Resisting Poisoning Attacks in Federated Learning via Dual-Domain Distance and Trust Assessment

Zhaoqi Wang, *Graduate Student Member, IEEE*, Zijian Zhang, *Senior Member, IEEE*, Zhen Li, Yan Wu, *Graduate Student Member, IEEE*, Ye Liu, Meng Li, *Senior Member, IEEE*, Xin Li, Yong Liu, Jincheng An, Wei Liang, and Liehuang Zhu, *Senior Member, IEEE*

***Abstract*—Subsequently, by executing various attacks on benchmark datasets such as MNIST, we construct Federated Learning Malicious Parameter Identification (FLMPID) dataset to enable malicious client detection. Building on this dataset, we propose FORTRESS (Federated POisoning-Resistance Defense via Dual-Domain Distance and TRust AssESSment), a framework designed to detect and mitigate malicious updates from clients. FORTRESS employs a unique encoder-decoder architecture. The encoder utilizes dual-domain distance metrics on weights and gradients to extract hidden representations, while the decoder leverages Actor-Critic (AC) reinforcement learning for trust assessment. We evaluated FORTRESS under multiple attack scenarios and demonstrated its defense effectiveness, making it a promising solution for enhancing the security of FL systems.**

***Index Terms*—Federated learning, poisoning attacks, anomaly detection, reinforcement learning, cosine-constrained substitution attack.**

I. Introduction

FEDERATED Learning (FL) is a distributed collaborative framework that addresses data decentralization by enabling participants to retain their personal data during model training. Unlike traditional centralized machine learning, FL mitigates the "Data Silos" issue, particularly in industries reluctant to share sensitive information. With regulations such as the General Data Protection Regulation (GDPR) in 2016 and the Artificial Intelligence Act (AI Act) in 2024, protecting user privacy has become critical. Introduced by Google as a privacy-enhancing solution [1], FL allows clients to send model parameters or gradients to a central server rather than sharing raw data. This approach has been widely adopted in sectors like medicine, finance, and social media to collaboratively train models while safeguarding data privacy [2]. However, the decentralized nature of FL exposes it to various attacks. Malicious participants can manipulate local models by altering parameters or modifying personal data, leading to model poisoning or data poisoning attacks [3]. These attacks degrade global model accuracy and may embed backdoors—malicious patterns that trigger incorrect outputs under specific conditions. Unlike general poisoning attacks, backdoors allow normal model behavior in most scenarios but trigger malicious outputs under specific conditions, compromising reliability.

To defend against such attacks, various Byzantine-robust frameworks have been developed. Early methods like GeoMedian [4] reduce outlier influence but are vulnerable to median shift attacks [5]. Approaches such as Krum [6] and Trim-Mean [7] exclude malicious clients but require prior knowledge of their number and are susceptible to adversarial strategies. FoolsGold [8] downweights clients exhibiting highly similar updates, assuming colluding attackers produce correlated updates, but struggles when attackers introduce diversity. Recent approaches, such as FLTrust [9], assign trust scores using cosine similarity with a trusted server. Clustering-based methods, including FLAME [10], FreqFed [11], and DFLDual [12], rely on the assumption that malicious clients comprise less than half of the population, which reduces their effectiveness in scenarios with a higher proportion of adversaries.

In this study, we first identify a novel attack that bypasses FLTrust [9], the current state-of-the-art (SOTA) defense mechanism in FL. This attack works by crafting malicious substitute models with high cosine similarity to benign local models, thereby circumventing FLTrust's defense mechanism and degrading the global model. To bridge this gap, we create a dataset for simulating poisoning attacks in FL, constructed using MNIST [13], Fashion-MNIST (F-MNIST) [14], and CIFAR10 [15] under a range of attack scenarios. This dataset includes both malicious and benign participant updates, enabling research into distinguishing between them, simplifying anomaly detection into a classification problem,

Received 16 January 2025; revised 30 May 2025 and 3 July 2025; accepted 9 July 2025. Date of publication 15 July 2025; date of current version 21 July 2025. This work was supported in part by the National Natural Science Foundation of China under Grant U2468205 and Grant 62372149, in part by China Scholarship Council (CSC), and in part by the Key Laboratory of Knowledge Engineering with Big Data (the Ministry of Education of China), under Grant BigKEOpen2025-04. The associate editor coordinating the review of this article and approving it for publication was Dr. Daisuke Mashima. *(Corresponding authors: Zijian Zhang; Meng Li.)*

Zhaoqi Wang, Zijian Zhang, Zhen Li, Yan Wu, and Liehuang Zhu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: wang_zhaoqi@bit.edu.cn; zhangzijian@bit.edu.cn; zhen.li@bit.edu.cn; wuyan.bit@gmail.com; liehuangz@bit.edu.cn).

Ye Liu is with the School of Computing and Information Systems, Singapore Management University, Singapore 188065 (e-mail: yeliu@smu.edu.sg).

Meng Li is with the Key Laboratory of Knowledge Engineering with Big Data, Ministry of Education, the School of Computer Science and Information Engineering, and the Intelligent Interconnected Systems Laboratory of Anhui Province, Hefei University of Technology, Hefei 230002, China, and also with the HIT Center, University of Padua, 35122 Padua, Italy (e-mail: mengli@hfut.edu.cn).

Xin Li is with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: xinli@bit.edu.cn).

Yong Liu is with Qi An Xin Technology Group Inc., Beijing 100000, China, and also with the Zhongguancun Laboratory, Beijing 100000, China (e-mail: liuyong03@qianxin.com).

Jincheng An is with the QAX Security Center, Qi An Xin Technology Group Inc., Beijing 100000, China (e-mail: anjincheng@qianxin.com).

Wei Liang is with the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China (e-mail: wliang@hnust.edu.cn).

Digital Object Identifier 10.1109/TIFS.2025.3589061

and providing a foundation for evaluating defense methods. Building on this dataset, we propose FORTRESS, a robust framework that combines anomaly detection with adaptive aggregation to defend against diverse attacks. FORTRESS identifies malicious updates by comparing client-uploaded weights and gradients per round through a dual-distance approach. Based on this, it assigns trust scores for aggregation to enhance global model robustness. Unlike existing methods, FORTRESS requires no prior knowledge of malicious participants or attack strategies, adapting to a wide range of scenarios.

We evaluate our defense strategy under multiple attack settings, including our CCS attack, model poisoning, and data poisoning attacks. To simulate real-world scenarios, we conduct experiments in non-independent and identically distributed (non-IID) environments. Experimental results demonstrate that FORTRESS outperforms other defense methods in both accuracy and robustness across most cases, showcasing its adaptability to different attacks and varying proportions of malicious clients. For instance, under the CCS attack on the Fashion-MNIST dataset with 60% clients be malicious, FORTRESS achieves an accuracy of 74.29%, higher than FLTrust (60.3%) and FedAvg (10.18%). We summarize our main contributions as follows:

- We propose an attack targeting cosine similarity-based defense mechanisms, revealing the inherent vulnerabilities of such approaches.
- We propose a novel defense mechanism capable of defending against model and data poisoning attacks in non-IID environments, mitigating attacks without assuming prior knowledge of malicious participant counts or attack strategies.
- We conduct comprehensive experiments with formal robustness analysis to evaluate our method, demonstrating its effectiveness and resilience against threats.

