FedBAP: Backdoor Defense via Benign Adversarial Perturbation in Federated Learning

Xinhai Yan
School of Cyber Science and Engineering, Wuhan University
Wuhan, China
yanxinhai@whu.edu.cn

Libing Wu*
School of Cyber Science and Engineering, Wuhan University
Wuhan, China
wu@whu.edu.cn

Zhuangzhuang Zhang*
School of Cyber Science and Engineering, Wuhan University
Wuhan, China
zhzhuangzhuang@whu.edu.cn

Bingyi Liu
Department of Computer and Artificial Intelligence, Wuhan University of Technology
Wuhan, China
byliu@whut.edu.cn

Lijuan Huo
School of Cyber Science and Engineering, Wuhan University
Wuhan, China
lijuanhuo@whu.edu.cn

Jing Wang
School of Software Engineering, Huazhong University of Science and Technology
Wuhan, China
cswjing@hust.edu.cn

ABSTRACT

Federated Learning (FL) enables collaborative model training while preserving data privacy, but it is highly vulnerable to backdoor attacks. Most existing defense methods in FL have limited effectiveness due to their neglect of the model's over-reliance on backdoor triggers, particularly as the proportion of malicious clients increases. In this paper, we propose FedBAP, a novel defense framework for mitigating backdoor attacks in FL by reducing the model's reliance on backdoor triggers. Specifically, first, we propose a perturbed trigger generation mechanism that creates perturbation triggers precisely matching backdoor triggers in location and size, ensuring strong influence on model outputs. Second, we utilize these perturbation triggers to generate benign adversarial perturbations that disrupt the model's dependence on backdoor triggers while forcing it to learn more robust decision boundaries. Finally, we design an adaptive scaling mechanism to dynamically adjust perturbation intensity, effectively balancing defense strength and model performance. The experimental results demonstrate that FedBAP reduces the attack success rates by 0.22%-5.34%, 0.48%-6.34%, and 97.22%-97.6% under three types of backdoor attacks, respectively. In particular, FedBAP demonstrates outstanding performance against novel backdoor attacks.

CCS CONCEPTS

• **Security and privacy → Distributed systems security**.

KEYWORDS

Federated Learning, Backdoor Attacks, Backdoor Triggers, Defense

*Corresponding authors.

MM '25, October 27–31, 2025, Dublin, Ireland.

ACM ISBN 979-8-4007-2035-2/2025/10…$15.00
https://doi.org/10.1145/3746027.3754814

ACM Reference Format:
Xinhai Yan, Libing Wu, Zhuangzhuang Zhang, Bingyi Liu, Lijuan Huo, and Jing Wang. 2025. FedBAP: Backdoor Defense via Benign Adversarial Perturbation in Federated Learning. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM '25), October 27–31, 2025, Dublin, Ireland.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3746027.3754814

1 INTRODUCTION

As data privacy and security concerns become increasingly prominent, Federated Learning (FL) [20] has emerged as a promising distributed machine learning paradigm, attracting widespread attention. Unlike traditional centralized training, FL enables multiple clients to train models locally and upload only model updates to a central server for aggregation, thereby facilitating collaborative learning across devices or institutions while preserving data privacy [40]. Since FL allows efficient training without exposing raw data, it has been widely adopted in various domains, including smart devices [9], healthcare [27], IoT device [21], and financial risk management [4].

However, this decentralized training paradigm also introduces new security challenges, particularly backdoor attacks [8], which pose a significant threat to FL systems. Specifically, during the FL training process, malicious clients can inject stealthy backdoors into local models through methods such as data poisoning or model replacement. These backdoors can then propagate to the global model through the server's model aggregation. Once the attack succeeds, the global model will output the attacker's designated target class when encountering a specific trigger, while maintaining high accuracy on normal inputs, making the attack extremely difficult to detect and defend against [1, 29, 30]. Furthermore, due to the decentralized nature of FL, the server is unable to directly access or audit the clients' training processes, making backdoor attacks even more elusive and harder to detect [8].

Existing defense mechanisms can be broadly categorized into two approaches [25]: mitigating the impact of infected models [10, 12, 23, 26, 37, 42] and filtering out malicious models or parameters [3, 7, 11, 13, 16, 25, 31, 33, 35, 38, 41]. However, both of these approaches have significant limitations. The former is challenged

by the difficulty of precisely identifying backdoor features, often relying on intrusive interventions that can disrupt the model's ability to learn normal tasks, leading to a decline in main task performance. The latter typically relies on feature statistics or anomaly detection, making it susceptible to failures when dealing with complex or stealthy backdoor strategies, potentially mislabeling legitimate updates, and impacting system stability. Furthermore, due to the non-independent and identically distributed (non-IID) nature of client data in FL, these defense strategies struggle to accommodate the diverse backdoor attack patterns encountered by different clients. Therefore, existing backdoor defense methods still face the following challenges:

Challenge 1 (Identification of Backdoor Attack Features). The stealthy characteristics of backdoor features fundamentally limit current defenses' capability in both accurate feature identification and subsequent complete backdoor removal.

Challenge 2 (Balancing Backdoor Attack Defense and Main Task Accuracy). Current backdoor defense methods require excessive interference with the model aggregation process for backdoor removal, consequently degrading the main task's accuracy.

To address the above challenge, we focus on the nature of backdoor attacks, namely that backdoor attacks cause the model to overly rely on triggers rather than the global semantic features. Driven by this insight, we propose FedBAP, a Benign Adversarial Perturbation based defense framework against backdoor attacks in FL. Our goal is to eliminate the model's dependence on backdoor triggers, thereby bypassing the challenging process of backdoor identification and ensuring the model's exclusive focus on globally representative features. To achieve this, we first propose a perturbation trigger generation mechanism that significantly influences the model's backdoor decision boundary by generating perturbation triggers that are very similar in position, and size to the backdoor triggers. Second, we design benign adversarial perturbations generation mechanism that utilize perturbation triggers for adversarial training on the client, to reduce the model's dependence on backdoor triggers and encourage it to learn more robust data features, thereby addressing **Challenge 1**. Finally, we propose an adaptive scaling mechanism that achieves a balance between defense strength and model performance by dynamically adjusting the intensity of benign adversarial perturbations, thus addressing **Challenge 2**. Experimental results show that FedBAP effectively mitigates backdoor attacks while preserving model accuracy, demonstrating its effectiveness as a defense strategy.

Our main contributions are as follows:

- We propose FedBAP, a Benign Adversarial Perturbation based defense framework against backdoor attacks in FL. FedBAP effectively mitigates backdoor threats while preserving the model's main task performance.
- We propose a novel perspective that introduces a fundamentally new defense approach rooted in the model's training behavior itself, focusing on the essential dependency nature of models on backdoor features, thereby effectively eliminating backdoors.
- We conduct extensive experiments under various federated backdoor attack scenarios, demonstrating that FedBAP significantly outperforms existing defense methods and, in some cases, even enhances the model's accuracy on the main task.

2 RELETED WORK

Extensive research [1, 6, 14, 15, 17–19, 22, 28, 34, 39, 43] has demonstrated that FL is highly susceptible to backdoor attacks. To mitigate these threats, researchers have proposed various defense mechanisms. Existing work on defending against targeted attacks in FL can be broadly categorized into two main approaches [25]: mitigating the impact of infected models and filtering out infected models or parameters.

Mitigating the Impact of Infected Models. Several methods aim to reduce the influence of compromised models without explicitly removing them. FLIP [37] produces an adversarial training-based approach to generate reverse-triggered augmented data, mitigating backdoor threats by exposing the model to counterexamples. LeadFL [42] enhances model resilience through Hessian-based regularization, improving gradient stability and reducing the impact of malicious updates. FedGame [12] formulates a game-theoretic defense that models the interaction between defenders and attackers, optimizing global model updates by reverse-engineering backdoor triggers and target classes. Lockdown [10] isolates malicious parameters using subspace training, which includes initialization, search, pruning, and aggregation to defend against backdoor attacks while reducing computational complexity. CrowdGuard [26] further improves backdoor resilience by leveraging client feedback, analyzing hidden-layer neuron behavior, and performing iterative pruning to remove poisoned models. FLAME [23] employs dynamic model clustering, adaptive weight clipping, and differential privacy noise injection to detect and suppress adversarial updates.

Filtering Out Infected Models or Parameters. Another line of work focuses on identifying and excluding compromised updates through robust aggregation and anomaly detection. FLTrust [3] adopts a trust-based framework by leveraging a small clean dataset to assign trust scores to client updates, ensuring that malicious contributions are effectively downweighted during aggregation. FPD [31] produces a four-module defense mechanism incorporating reliable client selection, similarity-based anomaly detection, and update denoising to filter out both colluding and non-colluding adversarial clients. AGRAMPLIFIER [7] strengthens Byzantine-robust aggregation rules by employing local update amplification techniques, improving robustness against adversarial updates. FLDetector [38] utilizes consistency detection, Euclidean distance-based suspicious scoring, and k-means clustering to isolate malicious clients. FLARE [33] analyzes penultimate layer representations to assess update credibility and mitigate poisoning effects. FLShield [13] addresses verification challenges in FL through representative model generation and class-wise loss impact measurement, ensuring the integrity of updates before aggregation. BackdoorIndicator [16] injects OOD-based indicator tasks into the global model to proactively detect any backdoor-poisoned local model uploads.

