AdaptiveShield: Dynamic Defense Against Decentralized Federated Learning Poisoning Attacks

Yebo Feng, Baichuan Zheng, Teng Li, *Member, IEEE*, Cong Wu, *Member, IEEE*, Zhuo Ma, *Senior Member, IEEE*, Yulong Shen, *Senior Member, IEEE*, and Jianfeng Ma, *Member, IEEE*

***Abstract*—Federated learning allows decentralized devices to collaboratively train a shared model while keeping datalocal, enhancing the privacy and security of the training process. However, it is vulnerable to poisoning attacks, where malicious participants inject false data to corrupt the global model. To address this, we propose AdaptiveShield, a dynamic hybrid defense approach designed to protect decentralized federated learning against such attacks. AdaptiveShield employs dynamic detection strategies that consider multiple risk factors to assess the maliciousness index and dynamically adjust the detection thresholds, which is able to adapt to various attack scenarios. In addition to attack detections, AdaptiveShield minimizes the negative impact on the global model from missed attackers by dynamically adjusting hyperparameters, thereby enhancing the robustness of the defense. It also dissociates user identities from their uploaded local models through a hierarchical shuffle mechanism, providing an extra layer of privacy protection for both the users and their local models. We evaluate AdaptiveShield across various experimental environments, attack settings, and datasets, demonstrating that it outperforms state-of-the-art approaches by achieving over 0.1 improvement in training accuracy while incurring negligible time overhead.**

Received 24 March 2025; revised 18 November 2025; accepted 29 November 2025. Date of publication 9 December 2025; date of current version 12 March 2026. This work was supported in part by the National Key Research and Development Program of China under Grant 2023YFB2904000, in part by the Natural Science Basic Research Program of Shaanxi under Grant 2025JC-JCQN-073, in part by the National Natural Science Foundation of China under Grant 62272370, in part by Young Elite Scientists Sponsorship Program by CAST under Grant 2022QNRC001, in part by the China 111Project under Grant B16037, in part by the Qinchuangyuan Scientist + Engineer Team Program of Shaanxi under Grant 2024QCY-KXJ-149, in part by Songshan Laboratory under Grant 241110210200, in part by the Open Foundation of Key Laboratory of Cyberspace Security, Ministry of Education of China under Grant KLCS20240405 , in part by the Fundamental Research Funds for the Central Universities under Grant QTZX23071, in part by the National Research Foundation, Singapore, and DSO National Laboratories under the AI Singapore Programme AISG under Award AISG2-GC-2023-008, in part by the National Research Foundation, Singapore, and the Cyber Security Agency under its National Cybersecurity R&D Programme under Grant NCRP25-P04-TAICeN, in part by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme, and in part by Ripple under its University Blockchain Research Initiative (UBRI) [DOI: 10.1109/MPOT.2022.3198929]. *(Corresponding author: Teng Li.)*

Yebo Feng is with the College of Computing and Data Science, Nanyang Technological University, Singapore 639798 (e-mail: yebo.feng@ntu.edu.sg).

Baichuan Zheng, Teng Li, Zhuo Ma, and Jianfeng Ma are with the School of Cyber Engineering, Xidian University, Xi'an 710071, China (e-mail: bczheng@stu.xidian.edu.cn; litengxidian@gmail.com; mazhuo@mail.xidian.edu.cn; jfma@mail.xidian.edu.cn).

Cong Wu is with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong 999077, China (e-mail: congwu@hku.hk).

Yulong Shen is with the School of Computer Science and Technology, Xidian University, Xi'an 710126, China (e-mail: ylshen@mail.xidian.edu.cn).

Digital Object Identifier 10.1109/TDSC.2025.3641896

***Index Terms*—Decentralized federated learning, poisoning attack, dynamic defense, privacy preserving.**

I. Introduction

FEDERATED learning is a machine learning technique that enables multiple decentralized devices to collaboratively train a shared model while keeping their data local, enhancing privacy and security [2], [3], [4]. Its training architecture facilitates distributed local training and centralized model aggregation, ensuring that local data remains undisclosed during training, with only model parameters being uploaded and shared. Additionally, by collaborating with multiple devices, federated learning leverages the diverse data characteristics of different participants to enhance the model's generalization capabilities, achieving robust performance [5]. This approach has gained significant popularity in industries such as healthcare [6], finance [7], and telecommunications [8], where data privacy and security are crucial [9], [10].

However, the privacy feature of federated learning can also introduce vulnerabilities [11], particularly in the form of poisoning attacks [12], [13]. In these attacks, malicious participants may deliberately inject false or misleading data during the local training phase or manipulate model updates before aggregation. Such actions can corrupt the global model, leading to degraded performance, incorrect predictions, or even malicious outcomes [14], [15]. The decentralized and privacy-preserving nature of federated learning makes it challenging to detect and tolerate these attacks [16], as the training data and processes of individual participants remain hidden. Consequently, ensuring robust security measures and developing effective detection and tolerance approaches are critical for maintaining the integrity and trustworthiness of federated learning systems [17], [18], [19].

Researchers have proposed various defense approaches against poisoning attacks in federated learning, categorized into poisoning tolerance, poisoning detection, and hybrid approaches [20]. *Poisoning tolerance approaches* modify the global model aggregation algorithm to mitigate the impact of poisoned local models on the global model [21], [22]. However, continuous participation by attackers means the global model remains persistently affected, making this an incomplete defense [23]. *Poisoning detection approaches* identify and remove attackers by analyzing the local models uploaded by clients [24]. However, this defense's effectiveness depends on knowledge of past attack characteristics, making it challenging to defend

against novel attacks and lacking adaptability [25]. *Hybrid approaches* combine elements of poisoning tolerance and detection, reducing the impact of potential attackers on the global model while removing identified attackers [26], [27]. However, existing hybrid approaches face two significant challenges: most research is based on centralized federated learning, introducing a single point of failure and being unsuitable for decentralized federated learning (DFL) [28], and they lack the capability to dynamically adjust defenses in response to evolving attack methods [29]. Therefore, proposing a dynamic defense approach that is both adaptive and suitable for DFL is crucial to counter poisoning attacks effectively [30].

To address this gap, we propose AdaptiveShield, a dynamic, hybrid defense approach against poisoning attacks in DFL. AdaptiveShield offers three key advantages over existing methods: 1) it tackles dynamically changing attacks by leveraging adaptive detection approaches to navigate ever-evolving operation environments; 2) it minimizes the negative impact on the global model from missed attackers by dynamically adjusting hyperparameters, thereby enhancing the robustness of the defense; and 3) it dissociates user identities from their uploaded local models, providing an additional layer of protection for user privacy.

AdaptiveShield leverages multiple techniques to achieve its design goals. It first employs a hierarchical shuffle system with differential privacy to process users' uploaded local models, reordering parameters within each user group and shuffling the ownership of local model parameters to eliminate identity information and enhance user privacy. Then, it assesses the maliciousness index based on biases between models, users' historical behaviors, and the specific situation of the user group. These assessments are used to dynamically adjust the attacker detection threshold, fine-tune learning rates, and configure training hyperparameters. By considering all risk factors, AdaptiveShield provides comprehensive protection to global models throughout the entire federated learning pipeline, effectively tackling poisoning attacks.

We systematically evaluate AdaptiveShield using various experimental environments, attack settings, and datasets. In the presence of poisoning attacks, the global model trained with AdaptiveShield achieves an accuracy improvement of over 0.1 compared to state-of-the-art approaches, while significantly reducing time consumption. Even under severe attacks, the global model trained with AdaptiveShield only experiences an accuracy decrease of around 3% compared to scenarios without attacks.

To sum up, this paper makes the following contributions:

- We propose a hybrid defense system to protect DFL against poisoning attacks, surpassing state-of-the-art approaches in defense efficacy, time efficiency, and adaptability.
- We design a dynamic maliciousness assessment mechanism for federated learning that considers multiple risk factors. This mechanism adapts to different attack patterns and settings, enhancing the system's resilience and flexibility.
- We introduce a grouped shuffle mechanism for federated learning that conceals the identity information of users' local models, adding an extra layer of privacy protection for users.

TABLE I
DIFFERENCES BETWEEN ADAPTIVESHIELD AND EXISTING SOLUTIONS IN TERMS OF TECHNIQUES USED

Approach	[32]	[37]	[38]	[36]	[35]	[16]	Adaptive-Shield
DFL	✗	✗	✗	✓	✓	✓	✓
Privacy Enhancement	✗	✓	✗	✗	✗	✗	✓
Poisoning Tolerance	✓	✓	✗	✓	✓	✓	✓
Poisoning Detection	✗	✗	✓	✗	✗	✓	✓

- AdaptiveShield increases processing time by about 1% compared to the non-deployed defense algorithm. The training accuracy deviates by no more than 5% from the non-attacked model and improves by over 10% compared to existing defenses under poisoning attacks.

II. RELATED WORK

Poisoning attacks, first introduced before federated learning [31]. Defense approaches against these attacks are generally categorized into poisoning tolerance, poisoning detection, and hybrid approaches [20]. In this section, we summarize their characteristics and compare them with AdaptiveShield. Table I provides an overview of the techniques they used.

Poisoning tolerance approaches modify the global model aggregation algorithm to mitigate the impact of poisoned local models on the global model [21], [22]. For example, Blanchard et al. [32] proposed Krum, a resilient aggregation approach for enhancing stochastic gradient descent in federated learning; Andreina et al. [33] developed BaFFLe to protect the global model. Yin et al. [34] introduced trimmed mean aggregation for global model updating. El-Mhamdi et al. [35] established an equivalence between Byzantine collaborative learning and averaging agreement, proposing two optimal algorithms for robust decentralized learning in heterogeneous, asynchronous non-convex settings. Guo et al. [36] proposed Ubar, a two-stage Byzantine-resilient aggregation rule for decentralized SGD that first filters neighbors by parameter distance and selects updates via local training sample evaluation. Abbas et al. [37] combined byzantine-tolerant aggregation with homomorphic encryption to safeguard encrypted models against poisoning attacks. However, these tolerance algorithms merely lower the influence of attackers without solving the problem fundamentally.