Page Organization The remainder of this paper is organized as follows. Section II reviews related work on attacks and defenses in FL. Section III briefly reviews some preliminaries. Section IV formulates the problem, detailing the system model, threat model, and design objectives. Section V presents the Cosine-Constrained Substitution (CCS) Attack and provides theoretical analysis. Section VI introduces the FORTRESS defense framework. Section VII evaluates the performance of FORTRESS. Section VIII concludes this paper.

II. Related Work

Federated Learning (FL) enables multiple clients to collaboratively train a global model without exchanging private datasets, addressing privacy concerns in distributed machine learning. In a centralized FL setup, a server coordinates the training process by broadcasting the global model to clients and aggregating their updates to refine the model. In contrast, decentralized FL allows clients to share and aggregate model updates directly with one another, achieving convergence to a global model through iterative communication. However, the challenges of non-IID data distributions across clients and potential adversarial behaviors in real-world scenarios make FL vulnerable to various attacks. [1].

One prominent threat in FL is Poisoning Attacks, which can be broadly categorized into two categories: **data poisoning** that manipulates training samples (annotated as (1) in Fig. 1), and **model poisoning** that directly alters gradient updates or local model weights (denoted as (2) in Fig. 1). During local training phases, malicious clients strategically deploy these attacks: data poisoning corrupts training data, while model poisoning injects biased parameters. Crucially, both benign and poisoned local models are aggregated without discrimination, leading to global model contamination. This poisoned global model is then broadcasted to all participants in the subsequent training round, creating a attack cycle where the poisoning effects propagate and amplify across iterations.

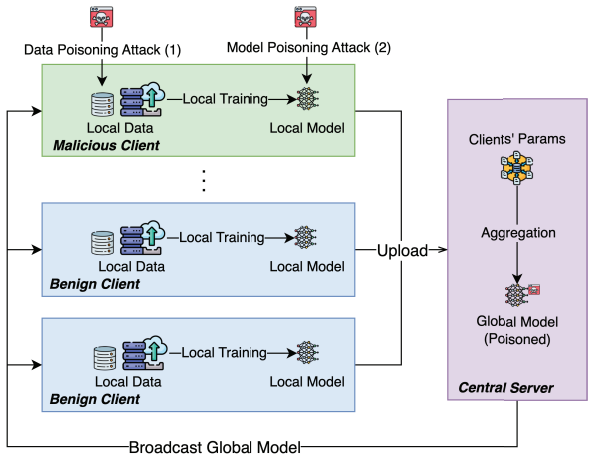


Fig. 1. An illustration of poisoning attacks in federated learning.

Data poisoning involves malicious clients manipulating their local training data through methods such as label alteration through systematic misclassification protocols (e.g., *label flipping* via predefined class permutation matrices) or injection of adversarially perturbations that preserve sample perceptibility, ultimately degrading the global model's accuracy or inducing non-convergent training dynamics [18], [19]. A particularly insidious form is the *backdoor attack*, where hidden triggers are implanted in the model to produce adversary-specified outputs when activated while maintaining normal performance on clean data. This stealthy and targeted nature renders backdoor attacks significantly more challenging to detect and mitigate compared to untargeted poisoning attacks. Advanced backdoor techniques enhance stealth through malicious-benign update mixing or distributed trigger deployment, including: *blending backdoors* [20] train separate clean and backdoored models, then create a blended model through weighted parameter averaging; *constrain-and-scale backdoors* [21] optimize malicious models by simultaneously minimizing task-specific loss and restricting parameter deviations from benign models through p-norm penalty terms in the objective function; *distributed backdoors* [22] allocate distinct trigger fragments to colluding clients, which collectively assemble into a complete trigger pattern through model aggregation. These implementations retain core attack mechanisms but simplify synchronization protocols, which reduces detectability during aggregation.

Model poisoning attacks manipulate gradients or model updates through geometric transformations in parameter space. *Gradient Ascent Attacks* [23], for instance, deliberately reverse gradient directions through sign inversion operators applied during local optimization steps to maximize loss function curvature. Countermeasures include robust aggregation methods such as: *Geo-Median* [4] utilizing geometric medians to reduce outlier impacts; the *Krum* algorithm [6] selecting updates

with minimal squared distance sums; and the *Trimmed-Mean* method [7] excluding extreme updates before averaging. While effective against basic attacks, these methods falter against sophisticated strategies due to their dependency on accurate estimations of malicious participant counts and behaviors. The *p-Norm Attack* [24] perturbs gradients to manipulate p-norm-based aggregation rules, ensuring Byzantine gradient selection during aggregation. Accompanying defense method Bulyan combines Krum with trimmed mean variants. *Median Shift* and *Krum Median Shift* Attacks [5] craft malicious updates using Gaussian-distributed parameters. The mean is maximally shifted relative to benign parameters' standard deviations, preserving original variance to subtly distort aggregation. These attacks underscore existing defenses' limitations in addressing advanced adversarial manipulations.

More recent approaches have introduced novel mechanisms to enhance robustness. *FLTrust* [9] pioneered a groundbreaking trust-weighted aggregation framework through its innovative dual mechanisms: (1) cosine similarity measurement between client updates and server-computed reference gradients, and (2) dynamic aggregation weight assignment based on trust scores. This seminal work established critical foundations for Byzantine-robust FL by addressing some poisoning attacks mentioned before through its trust-anchored paradigm. However, FLTrust relies on the server maintaining a small-scale but class-complete dataset which makes it less suitable for real non-IID scenarios. *FLAME* [10] counters backdoor attacks through clustering-based filtering and noise injection, identifying poisoned models as outliers. Similarly, *DFLDual* [12] leverages dual-domain client clustering and trust bootstrapping to exclude malicious clients. However, these methods face limitations: FLAME primarily targets backdoor attacks, while FLTrust and DFLDual's reliance on cosine similarity leaves it vulnerable to carefully designed perturbations.

Despite advances, existing defenses still face challenges in balancing robustness, scalability, and performance in the presence of sophisticated adversarial behaviors. This highlights the importance of continued research to develop more effective and adaptive solutions for securing Federated Learning systems.

III. Preliminaries

This section provides an overview of the key concepts and techniques used in this paper.

A. Federated Learning

Federated Learning (FL) aims to solve the following optimization problem:

$$w^* = \arg\min_w F(w), \quad F(w) = \mathbb{E}_{D\sim X}[f(D,w)]. \tag{1}$$

Here, w represents the global model parameters, D is the training data, and $f(D,w)$ is the loss function. The goal of FL is to minimize the global loss function $F(w)$ by aggregating local updates from multiple clients without sharing raw data.

B. Actor-Critic Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning where an agent learns to make decisions by interacting with an environment. Unlike supervised learning, RL can handle out-of-distribution (OOD) data by exploring unseen states and adapting its policy. Actor-Critic methods [25], a class of RL algorithms, combine two components: the actor, which updates the policy, and the critic, which evaluates actions using value functions. This method leverage Temporal Difference (TD) error [26], with the TD target defined as:

$$Q_{\text{target}} = r + \gamma \cdot Q_{\text{next}} \cdot (1 - \mathbb{I}(\text{done})). \tag{2}$$

Here, r is the immediate reward, γ the discount factor, Q_{next} the next state's Q-value, and $\mathbb{I}(\text{done})$ an indicator for episode termination. The TD error, as the difference between the target and current Q-value, stabilizes updates and accelerates convergence. Actor-Critic methods are particularly effective in environments with continuous state and action spaces.

