update reversal attack, label-flipping attack, and our loss-taking inverse attack on global model accuracy.

3) Impact of clustering and outlier detection on overall system performance: Although clustering and outlier detection may cause a small number of benign clients to be mistakenly classified as malicious and excluded from the model's voting aggregation process, as shown in the table III, the global model's accuracy does not significantly decrease. For example, on the CIFAR-10 dataset, the accuracy of the FRL

TABLE III: The numbers in the table represent the accuracy (%) of the global model for FRL and FRL++ when there are no malicious clients.

Dataset	FRL	FRL++
CIFAR10	84.74	84.43
MNIST	99.15	99.02
FEMNIST	72.42	71.63



(a) w/o rollback (b) Only clustering

Fig. 10: Ablation Experiment: (a) shows the convergence of the global model when there is no voting rollback mechanism. (b) shows the convergence of the global model when only clustering is used.

model is 84.74%, while the accuracy of the FRL++ model is 84.43%. Therefore, FRL++ maintains good performance of the global model while enhancing its robustness against poisoning attacks.

4) Ablation experiment: We conducted ablation experiments to explore the necessity of the defense scheme components and mechanisms (clustering, outlier detection, and voting rollback). The experiments used the CIFAR10 dataset and a representative scenario with 10% malicious clients. The results are shown in Fig. 10, where the blue line represents the baseline FRL with no malicious attackers, the orange line represents the defense scheme without the voting rollback strategy, and the purple line represents the defense scheme using only clustering.

Without voting rollback: As seen in Fig. 10a, when there is no voting rollback, the accuracy of the global model significantly drops in some rounds due to the omissions in clustering and outlier detection. For example, around the 1250th round, the global model's accuracy drops to approximately 50%. Although the accuracy of the global model will eventually converge to a level close to that without attacks, this significant drop near the final rounds of FRL could have catastrophic consequences.

Using only clustering: As shown in Fig. 10b, when only clustering is used, the global model's accuracy experiences a noticeable drop, slowing down the convergence of the global model. This highlights the necessity of outlier detection. Additionally, the lack of voting rollback not only increases fatal omissions but also affects the final accuracy. This proves that relying solely on clustering cannot eliminate the influence of malicious clients.

TABLE IV: Comparison of Accuracy, Computational, and Communication Costs of Aggregation Methods on MNIST and CIFAR-10. Abbreviations: Acc. = Accuracy, Srv. = Server-side computation time, Up = Upload size per client. Results are reported under 10% malicious client ratio.

Dataset	Method	Acc.(%)	Srv.(s)	Up(MB)
MNIST	FedAvg	10.02	0.15s	6.23
	Trimmed-mean	90.58	0.13s	6.23
	Multi-Krum	91.24	0.86s	6.23
	FRL	92.63	0.12s	4.05
	FRL++	98.98	5.22s	4.05
CIFAR-10	FedAvg	10.33	0.28s	20.13
	Trimmed-mean	57.96	0.25s	20.13
	Multi-Krum	60.83	1.42s	20.13
	FRL	11.94	0.30s	14.00
	FRL++	83.92	19.45s	14.00

5) Comparative Analysis of Computational and Communication Efficiency: We compare FRL++ with FRL, Multi-Krum, FedAVG, and Trimmed-Mean. The experimental settings for Multi-Krum, FedAvg, and Trimmed-Mean are consistent with those used in [14].

Experimental results: Table IV presents the results of our efficiency analysis experiments. We observe that: (i) FRL++ effectively mitigates the impact of poisoning attacks, achieving the highest accuracy under adversarial settings across both datasets. However, this improvement in robustness comes at the cost of increased server-side computational overhead, primarily due to the added clustering, outlier detection, and rollback mechanisms; (ii) In contrast, traditional defenses for FL such as Trimmed-mean and Multi-Krum also incur significant server-side computation (Multi-Krum also exhibit quadratic complexity in the number of clients) yet they provide weaker protection. This highlights a broader tradeoff: stronger defense mechanisms generally demand higher computational costs; and (iii) The comparison across MNIST (small model and fewer data) and CIFAR-10 (larger model and more data) shows that both communication and server-side computation costs scale consistently with the theoretical complexities described in SectionVI-D, growing approximately quadratically with model and dataset size. This observation suggests that incorporating parameter-efficient fine-tuning methods such as LoRA could substantially reduce the overhead of FRL++, making it practical for deployment in real-world federated settings.