These approaches have proven effective in various settings. However, they overlook the fundamental characteristic of backdoor attacks, where the model develops a strong dependence on backdoor triggers. As the intensity of backdoor attacks increases, the defensive performance of these methods will significantly decrease.

3 PRELIMINARIES

3.1 Federated Learning

We consider a standard FL setup where a set of clients S aim to collaboratively train a global model w with the coordination of a server. Let $\mathcal{D}_i$ be the private training dataset held by the client i, where $i \in S$. In the t-th communication round, the server first randomly selects a set of clients S_t, where $|S_t| \leq |S|$. The server then distributes the current version of the global model w_t to the selected clients. Each selected client $i \in S_t$ first uses the global model to initialize its local model, then trains its local model on its local training dataset, and finally uploads the local model update to the server. We use g_t^i to denote the local model update of the client i in the t-th communication round. The server aggregates the received updates on model weights and updates the current global model weights as follows:

$$w_{t+1} = w_t + \mathcal{A}(\{g_t^i | i \in S_t\}) \tag{1}$$

where $\mathcal{A}$ is an aggregation rule adopted by the server. For instance, a widely used aggregation rule FedAvg [20] takes an average over the local model updates uploaded by clients.

3.2 Threat Model

Attacker's goals. As the existing studies on FL backdoor attacks, an attacker's goal is to enforce models to classify data samples with triggers embedded to specific incorrect labels while keeping a high accuracy for samples without triggers embedded.

Attacker's capabilities. We assume that the server is honest. Following threat models in previous studies [1, 6, 18, 34, 36, 39], we consider an attacker that can compromise a certain number of clients. Specifically, the attacker can access to the training datasets and global model updates of these compromised clients, allowing manipulation of their uploaded updates. Furthermore, malicious clients under the attacker's control can communicate and synchronize their attack strategies. The attacker also has access to a snapshot of the global model in each round and can directly modify both model weights and datasets on compromised clients.

4 METHOD

4.1 High-level Description

As shown in Figure 1, FedBAP consists of three key mechanisms: Perturbation Triggers Generation, Benign Adversarial Perturbation Generation, and Adaptive Scaling. The detailed workflow of FedBAP is described in Algorithm 1, and the complete process is outlined as follows:

- **Step 1: Initialization.** Before the start of FL, the server initializes the global model and the scaling factor, which are then distributed to all clients. The scaling factor is a dynamic parameter produced to adjust the strength of benign adversarial perturbations after the defense is activated.
- **Step 2: Perturbation Trigger Generation.** At the designated start round, each client executes the *MaskGen* and *PatternGen* algorithms based on the preliminary converged global model to generate perturbation triggers. These triggers are later used to guide the generation of benign adversarial perturbation.

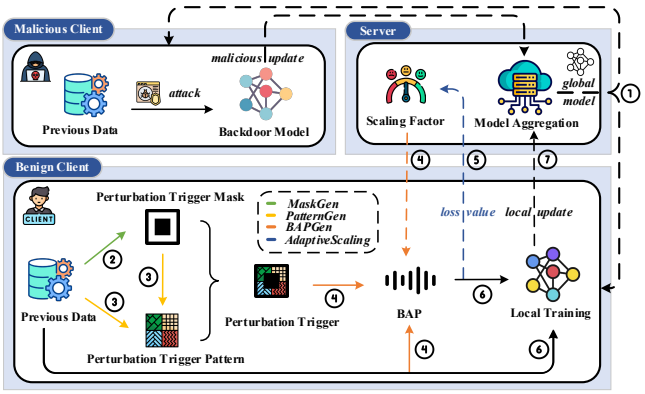


Figure 1: Overview of FedBAP. At the defense start round, the server distributes the global model to clients (①). Clients generate perturbation triggers (② and ③), apply BAPGen to create perturbations and upload loss values (④). The server updates the scaling factor (⑤). Clients then train locally and send model updates to the server (⑥ and ⑦).

- **Step 3: Benign Adversarial Perturbation Generation.** After the defense starts, clients use the perturbation triggers and the current scaling factor to execute the *BAPGen* algorithm, generating benign adversarial perturbations. And benign adversarial perturbation loss values are also uploaded once defense is activated.
- **Step 4: Local Training.** Each client trains a local model using its private data and subsequently uploads the corresponding model update.
- **Step 5: Adaptive Scaling.** After the defense is triggered, the server adjusts the scaling factor by executing the *AdaptiveScaling* algorithm based on the received Benign adversarial perturbation loss values and the current scaling factor.
- **Step 6: Model Aggregation and Distribution.** The server aggregates local updates into a new global model and distributes it to clients.

4.2 Perturbation Trigger Generation

The main objective of the perturbation trigger generation mechanism is to create a trigger that matches the size and position of the backdoor trigger, but has a greater impact on the model predictions. Unlike malicious triggers, perturbation triggers are designed to steer gradient updates away from backdoor-reliant behaviors and toward more robust feature representations. According to DEFINITION 1, each perturbation trigger consists of a mask that defines its spatial region and a pattern that specifies the perturbation applied. These components are carefully derived to maintain consistency with attacker-defined triggers while maximizing their ability to disrupt shortcut learning. The following section details the construction methodology.

DEFINITION 1. *The perturbation trigger consists of two components [32]:*

- *M, which represents the trigger mask. It is a two-dimensional matrix with the same height and width as the original image,*

Algorithm 1 Overview of FedBAP

FedBAP:

Parameter: Local epochs E_1, learning rate η_1, start round t_s, scaling factor c_{t_s}, number of clients m, number of communication rounds N.

Output: Model w_N

init w_0
for t = 1 to N **do**
  $S_t \leftarrow$ a random set of m clients
  // Client Side;
  if $t = t_s$ **then**
    $M \leftarrow MaskGen(w_t)$
    $\Delta x \leftarrow PatternGen(w_t, M)$
  end if
  for $i \in S_t$ **in parallel do**
    $w' \leftarrow w_t$
    $loss_t^i \leftarrow 0$
    if $t \geq t_s$ **then**
      $loss_t^i, w' \leftarrow BAPGen(c_t, w', i, M, \Delta x_i)$
    end if
    for e = 1 to E_1 **do**
      for $(x, y) \in \mathcal{D}_i$ **do**
        $w' \leftarrow w' - \eta_1 \nabla_{w'} \ell((w'(x), y)$
      end for
    end for
    $g_t^i \leftarrow w' - w_t$
  end for
  // Server Side;
  if $t \geq t_s$ **then**
    $loss_t \leftarrow \frac{1}{|S_t|} \sum_{i \in S_t} loss_t^i$
    $c_{t+1} \leftarrow AdaptiveScaling(loss_t, c_t, t)$
  end if
  $w_{t+1} \leftarrow w_t + \frac{1}{|S_t|} \sum_{i \in S_t} g_t^i$
end for
return w_N

where the same mask value is applied across all color channels. The values of M range from 0 to 1, determining the proportion of the original image that the trigger can cover.

- *Δx, which represents the trigger pattern. It is a three-dimensional matrix with the same height, width, and number of channels as the original image.*

Let $A(\cdot)$ denote the process of embedding a trigger into the original image x. The embedding operation $A(x, M, \Delta x)$ is defined as:

$$A(x, M, \Delta x) = (1 - M) \cdot x + M \cdot \Delta x \tag{2}$$

where the mask M selectively mixes the original image x with the perturbation trigger pattern Δx.

Perturbation Trigger Mask Generation. The first step is to obtain the mask of the perturbation trigger. Ideally, this mask should closely resemble that of the backdoor trigger, ensuring that it accurately interferes with the model's decision path during adversarial training without straying from the critical region of the backdoor feature. This alignment guarantees that the perturbation effectively aids the model in correcting its over-reliance on the backdoor trigger. According to DEFINITION 2, in a backdoored model, the backdoor trigger has a significantly smaller backdoor distance compared to non-backdoor triggers. Thus, our goal is to identify a trigger that successfully activates the backdoor while minimizing the backdoor distance. Formally, our optimization objective is defined as:

$$\arg\min_{M,\, \Delta x} \ell\left(w(A(x, M, \Delta x)), y_t\right) + \lambda \cdot \|M\| \tag{3}$$

where $\ell(\cdot)$ denotes the cross-entropy loss function, $w(\cdot)$ represents the neural network, x is the input image, M is the trigger mask, Δx is the trigger pattern, y_t is the target class, λ is a regularization coefficient controlling the trade-off between classification loss and the sparsity of the mask M.

DEFINITION 2. *Given an image from the source class i, a trigger composed of a mask M and a pattern Δx can flip the image's label to the target class j. The class-wise distance $d_{i \to j}$ is measured as $\|M_{i \to j}\|$, where $\|\cdot\|$ denotes the L_1 norm, representing the number of pixels that need to be modified to flip the label from class i to class j. This metric quantifies the difficulty of transitioning from class i to class j [32]. Based on this, we introduce the concept of backdoor distance. Given an image from an arbitrary source class, if a backdoor exists, the backdoor distance $d_{\forall \to t}$ is defined as $\|M_t\|$, where class t is the target class. Since the existence of the backdoor makes it easier to flip samples from any class to the target class, we have the inequality:*

$$\|M_t\| \ll \|M_{i \to j}\|, \text{ s.t. } j \neq t \tag{4}$$

indicating that flipping a sample to the backdoor target class is significantly easier than flipping it to any other classes.