C. Dirichlet Distribution

In federated learning environments, the Dirichlet distribution is widely used to simulate real-world non-IID data distributions [27]. This approach effectively models the data heterogeneity among different clients, thereby more accurately reflecting the data distribution characteristics encountered in practical applications.

The Dirichlet distribution is parameterized by a vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_K)$, where each $\alpha_i > 0$. Its probability density function is defined as:

$$f(x_1,\ldots,x_K;\alpha_1,\ldots,\alpha_K) = \frac{\Gamma\left(\sum_{i=1}^K \alpha_i\right)}{\prod_{i=1}^K \Gamma(\alpha_i)} \prod_{i=1}^K x_i^{\alpha_i - 1}, \tag{3}$$

where the variables $x_1, \ldots, x_K$ must satisfy:

$$\sum_{i=1}^K x_i = 1, \quad x_i \geq 0 \quad \forall i. \tag{4}$$

Specifically, by adjusting the concentration parameter α of the Dirichlet distribution, researchers can simulate different levels of data heterogeneity across clients, from iid ($\alpha \to \infty$) to extreme non-iid ($\alpha \to 0$) scenarios, enabling systematic evaluation of federated learning algorithms.

D. Common Distance Metrics

Most poisoning attack detection methods rely on distance measures to identify malicious behaviors. In this section, we introduce several common distance metrics that will be used in our subsequent analysis.

1) Cosine Similarity: A measure of similarity between two vectors based on the cosine of the angle between them:

$$\cos(\theta) = \frac{\mathbf{v_1} \cdot \mathbf{v_2}}{|\mathbf{v_1}||\mathbf{v_2}|}, \tag{5}$$

where $\mathbf{v_1}, \mathbf{v_2} \in \mathbb{R}^n$ denote the input vectors. The result ranges from -1 to 1, where 1 means the vectors are identical in direction, 0 means they are orthogonal, and -1 means they point in opposite directions.

2) Wasserstein Distance: : A metric that quantifies the distance between two probability distributions by measuring the minimum cost of transforming one distribution into another. For probability distributions P and Q with cumulative distribution functions F_P and F_Q respectively, the 1st Wasserstein distance is defined as:

$$W_1(P,Q) = \int_{-\infty}^{\infty} |F_P(x) - F_Q(x)| dx. \tag{6}$$

This formulation is particularly effective for capturing distributional shifts in continuous spaces.

3) Entropy Distance: Based on information entropy, this metric quantifies the difference in information content by calculating the absolute difference of their entropy values:

$$Entropy(\mathbf{v_1}, \mathbf{v_2}) = |H(\mathbf{v_1}) - H(\mathbf{v_2})|, \tag{7}$$

$$H(\mathbf{v}) = -\sum_i p_i \log(p_i). \tag{8}$$

Here, $H(\mathbf{v})$ represents the entropy of vector $\mathbf{v}$, and p_i denotes the probability of each unique value in the vector.

4) Norm Distance: (L_p): Measures the difference between two normalized vectors using various p-norms, each with its own sensitivity to deviations:

- $L_{0.5}$: Sensitive to small but widespread deviations, which are common in malicious updates [28].
- L_1 (Manhattan Distance): Captures the overall magnitude of deviations by summing the absolute differences between corresponding elements.
- L_2 (Euclidean Distance): Emphasizes larger deviations while downplaying smaller ones by computing the square root of the sum of squared differences.
- L_∞ (Chebyshev Distance): Detects extreme outliers by identifying the maximum absolute difference between corresponding elements.

E. Trust Assessment

The trust assessment mechanism was first proposed in FLTrust [9], where the trust score for each client is updated dynamically in each round. The trust score update rule is:

$$TS_i^t = TS_i^{t-1} + \max(0, \Delta TS_i^t), \tag{9}$$

where ΔTS_i^t is the trust score change of client i in round t. The non-negative constraint ensures that trust scores do not decrease. The trust score TS_i^t is then used to compute the aggregation weight α_i^t for global model updates:

$$\mathbf{w}^t = \mathbf{w}^{t-1} + \sum_i \alpha_i^t \cdot \Delta \mathbf{w}_i^t, \quad \alpha_i^t = \frac{TS_i^t}{\sum_j TS_j^t}. \tag{10}$$

IV. Problem Formulation

In this section, we first present the system model and threat model for federated learning. We then introduce the goals of the defense mechanism.

A. System Model

We formalize a federated learning ecosystem comprising N heterogeneous clients and a central aggregator. Each client i maintains a private dataset $\mathcal{D}_i$ and executes local model training through iterative interactions with the server. The server's primary function involves reconciling these client-submitted weights $\{\theta_i^{(t)}\}_{i=1}^N$ through aggregation protocols to produce the global model $\theta^{(t)}$. Clients exhibit two behavioral types: benign participants faithfully execute the learning protocol by transmitting genuine parameters, whereas malicious actors strategically manipulate their $\theta_i^{(t)}$ submissions to subvert model convergence.

B. Threat Model

In this paper, we follow the attack model in previous works [9], [12]. Specifically, we assume that malicious clients manipulate their parameters to achieve specific adversarial goals. The percentage of attackers among all participants is not specified, with benign clients representing the majority of data distribution. The attack method may be collaborative or individual. Typically, attackers possess basic knowledge similar to all participants, such as information about the global model. Additionally, malicious clients may have access to other participants' parameters to implement certain attacks. They are aware of the aggregation rules but do not know the specific parameters of the defense methods. Attackers can manipulate their personal data or model parameters to achieve specific goals. Attackers may choose any established attack strategy, though unaware of defense specifics.

C. Defender's Capability and Knowledge

As the protocol enforcer, the aggregator maintains full visibility into the weight vectors $\theta_i^{(t)}$ directly submitted by clients. The server computes gradient updates: $\Delta\theta_i^{(t)} = \theta_i^{(t)} - \theta^{(t-1)}$ by comparing current submissions to $\theta^{(t-1)}$. This computation enables continuous monitoring of parameter dynamics across rounds. The defense infrastructure incorporates a diagnostic module powered by a virtual client with verification dataset $\mathcal{D}_v$, which facilitates comparative assessment of model behaviors. However, three fundamental constraints govern the defender's operational scope: absence of prior knowledge about the instantaneous number of malicious clients $M^{(t)}$, blindness to specific attack vectors $\mathcal{A}^{(t)}$ employed by adversaries, and potential distributional divergence between $\mathcal{D}_v$ and the union of client data $\bigcup_{i=1}^N \mathcal{D}_i$ that limits diagnostic accuracy.

D. Design Objectives

The defense framework is designed with the following key objectives:

- Effectiveness: Capable of effectively counteracting data poisoning and model poisoning attacks, ensuring the defense mechanism remains robust across varying numbers of malicious clients and different types of attack methods.
- Fidelity: Ensures that the defense mechanism does not significantly impact the convergence or accuracy of the global model in the absence of malicious clients.
- Adaptability: The defense mechanism can adapt to emerging attack methods and remain effective in complex attack scenarios without compromising its robustness.