6) FRL++ Robustness against targeted poisoning attack: It is worth noting that while this paper primarily focuses on defending against non-targeted poisoning attacks, for the sake of experimental completeness, we also evaluated the robustness of FRL++ against targeted poisoning attacks, such as semantic backdoor attacks [19]. The scaling attack proposed by Bagdasaryan et al. [19] cannot be implemented in the FRL++ scenario because clients upload edge ranking vectors, and each ranking is unique, making it impossible to scale the

ranking vectors.

Experimental results: As shown in Table V, we compared the accuracy on the main task and the backdoor task for malicious client proportions of 2%, 5%, and 10%. Since FRL itself already offers some defense against targeted backdoor attacks, our FRL++ framework continues to defend against the backdoor attack while slightly enhancing the defense capability. Due to space limitations, the experimental setup is provided in Appendix D.

VIII. Related Work

In this section, we will introduce the poisoning attacks currently faced by FL VIII-A and the methods and techniques for defending against these poisoning attacks VIII-B.

A. Poisoning Attacks in FL

The distributed training approach of federated learning provides attackers with opportunities to launch poisoning attacks. Attackers can control or manipulate a subset of clients and their local data, uploading carefully crafted malicious models to compromise the integrity of the training process and the reliability of the final global model.

Poisoning attack methods are generally classified into targeted poisoning attacks [25], [26], [16], [27] and non-targeted poisoning attacks [18], [28], [29], [30], [31]. Targeted poisoning attacks, also known as backdoor attacks, aim to manipulate the global model so that it produces a desired output for specific inputs selected by the attacker. Non-targeted poisoning attacks, on the other hand, aim to disrupt the convergence of the global model. Once successful, they can be more destructive than targeted attacks.

B. Defenses against Poisoning Attacks in FL

Given the threat of poisoning attacks to FL, many defense measures have been developed to counter potential poisoning attacks in FL. Currently, defense strategies against poisoning attacks in FL can be broadly classified into three categories. The first category is robust aggregation [31], [32], [33], [34], [35], [36], where the server applies specific rules during the aggregation of client updates to mitigate the impact of malicious updates, such as methods like Trimmed-mean [36], Krum [31], and Median [36]. Using robust aggregation to defend against poisoning attacks is computationally simple and suitable for detecting attacks that significantly affect model updates. However, if the attacker employs a stealthy poisoning attack with minimal changes in the malicious updates, the defense's effectiveness will be greatly reduced.

The second category involves using anomaly detection methods to filter out potential malicious updates before aggregation [37]. These methods use carefully designed statistical indicators to filter the models uploaded by clients, with possible techniques including clustering [38] and trimming [38], among others. However, anomaly detection methods contradict the main idea of FL. Since FL leverages the diversity of NON-IID training data from different users (including rare or low-quality data), discarding model updates that differ from the global model is contrary to this principle.

The third category involves post-processing the global model after aggregation, with main techniques including model distillation [39], model pruning [40], and adding noise [41], among others. The downside of these methods is that they tend to have poor interpretability and may significantly impact the performance of the global model.

IX. Conclusion and Limitations

This paper re-examines the robustness of FRL against poisoning attacks and conducts research based on three research questions. First, we pointed out the flaws in the theoretical and experimental analyses of FRL's robustness in previous studies and demonstrated through experiments that FRL is still vulnerable to simple and direct poisoning attacks. Second, we re-evaluated the impact of the transformation of FRL's update space from continuous to discrete, highlighting an inherent advantage of FRL in defending against poisoning attacks: it is difficult for attackers to circumvent defenses by adding constraint terms to implement covert attacks. Therefore, leveraging this advantage of FRL, we introduced a defense mechanism called FRL++, which further reduces the discrete update space to a secure range, enhancing the robustness of FRL against poisoning attacks. Extensive experiments have validated the effectiveness of this mechanism.