Each client iterates over all possible target classes and optimizes the mask M according to Equation 3. At the end of each epoch, the client evaluates the current trigger mask. If the backdoor accuracy of the current trigger is not lower than the predefined threshold, and the backdoor distance of the current mask is smaller than the client's best recorded mask, then the client's best mask is updated to the current mask. Once all clients have obtained their optimal masks, the server aggregates the masks by computing the average mask across all clients. The server then binarizes the aggregated mask by setting values greater than or equal to 0.5 to 1, and values less than 0.5 to 0. The detailed procedure of the Perturbation Trigger Mask Generation Algorithm *MaskGen* is presented in the appendix.

Perturbation Trigger Pattern Generation. After obtaining the mask, the next step is to determine the perturbation trigger pattern. We aim to maximize the perturbation trigger's impact on the model's output by increasing the difference in the penultimate layer representations (PLR). According to DEFINITION 3, differences in the PLR space translate into corresponding differences in output probabilities. Therefore, when the distance between two PLRs is large, the corresponding output probability distributions exhibit significant divergence. If the perturbation trigger maximally perturbs the PLR, it can induce a substantial change in the model's output.

DEFINITION 3. *Let the neural network be defined as $f : \mathbb{R}^{d_1 \times d_2 \times d_3} \to \mathbb{R}^c$, mapping an input $x \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ to a c-dimensional probability vector $q \in \mathbb{R}^c$, which c is the number of classes. We denote the mapping from the input to the penultimate layer representations*

(PLR) as $g : \mathbb{R}^{d_1 \times d_2 \times d_3} \rightarrow \mathbb{R}^o$. *The output of* g *is the PLR, denoted as* $r \in \mathbb{R}^o$ *[33]. Finally, we define* $\sigma : r \in \mathbb{R}^o \rightarrow q \in \mathbb{R}^c$ *as the mapping function from the PLR to the output probability vector. We use* $\Omega = [\omega_1, \omega_2, \cdots, \omega_c]$ *to represent the weight connecting the penultimate layer to the last layer where* $\omega_k \in \mathbb{R}^o$ *denotes the weights connecting to the k-th neuron of the output layer. We have*

$$\left\|q^1 - q^2\right\|_2 \leq \|\Omega\|_2 \|r_1 - r_2\|_2 \tag{5}$$

where r_1 *and* r_2 *are the PLR of two input* x_1 *and* x_2 *respectively,* q^1 *and* q^2 *are the output probability vector for input* x_1 *and* x_2 *respectively,* $\left\|q^1 - q^2\right\|_2$ *denotes the Euclidean distance between* q^1 *and* q^2, $\|r_1 - r_2\|_2$ *denotes the Euclidean distance between* r_1 *and* r_2.

In FL, client data is typically non-IID, causing backdoor behaviors to vary across clients. As a result, each client must independently generate a customized perturbation trigger using local data. To optimize the trigger's pattern, we leverage the PLR, which reflects the model's output before the final prediction. Let r_1 and r_2 be the PLRs before and after applying the trigger. We minimize their cosine similarity to maximize the trigger's impact, encouraging it to induce a strong shift in the model's output. This drives the model to react to the trigger, facilitating the suppression of backdoor reliance. Once optimized, the pattern is stored for generating benign adversarial perturbations. The full procedure is detailed in the *PatternGen* algorithm in the appendix.

4.3 Benign Adversarial Perturbation Generation

To effectively defend against backdoor attacks, we are dedicated to breaking the model's reliance on backdoor triggers and encouraging it to learn the global features of the data. To achieve this goal, we design a benign adversarial perturbation generation mechanism. This mechanism leverages perturbation triggers generated by our trigger generation module to conduct adversarial training on clients, thereby reducing the model's reliance on backdoor triggers and encouraging the learning of more robust data features.

Benign adversarial perturbations differ fundamentally from traditional adversarial perturbations. While the latter are crafted to degrade model performance, the former aim to refine the model's decision boundary and enhance robustness against backdoor attacks. This is achieved by steering the model away from spurious, backdoor-associated features and promoting the learning of more robust and generalizable representations. By persistently applying pressure against backdoor dependencies, benign perturbations reduce the model's reliance on shortcut-based decision rules and strengthen its overall resilience.

To generate benign adversarial perturbation in practice, we introduce a client-side training algorithm that incorporates perturbation triggers into the local data. Each client embeds the predefined perturbation trigger into a subset of clean samples and enforces correct label prediction through adversarial training. In this process, the perturbation acts as a targeted optimization signal that reshapes the model's internal representations. To further amplify its effectiveness, we apply a scaling factor to control the perturbation's magnitude. This factor ensures that the perturbation maintains sufficient influence during training, particularly when the model begins adapting and the adversarial loss starts to decrease. Finally, the algorithm returns both the hardened model and the loss value of the benign adversarial perturbation. The detailed procedure of our Benign Adversarial Perturbation Generation *BAPGen* algorithm is presented in the appendix.

4.4 Adaptive Scaling

As training progresses, the impact of benign adversarial perturbation tends to weaken. This is because benign adversarial perturbation is essentially a gradient update generated through adversarial training, and as the model converges, the corresponding adversarial loss value decreases. However, this decline can be problematic, as attackers may continuously reinforce the model's reliance on backdoor triggers. It is therefore essential to regulate the perturbation strength within an effective range to prevent the model from reverting to decision rules based on backdoor trigger dependencies, while preserving robust generalization.

To address this, we propose adaptive scaling, a mechanism that dynamically adjusts the strength of benign adversarial perturbation based on its real-time influence on the model. Leveraging the adversarial nature of benign adversarial perturbation, we use its training loss value as a proxy for effectiveness: a high loss indicates strong gradient influence, while a decreasing loss suggests that the model is adapting and the defense is losing effectiveness. Adaptive scaling counteracts this by adjusting a scaling factor that amplifies the perturbation magnitude whenever its effect weakens. This mechanism maintains benign adversarial perturbation within a desirable range, ensuring continued suppression of backdoor dependencies throughout training without compromising performance on clean inputs.

Specifically, the scaling factor is governed by three key components. The first is the scaling step size δ, which is a hyperparameter that controls the magnitude of adjustment. The second is the loss ratio α_t, measuring the relative change between the current and previous loss values to capture short-term fluctuations. The third is the smoothing factor β, designed to reduce sensitivity to transient variations and reflect the overall trend of the loss. To compute β, we use a sliding window that averages recent loss ratios, enabling the mechanism to detect consistent upward or downward trends and stabilize the adjustment of the scaling factor. The complete adaptive scaling procedure *AdaptiveScaling* is detailed in the appendix.

5 EXPERIMENTS

5.1 Experimental Setup

FL Settings. We consider two widely-used benchmark datasets CIFAR-10 and CIFAR-100 to evaluate FedBAP. We use the method in [5] to distribute the training images to the clients. For CIFAR-10, we employ the commonly used ResNet-18 and VGG-19 architectures as the global model, while for CIFAR-100, we adopt ResNet-18 as the global model. By default, our federated training setup consists of 100 clients, 10% of which are malicious. In each training round, 10% of clients are randomly selected to perform local model training. We assume a non-IID data distribution with a concentration parameter h of 0.9, following previous works [6, 18, 39]. The FL training process consists of 200 communication rounds, with each selected client training the local model for 2 epochs. The start round for FedBAP is set to 100, and unless otherwise specified, the scaling step size δ is set to 1. More setup details can be found in the appendix.

Table 1: Performance of different defenses on non-IID datasets.