V. The Cosine-Constrained Substitution Attack

As previously discussed, cosine similarity-based defense mechanisms (e.g., FLTrust [9] and DFLDual [12]) and p-norm distance-based methods (e.g., Geo-Median [4], Krum [6], Bulyan [24]) constitute current primary defense paradigms in federated learning. Our proposed Cosine-Constrained Substitution (CCS) Attack specifically targets these defense frameworks by constructing poisoned model weights during the training phase. The attack objectives are twofold: (1) to degrade model convergence speed or prevent global model convergence entirely, and (2) to maintain evasion from detection mechanisms through carefully crafted perturbations.

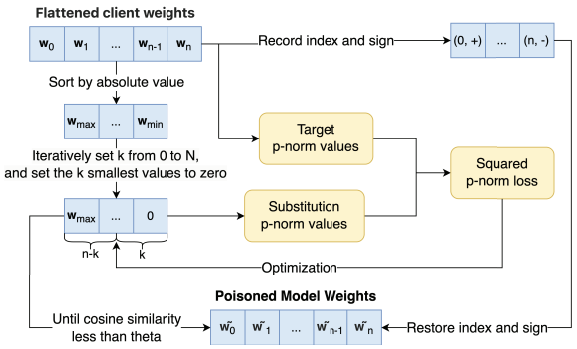


Fig. 2. Overview of the CCS attack.

To operationalize this attack, Fig. 2 illustrates the dual-constrained optimization process. Specifically, it generates substitute vectors $\mathbf{v}_2$ through a dual-constrained optimization process. These vectors preserve a cosine similarity above threshold θ with benign vectors $\mathbf{v}_1$ while adhering to specific ℓ_p norm constraints. This strategic design allows the attack to circumvent common defense mechanisms precisely by exploiting their reliance on directional alignment (via cosine similarity) and magnitude consistency (via p-norm comparisons). Compared to conventional model poisoning attacks, CCS demonstrates stealth through parameter-space constrained modifications: iteratively (1) selective zeroing of k parameters with smallest absolute values, and (2) addition of minimal perturbations to remaining params for maintaining p-norm distances, until the cosine similarity falls below threshold θ. This approach preserves statistical characteristics critical for defense mechanisms while introducing attacks. The attack's effectiveness stems from its exploitation of fundamental limitations in current defenses, as identified in experiments described later.

A. Specification of CCS Attack

As the Algorithm 1 shows, the CCS attack is implemented by constructing a substitute vector $\mathbf{v}_2 \in \mathbb{R}^n$ that maximizes the number of zero elements while maintaining a minimum cosine similarity threshold. Here, $\mathbf{v}_1 \in \mathbb{R}^n$ represents the original vector obtained by flattening the model parameters into a feature vector, and n denotes the dimension n of the vector. The attack designs $\mathbf{v}_2$ to maintain cosine similarity above θ with $\mathbf{v}_1$, while satisfying specific norm constraints such as ℓ_1, ℓ_2, or other norms. The algorithm automatically determines the maximum number of elements k that can be set to zero while meeting these constraints. After optimization, we reshape $\mathbf{v}_2$ to the original model structure and upload it.

This method effectively determines the maximum number of elements that can be zeroed out while still maintaining the required similarity threshold and matching the target norms. By retaining the largest elements and adjusting their values minimally, the direction of $\mathbf{v}_2$ remains sufficiently close to that of $\mathbf{v}1$. At the same time, the ℓp_i norm constraints are satisfied, enabling the substitute vector $\mathbf{v}_2$ to bypass defenses based on these norms. This adaptive approach ensures we achieve the highest possible sparsity while still maintaining the effectiveness of the attack.

B. Theoretical Analysis

Theorem 1: Consider following optimization problem:

$$\min_{\tilde{\mathbf{v}}} \quad f(\tilde{\mathbf{v}}) = \sum_i \left(\|\tilde{\mathbf{v}}\|_{p_i} - d_i\right)^2 \tag{11}$$

Algorithm 1 Cosine-Constrained Substitution Attack

Require: Original vector $v_1 \in \mathbb{R}^n$, cosine similarity threshold θ, desired norms $\{d_i\}$, learning rate η, maximum epochs E
Ensure: Substitute vector $v_2 \in \mathbb{R}^n$

$k \leftarrow 0$
$s \leftarrow \text{sign}(v_1)$
$v_1^{\text{abs}} \leftarrow |v_1|$
$v_1^{\text{sorted}} \leftarrow \text{sort}(v_1^{\text{abs}}, \text{descending})$
while $k \leq n$ **do**
  $\tilde{v} \leftarrow v_1^{\text{sorted}}[: n-k]$
  Initialize $\tilde{v}_{\text{opt}} \leftarrow \tilde{v}$
  for *epoch* $= 1$ to E **do**
    loss $\leftarrow \sum_i (\|\tilde{v}_{\text{opt}}\|_{p_i} - d_i)^2$
    $\tilde{v}_{\text{opt}} \leftarrow \tilde{v}_{\text{opt}} - \eta \cdot \nabla(\text{loss})$
    $\tilde{v}_{\text{opt}} \leftarrow \max(\tilde{v}_{\text{opt}}, 0)$
  end for
  $v_2 \leftarrow \text{zeros}(n)$
  Map $\tilde{v}_{\text{opt}}$ back to original positions in v_2
  $v_2 \leftarrow v_2 \cdot s$
  $cos \leftarrow \frac{v_1 \cdot v_2}{\|v_1\|\|v_2\|}$
  if $cos < \theta$ **then**
    Return the last valid k and its corresponding substitute v_2
  end if
  $k \leftarrow k + 1$
end while
return: v_1

$$\text{subject to} \quad \tilde{\mathbf{v}} \geq \mathbf{0}, \tag{12}$$

where $p_i \geq 1$ and d_i are the desired norm values. Then, this optimization problem is solvable in the sense that a local minimum exists, which is sufficient for our purposes.

Proof: First, note that the objective function $f(\tilde{\mathbf{v}})$ is continuously differentiable with respect to $\tilde{\mathbf{v}}$. Although the function may be non-convex, standard gradient-based optimization methods can be employed to find a local minimum.

We compute the gradient of the objective function with respect to $\tilde{\mathbf{v}}$:

$$\nabla_{\tilde{\mathbf{v}}} f(\tilde{\mathbf{v}}) = 2 \sum_i \left(\|\tilde{\mathbf{v}}\|_{p_i} - d_i\right) \nabla_{\tilde{\mathbf{v}}} \|\tilde{\mathbf{v}}\|_{p_i}. \tag{13}$$

To find $\nabla_{\tilde{\mathbf{v}}} \|\tilde{\mathbf{v}}\|_{p_i}$, we proceed as follows. Let $S = \sum_{j=1}^{n-k} \tilde{v}_j^{p_i}$, so that $\|\tilde{\mathbf{v}}\|_{p_i} = S^{1/p_i}$. Then,

$$\nabla_{\tilde{\mathbf{v}}} \|\tilde{\mathbf{v}}\|_{p_i} = \frac{\partial}{\partial \tilde{\mathbf{v}}} \left(S^{1/p_i}\right) \tag{14}$$

$$= \frac{1}{p_i} S^{\frac{1}{p_i}-1} \nabla_{\tilde{\mathbf{v}}} S \tag{15}$$

$$= \frac{1}{p_i} S^{\frac{1}{p_i}-1} \left(p_i \tilde{v}_1^{p_i-1}, \ \ldots, \ p_i \tilde{v}_{n-k}^{p_i-1}\right)^\top \tag{16}$$

$$= S^{\frac{1}{p_i}-1} \left(\tilde{v}_1^{p_i-1}, \ \ldots, \ \tilde{v}_{n-k}^{p_i-1}\right)^\top. \tag{17}$$

Therefore, the gradient of the objective function is:

$$\nabla_{\tilde{\mathbf{v}}} f(\tilde{\mathbf{v}}) = 2 \sum_i \left(\|\tilde{\mathbf{v}}\|_{p_i} - d_i\right) S^{\frac{1}{p_i}-1} \left(\tilde{v}_1^{p_i-1}, \ \ldots, \ \tilde{v}_{n-k}^{p_i-1}\right)^\top. \tag{18}$$

Given that $\tilde{\mathbf{v}} \geq \mathbf{0}$ and the initial values are positive, we can use gradient descent methods to iteratively update $\tilde{\mathbf{v}}$. After

each iteration, we project any negative components back to zero to satisfy the constraint $\tilde{\mathbf{v}} \geq \mathbf{0}$.

