The Defense Against Backdoor Attacks Using Trigger Inversion and Data Augmentation in Clustered Federated Learning

Hailong Tang, Lei Shi(✉), Yingfei Zhu, Cheng Gu, and Juan Xu

School of Computer Science and Information Engineering,
Hefei University of Technology, Hefei 230009, China
shilei@hfut.edu.cn

Abstract. Clustered Federated Learning (CFL) is a paradigm of Federated Learning (FL) that enhances model training accuracy and efficiency by clustering clients with similar characteristics and training models within each cluster. However, the structural characteristics of CFL make it more susceptible to backdoor attacks, where the attacker embeds a malicious trigger in clean data and modifies the label to the target label, resulting in malicious outputs when the trigger is activated. To address this threat, this paper proposes a defense method combining trigger inversion and data augmentation. First, we conduct a comparative analysis of the impact of backdoor attacks in FL and CFL, revealing that CFL is more susceptible to such attacks. Then, based on this analysis, the proposed defense method employs reverse engineering on the model compromised by a backdoor attack to detect trigger patterns and generate a potential trigger. This trigger is then applied to augment client data, neutralizing the effects of the malicious trigger. Experimental results demonstrate that the proposed defense method can reduce the attack success rate to nearly zero while maintaining high model accuracy, showcasing its robustness in defending against backdoor attacks.

Keywords: Clustered federated learning · Backdoor attack · Backdoor defense · Trigger inversion · Data augmentation

1 Introduction

Clustered Federated Learning (CFL) is a type of Federated Learning (FL) [1,2], which aggregates clients with similar features or data distributions into the same cluster and trains models independently within each cluster. This approach effectively addresses the challenges posed by data heterogeneity [3,4] in FL. Compared to traditional FL, CFL enables independent model training within each cluster, allowing the model to align with the local data distribution, which in turn enhances personalization, training efficiency, and convergence speed [5]. However,

Z. Cai et al. (Eds.): WASA 2025, LNCS 15686, pp. 331–343, 2025.
https://doi.org/10.1007/978-981-96-8725-1_27

while enhancing model performance, the structure of CFL also introduces new security risks. Since model updates between clusters are independent, malicious clients are more likely to launch attacks within clusters, particularly data poisoning and backdoor attacks [6–8]. These attacks are often highly destructive and can influence model updates in a way that is difficult to detect, thereby compromising the model's overall performance.

In recent years, significant progress has been made in the defense against backdoor attacks, with existing methods including differential privacy, gradient clipping [9,10], model behavior-based detection [11], and anomaly detection [12,13]. For instance, Sun et al. [14] mitigate backdoor attacks using weak differential privacy and gradient norm clipping, though at the cost of reduced model accuracy. Andreina et al. [15] proposed the introduction of feedback loops in federated learning, utilizing data from multiple clients to assess the authenticity of model updates, thus effectively detecting and defending against backdoor attacks. However, this method cannot defend against backdoor attacks based on trigger insertion, and it is limited to detecting malicious models without correcting or eliminating existing backdoor mechanisms. While these methods have shown some effectiveness in FL, they fail to address model contamination after aggregation or utilize the information from the contaminated models, making it difficult to eliminate the impact of malicious clients. In CFL, these methods have a more significant negative impact on model accuracy. Attackers can manipulate a small number of malicious clients to achieve high attack success rates while maintaining relatively high model accuracy. Therefore, addressing backdoor attacks in CFL has become a critical challenge in current research.

We compare the performance of backdoor attacks in CFL and FL. The experimental results show that the success rate of backdoor attacks in CFL is approximately 62.89% higher than in FL, indicating that the clustering mechanism in CFL enhances the persistence and effectiveness of these attacks. Therefore, we propose a defense method combining trigger inversion and data augmentation to mitigate backdoor attacks in CFL. The experimental results validate the effectiveness and robustness of the proposed defense method under different poisoning rates. On both the MNIST and EMNIST datasets, the defense significantly reduces the backdoor attack success rate to near zero at all poisoning rates, while maintaining stable model accuracy.

The main contributions of this paper are summarized as follows:

- We demonstrate that CFL is more vulnerable to backdoor attacks than FL.
- We propose a defense mechanism against backdoor attacks that combines trigger inversion with client data augmentation.
- Through experiments, we demonstrate that our defense method is more effective in mitigating backdoor attacks in CFL compared to other methods.