However, our study primarily focuses on poisoning attacks and does not address several other important issues related to FRL and FRL++ in the context of logical reasoning, such as their performance under non-IID data distributions, their effectiveness in other modalities, and their impact on communication efficiency. In addition, our experiments follow the original FRL setting and do not explore the applicability of these methods to larger models (e.g., Transformer-based architectures) or other modalities (e.g., natural language). Given the additional overhead introduced by FRL++, this may raise practical concerns. Nevertheless, considering that many current federated learning approaches for large models rely on parameter-efficient tuning methods (e.g., LoRA), which can even require fewer trainable parameters than full fine-tuning of a ResNet, this opens up possibilities for applying FRL(++) to larger-scale models and more complex tasks. We plan to further explore this direction in future work.

X. Acknowledgements

This research is supported by the Strategic Priority Research Program of the Chinese Academy of Sciences, Grant No. XDB0690303.

References

[1] "General data protection regulation, 2018," https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[2] F. Zhou, S. Liu, H. Fujita, X. Hu, Y. Zhang, B. Wang, and K. Wang, "Fault diagnosis based on federated learning driven by dynamic expansion for model layers of imbalanced client," *Expert Systems with Applications*, vol. 238, p. 121982, 2024.

[3] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *International journal of medical informatics*, vol. 112, pp. 59–67, 2018.

[4] N. N. Thilakarathne, G. Muneeswari, V. Parthasarathy, F. Alassery, H. Hamam, R. K. Mahendran, and M. Shafiq, "Federated learning for privacy-preserved medical internet of things," *Intell. Autom. Soft Comput*, vol. 33, no. 1, pp. 157–172, 2022.

[5] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, "Fedhealth: A federated transfer learning framework for wearable healthcare," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 83–93, 2020.

[6] D. C. Nguyen, M. Ding, P. N. Pathirana, and A. Seneviratne, "Blockchain and ai-based solutions to combat coronavirus (covid-19)-like epidemics: A survey," *Ieee Access*, vol. 9, pp. 95 730–95 753, 2021.

[7] R. Kumar, A. A. Khan, J. Kumar, N. A. Golilarz, S. Zhang, Y. Ting, C. Zheng, W. Wang *et al.*, "Blockchain-federated-learning and deep learning models for covid-19 detection using ct imaging," *IEEE Sensors Journal*, vol. 21, no. 14, pp. 16 301–16 314, 2021.

[8] X. Hou, J. Wang, C. Jiang, X. Zhang, Y. Ren, and M. Debbah, "Uav-enabled covert federated learning," *IEEE Transactions on Wireless Communications*, 2023.

[9] M. Fu, Y. Shi, and Y. Zhou, "Federated learning via unmanned aerial vehicle," *IEEE Transactions on Wireless Communications*, 2023.

[10] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for iot," in *2019 IEEE 39th International conference on distributed computing systems (ICDCS)*. IEEE, 2019, pp. 756–767.

[11] 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.

[12] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and trends® in machine learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[13] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1354–1371.

[14] H. Mozaffari, V. Shejwalkar, and A. Houmansadr, "Every vote counts:{Ranking-Based} training of federated learning to resist poisoning attacks," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 1721–1738.

[15] V. Tolpegin, S. Truex, M. E. Gursoy, and L. 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*. Springer, 2020, pp. 480–501.

[16] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *International conference on learning representations*, 2019.

[17] 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.

[18] 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.

[19] 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.

[20] T. Krauß, J. König, A. Dmitrienko, and C. Kanzow, "Automatic adversarial adaption for stealthy poisoning attacks in federated learning," in *To appear soon at the Network and Distributed System Security Symposium (NDSS)*, 2024.

[21] H. Li, Q. Ye, H. Hu, J. Li, L. Wang, C. Fang, and J. Shi, "3dfed: Adaptive and extensible framework for covert backdoor attack in federated learning," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 1893–1907.