While the optimization problem may be non-convex, a local minimum is sufficient for our application. In practice, gradient-based optimization algorithms effectively identify such solutions, allowing us to construct a substitute vector $\tilde{\mathbf{v}}$ that closely satisfies the desired norm constraints.

Theorem 2: Let $\mathbf{v} \in \mathbb{R}^n$ be a non-zero vector, and let $\theta \in (0, 1)$ be a cosine similarity threshold. Suppose we sort the components of $\mathbf{v}$ in descending order of their absolute values and construct a sequence of vectors $\mathbf{v}'^{(k)}$ by zeroing the smallest k components and optimizing the remaining $(n-k)$ components as described in Theorem 1. Then, the cosine similarity $\cos\left(\mathbf{v}, \mathbf{v}'^{(k)}\right)$ decreases monotonically as k increases. Consequently, for all $k > k_{\max}$, we have a maximal integer $k_{\max}$ such that:

$$\cos\left(\mathbf{v}, \mathbf{v}'^{(k)}\right) \begin{cases} \geq \theta, & \text{if } k = k_{\max}, \\ < \theta, & \text{if } k \neq k_{\max}. \end{cases} \tag{19}$$

Proof: We aim to show that as k increases, $\cos\left(\mathbf{v}, \mathbf{v}'^{(k)}\right)$ decreases.

First, express the cosine similarity explicitly:

$$\cos\left(\mathbf{v}, \mathbf{v}'^{(k)}\right) = \frac{\sum_{i=1}^{n} v_i v_i'^{(k)}}{\|\mathbf{v}\|_2 \|\mathbf{v}'^{(k)}\|_2}. \tag{20}$$

Since $v_{(i)}'^{(k)} = 0$ for $i > n-k$, the numerator and denominator becomes:

$$\sum_{i=1}^{n} v_i v_i'^{(k)} = \sum_{i=1}^{n-k} v_{(i)} v_{(i)}'^{(k)}, \tag{21}$$

$$\|\mathbf{v}'^{(k)}\|_2 = \left(\sum_{i=1}^{n-k} \left(v_{(i)}'^{(k)}\right)^2\right)^{1/2}. \tag{22}$$

As k increases, the number of non-zero terms $n-k$ decreases, so both the numerator and denominator are sums over fewer terms. For the numerator, consider the difference between k and $k+1$. Define:

$$N(k) = \sum_{i=1}^{n-k} v_{(i)} v_{(i)}'^{(k)}, \quad N(k+1) = \sum_{i=1}^{n-k-1} v_{(i)} v_{(i)}'^{(k+1)}. \tag{23}$$

The difference between the two is:

$$N(k) - N(k+1) = v_{(n-k)} v_{(n-k)}'^{(k)}. \tag{24}$$

Since $v_{(n-k)}'^{(k)} \geq 0$ and $v_{(n-k)}$ share the same sign due to the reconstruction step, the product $v_{(n-k)} v_{(n-k)}'^{(k)} \geq 0$. Therefore: $N(k+1) \leq N(k)$.

For the denominator, we similarly define:

$$D(k) = \|\mathbf{v}'^{(k)}\|_2 = \left(\sum_{i=1}^{n-k} \left(v_{(i)}'^{(k)}\right)^2\right)^{1/2}, \tag{25}$$

Since $v_{(n-k)}'^{(k+1)} = 0$, substituting $k+1$ into the definition of $D(k)$ gives:

$$D(k+1) \leq D(k). \tag{26}$$

Thus the cosine similarity at k is:

$$\cos\left(\mathbf{v}, \mathbf{v}'^{(k)}\right) = \frac{N(k)}{\|\mathbf{v}\|_2 D(k)}. \tag{27}$$

TABLE I
KEY NOTATIONS

Notation	Description	Notation	Description
D_{train}	Task dataset	D_w	Wasserstein distance
D_d	FLMPID dataset	D_e	Entropy distance
D_i	Dataset for client i	μ	Normalization factor
D_r	Server's root dataset	η	Scaling factor
R_l	Number of iterations	f	Distance features
N	Number of clients	$Encoder$	Encoder model
α	Dirichlet parameter	$Decoder$	Decoder model
R_g	Number of FL rounds	TS	Cumulative scores
A	Attack method	ΔTS	Trust score change
S	Number of attack	ϵ	Exploration rate
t	Current iteration index	T	Temperature
w	Global weight	λ	Distillation weight
w_i	Client weight	w_r	Virtual client weight
Δw_i	Client update	Δw_r	Virtual client update

By substituting $k+1$ into the definition, following from the Cauchy-Schwarz inequality, we have:

$$N(k+1) = \langle \mathbf{v}_{(n-k-1)}, \mathbf{v}'^{(k+1)} \rangle \tag{28}$$

$$\leq \|\mathbf{v}_{(n-k-1)}\|_2 \|\mathbf{v}'^{(k+1)}\|_2. \tag{29}$$

As one component of $\mathbf{v}'^{(k)}$ is zeroed out, we have $\|\mathbf{v}_{(n-k-1)}\|_2 \leq \|\mathbf{v}_{(n-k)}\|_2$ and $\|\mathbf{v}'^{(k+1)}\|_2 \leq \|\mathbf{v}'^{(k)}\|_2$, which implies:

$$\frac{N(k+1)}{D(k+1)} \leq \frac{N(k)}{D(k)}. \tag{30}$$

Thus, the cosine similarity satisfies:

$$\cos\left(\mathbf{v}, \mathbf{v}'^{(k+1)}\right) \leq \cos\left(\mathbf{v}, \mathbf{v}'^{(k)}\right). \tag{31}$$

Combining these observations, we conclude that $\cos\left(\mathbf{v}, \mathbf{v}'^{(k)}\right)$ decreases as k increases. Since $\cos\left(\mathbf{v}, \mathbf{v}'^{(0)}\right)$ is maximal and decreases with k, there exists a maximal integer $k_{\max}$ that satisfies (19).

VI. THE FORTRESS

In this section, we first introduce the FLMPID dataset, which is designed to facilitate the training and evaluation of defense mechanisms against malicious participants in federated learning. We then present the overview of the FORTRESS framework and details of the proposed defense mechanism. The main notations are shown in Table. I.