The structure of the paper is as follows. Section 2 presents the problem formulation and attack description. Section 3 introduces the defense method. Section 4 details the experimental setup and analyzes the results related to both the attack and defense. Finally, Sect. 5 concludes the paper and discusses future research directions.

2 Problem Formulation and Attack Description

Consider a Clustered Federated Learning (CFL) scenario, as illustrated in Fig. 1, consisting of a central parameter server and multiple clients with Non-Independent and Identically Distributed (Non-IID) data. Assume that the clients can be divided into n clusters based on their data distributions, where D_i $(i = 1, \ldots, n)$ represents the set of clients sharing the same data distribution. The notations used in the paper are summarized in Table 1.

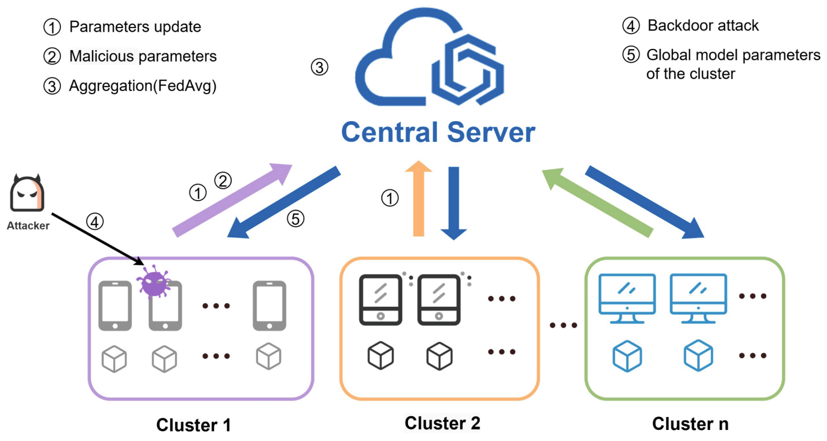


Fig. 1. Threat model.

In our threat model, we assume that the clustering process has been completed, and one local client is maliciously controlled by an attacker to perform a backdoor attack in CFL. The attacker manipulates the model by injecting poisoned samples during local training. The goal of the attack is to implant a backdoor into the model, causing it to make predefined incorrect predictions when encountering samples with the trigger, while maintaining high accuracy on benign samples. Since the global server cannot distinguish between trusted and malicious clients, the attacker's malicious behavior may be incorporated into the model, leading to erroneous predictions under specific trigger conditions. Specifically, the attack process can be divided into the following steps:

1. **Generating poisoned samples**: The malicious client controlled by the attacker modifies a portion of the client's benign samples according to the poisoning rate r, embeds a trigger, and changes their labels to the target label, thereby generating poisoned samples.
2. **Training with poisoned data**: The malicious client incorporates the poisoned samples into the local training process to update the model.

Table 1. Glossary of notations.

Notation	Description
α	The Dirichlet distribution parameter controlling the non-IID level
w_0	The initial cluster model weights before defense
W_r	The cluster model weights at round r
η	The learning rate (step size) for model updates
x_{aug}	The augmented sample with the trigger applied
τ	Gradient update threshold for model updates
m	The mask indicating the regions where the trigger is applied
δ	The perturbation representing the trigger pattern updated during the optimization process
r	The poisoning rate of malicious client data

3 Defense

3.1 Defense Model

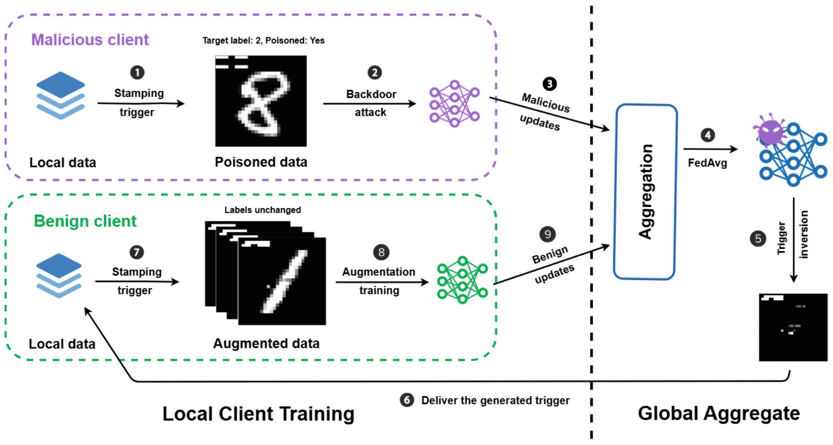


Fig. 2. Defence framework.