[22] C. Spearman, "Demonstration of formulæ for true measurement of correlation," *The American Journal of Psychology*, vol. 18, no. 2, pp. 161–169, 1907. [Online]. Available: http://www.jstor.org/stable/1412408

[23] S. Andreina, G. A. Marson, H. Möllering, and G. Karame, "Baffle: Backdoor detection via feedback-based federated learning," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 852–863.

[24] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[25] 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.

[26] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*. PMLR, 2019, pp. 634–643.

[27] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "M," *arXiv preprint arXiv:1911.07963*, 2019.

[28] C. Xie, O. Koyejo, and I. Gupta, "Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation," in *Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 261–270.

[29] 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.

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

[31] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.

[32] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, 2017.

[33] 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.

[34] S. Rajput, H. Wang, Z. Charles, and D. Papailiopoulos, "Detox: A redundancy-based framework for faster and more robust gradient aggregation," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[35] C. Xie, S. Koyejo, and I. Gupta, "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6893–6901.

[36] 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.

[37] S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen, "Abnormal client behavior detection in federated learning," *arXiv preprint arXiv:1910.09933*, 2019.

[38] T. D. Nguyen, P. Rieger, R. De Viti, H. Chen, B. B. Brandenburg, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen *et al.*, "{FLAME}: Taming backdoors in federated learning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1415–1432.

[39] W. Li, C.-h. Wang, G. Cheng, and Q. Song, "International conference on machine learning," *Transactions on machine learning research*, 2023.

[40] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.

[41] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

APPENDIX

A. Analysis of Voting System

Let us define the voting system: There are m clients, where m is a positive integer. They will select k edges from m edges and the number of malicious clients is s, $s << m$.

First Voting Method (Multiple Selection Voting): Each voter selects n edges, and each selected edge receives one vote. The n edges with the highest total votes are elected.

Second Voting Method (Weighted Ranking Voting): Each client ranks n edges in order of preference. In the ranking of an edge, the edge ranked i^{th} receives $n - i + 1$ points, where

$i = 1, 2, \ldots, n$. The n edges with the highest total points are elected.

Manipulation Strategy of the malicious clients: The malicious clients S aim to maximize the election chances of their preferred edges (denoted as set C_S) and minimize the chances of the edges they oppose (denoted as set C_N). In the first method, they can only increase the vote count of edges in C_S by voting for them; they cannot decrease the vote counts of edges in C_N. In the second method, they can assign the highest scores to edges in C_S and the lowest scores to edges in C_N, thereby both boosting their preferred edges and diminishing the scores of the opposed edges.

Influence in the First Method: For any candidate c, the total number of votes is:

$$V_c = V_c^s + V_c^{Others} \tag{8}$$

Where V_c^s is the number of votes from the malicious clients S, which can be s (if they vote for c) or 0 (if they do not). $V_c^{Others}$is the number of votes from other voters. The maximum increase in votes that the malicious clients can contribute to any edge is s.

Influence in the Second Method: For any candidate c, the total points are:

$$P_c = P_c^s + P_c^{Others} \tag{9}$$

Where P_c^s is the total points from the malicious clients S, ranging from $s1$ (if ranked last) to $s * n$ (if ranked first). P_c^{Others} is the total points from other voters. By adjusting their rankings, the malicious clients can manipulate each edge's points within the following range: For supported edges $c \in C_s$, they can set $P_c^s = s * n$ (each member gives the maximum score). For opposed edges $c \in C_n$, they can set $P_c^s = s * 1$ (each member gives the minimum score). Thus, the influence of the malicious clients on each candidate's points is:

$$\Delta P_c^s = s * (n - 1) \tag{10}$$

Since $n > 1$, it follows that:

$$s * (n - 1) >> s \tag{11}$$

Assuming that the votes or points from other clients are relatively evenly distributed, and the typical difference between edges is some constant d:

First Method: malicious clients needs $s > d$ to potentially change the election outcome. Second Method: malicious clients need $s \cdot (n - 1) > d$ to change the outcome. Because $n - 1$ amplifies the influence of malicious clients, they have a greater capacity to alter the results in the second method.