A. Introduction of the FLMPID Dataset

To simplify the detection of malicious clients as a classification problem, we introduce the FLMPID dataset. It is constructed based on the original FL task datasets (see details in the Experimental Setup section), dividing the training data into N parts via a Dirichlet distribution to reflect realistic non-IID data among participants. Using the standard FedAvg method, we simulate malicious clients manipulating their transmitted parameters. After several iterations, we collect global and client model parameters, including weights and gradients per round, to construct the FLMPID dataset. By leveraging the FLMPID dataset, we extract distance features from benign samples and various malicious attack patterns and integrate them into a multi-labeled training dataset (where data with different attack/normal labels are combined for training). This enables the pre-training of FORTRESS models and the application of transfer learning to defend against poisoning attacks in real-world federated learning scenarios.

Algorithm 2 FLMPID Dataset Generation

Require: Training dataset D_{train}; Number of clients N; Dirichlet parameter α; FL rounds R_g; attack methods A_{types}; Number of attack simulations S
Ensure: FLMPID dataset D_d

Initialize empty dataset $D_d = \{\}$
$\{D_1, \ldots, D_n\} \leftarrow \text{DirichletSplit}(D_{train}, N, \alpha)$
for a in A_{types} **do**
  for $i = 1$ to S **do**
    $w^0 \leftarrow$ random initialization
    for $t = 1$ to R_g **do**
      // Select benign and malicious clients
      $C_b, C_m \leftarrow \text{SelectClients}(N)$
      for j in C_b **do**
        $w_j^t \leftarrow \text{LocalTraining}(w^{t-1}, D_j, R_l)$
        $\Delta w_j^t = w_j^t - w^{t-1}$
      end for
      for j in C_m **do**
        $w_j^t \leftarrow \text{SimulateAttack}(w^{t-1}, D_j, R_l, a)$
        $\Delta w_j^t = w_j^t - w^{t-1}$
      end for
      $f^t = CollectParams(\{\Delta w_j^t\})$
      $D_d = D_d \cup \{(f^t, a)\}$
      $w^t \leftarrow \text{FedAvg}(\{w_j^t\}_{j \in C_b \cup C_m})$
    end for
  end for
end for
return: D_d

B. Introduction of the FORTRESS Framework

As illustrated in Fig. 3, the FORTRESS framework operates through nine coordinated steps: The process (1) initiates by flattening client model weights and computing their deviations from virtual reference weights. These deviations feed into an encoder (2) that extracts multi-dimensional distance features, generating latent representations of client behavior patterns. A decoder then interprets these representations (3) to produce dynamic trust scores, which undergo temporal aggregation with historical metrics (4) forming cumulative trust values. During model aggregation (5), client updates are weighted proportionally to their cumulative trust scores before reconstructing the original architecture (6) for global deployment (7). In training phases (8), global model performance metrics (e.g., accuracy) generate optimization rewards that drive iterative refinements (9) of the decoder's actor-critic networks via gradient-based policy updates. The complete workflow is formalized in Algorithm 3.

1) Preparation and Training Phase: As mentioned earlier, the aggregation server requires a small, clean root dataset for comparison, similar to the approach used in the FLTrust defense [9]. However, we do not require this root dataset to represent the complete data distribution across all clients. We propose three methods for constructing this root dataset: (a) utilizing the central server's existing local dataset, originally intended for evaluating the accuracy of the aggregated global model; (b) requesting clients to provide a small, desensitized dataset, which can be generated using privacy-preserving techniques such as Differential Privacy (DP) [29], Generative Adversarial Networks (GANs) [30], or similar methods; (c) using a batch from the training client itself during local aggregation in decentralized FL.

Algorithm 3 FORTRESS Framework

Require: N clients with local training datasets D_i; a server with root dataset D_r; global iteration count R_g; local iteration count R_l; current iteration index t; client weight w_i; global weight w; encoder *Encoder*; decoder *Decoder*; cumulative trust scores TS; distance features f.
Ensure: Global weight w

$w^0 \leftarrow$ random initialization
Initialize cumulative trust scores $TS \leftarrow 0$
for $t = 1, 2, \cdots, R_g$ **do**
  // Step 1: Local Training Phase
  Server sends w^{t-1} to all clients
  for each client $i \in \{1, \cdots, N\}$ in parallel **do**
    $w_i^t \leftarrow \text{LocalTraining}(w^{t-1}, D_i, R_l)$
    Send $\{w_i^t, |D_i|, R_l\}$ to server
  end for
  $w_r^t \leftarrow \text{LocalTraining}(w^{t-1}, D_r, R_l)$
  // Step 2: Calculate Distance Features
  for each client $i \in \{1, \cdots, N\}$ in parallel **do**
    $\Delta w_i^t = \frac{R_l \cdot |D_i|}{\sum_{j=0}^{N} R_l \cdot |D_j|} \cdot (w_i^t - w^{t-1})$
  end for
  $\Delta w_r^t = \frac{R_l \cdot |D_r|}{\sum_{j=0}^{N} R_l \cdot |D_j|} \cdot (w_r^t - w^{t-1})$
  for each client $i \in \{1, \cdots, N\}$ **do**
    $f_{i,\text{gradients}}^t = CalculateDistances(\Delta w_i^t, \Delta w_r^t)$
    $f_{i,\text{weights}}^t = CalculateDistances(w_i^t, w_r^t)$
    $f_i^t \leftarrow [f_{i,\text{weights}}^t, f_{i,\text{gradients}}^t]$
  end for
  // Step 3: Calculate Trust Score and Aggregation
  $\Delta TS^t = Decoder(Encoder(f_{1:N}))$
  for each client $i \in \{1, \cdots, N\}$ **do**
    $TS_i^t = TS_i^{t-1} + \max(0, \Delta TS_i^t)$
  end for
  $\Delta w^t = \sum_{i=1}^{N} \frac{TS_i^t}{\sum_{j=1}^{N} TS_j^t} \Delta w_i^t$
  $w^t \leftarrow w^{t-1} + \Delta w^t$
end for
return: w

The following explanation assumes a centralized FL setting, where the trusted aggregation server orchestrates the training process and uses the root dataset for comparison. The server also holds pre-trained encoder and decoder models, which serve as the core of the entire FORTRESS framework. In each training round, all clients, including a virtual client hosted on the server, train their local models. After training, clients send their parameters to the server, including model weights, data size, iteration count, and updates (computed as the difference from the previous round's global weights). To reduce data transmission, updates are used as a substitute for gradients, as they are mathematically equivalent.

2) Distance Feature Calculation: To ensure fair comparison between clients with different workloads, we first normalize updates based on each client's data size and iteration count. We then compute 12 distance metrics separately for weights and gradients (represented as updates) between each client and the virtual client, using the latter as a baseline, and combine the two types of distance features. These metrics