To address backdoor attacks in CFL, we propose a defense method that combines trigger inversion and data augmentation. The entire defense process is illustrated in Fig. 2. Backdoor attacks typically lead to abnormal model updates, so we monitor the parameter updates of each neural network layer in the server-side clustered model during each training round to identify potential anomalies. When a model update deviates from the expected range, it may indicate that the model has been affected by a backdoor attack, thus activating the defense mechanism.

Once an anomaly is detected, we initiate defense measures, including trigger inversion and data augmentation. The core idea of trigger inversion is to analyze the behavior of the attacked model, reverse-engineer the malicious patterns embedded by the attacker in the data, and generate a trigger. This trigger reveals the malicious features introduced by the attacker and can interfere with the attacker's strategy. The generated trigger is then applied to the datasets of the clients, enhancing their ability to identify and resist backdoor attacks during training. Algorithm 1 outlines the implementation process of this defense method.

Algorithm 1. Defense method

```
Input: total training round $R$, local dataset $D : \{x, y\}$, clients $K$
1: initialize learning rate $\eta$, gradient update threshold $\tau$, initial model parameters $w_i$
2: for each training round $r \in [1, R]$ do
3:     for each client $k$ in $K$ do
4:         Client $k$ does:
5:         $w_{r+1}^k \longleftarrow$ Local_Update $(w_r, D_k, mask, pattern)$
6:     end for
7:     Server does:
8:     Clustering: Calculate weight similarity. The clients are divided into two parts
9:     using the aggregation hierarchical clustering algorithm;
10:    $w_{r+1} \leftarrow \eta \sum_{k=1}^{N} w_{r+1}^k$    ▷ The server aggregates weights from clients in the cluster.
11:    if $\exists l, \left\| \Delta w_{r+1}^{(l)} \right\| > \tau$ then   ▷ Check if the gradient update for layer $l$ is abnormal.
12:        mask, pattern $\longleftarrow$ Trigger_Inversion $(w_{r+1})$   ▷ Generate a trigger through inversion based on the poisoned model.
13:    end if
14:    $w_{r+1}, mask, pattern \longrightarrow$ clients $K_{cluster}$   ▷ Distribute updated model parameters, trigger mask, and pattern to the clients in the cluster.
15: end for
16: function LOCAL_UPDATE$(w_r, D_k, mask, pattern)$
17:    if $mask\ is\ Null$ then
18:        $w_{r+1}^k \longleftarrow$ Local_Model_Train$(x, y)$
19:    else
20:        $x_{aug} = (1 - mask) \cdot x + mask \cdot pattern$
21:        $w_{r+1}^k \longleftarrow$ Data_Augmentation_Train $(x_{aug}, y)$   ▷ Training on augmented data without changing the ground truth labels.
22:    end if
23: end function
```

3.2 Trigger Inversion

In this section, we provide a detailed discussion on the principles, implementation steps, and application of trigger inversion in the context of CFL. Trigger inversion is an effective technique for detecting backdoor attacks by performing reverse optimization on the attacked model to derive the malicious triggers implanted by the attacker. The core idea is to generate triggers corresponding to each label through the inversion process, thereby revealing potential malicious patterns. Since backdoor attacks typically involve making subtle modifications to the input data, causing misclassification to the target label, the modification needed for misclassification to the target label is significantly smaller than that required for other unaffected labels. Trigger inversion leverages this characteristic to detect the model's sensitivity to input, thus identifying potential backdoor attacks.

Trigger Generation. In this phase, we first randomly initialize the trigger image or mask m, typically using random values or a zero image as the initial state. Next, we adopt a reverse optimization strategy to iteratively update the perturbation value δ of the trigger, guiding the model's output toward the specified target label. For this purpose, we use the following formula to iteratively optimize the input data:

$$x_{s \to t} = (1 - m) \cdot x_s + m \cdot \delta \tag{1}$$

where x_s represents the original input data, m is the mask that controls the degree of the trigger's application, δ is the perturbation value of the trigger updated during optimization, and $x_{s \to t}$ denotes the modified input after transitioning from state s to target state t. To evaluate the model's performance, we define the following loss function:

$$Loss = \mathcal{L}(F(x_{s \to t}), y_t) + \lambda \cdot \|m\| \tag{2}$$

where $\mathcal{L}(\cdot)$ represents the loss function, $F(x_{s \to t})$ is the model's output after applying the trigger to the modified input $x_{s \to t}$, y_t is the target label, λ is a hyperparameter used to balance the contribution of the regularization term, and $\|m\|$ is the $L1$ norm of the mask m, which is used to constrain the complexity of the mask. By minimizing the target loss function, we iteratively optimize the perturbation value δ so that the input $x_{s \to t}$ is misclassified as the target label y_t after applying the trigger. Figure 3 illustrates the process of trigger generation.