		ResNet-18 (CIFAR-10)			VGG-19 (CIFAR-10)			ResNet-18 (CIFAR-100)		
		BadNets	LP	A3FL	BadNets	LP	A3FL	BadNets	LP	A3FL
FedAvg	BBSR	99.37%	93.98%	100.00%	75.66%	95.91%	100.00%	99.53%	83.27%	100.00%
	ABSR	97.88%	91.29%	100.00%	60.16%	85.06%	100.00%	98.50%	76.57%	100.00%
	ACC	85.77%	75.03%	86.18%	78.26%	73.10%	78.95%	59.39%	43.05%	58.36%
Krum	BBSR	11.06%	97.51%	49.57%	100.00%	29.01%	99.31%	4.39%	97.34%	5.40%
	ABSR	**1.27%**	93.96%	10.42%	49.19%	**2.28%**	31.88%	1.24%	64.39%	1.31%
	ACC	44.00%	64.37%	45.10%	33.38%	47.76%	36.10%	24.03%	31.16%	26.64%
MultiKrum	BBSR	84.23%	87.11%	100.00%	59.97%	95.97%	100.00%	9.30%	96.84%	100.00%
	ABSR	56.08%	20.53%	99.71%	22.18%	61.82%	59.74%	4.23%	91.36%	99.94%
	ACC	78.84%	71.28%	79.35%	70.06%	67.47%	70.62%	51.32%	38.51%	51.03%
FLTrust	BBSR	96.24%	80.44%	100.00%	12.86%	14.72%	100.00%	48.01%	91.41%	100.00%
	ABSR	88.57%	29.26%	100.00%	8.66%	6.00%	100.00%	64.38%	42.52%	100.00%
	ACC	77.89%	67.70%	78.63%	63.15%	67.66%	67.52%	48.01%	34.99%	48.42%
RLR	BBSR	96.62%	9.56%	100.00%	62.57%	16.11%	100.00%	97.75%	35.70%	100.00%
	ABSR	70.32%	3.63%	97.64%	38.31%	6.75%	100.00%	93.01%	0.94%	100.00%
	ACC	79.55%	65.58%	78.86%	62.57%	63.21%	63.52%	46.50%	35.70%	47.21%
FLAME	BBSR	98.77%	97.62%	100.00%	91.08%	98.09%	100.00%	49.07%	91.85%	**2.93%**
	ABSR	93.39%	95.61%	100.00%	79.17%	71.49%	100.00%	**0.56%**	86.69%	**1.26%**
	ACC	78.00%	68.77%	79.46%	64.68%	66.29%	69.68%	49.07%	36.68%	48.19%
FLIP	BBSR	15.83%	36.89%	100.00%	9.97%	11.14%	100.00%	82.45%	20.64%	100.00%
	ABSR	4.37%	8.60%	100.00%	2.48%	3.32%	100.00%	5.92%	4.77%	100.00%
	ACC	85.95%	74.67%	85.75%	79.17%	73.87%	78.38%	57.41%	43.56%	58.23%
FedBAP (Ours)	BBSR	**2.51%**	**4.22%**	**5.89%**	**5.78%**	**5.50%**	**12.13%**	**1.11%**	**0.96%**	4.48%
	ABSR	1.55%	**2.26%**	**2.40%**	**2.26%**	2.84%	**2.78%**	0.58%	**0.38%**	2.55%
	ACC	**89.98%**	**78.64%**	**89.29%**	**87.24%**	**78.40%**	**86.32%**	**63.12%**	**47.22%**	**63.05%**

Baselines. We consider three types of state-of-the-art (SOTA) targeted attacks: BadNets [8], LP [43], and A3FL [36]. The performance of FedBAP is compared with six SOTA defenses: Krum [2], MultiKrum, FLTrust [3], RLR [24], FLAME [23], and FLIP [37] . Detailed descriptions of these comparison schemes can be found in the appendix.

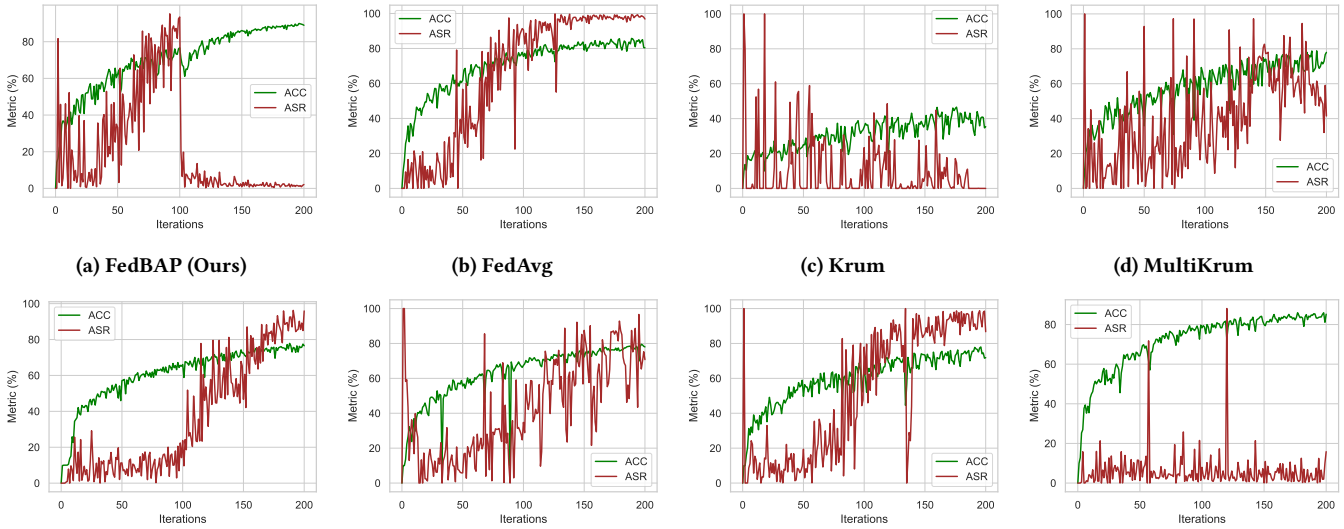
Evaluation Metrics. We consider Average Backdoor Success Rate (ABSR), Best Backdoor Success Rate (BBSR), and Main Task Accuracy (ACC) as evaluation metrics to assess the effectiveness of FedBAP. ABSR represents the average proportion of backdoor samples misclassified as the attack target label over the last 20 rounds. It reflects the effectiveness of the defense. BBSR denotes the highest proportion of backdoor samples misclassified as the attack target label within the last 20 rounds. It reflects the stability of the defense. ACC measures the accuracy on benign samples, representing the proportion of correctly classified benign inputs. It reflects the usability of the defense.

5.2 Overall Performance

Table 1 presents the performance of different defenses against three backdoor attacks across various model architectures and datasets in non-IID settings, while the results for IID settings are provided in the appendix. Figure 2 illustrates the backdoor success rates (BSR) and ACC curves of different defenses against the BadNets attack on ResNet-18 with CIFAR-10, while the results for the LP and A3FL attacks are shown in the appendix.

As shown in Table 1, FedBAP consistently achieves the lowest ABSR across most attack scenarios, highlighting its strong effectiveness in mitigating diverse backdoor threats. In terms of stability, FedBAP maintains the lowest BBSR across most cases, with minimal fluctuations in BSR over time. The BSR curves in Figure 2 show that FedBAP remains stable and does not exhibit sudden spikes, which is critical in ensuring reliable defense performance. Regarding usability, FedBAP consistently achieves the highest ACC across all attack scenarios, outperforming other baselines. This can be attributed to the nature of benign adversarial perturbations, which serve as a form of robust training that improves generalization and enhances the model's resilience to abnormal or noisy data.

The results consistently demonstrate that FedBAP outperforms existing defenses by effectively mitigating backdoor attacks while maintaining high main task accuracy. Its ability to provide stable and robust defense in both IID and non-IID settings highlights its practical applicability in FL scenarios. These findings confirm that FedBAP is a highly effective and generalizable defense against various types of backdoor attacks.



(a) FedBAP (Ours) (b) FedAvg (c) Krum (d) MultiKrum

(e) FLTrust (f) RLR (g) FLAME (h) FLIP

Figure 2: ResNet-18 trained with different defenses on the non-IID CIFAR10 dataset against the BadNets attack.

Table 2: Ablation study.

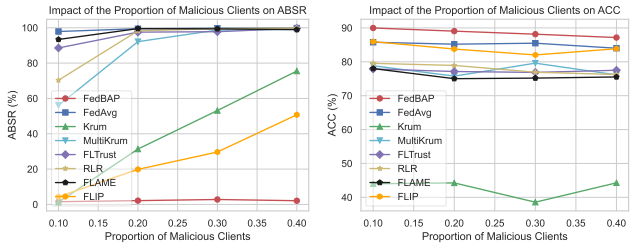
	w/o BAP	w/o AS & PG	w/o PG	w/o AS	FedBAP
BBSR	99.37	99.17	5.32	98.97	**2.51**
ABSR	97.88	97.92	2.25	97.28	**1.55**
ACC	85.77	88.61	89.15	89.5	**89.98**

5.3 Ablation Study

We conduct ablation studies on the CIFAR-10 dataset under the BadNets attack to evaluate the contribution of each component of FedBAP, including Benign Adversarial Perturbation (BAP), Adaptive Scaling (AS), and Perturbation Pattern Generation (PG). Notably, in the w/o PG setting, the learned perturbation pattern is replaced with a randomly initialized one instead of executing PatternGen. As shown in Table 2, removing any of these components results in a significant decline in defense performance, indicating that each module plays an essential role in the effectiveness of the overall framework.

5.4 Impact of the Proportion of Malicious Clients

We evaluate the impact of the proportion of malicious clients using the CIFAR-10 dataset. Specifically, we vary the proportion of malicious clients from 0.1 to 0.4 under three representative attacks: BadNets, LP, and A3FL. It is worth noting that we increase the δ of FedBAP to 1.5 under LP and to 3 under A3FL, as these are more sophisticated attack strategies. Figures 3, 4, and 5 respectively illustrate the ABSR and ACC trends of various defenses under the BadNets, LP, and A3FL attacks. The results show that increasing the proportion of malicious clients leads to a significant drop in the performance of most defense methods, except FedBAP. FedBAP



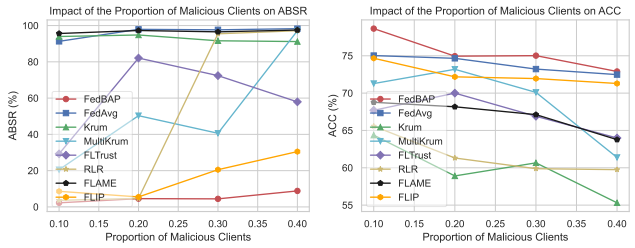
(a) ABSR (b) ACC

Figure 3: Impact of the malicious clients proportion on different defenses under the BadNets attack.

consistently achieves the lowest ABSR across all settings, staying below 6% under LP and A3FL, and under 3% for BadNets. It also maintains higher ACC than other defenses, demonstrating strong robustness across different attack scenarios. This is because FedBAP encourages the model to learn robust features rather than overfitting to malicious patterns, thus maintaining stability even under stronger attack intensities.