Poisoning detection approaches identify malicious participants in the federated learning system and further remove them to resolve the attack. For instance, Zhang et al. [24] introduced FLDetector, which identifies malicious users by assessing local model update consistency. Zhou et al. [39] proposed MSFL, a framework for dynamic malicious user detection and exclusion via regrouping and behavior evaluation. Yan et al. [38] utilized federated gradient norm vectors to quantify local model disparities and detect malicious clients. However, these detection approaches often fail to provide adequate defense prior to attacker identification and are primarily designed for centralized federated learning, limiting their applicability to DFL.

Fang et al. [16] proposed BALANCE, a Byzantine-robust DFL method that filters malicious neighbor updates by constraining directional deviations from a client's local model, achieving optimal convergence rates in both strongly convex and non-convex settings with no extra communication cost versus FedAvg.

Hybrid approaches combine poisoning tolerance and detection for comprehensive defense. Wang et al. [26] employed an adversarial input filtering algorithm utilizing consistent features and a global model repair algorithm based on neuronal reverse learning to mitigate poisoning attacks. Li et al. [27] conducted a statistical analysis of neighbor nodes' local models to detect attacks and defend against poisoning. However, most existing hybrid defense approaches are tailored for centralized federated learning, impeding their adaptation to DFL architectures.

III. Background and Threat Model

A. Decentralized Federated Learning and Poisoning Attack

While centralized federated learning has been well-studied, it creates a single point of failure and requires participants to place complete trust in the central aggregator. This trust requirement becomes problematic in cross-organizational collaborations, such as healthcare institutions sharing sensitive patient data or competing businesses collaborating on shared AI models. To address these critical limitations in traditional centralized approaches, Decentralized Federated Learning (DFL) emerges as a robust alternative, especially in scenarios where a single trusted aggregation server is impractical or introduces unacceptable security vulnerabilities.

DFL represents an advanced paradigm where multiple clients collaboratively train a global model without relying on a fixed central server. In this setup, server roles are dynamically assigned among participating nodes through random selection [40]. This approach significantly enhances privacy, as no single entity has access to all data, and mitigates the risks associated with a single point of failure [41], [42]. DFL addresses key challenges such as efficient model aggregation, communication overhead, and synchronization through innovative aggregation techniques and robust decentralized communication protocols [2]. By leveraging these mechanisms, DFL improves privacy and system resilience, providing a scalable solution for complex federated learning environments.

The shift to DFL is driven by three key factors. First, regulatory compliance, particularly in sectors like healthcare and finance, often restricts data sharing through centralized entities. Second, organizations are increasingly reluctant to grant a single entity complete control over model aggregation due to competitive and security concerns. Third, the growing scale of federated learning deployments necessitates distributed computational resources to handle model aggregation efficiently.

However, DFL introduces new security challenges, particularly in server selection and trust. Our approach of random server selection for epoch-level aggregation, while addressing computational distribution, requires additional safeguards against compromised servers. Building upon existing centralized FL security measures, we propose extending Byzantine fault tolerance mechanisms and incorporating verifiable computation protocols to ensure aggregation integrity without requiring complete trust in any single server.

Our research bridges the gap between centralized FL security measures and the practical requirements of decentralized deployments, offering a framework that maintains security guarantees while addressing the real-world constraints that make centralized solutions inadequate.

Poisoning attacks in federated learning involve malicious participants deliberately injecting deceptive or harmful models into the training process to compromise the integrity of the global model [43]. These attacks can be categorized into two primary types: label-flipping attacks and backdoor attacks [20]. In label-flipping attacks, adversaries alter the labels of their training data, causing the global model to learn incorrect associations and significantly reducing its classification accuracy. In backdoor attacks, attackers implant a concealed trigger in the data or intentionally modify data labels to induce particular behaviors. When this trigger is activated during inference, it causes the model to generate incorrect outputs for specific inputs while maintaining accuracy in other situations. Both types of poisoning attacks aim to degrade model performance or exploit vulnerabilities, posing significant challenges to maintaining the security and reliability of federated learning systems [44].

B. Threat Model

Attackers' goal: Poisoning attackers aim to undermine the system by degrading the global model's accuracy and reliability. They may manipulate the local model to produce outcomes that benefit them, such as poor performance on specific tasks. Additionally, attackers might introduce malicious data into their local training sets to poison the global model, skewing its learning process and compromising performance.

Attackers' capabilities: Attackers can execute their malicious objectives by sending erroneous updates to the aggregating server, disrupting the aggregation process and corrupting the global model. They can also manipulate data labels in their local training sets, introducing noise or bias that degrades the quality of the training data and impacts the model's accuracy and reliability.

Training data distribution: Since users gather data from diverse environments, their data distributions may vary. Consequently, we consider that the training data can be either independent and identically distributed (IID) or non-independent and identically distributed (non-IID).

IV. Methodology

AdaptiveShield's workflow, depicted in Fig. 1, comprises four primary stages: user configuration, hierarchical shuffle, maliciousness assessment, and configuration updating. User configuration deploys necessary tool kits and training hyperparameters. Hierarchical shuffle enhances privacy through user and model shuffling mechanisms. Maliciousness assessment quantifies user group maliciousness within each epoch by evaluating deviation and anomaly indices. Configuration updating involves sharing user group maliciousness indices, dynamically

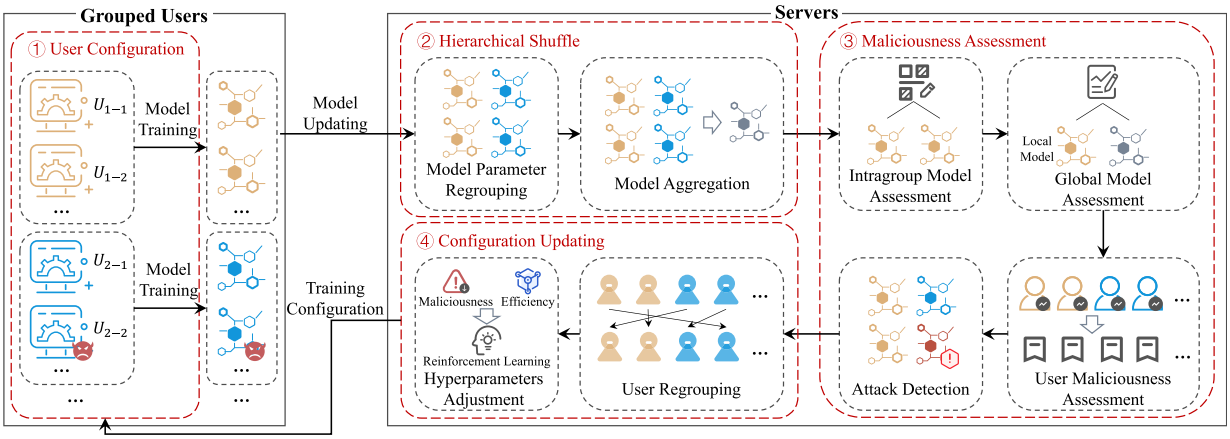


Fig. 1. The workflow of AdaptiveShield.

determining malicious user detection thresholds, and adjusting training hyperparameters accordingly.

Our framework utilizes a dynamic, two-tier communication structure. In each round, the network is partitioned into temporary groups. Within each group, communication follows a star topology where clients report directly to a short-lived group server. Subsequently, these group servers communicate with a randomly elected epoch aggregator. Our analysis assumes an efficient inter-server communication channel, effectively a fully-connected or star topology, where the epoch aggregator can directly poll each group server. This model was chosen to facilitate our core defense mechanisms, such as group-level anomaly detection and the centralized calculation of the global rejection threshold.

The following sections elucidate AdaptiveShield's detailed workflow.

A. User Configuration

Pre-training configuration is critical for learning framework efficiency and quality. Key steps include:

Model selection: Choose based on data characteristics and training requirements. Linear models suit linearly separable data, SVMs for small high-dimensional datasets, decision trees for hierarchical data, and neural networks for complex structures.

Hyperparameter initialization: Set crucial parameters like learning rate and batch size. Learning rate affects update speed, while batch size impacts stability and efficiency. Proper initialization enhances model performance and stability.

Framework configuration: Establish user participation criteria, completion conditions, and communication protocols for decentralized training, ensuring efficient server-user collaboration.

Synchronization: Align model and hyperparameters across servers. New users retrieve these from servers before local iterations, ensuring seamless integration and training continuity.

This configuration phase is fundamental to the zone-centered learning framework, establishing a robust foundation for subsequent training processes and influencing the accuracy and efficacy of outcomes.

B. Hierarchical Shuffle

The hierarchical shuffle architecture implements comprehensive defense mechanisms through dynamic node redistribution and model parameter obfuscation, working synergistically to enhance system security and preserve user privacy in federated learning environments. Through periodic node reassignment, the architecture effectively mitigates various attack vectors. This process prevents malicious entities from establishing persistent control over any single network segment, thereby neutralizing targeted and Sybil attacks. It also enables the early detection and isolation of potentially compromising behavior while ensuring equitable resource distribution. For parameter protection, the architecture employs a multi-layered approach beginning with Laplace noise ensuring that even in cases of server compromise, attackers cannot reconstruct individual user models or establish user-model correlations. This systematic integration of protective measures maintains security integrity even when individual layers are compromised. The architecture combines several defenses: user-level random server assignment, parameter-level differential privacy, transmission-phase parameter randomization, and server-level model update shuffling. Together, these layers provide a robust defense against both targeted attacks and privacy violations.

Formal Privacy Guarantee and Resilience to Attacks: The primary privacy advantage of our hierarchical shuffle stems from the principle of *privacy amplification by shuffling*, which we formalize in Theorem IV.1.

Theorem IV.1 (Privacy Amplification via Hierarchical Shuffle): Assume a group of n users, where each user i applies a local mechanism $\mathcal{M}_i$ that is $\varepsilon_{\text{local}}$-differentially private (DP). The hierarchical shuffle mechanism, by applying a random permutation to the n outputs before they are observed by the server, ensures the resulting combined mechanism is $(\varepsilon_{\text{global}}, \delta)$-centrally differentially private for any $\delta > 0$. The amplified privacy guarantee is approximately bounded by $\varepsilon_{\text{global}} = O(\varepsilon_{\text{local}}/\sqrt{n})$, demonstrating that the central privacy loss is reduced by a factor proportional to the square root of the size of the shuffle group.