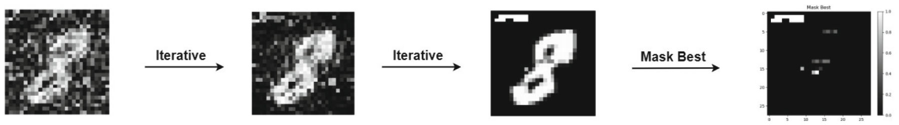


Fig. 3. Trigger generation.

Through this reverse optimization process, we can effectively infer the original trigger that the attacker may have used. This method not only enhances the detection capability against potential backdoor attacks but also provides theoretical support for designing more targeted defense strategies. Algorithm 2 illustrates the implementation process of trigger inversion.

Algorithm 2. Trigger inversion

```
Input: global model parameters $W_r$, the classes of image classification $C$
Output: $mask, pattern$
 1: initialize the maximum number of iterations $T = 1000$
 2: function TRIGGER_INVERSION($w_r$)
 3:     for each $class\ c\ in\ C$ do
 4:         $m_c, \delta_c \longleftarrow$ random init with shape of $x$
 5:         for each $step \in [1, T]$ do
 6:             $x_{aug} = (1 - m_c) \cdot x + m_c \cdot \delta_c$
 7:             Evaluate the recognition performance or calculate the loss for $W_r$, $x_{aug}$,
 8:             and $c$.
 9:             Perform appropriate scaling on $m_c$ and $\delta_c$   ▷ Enhance numerical stability
    or adjust the mask's range to a specific interval.
10:         end for
11:         Add $m_c$, $\delta_c$ to set $S$
12:     end for
13:     Select optimal $m_{opt}$, $\delta_{opt}$ from set $S$ based on MAD.
14:     return $m_{opt}$, $\delta_{opt}$
15: end function
```

Anomaly Detection. We apply the aforementioned optimization method to reverse-engineer triggers for each poisoned label and compute their $L1$ norms. Next, we identify triggers that appear as outliers in the distribution, which typically have smaller $L1$ norms.

After generating the triggers, we use the Median Absolute Deviation (MAD) method for anomaly detection to identify potential outliers in the model's output, which may indicate contamination. Specifically, we first compute the median of the model's outputs, then calculate the absolute deviation of each output from the median, and ultimately obtain the MAD value. Since the perturbation required to misclassify an input as a poisoned label is generally smaller than the perturbation needed to misclassify it as any other unaffected label, the anomalous triggers tend to be located at the lower end of the distribution.

3.3 Data Augmentation

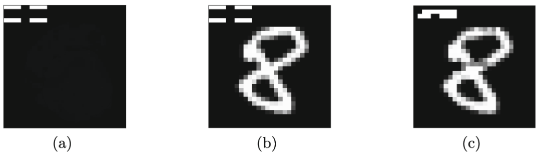


Fig. 4. (a) is the truth trigger, (b) is the poisoned data from the malicious client, and (c) is the augmented data after trigger inversion.

The data augmentation training process involves several key steps. First, after receiving the trigger sent by the server, the client applies it to the benign data, making appropriate modifications while ensuring that the labels remain unchanged to guarantee the effectiveness of the training. Next, the client uses the modified data for data augmentation training, aiming to counteract the impact of backdoor attacks and improve the model's robustness. By learning from these augmented samples, the model gradually enhances its ability to recognize and defend against potential attacks. After the training is complete, the client sends the updated model parameters back to the server, which is responsible for final evaluation and optimization to verify the effectiveness of the adversarial training. This training significantly strengthens the system's ability to handle potential backdoor attacks, improving both the performance and security of the model. Figure 4 shows the poisoned data and the augmented data.