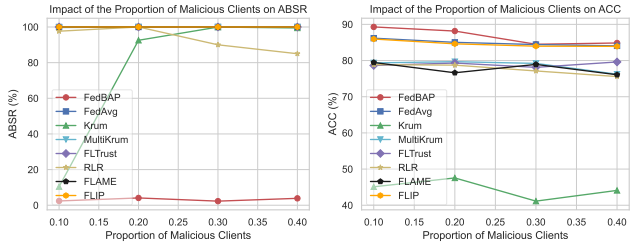
5.5 Impact of the non-IIDness

To investigate the effect of data heterogeneity, we evaluate all methods under both IID and non-IID settings, where $h = 1.0$ corresponds to the IID setting and smaller h values (e.g., $h = 0.9$ and $h = 0.5$) indicate increasing levels of non-IID distribution. We conduct experiments on the CIFAR-10 dataset under the BadNets attack. As shown in Figure 6, most baseline methods suffer significant degradation in ABSR and ACC as the degree of non-IIDness increases. In contrast, FedBAP consistently maintains low attack success rates and exhibits only a minor drop in ACC under non-IID settings. This demonstrates that FedBAP is not only robust against backdoor



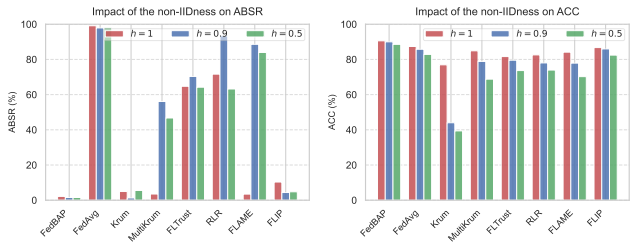
(a) ABSR (b) ACC

Figure 4: Impact of the malicious clients proportion on different defenses under LP attack.



(a) ABSR (b) ACC

Figure 5: Impact of the malicious clients proportion on different defenses under A3FL attack.



(a) ABSR (b) ACC

Figure 6: Impact of the non-IIDness on different defenses under BadNets attack.

attacks, but also resilient to performance degradation caused by data heterogeneity.

5.6 Impact of the Scaling Step Size

To evaluate the impact of the scaling step size δ on the effectiveness of our defense, we compare the ACC and BSR across different values of δ using the CIFAR-10 dataset under A3FL attacks, where the proportion of malicious clients is set to 40%. As shown in Figure 7, we present results starting from round 110, as the defense mechanism is activated at round 100. Experimental results show that increasing δ effectively enhances backdoor suppression but may also degrade model performance. When $\delta < 3$, the backdoor effect tends to re-emerge during training. In contrast, BSR drops below 5% and remains suppressed when $\delta = 3$. However, further increasing δ to 4

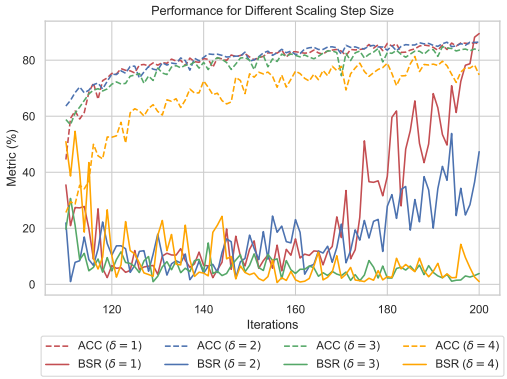


Figure 7: Impact of the scaling step size.

leads to a noticeable decline in ACC, indicating potential instability. These results suggest that a small benign adversarial perturbation may be insufficient to eliminate the backdoor, while an overly large one can harm the model's performance. Therefore, choosing an appropriate δ is crucial for balancing robustness and utility. Among the evaluated settings, $\delta = 3$ strikes the best trade-off, offering effective backdoor suppression while maintaining stable accuracy on the main task.

6 CONCLUSION

In this paper, we propose FedBAP, a robust FL defense mechanism against backdoor attacks. In FedBAP, we propose a novel defense approach that stems from the behavior of model training itself, focusing on the inherent reliance of the model on backdoor features. FedBAP utilizes perturbation trigger to generate benign adversarial perturbations and employs adaptive scaling to control the perturbation magnitude, thereby mitigating the influence of malicious model updates while preserving overall model accuracy. This method of weakening reliance from the source, guiding training locally, and stabilizing the system through dynamic adjustment ensures both enhanced defense effectiveness and the preservation of model accuracy. Our extensive evaluation on two benchmark datasets, three backdoor attack strategies, and six defense methods demonstrates that FedBAP effectively suppresses backdoor effects and achieves SOTA performance, consistently outperforming existing defense approaches in multiple scenarios. In future work, we plan to extend our method to asynchronous FL and personalized FL to address backdoor attacks faced in these FL scenarios.

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Program of China under Grant 2022YFB3104502, in part by the National Natural Science Foundation of China under Grant 62441237, Grant U24A20336, Grant 62272348, Grant 62202197, the Key Research and Development Program of Wuhan under Grant 2024050702030090, Wuhan Science and Technology Joint Project for Building a Strong Transportation Country under Grant 2023-2-7, and Wuhan City Joint Innovation Laboratory for Next-Generation Wireless Communication Industry Featuring Satellite-Terrestrial Integration under Grant 4050902040448.

REFERENCES

[1] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2938–2948.

[2] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems* 30 (2017).

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

[4] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. 2021. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems* 36, 6 (2021), 87–98.

[5] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local model poisoning attacks to Byzantine-Robust federated learning. In *29th USENIX Security Symposium (USENIX Security 20)*. 1605–1622.

[6] Pei Fang and Jinghui Chen. 2023. On the vulnerability of backdoor defenses for federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 11800–11808.

[7] Zirui Gong, Liyue Shen, Yanjun Zhang, Leo Yu Zhang, Jingwei Wang, Guangdong Bai, and Yong Xiang. 2023. Agramplifier: defending federated learning against poisoning attacks through local update amplification. *IEEE Transactions on Information Forensics and Security* 19 (2023), 1241–1250.

[8] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* (2017).

[9] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[10] Tiansheng Huang, Sihao Hu, Ka-Ho Chow, Fatih Ilhan, Selim Tekin, and Ling Liu. 2023. Lockdown: backdoor defense for federated learning with isolated subspace training. *Advances in Neural Information Processing Systems* 36 (2023), 10876–10896.

[11] Wenke Huang, Mang Ye, Zekun Shi, Guancheng Wan, He Li, and Bo Du. 2024. Parameter disparities dissection for backdoor defense in heterogeneous federated learning. *Advances in Neural Information Processing Systems* 37 (2024), 120951–120973.

[12] Jinyuan Jia, Zhuowen Yuan, Dinuka Sahabandu, Luyao Niu, Arezoo Rajabi, Bhaskar Ramasubramanian, Bo Li, and Radha Poovendran. 2023. Fedgame: A game-theoretic defense against backdoor attacks in federated learning. *Advances in Neural Information Processing Systems* 36 (2023), 53090–53111.

[13] Ehsanul Kabir, Zeyu Song, Md Rafi Ur Rashid, and Shagufta Mehnaz. 2024. Flshield: a validation based federated learning framework to defend against poisoning attacks. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2572–2590.

[14] Haoyang Li, Qingqing Ye, Haibo Hu, Jin Li, Leixia Wang, Chengfang Fang, and Jie Shi. 2023. 3dfed: Adaptive and extensible framework for covert backdoor attack in federated learning. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1893–1907.

[15] Minghui Li, Wei Wan, Yuxuan Ning, Shengshan Hu, Lulu Xue, Leo Yu Zhang, and Yichen Wang. 2024. Darkfed: A data-free backdoor attack in federated learning. *arXiv preprint arXiv:2405.03299* (2024).

[16] Songze Li and Yanbo Dai. 2024. BackdoorIndicator: Leveraging OOD Data for Proactive Backdoor Detection in Federated Learning. In *33rd USENIX Security Symposium (USENIX Security 24)*. 4193–4210.

[17] Tao Liu, Yuhang Zhang, Zhu Feng, Zhiqin Yang, Chen Xu, Dapeng Man, and Wu Yang. 2024. Beyond traditional threats: A persistent backdoor attack on federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 21359–21367.

[18] Xiaoting Lyu, Yufei Han, Wei Wang, Jingkai Liu, Bin Wang, Jiqiang Liu, and Xiangliang Zhang. 2023. Poisoning with cerberus: Stealthy and colluded backdoor attack against federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 9020–9028.