This formal guarantee provides inherent resilience against privacy attacks such as **model inversion** and **membership inference**. The success of these attacks is fundamentally limited

by the amount of information that can be leaked about any single participant's data. By its definition, differential privacy provides a quantifiable upper bound on this leakage. As established in Theorem IV.1, our shuffle mechanism achieves a significantly stronger central DP guarantee for a given level of utility compared to standard DP-FL without shuffling. This stronger formal bound directly implies a reduced risk of successful privacy attacks, as any adversary has a provably lower advantage in inferring information about an individual's data. Therefore, this formal proof offers a more robust and generalizable argument for the privacy benefits of our system than an empirical evaluation against a single attack vector, which we leave as an important direction for future work.

Once a user joins the training process, a hierarchical shuffle of user nodes and their data is executed to complicate the attacker's ability to recognize the overall framework and to enhance data privacy. The hierarchical shuffle mechanism is a two-stage process comprising a node shuffle for inter-group reassignment and a local model shuffle for intra-group anonymization. These stages are procedurally integrated into the training pipeline to disrupt attacks and protect user privacy.

When a user satisfies the criteria for participation in the training and requests to join the framework, the server initializes the user's maliciousness index and randomly assigns them to one of the available servers. Furthermore, after each training epoch, users evaluated as non-malicious are reassigned randomly, and dynamic learning hyperparameters, such as the learning rate, are adjusted according to the user's maliciousness index. The node shuffle is executed at the end of each training epoch, immediately following the maliciousness assessment and prior to the start of the next epoch. First, any users identified as malicious are removed from the active participant pool. Subsequently, the remaining set of benign users is randomly re-partitioned and reassigned to the various server groups. This dynamic regrouping ensures that no user remains in the same group for an extended period, which effectively prevents malicious actors from establishing a stable, colluding subgroup to coordinate attacks.

When a user participates in the training, they submit their local model to their assigned server. Once the server has collected and assessed the local model updates from users within the group, it conducts a local model shuffle. Subsequently, the server performs local model sharing and aggregation operations to obfuscate the association between the local model updates and their respective users. This process significantly complicates the attacker's efforts to infer the identity of the user associated with a particular local model update, thereby enhancing the protection of data privacy.

Following the node shuffle, the local model shuffle is performed by each server at the beginning of the aggregation phase for the current epoch. After a server has collected all local model updates from the users within its assigned group, but before performing the model aggregation, it executes the following steps to anonymize the contributions. The server first generates a random permutation of the user identities within its group. It then re-associates the collected model updates according to this permutation, effectively severing the direct link between each update and its original contributor. Specifically, for each parameter position i in the model, the server maintains an availability matrix W where $W_{p,k}$ indicates whether user p's parameter at position k is occupied. When receiving a parameter w_k from user p, the server randomly selects an unoccupied position $W_{p,m}(m \neq k)$ and stores w_k in that location. This process continues until all parameters from all users have been randomly redistributed across the available positions, effectively creating a many-to-many mapping between parameter positions and user identities. For instance, the third parameter from user p_1 might be stored in the third parameter position of user p_3. Subsequently, the corresponding parameter from user p_3 might be stored in a position associated with user p_2. This process establishes a chain of parameter relocation, which effectively obscures the original associations between users and their parameters. Only after this ownership has been shuffled does the server proceed with aggregation. This intra-group anonymization ensures that the aggregation process is performed on a set of dissociated updates, making it computationally infeasible to trace a specific model contribution back to an individual user for that epoch.

The server initially introduces Laplace noise to the local model data prior to transmission to ensure it adheres to the ε-differential privacy criterion. For a user p's local model W_p containing n parameters, the noise added to each parameter w_p^i is computed using the following equation:

$$Lap(w_p^i) = \frac{1}{2\lambda} exp\left(-\frac{\left|w_p^i\right|}{\lambda}\right), \tag{1}$$

where λ denotes the scale parameter of the Laplace distribution from which noise is sampled and subsequently incorporated into the function's output to ensure privacy.

The calculation for λ satisfies the ε-differential privacy requirements with equation $\lambda = \frac{\Delta f}{\varepsilon}$, where the query sensitivity is denoted as Δf. Subsequently, the local model parameters are transmitted in a randomized sequence. Specifically, during the transmission of local model information, the parameters submitted by each user are deliberately scrambled. The model shuffle mechanism demonstrates robust defense capabilities against sophisticated poisoning attacks through its dynamic parameter dissociation approach. By obscuring the temporal and spatial patterns of model updates, the mechanism effectively prevents adversaries from conducting adaptive poisoning attacks based on normal model characteristics. Specifically, attackers are unable to analyze the statistical properties or structural patterns of legitimate model updates, which significantly impairs their ability to craft poisoned parameters that could evade detection systems. The randomization protocol also disrupts potential timing-based attack strategies where adversaries alternate between uploading poisoned models and intercepted legitimate updates to avoid detection. This temporal obfuscation, combined with parameter-level shuffling, creates a dual-layer defense that forces attackers to operate without knowledge of the current model state or update patterns. Furthermore, the mechanism's inherent unpredictability in parameter assignment makes it challenging for attackers to maintain consistent control over specific

model components, thereby reducing the effectiveness of targeted poisoning attempts and increasing the probability of attack detection through statistical anomaly identification.

C. Maliciousness Assessment

Upon completion of local model collection and processing for the current epoch by all servers, a server is randomly chosen to serve as the model aggregation server for that epoch. The remaining servers then transmit their processed local model information to the selected aggregation server. After the data collection is finalized, the aggregation server performs the model aggregation, updates the global model accordingly, and disseminates the updated global model to the other servers.

Concurrently, each server conducts an assessment of posioning attack activity for the current epoch across each user group. This assessment involves evaluating user behavior based on two metrics: the deviation index and the anomaly index.

The deviation index quantifies the discrepancy between the weights of a local model in the current training epoch and the reference weights. This discrepancy is computed using metrics such as mean squared error (MSE). Instead of evaluating each user individually, which is impossible due to anonymity, this error is calculated for the entire group a user belongs to. A high Err score for a group means that at least one of its members likely submitted a malicious model that is far from the global model, S. This error gives us an immediate signal that poisoning is present within that group. By evaluating this error, the deviation index provides a direct measure of how updates to the local model influence the global model. Furthermore, since posioning attack activities often lead to deviations in the global model, this index offers an intuitive and prompt assessment of the extent to which local model updates diverge from the global model. For instance, using MSE, the deviation between the n parameters of the global model S and the n parameters of the local model s_i can be computed as follows:

$$Err(s_i, S) = \left\| \left(\sum_{j=1}^{n} (s_{i,j} - \hat{s}_{i,j})^2 \right)^{\frac{1}{2}} \right\|. \tag{2}$$

Deviation from the Maliciousness Index f_D is defined as follows:

$$f_D(s_i, t) = \left\{ \zeta \sum_{r=1}^{t} Err(s_i, S | s_i \in S), 0 \right\}_{\max} \Big/ t, \tag{3}$$

where ζ functions as a scaling factor that modulates the magnitude of the cumulative error.

The formulation of the deviation index in Equation (3) is designed to balance two competing objectives. The numerator captures the magnitude of the user's deviation from the global model. This ensures that even a single, egregious poisoning attempt is retained as a permanent part of the user's risk profile and is not diluted by subsequent, less deviant updates. Conversely, the denominator, t, introduces a temporal decay effect. This mechanism ensures that the penalty associated with a past transgression gradually diminishes over time if the user consistently provides benign updates. By combining these two elements, the deviation index penalizes users based on their peak malicious behavior while allowing for a gradual reduction of their risk score contingent on sustained good conduct.

The anomaly index quantifies the extent of posioning attack behavior based on the frequency of anomalous behavior exhibited by a user since joining the group. This index is assessed using machine learning algorithms designed to detect outliers or deviations that diverge from established deviation patterns. Given the emergence of adversarial attacks and other sophisticated attack methods, local models impacted by such attacks may show minimal deviation but significant performance discrepancies. Employing the anomaly index for evaluation can offer a more robust defense against these intricate attacks that seek to evade traditional deviation detection mechanisms.

Prior to calculating the anomaly index, a classifier C is trained using data from normal models to detect significant inconsistencies among local models within the group during the current cycle. In this study, the DBSCAN clustering algorithm is employed to identify anomalous local models.$Flag(s_i | s_i \in G_k)$ represent the classification outcome of the classifier C for the local model s_i in group G_k. If the local model is not classified as anomalous, the result is recorded as 1; otherwise, it is recorded as 0. The anomaly index f_A is then defined as follows:

$$f_A(s_i, t) = \left\{ t - \eta \sum_{r=1}^{t} \max\big(Flag(s_i | s_i \in G_k)\big), 0 \right\}_{\max} /t, \tag{4}$$

where η is a scaling factor that adjusts the magnitude of the sum of classification results.

Equation (4) is designed to generate an anomaly index by incorporating a weighted historical assessment of a user's classification results. The core intent is to reflect the persistence and severity of anomalous behavior over time. Specifically, the numerator functions as a historical marker, indicating whether the user has ever been classified as non-anomalous during the training process up to epoch t. If a user has consistently exhibited anomalous behavior, this marker will yield a value of 0, resulting in the maximum possible anomaly index, reflecting persistent maliciousness. If the user has, at any point, been classified as benign, the anomaly index is adjusted downwards by a factor determined by η and the total epochs t. This mechanism ensures that a history of anomalous classifications contributes significantly to the user's risk profile, while also allowing for a temporal scaling of this impact based on their overall engagement duration.