4 Experiments

4.1 Experiment Setup

We use a Convolutional Neural Network (CNN) model for both the MNIST and EMNIST datasets. The input size of the model is 28×28, with the MNIST dataset outputting 10 classes and the EMNIST dataset outputting 62 classes. The network consists of two convolutional layers and two fully connected layers. The convolutional layers use 5×5 kernels with a stride of 1, followed by max-pooling layers for dimensionality reduction. The fully connected layers map the features to 500 neurons, with the final output corresponding to the classification results. The activation function used throughout the network is ReLU, and the output layer applies log Softmax normalization for multi-class classification.

The experiment is conducted using the PyTorch framework. We create a non-IID dataset by applying a Dirichlet distribution ($\alpha = 0.5$) to the data, distribute it across 100 clients, and select 10 clients for the experiment, one of which is controlled by an attacker. In the attack setup, the SGD optimizer is used for training, and a backdoor attack is performed in a specific round. The learning rate for the backdoor attack is set to 0.05, while the learning rate for normal training is $\eta = 0.1$, with a batch size of 64. Higher poisoning rates can prevent the model from converging, so we use commonly adopted parameter settings of $r = 10\%$, 20%, and 30%. In the defense setup, the Adam optimizer is used for reverse engineering to generate the trigger. The value of $\tau = 1.0$ and the parameters of the pre-trained model $w_i = w_{10}$ (for MNIST) and $w_i = w_{200}$ (for EMNIST) are set in Algorithm 1. The experiment is run on a server equipped with an NVIDIA V100 (16 GB) GPU.

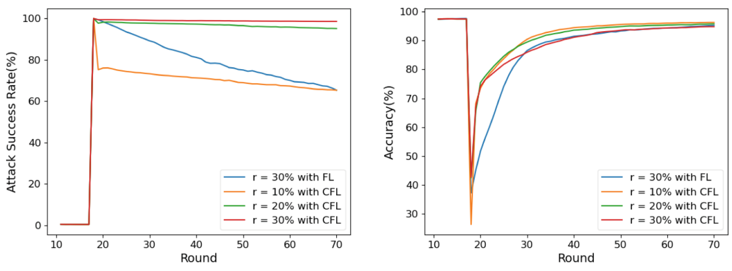
Evaluation Metrics. We use Attack Success Rate (ASR) and Main Task Accuracy (ACC) as evaluation metrics to measure the effectiveness of the defense.

1) Attack Success Rate (ASR): This metric measures the proportion of backdoor task samples with a trigger that are misclassified as as the attack's target label.
2) Main Task Accuracy (ACC): This metric indicates the proportion of benign samples correctly classified to their true labels.

4.2 Attack Result Analysis

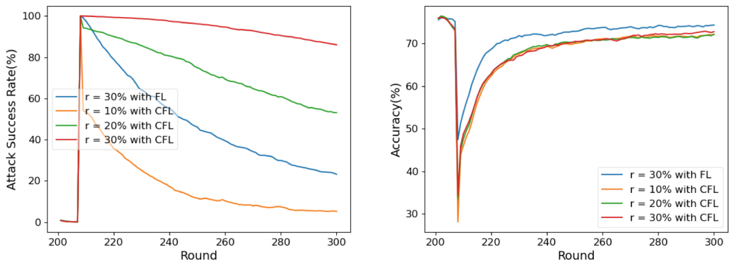
Our research demonstrates that backdoor attacks in CFL pose a greater threat compared to FL. This increased threat can be attributed to the unique client clustering mechanism in CFL, which provides attackers with more favorable conditions for executing their attacks. Compared to FL, CFL achieves better attack performance with the same or even lower poisoning rates.

In the experiments conducted on the MNIST and EMNIST datasets, we compare the performance of FL and CFL in backdoor attacks. The corresponding experimental results are shown in Figs. 5 and 6. The backdoor attack is performed in the 18th round of the MNIST dataset and the 208th round of the EMNIST dataset. At a poisoning rate of 30%, the ASR in CFL is 33.27% higher than in FL on the MNIST dataset, and 62.89% higher on the EMNIST dataset. Furthermore, in the CFL scenario, the attacker can achieve the same or even higher attack success rates than in FL, even at lower poisoning rates.



(a) Backdoor Task (b) Main Task

Fig. 5. Comparison of ASR and ACC between CFL and FL for MNIST.



(a) Backdoor Task (b) Main Task

Fig. 6. Comparison of ASR and ACC between CFL and FL for EMNIST.