[19] Xiaoting Lyu, Yufei Han, Wei Wang, Jingkai Liu, Yongsheng Zhu, Guangquan Xu, Jiqiang Liu, and Xiangliang Zhang. 2024. Lurking in the shadows: Unveiling stealthy backdoor attacks against personalized federated learning. In *33rd USENIX Security Symposium (USENIX Security 24)*. 4157–4174.

[20] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*. PMLR, 1273–1282.

[21] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, Nadarajah Asokan, and Ahmad-Reza Sadeghi. 2019. DÏoT: A federated self-learning anomaly detection system for IoT. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 756–767.

[22] Thuy Dung Nguyen, Tuan A Nguyen, Anh Tran, Khoa D Doan, and Kok-Seng Wong. 2023. Iba: Towards irreversible backdoor attacks in federated learning. *Advances in Neural Information Processing Systems* 36 (2023), 66364–66376.

[23] Thien Duc Nguyen, Phillip Rieger, Roberta De Viti, Huili Chen, Björn B Brandenburg, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, et al. 2022. FLAME: Taming backdoors in federated learning. In *31st USENIX Security Symposium (USENIX Security 22)*. 1415–1432.

[24] Mustafa Safa Ozdayi, Murat Kantarcioglu, and Yulia R Gel. 2021. Defending against backdoors in federated learning with robust learning rate. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 9268–9276.

[25] Zhen Qin, Feiyi Chen, Chen Zhi, Xueqiang Yan, and Shuiguang Deng. 2024. Resisting backdoor attacks in federated learning via bidirectional elections and individual perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 14677–14685.

[26] Phillip Rieger, Torsten Krauß, Markus Miettinen, Alexandra Dmitrienko, and Ahmad-Reza Sadeghi. 2022. Crowdguard: Federated backdoor detection in federated learning. *arXiv preprint arXiv:2210.07714* (2022).

[27] Micah J Sheller, G Anthony Reina, Brandon Edwards, Jason Martin, and Spyridon Bakas. 2018. Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation. In *International MICCAI Brainlesion Workshop*. Springer, 92–104.

[28] Chenghui Shi, Shouling Ji, Xudong Pan, Xuhong Zhang, Mi Zhang, Min Yang, Jun Zhou, Jianwei Yin, and Ting Wang. 2024. Towards practical backdoor attacks on federated learning systems. *IEEE Transactions on Dependable and Secure Computing* (2024).

[29] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. 2019. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963* (2019).

[30] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security*. Springer, 480–501.

[31] Wei Wan, Shengshan Hu, Minghui Li, Jianrong Lu, Longling Zhang, Leo Yu Zhang, and Hai Jin. 2023. A four-pronged defense against byzantine attacks in federated learning. In *Proceedings of the 31st ACM International Conference on Multimedia*. 7394–7402.

[32] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 707–723.

[33] Ning Wang, Yang Xiao, Yimin Chen, Yang Hu, Wenjing Lou, and Y Thomas Hou. 2022. Flare: defending federated learning against model poisoning attacks via latent space representations. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*. 946–958.

[34] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. 2019. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*.

[35] He Yang, Wei Xi, Yuhao Shen, Canhui Wu, and Jizhong Zhao. 2024. Roseagg: Robust defense against targeted collusion attacks in federated learning. *IEEE Transactions on Information Forensics and Security* 19 (2024), 2951–2966.

[36] Hangfan Zhang, Jinyuan Jia, Jinghui Chen, Lu Lin, and Dinghao Wu. 2023. A3fl: Adversarially adaptive backdoor attacks to federated learning. *Advances in Neural Information Processing Systems* 36 (2023), 61213–61233.

[37] Kaiyuan Zhang, Guanhong Tao, Qiuling Xu, Siyuan Cheng, Shengwei An, Yingqi Liu, Shiwei Feng, Guangyu Shen, Pin-Yu Chen, Shiqing Ma, et al. 2022. Flip: A provable defense framework for backdoor mitigation in federated learning. *arXiv preprint arXiv:2210.12873* (2022).

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

[39] Zhengming Zhang, Ashwinee Panda, Linyue Song, Yaoqing Yang, Michael Mahoney, Prateek Mittal, Ramchandran Kannan, and Joseph Gonzalez. 2022. Neurotoxin: Durable backdoors in federated learning. In *International Conference on Machine Learning*. PMLR, 26429–26446.

[40] Zhuangzhuang Zhang, Libing Wu, Jiong Jin, Enshu Wang, Bingyi Liu, and Qing-Long Han. 2025. Secure Federated Learning for Cloud-Fog Automation: Vulnerabilities, Challenges, Solutions, and Future Directions. *IEEE Transactions on Industrial Informatics* (2025), 1–13.

[41] Zhuangzhuang Zhang, Libing Wu, Chuanguo Ma, Jianxin Li, Jing Wang, Qian Wang, and Shui Yu. 2023. LSFL: A lightweight and secure federated learning scheme for edge computing. *IEEE Transactions on Information Forensics and Security* 18 (2023), 365–379.

[42] Chaoyi Zhu, Stefanie Roos, and Lydia Y Chen. 2023. LeadFL: Client self-defense against model poisoning in federated learning. In *International Conference on Machine Learning*. PMLR, 43158–43180.

[43] Haomin Zhuang, Mingxian Yu, Hao Wang, Yang Hua, Jian Li, and Xu Yuan. 2023. Backdoor federated learning by poisoning backdoor-critical layers. *arXiv preprint arXiv:2308.04466* (2023).

A ALGORITHMS

Algorithm 2 Perturbation Trigger Mask Generation

MaskGen:

Input: Global model w_t.

Parameter: Class numbers N_{class}, epochs E_2, learning rate η_2, backdoor accuracy threshold acc_{TH}, regularization coefficient λ.

Output: Trigger mask M.

$S \leftarrow$ the set of all clients
for $i \in S$ **in parallel do**
  for $y_t = 0$ to N_{class} **do**
    init $M_i^{y_t}$ and $\Delta x_i^{y_t}$ randomly
    for $e = 1$ to E_2 **do**
      for $(x, y) \in \mathcal{D}_i$ **do**
        $loss \leftarrow \ell\left(w_t(A(x, M_i^{y_t}, \Delta x_i^{y_t})), y_t\right) + \lambda \cdot \|M_i^{y_t}\|$
        $M_i^{y_t} \leftarrow M_i^{y_t} - \eta_2 \nabla_{M_i^{y_t}} loss$
        $\Delta x_i^{y_t} \leftarrow \Delta x_i^{y_t} - \eta_2 \nabla_{\Delta x_i^{y_t}} loss$
      end for
      compute backdoor accuracy acc
      if $acc \geq acc_{TH}$ and $\|M_i^{y_t}\| < \|M_i^{best}\|$ **then**
        $M_i^{best} \leftarrow M_i^{y_t}$
      end if
    end for
  end for
end for
for $j = 0$ to row_M **do**
  for $k = 0$ to col_M **do**
    $M[j][k] \leftarrow \mathbb{I}\left\{\frac{1}{|S|}\sum_{i \in S} M_i^{best}[j][k] \geq 0.5\right\}$
  end for
end for
return M

Algorithm 3 Perturbation Trigger Pattern Generation

PatternGen:

Input: Global model w_t, trigger mask M.

Parameter: Epochs E_3, learning rate η_3.

Output: Trigger patterns Δx.

$S \leftarrow$ the set of all clients
for $i \in S$ **in parallel do**
  init Δx_i randomly
  for $e = 1$ to E_3 **do**
    for $(x, y) \in \mathcal{D}_i$ **do**
      $r_1 \leftarrow w_t^{penultimate}(A(x, M, \Delta x_i))$
      $r_2 \leftarrow w_t^{penultimate}(x)$
      $\Delta x_i \leftarrow \Delta x_i - \eta_3 \nabla_{\Delta x_i} \frac{r_1 \cdot r_2}{\|r_1\| \|r_2\|}$
    end for
  end for
end for
return Δx

We present the detailed workflow of four key algorithms in FedBAP, namely *MaskGen*, *PatternGen*, *BAPGen*, and *AdaptiveScaling*.

Algorithm 4 Benign Adversarial Perturbation Generation

BAPGen:

Input: Scaling factor c_t, local model w', client i, trigger mask M, perturbation trigger Δx_i.

Parameter: Benign adversarial perturbation epochs E_4, learning rate η_4.

Output: Benign adversarial perturbation loss value $loss_t^i$, local model w'.

for $e = 1$ to E_4 **do**
  for $(x, y) \in \mathcal{D}_i$ **do**
    $loss \leftarrow c_t \cdot \ell(w'(A(x, M, \Delta x_i), y)$
    $w' \leftarrow w' - \eta_4 \nabla_{w'} loss$
  end for
end for
$loss_t^i \leftarrow$ the average value of $loss$ over E_4 epochs on $\mathcal{D}_i$
return $loss_t^i$, w'

Algorithm 5 Adaptive Scaling

AdaptiveScaling:

Input: Benign adversarial perturbation loss value $loss_t$, scaling factor c_t, round t.

Parameter: Scaling step size δ, start round t_s, window size k.