By synthesizing each user's deviation index f_D and anomaly index f_A, their maliciousness index can be expressed as follows:

$$TI = e^{(1-V) f_D(s_i,t) + V f_A(s_i,t)}, \forall s_i \in S, V \in [0, 1], \tag{5}$$

where V is a weighting parameter governing the relative contributions of the deviation index $f_D(s_i, t)$ and the anomaly index $f_A(s_i, t)$. This parameter interpolates between the two indices. At $V = 0$, the equation simplifies to $TI = e^{f_D(s_i,t)}$, indicating that the maliciousness index TI is solely determined by $f_D(s_i, t)$. Conversely, at $V = 1$, the equation becomes $TI = e^{f_A(s_i,t)}$, indicating exclusive reliance on $f_A(s_i, t)$. For V between 0 and 1, TI represents a weighted combination of both

indices. Thus, V modulates the influence of each index: values near 0 prioritize $f_D(s_i, t)$, values near 1 prioritize $f_A(s_i, t)$, and a value around 0.5 balances both indices equally.

In summary, deviation-based assessment focuses more on measuring behavioral deviations. This approach is relatively quick to execute and consumes fewer computational resources. However, its limitation lies in its insensitivity to changes in attack patterns, which may render it less effective against complex and variable attack methods. In contrast, the machine learning-based anomaly detection approach is more flexible. This approach aims to adaptively learn user behavior models and respond swiftly to various attack methods. Moreover, when encountering unknown attack methods, it can provide effective defense, thereby reducing the reliance on defenders' prior knowledge and enhancing the robustness and stability of the defense.

By combining the deviation index and anomaly index assessment approaches, this paper constructs a multi-level, multi-perspective, and highly adaptive defense approach. This approach provides more comprehensive coverage of various potential attack scenarios, enhancing both detection range and accuracy. It allows the system to dynamically adjust in response to changes in attackers' methods, leveraging the complementary strengths of multiple approaches. Consequently, it improves the system's adaptability to emerging attack methods, offering a more comprehensive and robust defense for DFL.

D. Configuration Updating

Upon completion of the user's maliciousness assessment, each server transmits the maliciousness index of each user to the designated aggregation server for calculating the rejection threshold. At this stage, the threshold for identifying a user as malicious can be determined as follows:

$$Thr = \mu + \alpha\sigma + \varphi, \tag{6}$$

where μ and σ are the mean and standard deviation of all indexes within the subgroup, respectively, while α and φ are pre-set hyperparameters to adjust the threshold settings based on real-world conditions. In the equation, the mean index μ establishes a baseline threshold that reflects the average index, thereby aligning with the subgroup's typical performance. The standard deviation σ accounts for index variability, making the threshold robust to natural fluctuations. The scaling factor α controls sensitivity to variability, allowing customization of the threshold's responsiveness. And φ fine-tunes the threshold independently of mean and variability for specific adjustments. Consequently, this equation creates an adaptive, robust, and customizable threshold, improving the accuracy and reliability of poisoning attack detection in DFL systems.

As training progresses, the maliciousness index for each user and the identification threshold for detecting malicious users will dynamically adjust with each epoch, reflecting changes in model and data characteristics. When a user's index surpasses this threshold, the user is classified as anomalous and will be subsequently removed.

Subsequently, utilizing the user's maliciousness index TI, AdaptiveShield employs a heuristic-based approach to dynamically adjust hyperparameters, such as the learning rate. Specifically, the learning rate assigned to each user for the next training epoch is made inversely proportional to their current maliciousness index. This strategy directly mitigates the influence of potentially malicious users on the global model update while preserving training efficiency, ensuring that the global model continues to learn and improve despite adverse conditions. Consequently, it enhances the overall robustness and accuracy of the system.

Subsequently, the server transmits the updated global model to the users, assigning varying learning rates based on the maliciousness index. Upon receiving this information, users update their local model weights and assess whether specific conditions, such as global model error thresholds and the number of training epochs, are met. If these conditions are satisfied, the training process is halted; otherwise, it continues.

The system then iterates through the phases of hierarchical obfuscation, maliciousness assessment, and configuration updating until the training termination criteria are fulfilled.

V. Evaluation

A. Experimental Setup

Our experiments are conducted using two real-world datasets. Each dataset is partitioned into a training set and a test set, with proportions of 80% and 20%, respectively.

The MNIST dataset contains 70,000 grayscale images of handwritten digits ($0-9$), each with a resolution of 28×28 pixels. It is widely utilized as a benchmark for evaluating image classification algorithms due to its simplicity and efficacy in testing fundamental machine learning models.

The CIFAR-10 dataset comprises 60,000 color images, each with a resolution of 32×32 pixels, divided into 10 distinct classes, including various vehicles and animals. It is commonly employed to assess the performance of more advanced image recognition algorithms, owing to its diverse and challenging array of images.

a) Implementation details: The experiments described in this paper are conducted on computers equipped with an Intel Core i7-10750H CPU, an NVIDIA GeForce RTX 2060 GPU, 16 GB of RAM, and the Windows 11 23H2 operating system. The implementation of AdaptiveShield is carried out using the Python-based PyTorch federated learning framework, along with the Scikit-learn machine learning library.

Our client-server configuration consisted of 100 clients distributed among 10 server-managed groups. To simulate a non-IID environment, each client's local dataset ($500-600$ entries) was partitioned using a Dirichlet distribution over class labels with a concentration parameter of $\alpha = 0.5$. The training procedure was run for 150 communication rounds, with clients performing 5 local epochs per round using Stochastic Gradient Descent (SGD) with a momentum of 0.9. The initial learning rate was set to 0.01, the local batch size was 32, and the optimization objective was Cross-Entropy Loss. For the AdaptiveShield defense, key hyperparameters were set as follows:

the deviation/anomaly weight $V = 0.5$, the standard deviation scaling factor $\alpha = 1$, and the differential privacy budget $\varepsilon = 5.0$ for our primary comparative experiments. For the model architectures, we employed well-established networks: the classic LeNet architecture for the MNIST dataset, and a variant of AlexNet adapted for smaller images, herein referred to as AlexCifarNet, for the CIFAR-10 dataset. The AlexCifarNet model features two blocks of sequential convolutional layers followed by max-pooling, and concludes with two fully connected layers.

The selection of key hyperparameters was guided by established practices in federated learning literature and a preliminary grid search to ensure robust and stable performance. The initial learning rate of 0.01 and a local batch size of 32 represent a standard trade-off between convergence speed and the stability of gradient updates. For the defense mechanism, the deviation/anomaly weight V was set to 0.5 to ensure a balanced contribution from both detection metrics, as its impact on performance will be further analyzed in a subsequent section. The threshold scaling factor α was set to 1 to establish a baseline sensitivity that flags participants whose maliciousness index exceeds one standard deviation from the group mean, providing a statistically grounded approach for identifying significant outliers.

In terms of experimental parameter settings, we configure 10 servers and 100 clients, with each client holding between 500 and 600 non-i.i.d. data entries for local training. Depending on the attack scenario, local batch sizes for client training are selected from among 16, 32, 64, and 128. Additionally, learning rates for local model training range from 0.001 to 0.1, tailored to the specific training characteristics of each client. The hyperparameters are set as follows: $\zeta = 0.01$, $\eta = 1$, $V = 0.5$, and $\alpha = 1$.

b) Baselines: To better illustrate the accuracy and robustness of AdaptiveShield, we compared its performance with that of state-of-the-art solutions such as Learn, Ubar, and BALANCE.

Learn [35]: This work addresses Byzantine collaborative learning in a fully decentralized, asynchronous, and heterogeneous setting. It establishes a theoretical equivalence between this complex learning task and a more abstract distributed computing problem termed averaging agreement. Leveraging this reduction, the authors propose two distinct, optimal algorithms. The first, Minimum-Diameter Averaging, identifies and averages a subset of vectors with the smallest diameter to achieve an asymptotically optimal averaging constant. The second, Reliable Broadcast-Trimmed Mean, uses reliable broadcast to ensure consistency before applying a coordinate-wise trimmed mean, providing optimal Byzantine resilience. This framework allows for the derivation of robust learning protocols directly from solutions to the simpler agreement problem.

Ubar [36]: To defend against Byzantine attacks in decentralized learning, this paper proposes a Uniform Byzantine-resilient Aggregation Rule (Ubar). This method introduces a two-stage filtering process executed by each benign node. In the first stage, a node shortlists candidate updates from its neighbors based on their parameter distance to its own model, using its local state as a trusted anchor. In the second stage, these shortlisted models are further validated by evaluating their performance on the node's local data. By only aggregating the models that prove to be beneficial, UBAR effectively isolates and discards malicious updates, ensuring the collaborative training process remains secure and efficient.

BALANCE [16]: This paper introduces a defense against Byzantine poisoning attacks in DFL, named Byzantine-robust averaging through local similarity in decentralization (BALANCE). The core mechanism leverages each client's own local model as a trusted similarity reference to validate models received from neighbors. A client accepts a peer's model for aggregation only if it is sufficiently close to its own, effectively filtering out malicious updates that deviate significantly. This approach provides theoretical convergence guarantees for both strongly convex and non-convex settings, demonstrating that its convergence rate under attack matches the optimal rate of its Byzantine-free counterparts, all without requiring prior knowledge of the network's malicious ratio.

c) Attack methods: To assess the effectiveness of safeguarding model performance and ensuring robustness, We use two representative poisoning attack methods to evaluate and test the defense efficacy of AdaptiveShield.

Label flipping attack: This attack represents a classic non-targeted poisoning attack approach. It involves randomly altering the labels of data, which impedes the global model's ability to learn accurate data classifications. Consequently, the global model is compromised through the introduction of the local malicious model, diminishing its overall usability.

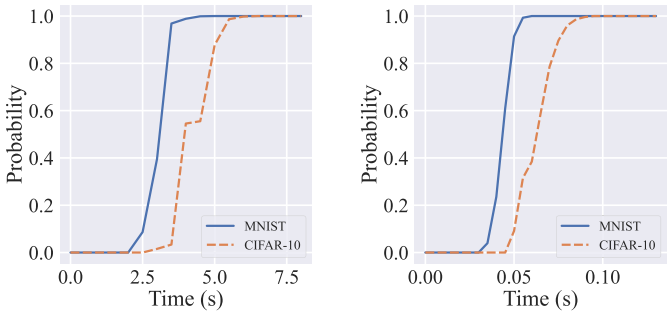
Backdoor attack: This attack is a prominent example of a targeted poisoning attack. It involves deliberately altering data labels to induce specific, undesirable behaviors in the global model when it encounters particular inputs. This manipulation undermines the credibility of the global model by causing it to produce incorrect or unexpected outputs under certain conditions.