Overall, the clustering mechanism in CFL enhances the persistence of backdoor attacks across both datasets, demonstrating a stronger attack effect, while the global aggregation strategy in FL somewhat mitigates the impact of these attacks.

4.3 Defense Result Analysis

We conduct defense experiments based on the backdoor attack scenario in CFL described in Sect. 4.2. Figures 7 and 8 show ACC and ASR of the backdoor task under different poisoning rates with and without defense measures on the MNIST and EMNIST datasets. Table 2 summarizes the baseline results under various poisoning rates for each dataset, comparing the model performance with and without defense. On the MNIST dataset, without any defense, the ASR significantly increases as the poisoning rate of the malicious client rises, reaching 98.62% at a 30% poisoning rate, while the ACC slightly drops to 94.80%. On the EMNIST dataset, the ASR reaches 86.09% when the poisoning rate increases to 30%. However, with the defense method in place, the ASR for both datasets remains close to zero at all poisoning rates, indicating that the defense method effectively mitigates backdoor attacks. Meanwhile, the ACC for both MNIST and EMNIST remains stable, reaching a maximum of 95.83% and 72.10%, respectively. These results demonstrate the outstanding effectiveness and robustness of the proposed defense method across different poisoning rates.

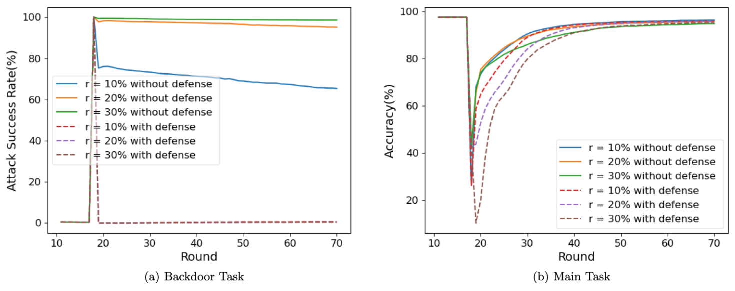


Fig. 7. Comparison of ASR and ACC for MNIST in the CFL scenario at different poisoning rates with and without defense.

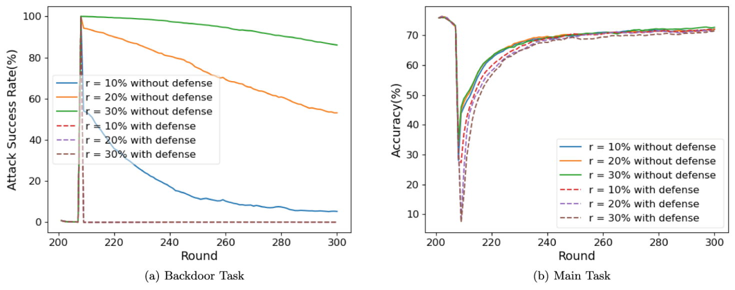


Fig. 8. Comparison of ASR and ACC for EMNIST in the CFL scenario at different poisoning rates with and without defense.

To provide a clearer comparison, the Differential Privacy (DP) method adds Gaussian noise to each client's update, reducing the impact of malicious updates while preserving the model's overall performance. In contrast, Clipped Differential Privacy Defense (CDPD) [14] mitigates the impact of malicious updates by clipping updates from malicious clients and adding Gaussian noise, thereby preventing backdoor attacks from compromising the model.

Table 2. Performance comparison under different poisoning ratios.

CFL Scenario		MNIST		EMNIST	
		ACC	ASR	ACC	ASR
No Defense	$r = 10\%$	96.26	65.34	72.06	5.23
	$r = 20\%$	95.73	95.16	72.12	53.14
	$r = 30\%$	94.80	98.62	72.73	86.09
Defense	$r = 10\%$	95.83	0.63	72.10	0.06
	$r = 20\%$	95.67	0.51	71.65	0.08
	$r = 30\%$	95.14	0.36	71.52	0.08

According to the experimental results in Table 3, our defense method shows significant results on both the MNIST and EMNIST datasets. On the MNIST dataset, the ASR of the backdoor attack drops to 0.36%, while the accuracy increases to 95.14%. On the EMNIST dataset, the ASR drops to 0.08%, and the accuracy remains at 71.52%. Compared to other defense methods, such as DP and CDPD, the proposed method not only significantly reduces the ASR but also demonstrates the stronger defense capability while maintaining high accuracy, thereby confirming its superiority.