Output: Updated scaling factor c_{t+1}.

compute $\alpha_t \leftarrow \begin{cases} 1, & \text{if } t = t_s \\ \frac{\text{loss}_t}{\text{loss}_{t-1}}, & \text{if } t > t_s \end{cases}$
compute $\beta \leftarrow \begin{cases} 1, & \text{if } t - t_s = 0 \\ \frac{\sum_{j=ts}^{t-1} \alpha_j}{t-ts}, & \text{if } 0 < t - t_s < k \\ \frac{\sum_{j=t-k}^{t-1} \alpha_j}{k}, & \text{if } t - t_s \geq k \end{cases}$
compute $c_{t+1} \leftarrow \begin{cases} c_t + \frac{\delta\beta}{\sqrt{\alpha_t}}, & \text{if } \alpha_t \geq 1 \\ c_t + \frac{\delta\beta}{\alpha_t^2}, & \text{if } \alpha_t < 1 \end{cases}$
return c_{t+1}

These algorithms constitute the core of our defense mechanism and are essential for enhancing the robustness and generalization of the global model. Their pseudocode is presented in Algorithms 2–5 to facilitate reproducibility and a better understanding of our approach.

B DETAILS OF EXPERIMENT SETUP

All experiments are conducted using the PyTorch framework on an NVIDIA RTX 3080 Ti GPU. The implementations of backdoor attacks, including BadNets [8] and LP [43], as well as defenses such as Krum [2], MultiKrum, FLTrust [3], RLR [24], FLAME [23], and FLIP [37], were directly used from implementations by LP [43]. The A3FL [36] attack is re-implemented for fair comparison, with reference to the original code. The detailed experimental hyperparameter settings are shown in Tables 3 and 4.

Table 3: Hyper-parameters settings in FL.

Experimental parameters	Parameter settings
Number of clients	100
Select clients proportion	0.1
Malicious clients proportion	0.1
Malicious data proportion	0.5
Trigger size	5×5
Global training rounds	200
Local epochs	2
Learning rate	0.01
Batch size	64

Table 4: Hyper-parameters settings in FedBAP.

	Experimental parameters	Parameter settings
E_1	Local epochs	2
η_1	Learning rate	0.01
t_s	Start round	100
E_2	Local epochs for *MaskGen*	100
η_2	Learning rate for *MaskGen*	0.1
acc_{TH}	Backdoor accuracy threshold	0.9
λ	Weight for backdoor distance	0.01
E_3	Local Epochs for *PatternGen*	100
η_3	Learning rate for *PatternGen*	10
E_4	Local epochs for *BAPGen*	10
η_4	Learning rate for *BAPGen*	0.01
δ	Scaling step size	1
k	Window size	5

C BASELINE ATTACKS

- **BadNets** [8] is one of the earliest and most influential backdoor attack methods. It demonstrates that deep neural networks trained in an outsourced or untrusted environment can be embedded with malicious behavior. A BadNet maintains high accuracy on clean data but misclassifies specific attacker-chosen inputs containing a trigger. The attack is stealthy and persistent—even retraining the model may not completely remove the backdoor. This foundational work highlights the vulnerability of DNNs to covert manipulation during training.
- **LP** [43] is a novel backdoor attack strategy by identifying backdoor-critical (BC) layers—a small set of layers primarily responsible for model vulnerability. By targeting only these layers, LP achieves comparable attack effectiveness to full-model attacks while significantly improving stealthiness against existing defenses.
- **A3FL** [36] is an adaptive backdoor attack that dynamically adjusts the trigger to align with the global model training dynamics in FL. Unlike traditional static triggers, A3FL optimizes the trigger to survive scenarios where the global model actively tries to unlearn it. It remains effective even under limited attack budgets and demonstrates high attack success rates against a broad range of state-of-the-art defenses.

D BASELINE DEFENSES

- **FedAvg** [20] is one of the most popular algorithms in FL. It is designed to address the challenges of training machine learning models in a decentralized manner, where data is distributed across multiple devices and cannot be shared due to privacy concerns. The core idea of FedAvg is to perform local model updates on each participating device and then aggregate these updates to form the global model. Specifically, each client computes model updates by training locally on its own data, and after a few local updates, the models are averaged on the central server. This process reduces the need for sharing raw data, as only model updates are exchanged.
- **Krum** [2] is a Byzantine-resilient aggregation algorithm designed to safeguard distributed learning, particularly in the context of Stochastic Gradient Descent. In a distributed system with n workers, Krum can tolerate up to f Byzantine workers. The algorithm works by selecting the model update that is closest to the majority of other updates, effectively identifying and excluding those that deviate significantly, which are assumed to be malicious. This approach ensures that even with the presence of faulty or adversarial updates, the global model can still converge correctly.
- **MultiKrum** is an extension of the Krum algorithm designed to provide greater robustness in distributed learning systems, particularly against Byzantine faults. While Krum aggregates model updates by selecting the one closest to the majority, MultiKrum improves upon this by considering multiple candidates rather than just one. Specifically, MultiKrum selects the k closest updates, where k is a predetermined number based on the level of fault tolerance required. By doing so, it ensures that even if more than one Byzantine worker is present, the global model can still be updated reliably. This multi-candidate approach increases resilience, offering better protection against adversarial behavior compared to the single-update selection in Krum.
- **FLTrust** [3] is a Byzantine-robust FL method that bootstraps trust from the server side by leveraging a small, clean root dataset. Unlike other defenses that solely rely on statistical filtering of client updates, FLTrust maintains a server model trained on trusted data and uses it to evaluate the alignment of client updates. It assigns trust scores based on the directional similarity between client updates and the server model update, normalizes all updates to limit their impact, and then performs trust-weighted aggregation. This approach significantly improves robustness against strong adaptive attacks, even when a large fraction of clients are malicious.
- **RLR** [24] is a defense mechanism designed to mitigate backdoor attacks in FL. Instead of filtering or removing suspicious client updates, RLR takes a different approach by dynamically adjusting the server-side learning rate for each dimension of the model update. This adjustment is based on the degree of agreement among clients on the sign of the update in each dimension. The core intuition is that dimensions with high disagreement are more likely to be manipulated by malicious clients, and thus should be updated more conservatively. By

Table 5: Performance of different defenses on IID datasets.

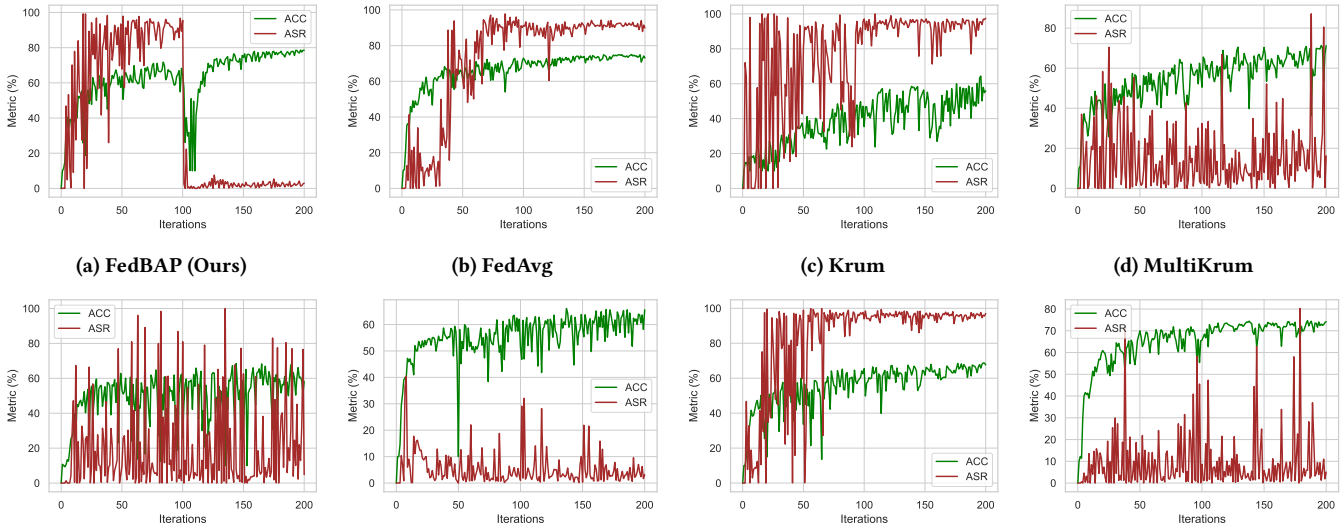
		ResNet-18 (CIFAR-10)			VGG-19 (CIFAR-10)			ResNet-18 (CIFAR-100)		
		BadNets	LP	A3FL	BadNets	LP	A3FL	BadNets	LP	A3FL
FedAvg	BBSR	99.71%	95.64%	100.00%	91.87%	95.43%	100.00%	99.60%	90.76%	100.00%
	ABSR	99.11%	95.54%	100.00%	83.86%	94.03%	100.00%	98.20%	88.95%	100.00%
	ACC	87.34%	74.42%	87.41%	82.34%	75.69%	82.33%	60.17%	43.08%	60.04%
Krum	BBSR	13.36%	98.83%	13.40%	10.20%	99.57%	7.94%	3.10%	96.17%	**1.69%**
	ABSR	4.94%	50.86%	5.71%	4.37%	27.46%	3.20%	0.65%	89.96%	**0.63%**
	ACC	76.92%	68.77%	77.21%	70.36%	65.46%	69.62%	37.53%	29.43%	34.56%
MultiKrum	BBSR	8.44%	97.69%	10.91%	6.14%	94.69%	5.75%	1.23%	94.28%	2.49%
	ABSR	3.51%	97.07%	4.87%	3.47%	55.86%	2.73%	**0.41%**	91.07%	0.81%
	ACC	84.88%	73.21%	84.02%	79.13%	72.97%	79.10%	52.77%	38.25%	51.83%
FLTrust	BBSR	78.27%	99.94%	100.00%	9.81%	96.71%	100.00%	91.98%	97.92%	100.00%
	ABSR	64.73%	94.37%	100.00%	7.82%	93.58%	100.00%	85.73%	86.93%	100.00%
	ACC	81.61%	73.51%	82.82%	70.36%	71.81%	71.67%	50.21%	38.75%	49.54%
RLR	BBSR	87.57%	34.99%	100.00%	51.16%	91.36%	100.00%	89.74%	26.98%	100.00%
	ABSR	71.63%	13.96%	100.00%	36.46%	87.80%	100.00%	74.25%	12.83%	100.00%
	ACC	82.56%	69.98%	82.04%	68.65%	68.66%	69.77%	48.50%	36.91%	47.20%
FLAME	BBSR	6.67%	93.51%	9.38%	7.23%	98.43%	7.50%	1.07%	87.92%	1.74%
	ABSR	3.47%	89.59%	3.91%	4.02%	90.12%	3.40%	0.53%	81.16%	0.92%
	ACC	84.12%	74.35%	84.49%	78.65%	74.73%	77.93%	50.68%	40.80%	51.13%
FLIP	BBSR	94.80%	14.34%	100.00%	7.12%	6.27%	100.00%	93.81%	4.01%	100.00%
	ABSR	10.33%	7.25%	100.00%	2.82%	3.48%	100.00%	5.27%	1.80%	100.00%
	ACC	86.72%	74.93%	**87.89%**	81.02%	75.64%	82.24%	59.22%	43.27%	59.14%
FedBAP (Ours)	BBSR	**4.59%**	**4.59%**	**4.83%**	**2.51%**	**5.03%**	**3.23%**	**0.95%**	**0.97%**	2.60%
	ABSR	**2.16%**	**2.75%**	**2.21%**	**1.84%**	**3.05%**	**2.26%**	0.55%	**0.55%**	1.79%
	ACC	**90.55%**	**80.81%**	86.58%	**87.73%**	**81.07%**	**88.24%**	**63.61%**	**46.51%**	**64.08%**