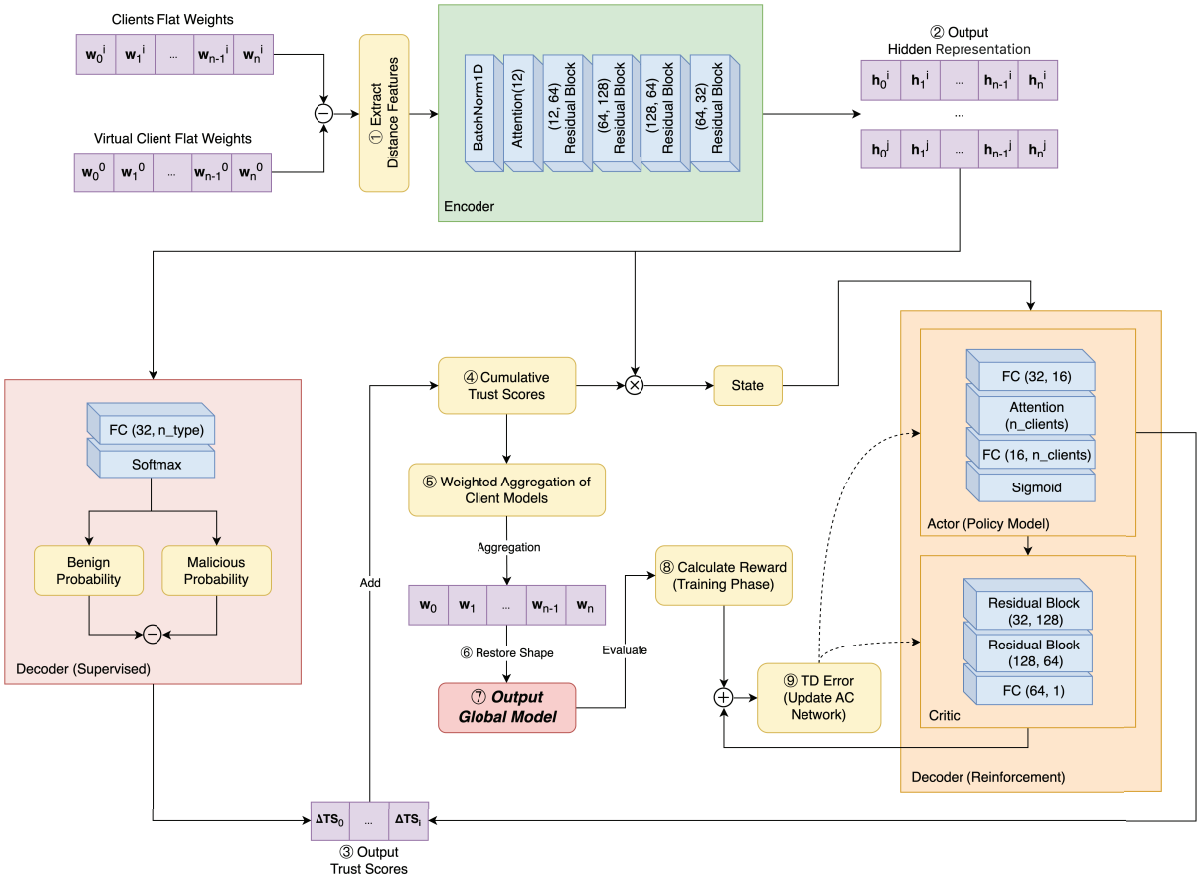


Fig. 3. the architectural overview of FORTRESS framework.

include **cosine dissimilarity**, **wasserstein distance**, **entropy distance**, various $\mathbf{L_p}$ **norms** ($L_{0.5}$, L_1, L_2, L_∞), and **statistical distances** (mean difference, median difference, standard deviation difference, min-max difference). This diverse set of metrics allows us to capture different aspects of potential malicious behavior: directional differences through cosine dissimilarity (i.e., the complement of cosine similarity), distributional shifts via wasserstein distance, unusual value patterns through entropy, magnitude variations through different norms, and overall statistical characteristics through statistical distances.

3) Extract Hidden Representations: As shown in Fig. 3, each round when server receives parameters from clients and calculates the distance features, the encoder generates hidden representation vectors ($\mathbf{h}_i$) for each client. These hidden representations are used to compute trust scores, which reflect the reliability of each client. The encoder consists of a *BatchNorm1d* layer for feature normalization, an *Attention* layer for feature weighting, and four *Residual Blocks* for extracting hidden representations. During the pre-training phase, we combine the encoder with a fully connected (FC) layer to output the client type and optimize the model using cross-entropy. When deployed, the FC layer is removed, and the encoder outputs the hidden representation. To enable the encoder to be effectively deployed in practical scenarios and to handle new attack methods, we adopt two strategies: fine-tuning [31] on a dataset simulated from attack scenarios and knowledge distillation [32] for adapting to new attack scenarios.

To enable the encoder to identify malicious participants from the parameter updates of all clients in different FL tasks, we fine-tune it as follows: First, the root dataset (D_r) is used to simulate attack scenarios and generate a fine-tuning dataset (D_f). This involves splitting D_r into benign data and malicious data, simulating weight updates, and calculating the distance features f between these updates. The resulting features, along with corresponding attack labels, form D_f. The pre-trained encoder is then fine-tuned using D_f. During this process, the feature extraction layers of the encoder are initialized with the pre-trained weights and remain frozen, while the remaining layers are updated. This refinement ensures that the encoder is better suited for practical deployment, enabling it to effectively distinguish malicious participants in diverse FL scenarios.

Knowledge distillation enhances the encoder's detection by transferring knowledge from a pre-trained teacher encoder to a student encoder. The process uses simulated attack data, where weight benign and malicious updates are generated and their distance features f are stored in the dataset D_d. The student encoder is trained with a total loss combining cross-entropy loss (L_{ce}) and knowledge distillation loss (L_{kd}), where L_{kd} is the KL divergence between the softened outputs of the teacher model and student model:

$$L_{kd} = \mathrm{KL}(\sigma(E_s(B)/T), \sigma(E_t(B)/T)), \tag{32}$$

and the total loss is:

$$L_{total} = (1-\lambda)L_{ce} + \lambda T^2 L_{kd}. \tag{33}$$

Here, $\sigma(x)$ represents the softmax function, T is the temperature parameter controlling the smoothness of the outputs, and λ balances the two loss terms. By minimizing L_{total}, the student encoder adapts to new attack scenarios while retaining detection capabilities.

4) Trust Score Calculation: To ensure computational effectiveness and robustness, we adopt a hybrid switching strategy between two mechanisms shown in Fig. 3. The server alternates between a **supervised mechanism** and a **reinforcement mechanism**. The supervised mechanism, operating during the 4 rounds of each 5-round cycle, is computationally efficient and performs well under normal conditions. It computes the trust score (ΔTS_i) for each client using hidden representations ($\mathbf{h}_i$), defined as the probability of the client being benign minus the maximum probability of it being malicious.

However, the supervised mechanism may struggle with persistent misclassification of certain benign clients due to their behavioral patterns. To address this limitation, we introduce the reinforcement mechanism every fifth round. This mechanism, based on an Actor-Critic reinforcement learning approach, is designed to provide additional insights and handle out-of-distributed client behaviors through exploration and long-term reward optimization. It explores alternative strategies and mitigates the biases of the baseline classifier, enabling adaptation to diverse attack methods. Additionally, the reinforcement mechanism aims to handle the complex scenarios where malicious clients may still contribute useful data. Instead of simply excluding these clients, it learns to assign small weights to their contributions, allowing the system to preserve potentially useful features while the larger-weighted contributions from benign clients help mitigate the impact of potential malicious manipulations.

By combining the strengths of both mechanisms, our hybrid approach enhances the robustness and performance of the federated learning system. It addresses challenges such as persistent classifier biases and unexpected client behaviors with greater flexibility and adaptability, achieving a balance between effectiveness and adaptability.