Non-omniscient attack [32]: This attack strategy subverts statistical defenses by exploiting the inherent variance among honest workers' gradients. Instead of submitting large, outlier gradients, the adversary crafts malicious updates that are intentionally subtle. These updates are designed to be closer to the mean of the honest gradients than the updates from a subset of benign workers whose gradients naturally deviate in the opposite direction. This causes robust aggregation rules, such as Krum and Trimmed Mean, to misidentify and discard the benign gradients as outliers, while accepting the malicious ones. The attack can be used to either degrade the model's overall performance or to inject a backdoor by systematically shifting the aggregated model in a desired direction.

This study implements two attack strategies to evaluate model vulnerability during training using MNIST and CIFAR-10 datasets. In the label-flipping attack, labels in the adversarial dataset are randomly altered. For the backdoor attack, specific modifications are made: in MNIST, the digit "5" is relabeled as "8", while in CIFAR-10, instances of "Dog" are reclassified as "Ship".

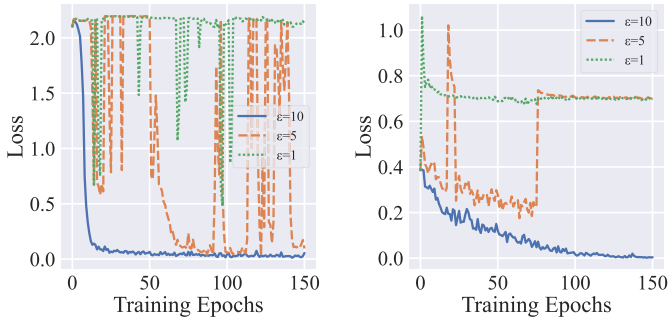
B. Defense Cost

To assess the efficiency of defense mechanisms and the impact of privacy constraints on model performance, we analyze the



(a) Local training. (b) Defense.

Fig. 2. CDFs of time consumption for local training and defense.



(a) MNIST. (b) CIFAR-10.

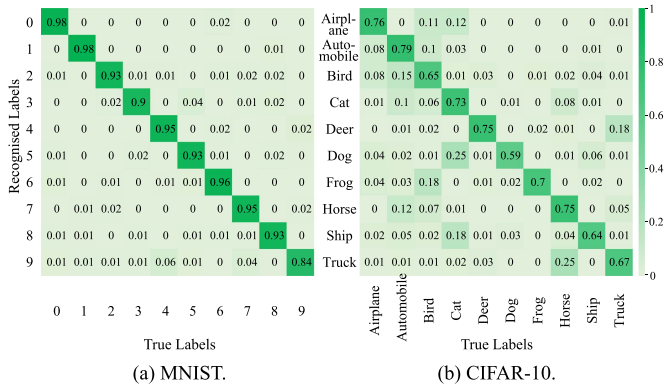
Fig. 3. The effect of the privacy budget ε on the convergence rate of the global model.

associated costs, including the time spent on local training and defense activities, as well as the loss of the global model. Key indicators of these costs are evaluated by measuring the time spent on training and defense, and examining how different privacy budgets ε affect the global model's convergence speed. We also plotted the cumulative distribution functions (CDFs) of global model training and defense across various datasets and assessed the variation in model loss over epochs with different privacy budgets.

Fig. 2 illustrates that the additional operational time required by AdaptiveShield has a negligible impact on global model training, with only a roughly 1% increase in time. Thus, AdaptiveShield successfully introduces effective system complexity with only a minor increase in system running time, while enhancing the privacy protection of local models, improving overall system security, and increasing the system's stability in response to attacks.

To enhance the privacy protection of user data, noise is added to the local model using differential privacy before the local model exchange. This added noise interferes with the aggregation of the global model. In this section of the experiments, the global model loss is measured separately for privacy budgets of 10, 5, and 1, respectively.

Fig. 3 demonstrates that the size of the privacy budget significantly affects the performance of the global model. With $\varepsilon = 10$, the global model achieves zero training loss on both the MNIST and CIFAR-10 datasets, indicating effective learning of



(a) MNIST. (b) CIFAR-10.

Fig. 4. Accuracy of the global model trained without attacks.

data features. However, when $\varepsilon = 5$, the global model's training loss shows variability due to the introduction of additional noise, which complicates feature learning and increases noise interference during local model aggregation. At $\varepsilon = 1$, the training loss remains persistently high across both datasets. Although a higher privacy budget significantly enhances data privacy by introducing more noise, this noise effectively obscures data features, thereby impeding the global model's ability to learn accurately. As a result, the global model's training loss remains elevated, and the global model encounters difficulties in converging.

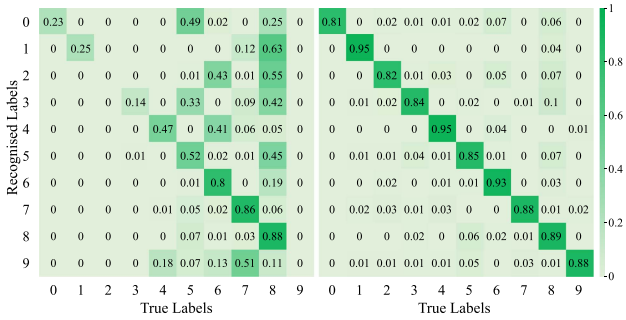
When the privacy budget is large, the level of privacy protection for user data is relatively low, as the noise added to the local model is minimal and has a negligible impact on global model updates. Conversely, as the privacy budget decreases, user data privacy protection improves, resulting in increased noise being added to the local model. This additional noise significantly elevates the global model's training loss, leading to a marked reduction in global model performance and a slower convergence rate.

C. Defense Efficacy

To demonstrate the robustness and applicability of AdaptiveShield, we evaluate its performance against label-flipping and backdoor attacks across different datasets by measuring the model's classification accuracy.

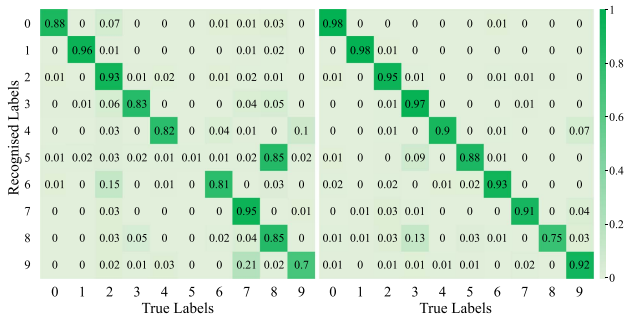
Fig. 4 presents the baseline classification performance of the global model, trained in an attack-free environment, on the MNIST and CIFAR-10 datasets. Fig. 5, 6 illustrate the classification accuracy of the global model before and after enabling the defense, using the MNIST dataset. In these figures, the horizontal axis represents the predicted label, while the vertical axis denotes the true label.

In the absence of attacks, the average accuracy of federated learning is 0.935. Under label-flipping attacks, without any defense measures, the average accuracy of the global model declines to 0.44. This significant drop occurs because the randomly modified labels disrupt the model's ability to correctly map and learn data features, resulting in poor classification performance. Consequently, the label-flipping attack successfully compromises the global model. However, after implementing



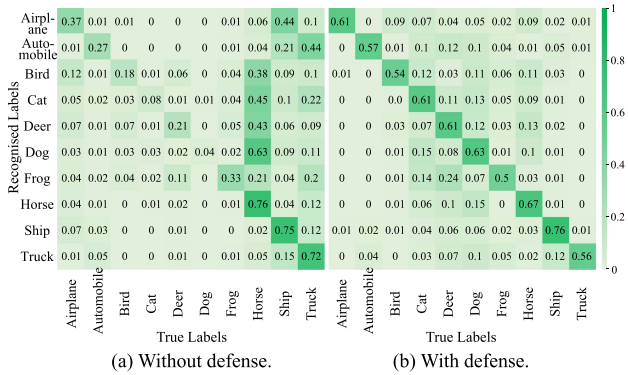
(a) Without defense. (b) With defense.

Fig. 5. Efficacy of defense against label flipping attacks, evaluated with MNIST dataset.



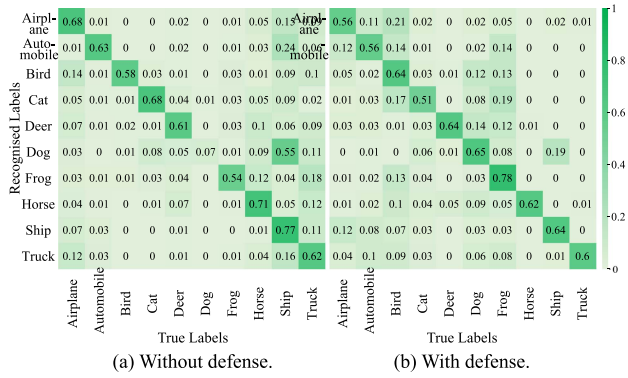
(a) Without defense. (b) With defense.

Fig. 6. Efficacy of defense against backdoor attacks, evaluated with MNIST dataset.



(a) Without defense. (b) With defense.

Fig. 7. Efficacy of defense against label flipping attacks, evaluated with CIFAR-10 dataset.



(a) Without defense. (b) With defense.

Fig. 8. Efficacy of defense against backdoor attacks, evaluated with CIFAR-10 dataset.