Table 6: Performance of different defenses on Fashion-MNIST.

	BadNets			A3FL			CerP		
	BBSR	ABSR	ACC	BBSR	ABSR	ACC	BBSR	ABSR	ACC
FedAvg	99.98%	99.93%	**92.46%**	99.99%	99.97%	91.90%	100.00%	100.00%	**92.38%**
Krum	99.63%	22.47%	79.59%	99.76%	39.65%	77.51%	74.33%	7.88%	75.75%
MultiKrum	100.00%	99.73%	91.14%	99.97%	99.90%	91.57%	100.00%	99.99%	91.82%
FLTrust	17.38%	13.41%	89.66%	98.53%	81.43%	89.93%	96.45%	79.85%	89.83%
RLR	100.00%	86.66%	90.94%	99.99%	86.66%	91.05%	100.00%	80.00%	91.24%
FLAME	99.48%	95.99%	91.19%	99.98%	99.95%	91.12%	100.00%	100.00%	91.35%
FLIP	11.16%	1.19%	91.92%	94.42%	15.36%	**92.02%**	100.00%	100.00%	92.21%
RoseAgg	99.68%	73.34%	91.18%	100.00%	99.71%	91.12%	100.00%	99.78%	92.04%
Snowball	99.97%	99.88%	91.84%	100.00%	99.98%	91.57%	100.00%	99.98%	91.67%
BackdoorIndicator	99.93%	50.42%	88.32%	99.98%	99.93%	88.88%	100.00%	99.41%	87.86%
FedBAP (Ours)	**1.77%**	**0.42%**	90.79%	**0.65%**	**0.21%**	90.83%	**0.43%**	**0.14%**	90.15%

selectively shrinking the learning rate in these contentious dimensions, RLR effectively suppresses the influence of poisoned updates while still allowing benign information to be aggregated. RLR does not require knowledge of which clients are malicious, and can be seamlessly integrated into standard FL protocols with minimal overhead.

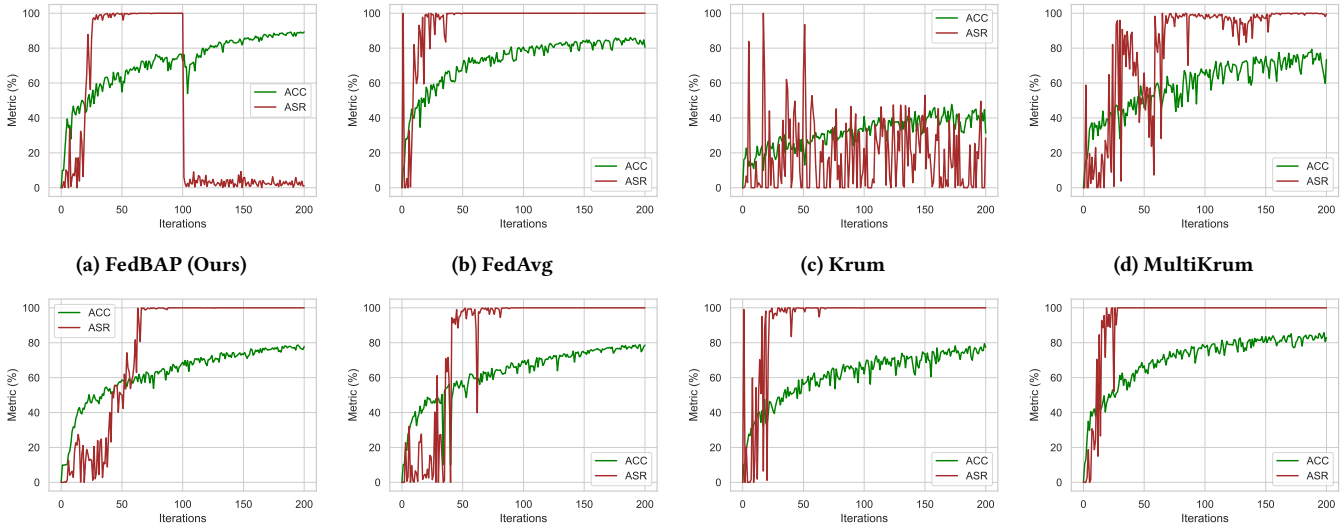
- **FLAME** [23] is a defense framework against backdoor attacks in FL that aims to eliminate malicious behavior while preserving model utility. Unlike filtering-based methods or



(a) FedBAP (Ours) (b) FedAvg (c) Krum (d) MultiKrum

(e) FLTrust (f) RLR (g) FLAME (h) FLIP

Figure 8: ResNet-18 trained with different defenses on the non-IID CIFAR10 dataset against the LP attack.



(a) FedBAP (Ours) (b) FedAvg (c) Krum (d) MultiKrum

(e) FLTrust (f) RLR (g) FLAME (h) FLIP

Figure 9: ResNet-18 trained with different defenses on the non-IID CIFAR10 dataset against the A3FL attack.

differential privacy approaches that may suffer from limited threat coverage or significant accuracy degradation, FLAME estimates the minimal amount of noise required to remove backdoors. It employs model clustering and weight clipping to reduce the necessary noise injection, thereby maintaining benign performance.

- **FLIP** [37] is a novel defense strategy based on trigger reverse engineering to mitigate backdoor attacks in FL. Unlike previous robust aggregation or certified robustness methods, FLIP focuses on hardening benign clients and analyzes the theoretical relationship between cross-entropy loss, attack success rate, and clean accuracy. It guarantees reduced attack success without harming benign performance.

E ADDITIONAL EXPERIMENTAL RESULT

Table 5 presents the performance of different defenses against three backdoor attacks across various model architectures and datasets in IID settings. The results in Table 5 clearly illustrate the performance of different defense strategies against backdoor attacks on IID datasets. The result demonstrate that FedBAP consistently

shows superior performance in both resisting backdoor attacks and maintaining accuracy across different settings.

Figure 8, 9 illustrates the BSR and ACC curves of different defenses against the LP and A3FL attacks on ResNet-18 with CIFAR-10. The curves in Figures 8 and 9 demonstrate that FedBAP effectively keeps the BSR within a low range while maintaining stability, without any sudden spikes. This indicates that FedBAP can consistently control the impact of attacks and avoid instability in the defense performance. Such stability is crucial for ensuring the reliability and long-term effectiveness of the defense system.

To further validate the effectiveness and generalizability of FedBAP, we extend our experiments to the Fashion-MNIST dataset. We additionally incorporate the CerP [18] attack and compare FedBAP with recently proposed defenses, including RoseAgg [35], Snowball [25], and BackdoorIndicator [16]. Specifically, we adopt ResNet-18 as the global model architecture, set the malicious client proportion to 30%, the total number of global communication rounds to 100, and the defense start round in FedBAP to 85. As shown in Table 6, FedBAP demonstrates strong and consistent performance across all attack types.