B. Analysis of Relationship Between Real Number Vector and Ranking Vector

Prove that minimizing the Euclidean distance $D(X, Y)$ to make X and Y converge does not guarantee that their rank vectors $R(X)$ and $R(Y)$ will also converge.

Lemma 1. *The Ranking Function is Discontinuous over R_n.*

Proof. Consider the ranking function $R(X)$. For any two elements X_i and X_j, if their values are very close, a tiny perturbation can change their relative order, thus altering the ranking. Let $X_i = a$ and $X_j = a + \delta$, where $\delta > 0$ is very small. If we apply a small perturbation ϵ to X_i, such that $X_i' = X_i + \epsilon$, when $\epsilon > \delta$, $X_i' > X_j$ leading to a change in ranking.

Therefore, the ranking function is discontinuous with respect to small changes in X.

Let $X \in R_n$ be a real-valued vector with all elements distinct to ensure unique rankings. Construct $Y = X + \epsilon v$, where $v \in \mathbb{R}^n$ is a vector, and $\epsilon > 0$ is a very small positive number. Analyze the Euclidean Distance between X and Y.

Since ϵ is very small, $D(X, Y)$ is also small, indicating that X and Y are numerically very close. Analyze the Difference in Rank Vectors $R(X)$ and $R(Y)$. Despite X and Y being close in value, due to the discontinuity of the ranking function, $R(X)$ and $R(Y)$ may differ.

Case 1: Large Differences between Elements If the differences between elements of X are large, the small perturbation ϵ is insufficient to alter the relative order of elements, resulting in $R(X) = R(Y)$.

Case 2: Small Differences between Elements If certain elements of X are very close in value, the small perturbation ϵ can change their relative order, leading to $R(X) \neq R(Y)$.

Construct a Counterexample Let $X = [a, a + \delta]$, where $\delta > 0$ is very small. Let $Y = X + \epsilon[-1, 1]$, so:

$$Y = [a - \epsilon, a + \delta + \epsilon]$$

When ϵ satisfies $\epsilon > \frac{\delta}{2}$, the ordering of elements in Y may change compared to X, depending on the relationship between ϵ and δ.

This demonstrates that even when $D(X, Y)$ is small, the rank vectors $R(X)$ and $R(Y)$ can be different.

Conclusion:

Since the ranking function is discontinuous over the real numbers, minimizing the Euclidean distance $D(X, Y)$ does not guarantee that the rank vectors $R(X)$ and $R(Y)$ will converge.

C. Algorithm Design Details

The algorithm 1 introduced in this section is the FRL++ defense enhancement framework, designed to protect federated learning (FL) systems from attacks by malicious clients and improve the robustness of FRL. Below are the key steps of the algorithm:

Server Initialization: The server first initializes the variables and aggregates client votes to decide whether a rollback to the previous round is needed.

Client Training: Each selected client U trains a local model R_u^t using its local dataset D_u and the current global model R_g^t.

Server Operations (After Client Training):

- BIRCH Clustering (Server-side): The server uses BIRCH to cluster the trained models and detect outliers based on the ranking vectors of client models. If a model does not belong