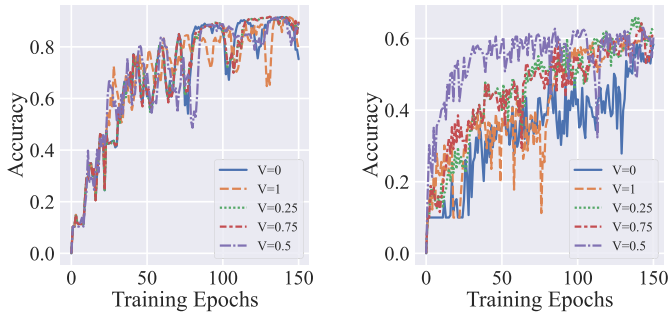
defense measures, the average accuracy of the global model increases to 0.88, restoring its efficacy to a level comparable to that achieved without any attacks. During each training epoch, the defense system effectively identifies the attacker and mitigates the impact of the poisoned local models, ultimately defending against the label-flipping attack and ensuring the accuracy and effectiveness of the global model.

Similarly, under a backdoor attack, without deploying any defensive measures, the overall accuracy of the global model is 0.77, which reflects reasonably good performance. However, due to the insertion of backdoor data, the classification accuracy for the digit "5" is only 0.01, while the accuracy for misclassifying it as "8" is 0.85. This indicates that the attacker has effectively executed a covert and damaging backdoor attack, which poses significant challenges for the dynamic defense approach. When employing AdaptiveShield under the backdoor attack, the global model's classification accuracy improves from 0.77 to 0.92. Additionally, the classification accuracy across all data types remains high, demonstrating that AdaptiveShield effectively counters backdoor attacks and maintains robust defense efficacy

As shown in Fig. 7, 8, AdaptiveShield demonstrates enhanced defense capabilities utilizing the CIFAR-10 dataset. Under the label-flipping attack, the global model trained on this dataset experiences a considerable decline in classification accuracy. However, with the deployment of AdaptiveShield, the accuracy of the global model is significantly improved.

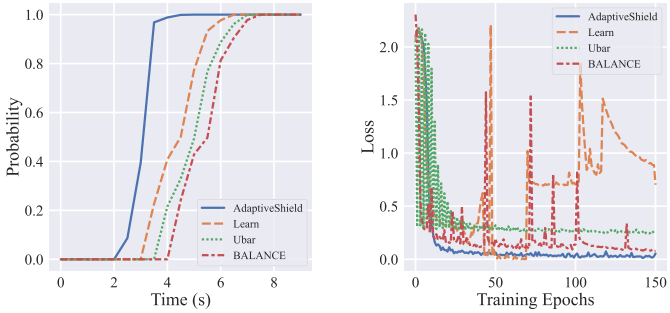
Under the backdoor attack, the overall accuracy of the global model trained on the CIFAR-10 dataset decreases by 17%. However, with the deployment of AdaptiveShield, the backdoor attack is effectively neutralized, and the classification accuracy across all data types is restored to levels comparable to those before the attack, with only about a 3% difference from the pre-attack accuracy.

To evaluate the robustness of AdaptiveShield with respect to hyperparameter configuration, the parameter V in equation (5) was varied at 0, 0.25, 0.5, 0.75, and 1. A statistical analysis was conducted to assess the global model's efficacy under these different hyperparameter settings. As shown in Fig. 9, the results indicate that on the MNIST dataset, variations in V have minimal impact on the global model's accuracy; however, defense efficiency decreases when $V = 0$ or 1. When V is within the range (0,1), the defense capability is enhanced. This trend is even more pronounced on the CIFAR-10 dataset. Nonetheless, as the number of training epochs increases, the global model accuracy under defense converges to levels observed in the absence of an attack. Overall, AdaptiveShield exhibits strong stability and adaptability against poisoning attacks across different hyperparameter settings and datasets, indicating its suitability for diverse environments.



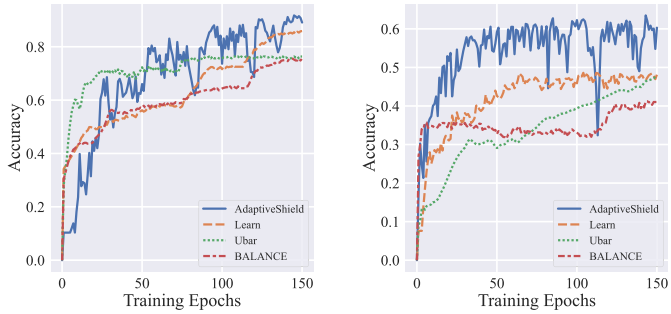
(a) MNIST. (b) CIFAR-10.

Fig. 9. Impact of V values on the effectiveness of defense.



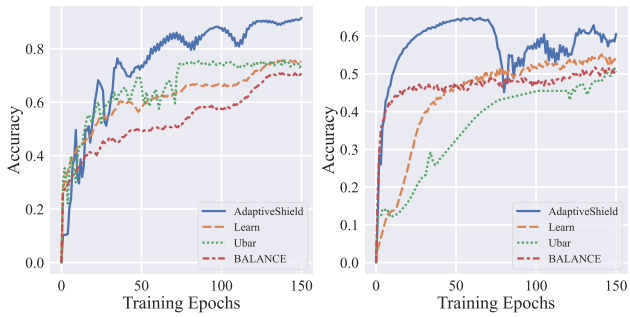
(a) Total time consumption of training and defense. (b) Convergence rate comparison.

Fig. 10. Comparisons of training plus defense time overhead and convergence rate.



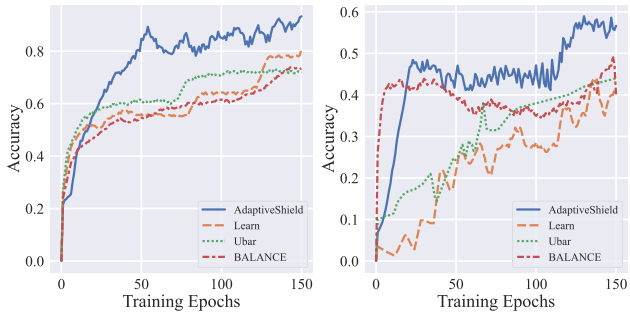
(a) MNIST. (b) CIFAR-10.

Fig. 11. Comparison of training efficiency with different poisoning attack defense solutions under label flipping attacks.



(a) MNIST. (b) CIFAR-10.

Fig. 12. Comparison of training efficiency with different poisoning attack defense solutions under backdoor attacks.



(a) MNIST. (b) CIFAR-10.

Fig. 13. Comparison of training efficiency with different poisoning attack defense solutions under non-omniscient attacks.

D. Comparison With Other Solutions

To further illustrate the advantages of AdaptiveShield in terms of time consumption and model performance, we compare its performance with other approaches by examining the CDFs and plotting model accuracy and loss curves across different datasets under various attacks.

The computational efficiency and convergence dynamics of the evaluated defense solutions are detailed in Fig. 10. The empirical evidence highlights the marked superiority of AdaptiveShield on both fronts. As shown in Fig. 10(b), our method achieves a rapid and stable convergence, maintaining a consistently low loss throughout the training process. In contrast, competing solutions exhibit significant drawbacks: Learn displays extreme volatility with recurrent loss spikes, indicating instability, while Ubar and BALANCE plateau at substantially higher loss values. Furthermore, Fig. 10(a) illustrates that AdaptiveShield incurs the lowest computational overhead, completing its training and defense cycle faster than all other methods. This dual advantage of superior convergence stability and reduced time consumption unequivocally establishes AdaptiveShield as a more efficient and robust defense mechanism.

As shown in Fig. 11, 12, 13, the accuracy of the global model produced by existing methods substantially declines under differential privacy constraints. In contrast, AdaptiveShield demonstrates a significant advantage, maintaining high global model accuracy while simultaneously preserving user privacy, identifying attackers, and effectively excluding them from the training process.

As illustrated in Fig. 11, AdaptiveShield solution demonstrates superior performance. While exhibiting some volatility in early training epochs, it consistently trends upward to achieve the highest final accuracy, converging at approximately 0.85. In comparison, Learn and BALANCE show more stable but ultimately lower performance, reaching accuracies of approximately 0.80 and 0.75, respectively. The Ubar method, despite a strong start, plateaus early and yields the lowest final accuracy

of the group at around 0.70, indicating a limited capacity to mitigate the persistent impact of poisoned data.

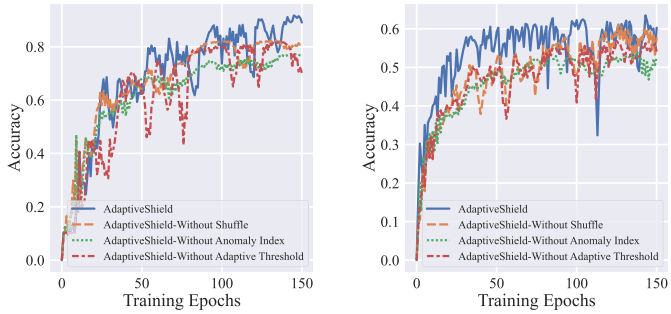
The performance advantages of AdaptiveShield are even more pronounced on the more challenging CIFAR-10 dataset. AdaptiveShield rapidly achieves and sustains a leading accuracy level of approximately 0.6, significantly outperforming all other solutions. The next-best method, Learn, struggles to maintain an accuracy of 0.5. The defensive capabilities of BALANCE and Ubar are substantially compromised, with their accuracies stagnating below 0.4 and 0.35, respectively. This demonstrates that while baseline methods offer some resistance, their effectiveness is severely diminished on complex datasets. In contrast, AdaptiveShield maintains high training efficiency and robustly defends the global model, ensuring significantly higher final accuracy under attack.

The comparative performance under backdoor attacks, presented in Fig. 12, unequivocally demonstrates the superior efficacy of our AdaptiveShield solution. On the MNIST dataset, AdaptiveShield consistently maintains the highest accuracy, converging near 0.9, while competing methods like Learn, Ubar, and BALANCE plateau significantly lower, around 0.75. This performance gap is magnified on the more complex CIFAR-10 dataset, where AdaptiveShield rapidly achieves and sustains an accuracy above 0.6. In stark contrast, the other defenses falter, struggling to surpass the 0.5 accuracy mark, with Ubar exhibiting particularly poor performance. These results affirm the robust defense capabilities and enhanced training efficiency of AdaptiveShield, especially in challenging scenarios where baseline methods are largely ineffective.