Table 3. Comparison of defense strategy effectiveness in the CFL scenario.

Baselines	MNIST		EMNIST	
	ACC	ASR	ACC	ASR
No Defense	94.80	98.62	72.73	86.09
DP	92.50	63.55	65.95	77.25
CDPD	**95.52**	2.58	65.85	0.38
Ours	95.14	**0.36**	**71.52**	**0.08**

5 Conclusion

This study reveals the vulnerabilities of CFL to backdoor attacks, highlighting that CFL faces greater security risks compared to FL. Through theoretical analysis and experimental validation, we find that the client clustering mechanism in CFL creates favorable conditions for attackers, thereby increasing the success rate of backdoor attacks. To address this issue, we propose a defense method based on trigger inversion and data augmentation, which effectively identifies and mitigates malicious inputs, significantly enhancing the security and robustness of CFL. Our research emphasizes the necessity of developing effective defense strategies and provides valuable insights for future security research in CFL.

Acknowledgments. This article was supported by the Anhui Provincial Natural Science Foundation (Grant No. 2308085MF212, No. 2308085MF221) and Joint Fund Project of Natural Science Foundation of Anhui Province (Grant No. 2208085US05, No. 2308085US05).

References

1. Xiong, Z., Cai, Z., Takabi, D., Li, W.: Privacy threat and defense for federated learning with non-I.I.D. data in AIOT. IEEE Trans. Ind. Inform. **18**(2), 1310–1321 (2022)
2. Xia, Q., Ye, W., Tao, Z., Wu, J., Li, Q.: A survey of federated learning for edge computing: research problems and solutions. High-Confidence Comput. **1**(1), 100008 (2021)
3. He, Z., Wang, L., Cai, Z.: Clustered federated learning with adaptive local differential privacy on heterogeneous IoT data. IEEE Internet Things J. **11**(1), 137–146 (2024)
4. Pang, J., Huang, Y., Xie, Z., Han, Q., Cai, Z.: Realizing the heterogeneity: a self-organized federated learning framework for IoT. IEEE Internet Things J. **8**(5), 3088–3098 (2021)
5. Sattler, F., Müller, K.R., Samek, W.: Clustered federated learning: model-agnostic distributed multitask optimization under privacy constraints. IEEE Trans. Neural Netw. Learn. Syst. **32**(8), 3710–3722 (2021)
6. Xu, H., Cai, Z., Xiong, Z., Li, W.: Backdoor attack on 3D grey image segmentation. In: 2023 IEEE International Conference on Data Mining (ICDM), pp. 708–717 (2023)
7. Chen, Z., Tian, P., Liao, W., Yu, W.: Towards multi-party targeted model poisoning attacks against federated learning systems. High-Confidence Comput. **1**(1), 100002 (2021)
8. Gao, Y., Li, Y., Zhu, L., Wu, D., Jiang, Y., Xia, S.T.: Not all samples are born equal: towards effective clean-label backdoor attacks. Pattern Recogn. **139**, 109512 (2023)
9. Liu, Y., Xie, Y., Srivastava, A.: Neural trojans. In: 2017 IEEE International Conference on Computer Design (ICCD), pp. 45–48 (2017)
10. Pang, L., Sun, T., Ling, H., Chen, C.: Backdoor cleansing with unlabeled data. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 12218–12227 (2023)

11. Chou, E., Tramèr, F., Pellegrino, G.: Sentinet: detecting localized universal attacks against deep learning systems. In: 2020 IEEE Security and Privacy Workshops (SPW), pp. 48–54 (2020)
12. Tran, B., Li, J., Mądry, A.: Spectral signatures in backdoor attacks. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS 2018, pp. 8011–8021. Curran Associates Inc., Red Hook (2018)
13. Kopčan, J., Škvarek, O., Klimo, M.: Anomaly detection using autoencoders and deep convolution generative adversarial networks. Transp. Res. Procedia **55**, 1296–1303 (2021)
14. Sun, Z., Kairouz, P., Suresh, A.T., McMahan, H.B.: Can you really backdoor federated learning? arXiv abs/1911.07963 (2019)
15. Andreina, S., Marson, G.A., Möllering, H., Karame, G.: Baffle: backdoor detection via feedback-based federated learning. In: 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), pp. 852–863 (2021)