Algorithm 1

Input: number of rounds T, number of total clients N,number of clients randomly sampled per round n, random seed $SEED$, learning rate η, loss function of i^{th} client L_i, number of local epochs E, number of layers of network model L, dataset of i^{th} client D_i, threshold for decreasing accuracy τ, threshold for outlier detection q
Output: Global Ranking Model R_g^T
Server:Initialization
$\theta^w \leftarrow$ **Initialize using SEED** , $R_g^1 \leftarrow$ **Initialize**
$last\_acc_{i \in N} \leftarrow 0$, $vote\_rollback_{i \in N} \leftarrow False$
for $t \in [1, T]$ **do**
  $U \leftarrow$ **set of n randomly selected clients out of N total clients**
  Client:Train
  for $u \in U$ **do**
    $\theta^w \leftarrow$ **Initialize using SEED**
    $R_u^t \leftarrow$ **TRAIN**$(R_g^t, D_u, \theta^w, L_u)$
  end for
  for $l \in [1, L]$ **do**
    $f_u \leftarrow True$
    Server:Clusering
    $clusters \leftarrow$ **BIRCH**$({R_{u \in U}^t}^l)$
    if ${R_{u \in U}^t}^l \notin$ **MAX**$(clusters)$ **then** $f_u \leftarrow False$
    end if
    Server:Outlier Detection
    standard_arrange $\leftarrow [1, 2, 3..,$ **EDGENUMBER**$(l)]$
    $\rho_{u \in U}^l \leftarrow$ **SPEARMAN**$({R_{u \in U}^t}^l, standard\_arrange)$
    $dist_{u \in U}^l \leftarrow 1 - \rho_{u \in U}^l$, $dist_{max}^l \leftarrow$ MAX$(dist_{u \in U}^l)$, $dist_{min}^l \leftarrow$ MIN$(dist_{u \in U}^l)$
    $\Delta \leftarrow dist_{max}^l - dist_{min}^l$
    if $dist_{u \in U}^l > dist_{max}^l * (1 - q * \Delta)$ **or** $dist_{u \in U}^l < dist_{min}^l * (1 + q * \Delta)$ **then** $f_u \leftarrow False$
    end if
    Server:FRL Aggregation
    $benign\_models \leftarrow$ **[ ]**
    if $f_{u \in U} = True$ **then**
      $benign\_models \leftarrow$ **ADD**$(benign\_models, {R_u^t}^l)$
    end if ${R_g^{t+1}}^l \leftarrow VOTE(benign\_models)$
  end for
  Client:Rollback Vote
  for $u \in U$ **do**
    $\theta^w \leftarrow$ **Initialize using SEED**
    $acc \leftarrow$ **TEST**$(R_l^t,$ **SUBSET**$(D_u))$
    if $acc_i last\_acc_u$ - τ **then** $vote\_rollback_u \leftarrow True$
    else $vote\_rollback_u \leftarrow False$
    end if
    $last\_acc_u \leftarrow acc$
  end for
  Server:Rollback Aggregation
  if SUM$(vote\_rollback_{u \in U}) > 1/2 * n$ **then** $R_g^t \leftarrow R_g^{t-1}$
  end if
end for

to the MAX cluster, it is flagged as a potential malicious model (set f_u to False).

- Outlier Detection (Server-side): The server calculates the Spearman correlation between each model's ranking vector and the standard edge ranking. It then calculates the distance between the correlations. If any client model's distance exceeds a defined threshold (determined by q), the model is considered an outlier and flagged as a potential malicious model.
- Aggregation (Server-side): If a model is deemed benign, it is added to the list of models to be aggregated. The server aggregates the benign models using a voting mechanism and updates the global model R_g^{t+1}.

Fig. 11: Semantic backdoor

TABLE V: The table displays a comparison of the final test accuracy of the global model and the accuracy on the backdoor task when FRL and FRL++ are faced with targeted poisoning attacks.

	final test			**backdoor**		
malicious rate	2%	5%	10%	2%	5%	10%
FRL	84.43	84.31	84.30	21.45	45.34	56.50
FRL++	84.09	84.03	83.95	16.62	39.26	49.30

- Client Rollback Check: After aggregation, the server sends the global ranking to the clients, who test the model on a subset of their dataset and compare the accuracy with the previous round's accuracy. If the accuracy drops by more than the threshold τ, the client votes for a rollback.
- Server Rollback Aggregation: If more than half of the clients request a rollback in this round, the server restores the global model R_g^t to the previous round's model R_g^{t-1}.
- Repeat: Steps 2 to 5 are repeated for each round until T rounds are completed.

D. Experimental Setup for Backdoor Attacks

We selected images with vertical striped walls in the background as the backdoor trigger. In the CIFAR-10 dataset, there are a total of 22 images that fit the semantic backdoor trigger, for example, as shown in Figure 11. We used 18 of these images with the trigger to train the backdoor, leaving the remaining 4 for testing the backdoor accuracy. The attacker's goal is to have the global model classify these images as birds (category label = 2) without reducing the accuracy on the main task. Furthermore, we applied random rotations and cropping to the 4 images used for testing the backdoor accuracy, ultimately expanding the number of test images to 1000.