Fig. 13 evaluates the defense mechanisms in a more realistic non-omniscient attack scenario, where our proposed AdaptiveShield again exhibits superior resilience and efficiency. On the MNIST dataset, AdaptiveShield overcomes a slightly slower start to establish a clear lead, ultimately converging to a final accuracy approaching 0.9. This performance significantly surpasses that of Ubar, Learn, and BALANCE, which fail to exceed an accuracy of 0.75. The performance delta is even more pronounced on the CIFAR-10 dataset. Here, AdaptiveShield demonstrates a rapid and sustained increase in accuracy, reaching nearly 0.6. In stark contrast, competing methods are severely hindered; BALANCE plateaus early, while Learn and Ubar exhibit extreme instability and fail to achieve effective learning. These findings underscore the robustness of AdaptiveShield's adaptive defense strategy, which remains highly effective even with incomplete information about the threat landscape.

E. Ablation Experiments

To isolate the contribution of each key component within the AdaptiveShield framework, we conducted a comprehensive ablation study. We evaluated the performance of the full system against three variants, each with a single core mechanism disabled: (1) AdaptiveShield-Without Shuffle, which removes the privacy-enhancing user and model shuffling; (2) AdaptiveShield-Without Anomaly Index, which relies solely on the model deviation for maliciousness assessment; and (3) AdaptiveShield-Without Adaptive Threshold, which employs a static rejection threshold instead of the dynamically adjusting



(a) MNIST. (b) CIFAR-10.

Fig. 14. Ablation analysis of component contributions to the robustness of AdaptiveShield.

one. All experiments were conducted under label flipping attack conditions.

The results, presented in Fig. 14, demonstrate that the complete AdaptiveShield architecture consistently outperforms all ablated variants on both the MNIST and CIFAR-10 datasets. The analysis reveals several key insights into the system's design.

The AdaptiveShield Without Adaptive Threshold variant exhibits the most significant performance degradation and volatility. The frequent and sharp drops in accuracy indicate that a static threshold is ill-suited to the dynamic federated learning environment. Its inflexibility renders it either too permissive, allowing attacks to succeed, or too restrictive, incorrectly penalizing benign clients. This highlights the critical role of the adaptive threshold in maintaining stability and resilience.

Disabling the anomaly index, as evaluated in the AdaptiveShield Without Anomaly Index variant, also leads to a marked decrease in performance. This finding underscores the importance of a multifaceted risk assessment; relying only on immediate model deviation is insufficient to counter attacks that may be subtle or build up over time. The historical context provided by the anomaly index is crucial for robust detection.

Finally, removing the hierarchical shuffle with the AdaptiveShield Without Shuffle variant results in a marginal but consistent reduction in final accuracy and stability compared to the full system. This suggests that while the core detection and adaptation mechanisms are the primary drivers of performance, the shuffle component provides an essential supplementary layer of defense by disrupting attacker collusion and complicating the task of crafting effective poisoned updates.

In summary, these experiments validate the synergistic design of AdaptiveShield. They confirm that each component, namely the hierarchical shuffle, the dual metric maliciousness assessment, and the adaptive threshold, contributes indispensably to the system's overall robustness and effectiveness against poisoning attacks.

VI. Discussion

A. Limitations

While AdaptiveShield demonstrates significant advancements in defending against poisoning attacks in DFL, several limitations warrant further consideration.

Trade-off Between Privacy and Utility: The hierarchical shuffle mechanism incorporates differential privacy (DP) to obfuscate user identities. While DP enhances privacy, stricter privacy budgets (e.g., $\varepsilon = 1$) introduce substantial noise, degrading model convergence and accuracy (as shown in Section V-B). Balancing privacy guarantees with model performance across diverse applications remains an open challenge.

Scope of Privacy Evaluation: We establishes a formal privacy guarantee through Theorem IV.1, which provides a strong theoretical foundation for the system's resilience against information leakage. However, the scope of the current evaluation is focused on model performance and defense efficacy against poisoning attacks. Consequently, empirical validation of the system's robustness against specific privacy attacks, such as model inversion or membership inference, was not performed. While the formal proof offers a quantifiable upper bound on potential privacy loss, empirical testing against these attack vectors remains an important direction for future investigation.

Scalability in Large-Scale Deployments: The experiments assume a moderate-scale DFL environment (100 clients, 10 servers). In larger deployments with thousands of participants, the hierarchical shuffle and dynamic reassignment mechanisms may incur non-negligible communication overhead. While the current time overhead is reported as minimal, scalability under bandwidth-constrained or asynchronous settings requires further investigation.

Vulnerability to Majority Collusion: The efficacy of AdaptiveShield, like other outlier-based defenses, relies on the assumption that benign participants form a statistical majority. When this assumption is violated, the defense not only fails but inverts: the malicious majority skews the global model, causing the system to incorrectly flag and penalize benign participants as outliers. Consequently, defining the precise operational boundary and designing defenses effective against majority collusion remain significant open challenges.

Impact of Communication Topology: While our design assumes efficient inter-server communication, its core defense principles can be adapted to other topologies. Under a constrained graph like a **line or ring**, the multi-hop information propagation would introduce a linear latency overhead, O(S), for coordinating the global defense, impacting scalability. Similarly, a **gossip-based protocol** would increase delay and complexity in achieving a consistent state for calculating the rejection threshold. Therefore, the efficiency of AdaptiveShield's global coordination is inherently coupled with the underlying inter-server network structure, a trade-off that merits future empirical investigation.

B. Future Work

To address these limitations and advance the field of secure federated learning, we propose the following research directions:

Adaptive Privacy-Preserving Mechanisms: Developing dynamic privacy budget allocation strategies, where ε is adjusted based on data sensitivity or attack severity, could optimize the privacy-utility trade-off. Exploring alternative privacy-preserving techniques, such as secure multi-party computation or homomorphic encryption, may further strengthen privacy without sacrificing model performance.

Empirical Characterization of Adversarial Risk: A critical line of inquiry is to bridge the gap between the formal privacy guarantees of Theorem IV.1 and their operational implications under adversarial interrogation. Future research will involve a systematic investigation of the system's practical resilience against canonical privacy threat models, particularly those involving reconstruction (e.g., model inversion) and distinguishability (e.g., membership inference) adversaries. The objective is not merely to measure attack success, but to characterize the relationship between the system's tunable privacy parameters and the quantifiable advantage an adversary can achieve. This will provide a more nuanced, empirical understanding of the privacy-utility frontier inherent to the AdaptiveShield architecture.

Efficiency Optimization for Large-Scale Systems: Investigating lightweight parameter shuffling protocols and asynchronous aggregation mechanisms would improve scalability. Additionally, integrating edge computing paradigms or leveraging blockchain for decentralized trust management could reduce communication bottlenecks in large-scale deployments.

Proactive Defense Against Adversarial Attacks: Combining AdaptiveShield with adversarial training or certified robustness frameworks could preemptively harden the global model against evasion and backdoor attacks. Exploring reinforcement learning-based strategies to simulate and counteract adaptive attackers would further enhance system resilience.

Modeling Defense Inversion under Majority Collusion: A critical avenue for future research involves modeling the systemic failure dynamics of AdaptiveShield when its core statistical assumption of a benign majority is violated. This inquiry would extend beyond simple threshold determination to characterize the phase transition where the global model's reference distribution becomes dominated by adversarial inputs. The objective is to develop a theoretical framework that can predict the conditions leading to defense inversion, wherein the mechanism begins to systematically penalize benign participants, thereby providing a more nuanced understanding of the system's resilience envelope.

VII. Conclusion

Federated learning enables different devices to collaboratively train a global model while keeping data local, enhancing users' privacy. It's popular in fields like healthcare and finance but vulnerable to poisoning attacks, where malicious participants inject false data. Robust defenses are crucial to ensure the correctness of the trained global model.

In this paper, we introduced AdaptiveShield, a dynamic and hybrid defense approach designed to protect decentralized federated learning systems against poisoning attacks. AdaptiveShield combines adaptive detection methods, hierarchical shuffle of local models, comprehensive maliciousness assessment, and dynamic hyperparameter adjustments to offer robust protection while preserving the privacy of users. Compared with existing solutions, it can effectively tackle dynamically changing poisoning attacks, minimize the negative impact on the global

model from missed attackers, and dissociate user identities from their uploaded local models.

Our evaluation demonstrated that AdaptiveShield significantly improves the accuracy of the global model under attack conditions, with improvements of over 0.1 compared to state-of-the-art methods, and reduces time consumption. Even in severe attacks, AdaptiveShield effectively maintains global model performance, with accuracy decreases limited to around 3% compared to scenarios without attacks. These performance gains are achieved under the operational assumption, common to outlier-based defenses, that benign participants form a statistical majority.

REFERENCES

[1] Y. Feng, J. Xu, and L. Weymouth, "University blockchain research initiative (UBRI): Boosting blockchain education and research," *IEEE Potentials*, vol. 41, no. 6, pp. 19–25, Nov./Dec. 2022.

[2] L. Yuan, Z. Wang, L. Sun, P. S. Yu, and C. G. Brinton, "Decentralized federated learning: A survey and perspective," *IEEE Internet Things J.*, vol. 11, no. 21, pp. 34617–34638, Nov. 2024.

[3] W. Huang et al., "Federated learning for generalization, robustness, fairness: A survey and benchmark," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 9387–9406, Dec. 2024.

[4] J. Wen, Z. Zhang, Y. Lan, Z. Cui, J. Cai, and W. Zhang, "A survey on federated learning: Challenges and applications," *Int. J. Mach. Learn. Cybern.*, vol. 14, no. 2, pp. 513–535, 2023.

[5] M. Gecer and B. Garbinato, "Federated learning for mobility applications," *ACM Comput. Surv.*, vol. 56, no. 5, pp. 1–28, 2024.

[6] H. Guan, P.-T. Yap, A. Bozoki, and M. Liu, "Federated Learning for Medical Image Analysis: A Survey," *Pattern Recognit.*, vol. 151, 2024, Art. no. 110424.

[7] R. Ye et al., "OpenFedLLM: Training large language models on decentralized private data via federated learning," in *Proc. 30th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2024, pp. 6137–6147.

[8] M. Al-Quraan et al., "Edge-native intelligence for 6G communications driven by federated learning: A survey of trends and challenges," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 3, pp. 957–979, Jun. 2023.

[9] N. Rodrıǵuez-Barroso, D. Jiménez-López, M. V. Luzón, F. Herrera, and E. Martıńez-Cámara, "Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges," *Inf. Fusion*, vol. 90, pp. 148–173, 2023.

[10] J. Yuan et al., "Privacy as a resource in differentially private federated learning," in *IEEE INFOCOM 2023-IEEE Conf. Comput. Commun.*, 2023, pp. 1–10.

[11] Y. Zhang et al., "A survey of trustworthy federated learning: Issues, solutions, and challenges," *ACM Trans. Intell. Syst. Technol.*, vol. 15, no. 6, pp. 1–47, 2024.

[12] X. Li, X. Yang, Z. Zhou, and R. Lu, "Efficiently achieving privacy preservation and poisoning attack resistance in federated learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 4358–4373, 2024.

[13] Y. Miao et al., "RFed: Robustness-enhanced privacy-preserving federated learning against poisoning attack," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 5814–5827, 2024.

[14] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning," in *Proc. IEEE Symp. Secur. Privacy*, 2022, pp. 1354–1371.

[15] Z. Xu, F. Jiang, L. Niu, J. Jia, B. Li, and R. Poovendran, "{ACE}: A model poisoning attack on contribution evaluation methods in federated learning," in *Proc. 33rd USENIX Secur. Symp.*, 2024, vol. 24, pp. 4175–4192.

[16] M. Fang et al., "Byzantine-robust decentralized federated learning," in *Proc. of the 2024 on ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, 2024, pp. 2874–2888. [Online]. Available: https://doi.org/10.1145/3658644.3670307

[17] M. Ye, W. Shen, B. Du, E. Snezhko, V. Kovalev, and P. C. Yuen, "Vertical Federated Learning for Effectiveness, Security, Applicability: A Survey," *ACM Comput. Surv.*, vol. 57, no. 9, pp. 1–32, 2024.

[18] A. Khraisat, A. Alazab, S. Singh, T. Jan, and A. Jr Gomez, "Survey on federated learning for intrusion detection system: Concept, architectures, aggregation strategies, challenges, and future directions," *ACM Comput. Surv.*, vol. 57, no. 1, pp. 1–38, 2024.

[19] A. Sharma and N. Marchang, "A review on client-server attacks and defenses in federated learning," *Comput. Secur.*, vol. 140, 2024, Art. no. 103801.

[20] Z. Tian, L. Cui, J. Liang, and S. Yu, "A comprehensive survey on poisoning attacks and countermeasures in machine learning," *ACM Comput. Surv.*, vol. 55, no. 8, pp. 1–35, 2022.

[21] S. Awan, B. Luo, and F. Li, "Contra: Defending against poisoning attacks in federated learning," in *Proc. Comput. Secur. – ESORICS 2021: 26th Eur. Symp. Res. Comput. Secu.*, Darmstadt, Germany, 2021, pp. 455–475.

[22] V. Shejwalkar and A. Houmansadr, "Manipulating the Byzantine: Optimizing model poisoning attacks and defenses for federated learning," *Netw. Distrib. Syst. Secur. Symp.*, 2021.

[23] H. Bansal, N. Singhi, Y. Yang, F. Yin, A. Grover, and K.-W. Chang, "CleanCLIP: Mitigating data poisoning attacks in multimodal contrastive learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 112–123.

[24] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "FLDetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 2545–2555.

[25] X. Qi, T. Xie, J. T. Wang, T. Wu, S. Mahloujifar, and P. Mittal, "Towards a proactive {ML} approach for detecting backdoor Poison samples," in *Proc. 32nd USENIX Secur. Symp.*, 2023,vol. 23, pp. 1685–1702.

[26] B. Wang et al., "Neural Cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 707–723.

[27] X. Li, Z. Qu, S. Zhao, B. Tang, Z. Lu, and Y. Liu, "LoMar: A local defense against poisoning attack on federated learning," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 437–450, Jan./Feb. 2023.

[28] A. Hard, K. Partridge, R. Mathews, and S. Augenstein, "Jointly learning from decentralized (federated) and centralized data to mitigate distribution shift," in *Proc. Neurips Workshop Distrib. Shifts*, 2021, pp. 1–8.

[29] T. Krauß, J. König, A. Dmitrienko, and C. Kanzow, "Automatic adversarial adaption for stealthy poisoning attacks in federated learning," *Netw. Distrib. Syst. Secur. Symp.*, 2024.

[30] E. Hallaji, R. Razavi-Far, M. Saif, B. Wang, and Q. Yang, "Decentralized federated learning: A survey on security and privacy," *IEEE Trans. Big Data*, vol. 10, no. 2, pp. 194–213, Apr. 2024.

[31] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proc. 29th Int. Cof. Int. Conf. Mach. Learn.*, Edinburgh, Scotland: Omnipress, 2012, pp. 1467–1474.

[32] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 119–129.

[33] S. Andreina, G. A. Marson, H. Möllering, and G. Karame, "BaFFLe: Backdoor detection via feedback-based federated learning," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst.*, 2021, pp. 852–863.

[34] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *35th Int. Conf. Mach. Learn.*, 2018, pp. 5650–5659.

[35] E. M. El-Mhamdi, S. Farhadkhani, R. Guerraoui, A. Guirguis, L.-N. Hoang, and S. Rouault, "Collaborative learning in the jungle (decentralized, byzantine, heterogeneous, asynchronous and nonconvex learning)," in *Proc. Adv. Neural Inf. Process. Syst.*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. W. Vaughan, Eds. 2021, vol. 34, pp. 25044–25057. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/d2cd

[36] S. Guo et al., "Byzantine-resilient decentralized stochastic gradient descent," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 6, pp. 4096–4106, Jun. 2022.

[37] A. Yazdinejad, A. Dehghantanha, H. Karimipour, G. Srivastava, and R. M. Parizi, "A robust privacy-preserving federated learning model against model poisoning attacks," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 6693–6708, 2024.

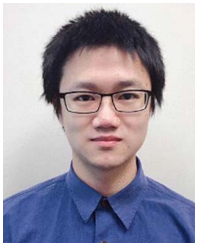
[38] G. Yan, H. Wang, X. Yuan, and J. Li, "DeFL: Defending against model poisoning attacks in federated learning via critical learning periods awareness," in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, no. 9, pp. 10711–10719.

[39] Z. Zhou, C. Xu, M. Wang, X. Kuang, Y. Zhuang, and S. Yu, "A multi-shuffler framework to establish mutual confidence for secure federated learning," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 5, pp. 4230–4244, Sept.–Oct. 2023.

[40] E. T. M. Beltrán et al., "Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 4, pp. 2983–3013, Fourthquarter 2023.

[41] S. Warnat-Herresthal et al., "Swarm learning for decentralized and confidential clinical machine learning," *Nature*, vol. 594, no. 7862, pp. 265–270, 2021.

[42] E. T. M. Beltrán et al., "Fedstellar: A platform for decentralized federated learning," *Expert Syst. Appl.*, vol. 242, 2024, Art. no. 122861.
[43] T. T. Nguyen et al., "Manipulating Recommender Systems: A Survey of Poisoning Attacks and Countermeasures," *ACM Comput. Surveys*, vol. 57, no. 1, pp. 1–39, 2024.
[44] Z. Wang, J. Ma, X. Wang, J. Hu, Z. Qin, and K. Ren, "Threats to training: A survey of poisoning attacks and defenses on machine learning systems," *ACM Comput. Surv.*, vol. 55, no. 7, pp. 1–36, 2022.



Yebo Feng received the PhD degree from the University of Oregon (UO) in 2023. He was the reviewer of *IEEE Transactions on Dependable and Secure Computing*, *ACM Transactions on Knowledge Discovery from Data*, and *IEEE Journal on Selected Areas in Communications*. His research interests include network security, AI security, and intrusion detection. He was the recipient of the Best Paper Award of 2019 IEEE CNS, Gurdeep Pall Graduate Student Fellowship of UO, and Ripple Research Fellowship.

Baichuan Zheng received the BS degree in 2024 in School of Cyber Engineering from Xidian University, China, where he is currently working toward the MS degree. His current research interests include network security, information security, and federated learning.

Teng Li (Member, IEEE) received the BS degree in School of Computer Science and Technology from Xidian University, China in 2013, and the PhD degree in 2018, in School of Computer Science and Technology from Xidian University, China where he is currently a professor with the School of Cyber Engineering, Xidian University, China. His current research interests include wireless and networks, distributed systems and intelligent terminals with focus on security and privacy issues.

Cong Wu (Member, IEEE) received the PhD degree from the School of Cyber Science and Engineering, Wuhan University, in 2022. He is currently a HKU-URC postdoctoral researcher with Department of Electrical and Electronic Engineering, University of Hong Kong. His research interests include blockchain, Web3, and distributed system security.

Zhuo Ma (Senior Member, IEEE) received the PhD degree in computer architecture from Xidian University, Xi'an, China, in 2010. He is currently a professor with the School of Cyber Engineering, Xidian University. His research interests include cryptography, machine learning in cyber security, and the Internet of things security.

Yulong Shen (Senior Member, IEEE) received the BS and MS degrees in computer science and the PhD degree in cryptography from Xidian University, Xi'an, China, in 2002, 2005, and 2008, respectively. He is currently a professor with the School of Computer Science and Technology, Xidian University, and also an associate director of Shaanxi Key Laboratory of Network and System Security. His research interests include wireless network security and cloud computing security.

Jianfeng Ma (Senior Member, IEEE) received the BS degree in computer science from Shaanxi Normal University in 1982, and the MS and PhD degrees in computer science computer science from Xidian University in 1992 and 1995, respectively. He is currently the directer of Department of Cyber Engineering and a professor in School of Cyber Engineering, Xidian University. He has authored or coauthored more than 150 journal and conference papers. His research interests include information security, cryptography, and network security.