In the supervised mechanism, the decoder uses a *FC* layer and a *Softmax* layer to calculate ΔTS_i:

$$\Delta TS_i = \max(p_{\mathrm{b}} - \max(p_{\mathrm{m}}), 0). \tag{34}$$

Here, p_{b} is the probability of a benign update, and p_{m} represents probabilities of malicious updates. In the reinforcement mechanism, the decoder employs an actor-critic framework. The policy model calculates ΔTS_i using a *Linear* and *Attention* layer, while the critic model evaluates actions using *Residual Blocks* and a fully connected layer. During the pre-training phase, the critic model is trained to evaluate the quality of actions and assist in optimizing the policy model. When deployed, only the policy model is used for decision-making. The exploration rate ϵ is updated dynamically as:

$$\epsilon = \epsilon_{\mathrm{end}} + (\epsilon_{\mathrm{start}} - \epsilon_{\mathrm{end}}) \cdot \exp\left(-\frac{r}{R_g}\right), \tag{35}$$

where $\epsilon_{\mathrm{start}}$ and ϵ_{end} are hyperparameters, r is the current step, and R_g is the total rounds. The total reward R_{final} combines accuracy reward R_{accu} and action reward R_{action}, scaled to $[-1, 1]$. Policy and critic losses are used to optimize the policy and critic models. During inference, the policy model directly calculates trust scores.

5) Aggregation and Model Update: Similar to the FLTrust defense, the cumulative trust score update process for both mechanisms is defined as in (9), which ensures that the trust score change is non-negative. The trust score in the current round is computed based on the previous round's score and the trust score change, determined by the decoder in the supervised mechanism or the policy model in the reinforcement mechanism. The updated trust score is then used to calculate the aggregation weight, determining each client's contribution to the global model update, as described in (10). This process ensures that clients with higher trust scores have a greater influence on the global model.

Through the entire FORTRESS framework, each participant is scored in every round based on their behavior, thereby weakening the influence of malicious participants on the global model and effectively defending against poisoning attacks. Additionally, the reinforcement mechanism addresses the supervised mechanism's tendency to overly penalize slightly suspicious but potentially useful updates, enabling the system to adaptively learn from diverse client behaviors and improve the global model's robustness and accuracy in non-IID settings.

C. Formal Robustness Analysis

We present the formal robustness guarantees of our framework under a set of natural assumptions. These assumptions characterize the separability of features distinguishing benign and malicious behaviors, the adequacy of the attack simulation dataset used for training, and the controlled decay of the reinforcement learning exploration rate. Under these conditions, we establish a rigorous bound on the difference between the learned global model and the optimal model.

Theorem 3 (Formal Robustness Guarantees): The Fortress framework satisfies robustness if following conditions are met:

The benign and malicious distance features exhibit a minimum separation, quantified by a positive constant γ, such that the norm of the difference between their expected feature vectors satisfies $\|\mathbb{E}[\mathbf{h}_b] - \mathbb{E}[\mathbf{h}_m]\| \geq \gamma$. The attack simulation dataset is sufficiently large to ensure reliable training, specifically, its size satisfies $|\mathcal{D}_{\mathrm{attack}}| \geq \Omega(d/\epsilon_{\mathrm{trust}}^2)$, where d is the feature dimension and $\epsilon_{\mathrm{trust}}$ is a parameter controlling trust score accuracy. The reinforcement learning exploration rate follows a decaying schedule given by $\epsilon = \epsilon_{\mathrm{end}} + (\epsilon_{\mathrm{start}} - \epsilon_{\mathrm{end}})e^{-r/R_g}$, with the final exploration rate ϵ_{end} bounded above by a safe threshold ϵ_{safe}. Under these assumptions, after T rounds of training, the global model $\mathbf{w}^{(T)}$ satisfies the following bound with probability at least $1 - \delta$:

$$\|\mathbf{w}^{(T)} - \mathbf{w}^*\| \leq \frac{C_1}{\sqrt{T}} + C_2 f \epsilon_{\mathrm{trust}} + C_3 \sqrt{\frac{\log(1/\delta)}{n}}, \tag{36}$$

where f denotes the fraction of malicious clients, and C_1, C_2, C_3 are constants depending on system parameters.

Proof: The security guarantee is established by carefully analyzing the reliability of the trust scoring, the stability of the aggregation process, and the bounding of malicious influence on the global model.

First, under the assumption of feature separability, the empirical risk minimization (ERM) based encoder-decoder model yields a high probability that the trust score difference for a benign client exceeds a threshold η. More precisely, this probability is lower bounded by an expression that exponentially decays with the size of the attack simulation dataset and

the squared feature gap γ^2, controlled by the feature variance bound σ_h^2:

$$\Pr\left(\Delta TS_i \geq \eta \mid y_i = 1\right) \geq 1 - \exp\left(-\frac{|\mathcal{D}_{\text{attack}}|\gamma^2}{2\sigma_h^2}\right).$$

Minimizing the cross-entropy loss further ensures that the trust score error ϵ_{trust} is bounded above by a quantity proportional to the square root of the ratio between the feature dimension d (scaled by the logarithm of the hypothesis space size $|\mathcal{H}|$ over the confidence parameter δ) and the attack dataset size:

$$\epsilon_{\text{trust}} \leq \sqrt{\frac{d\log(|\mathcal{H}|/\delta)}{2|\mathcal{D}_{\text{attack}}|}}.$$

Next, we consider the stability of the aggregation weights. Denoting the deviation of each client's weight from the ideal indicator by $\Delta w_i^{(t)} = w_i^{(t)} - \mathbb{I}[y_i = 1]$, the hybrid defense mechanism guarantees that the cumulative expected squared norm of these deviations over T rounds is bounded. Specifically, it is upper bounded by the sum of a term linear in T scaled by the square of ϵ_{trust} and a term diminishing with the square root of the reinforcement learning horizon R_g:

$$\sum_{t=1}^{T}\mathbb{E}[\|\Delta\mathbf{w}^{(t)}\|^2] \leq T\epsilon_{\text{trust}}^2 + \frac{C_{\text{RL}}}{\sqrt{R_g}},$$

where the constant C_{RL} depends on the size of the action space $\mathcal{A}$ and the discount factor γ_{RL} as $C_{\text{RL}} = \mathcal{O}\left(\sqrt{\frac{|\mathcal{A}|}{1-\gamma_{\text{RL}}}}\right)$.

The final component bounds the influence of malicious clients on the aggregated gradient. Letting $\mathbf{g}_{\text{mal}}$ denote the weighted sum of malicious gradients, the trust scores ensures that its norm does not exceed the product of the malicious client fraction f, the trust score error plus a statistical confidence term, and the norm of the true gradient at the optimum:

$$\|\mathbf{g}_{\text{mal}}\| \leq f\left(\epsilon_{\text{trust}} + \sqrt{\frac{\log(1/\delta)}{n}}\right)\|\nabla F(\mathbf{w}^*)\|.$$

Building on these results and standard assumptions in federated learning (e.g., smoothness, bounded variance, Polyak-Lojasiewicz condition) as in [1], the global model update satisfies a recursion that, with an appropriately chosen learning rate η_t, telescopes to

$$\mathbb{E}\|\mathbf{w}^{(T)} - \mathbf{w}^*\|^2 \leq \frac{4}{\mu^2 T}\left(\frac{L\sigma^2}{n_{\text{eff}}} + f^2\epsilon_{\text{trust}}^2 G^2\right) + \mathcal{O}\left(\frac{\log T}{T}\right).$$

When data distributions follow a Dirichlet allocation with parameter $\alpha = 0.5$, the expected per-client distribution divergence can be explicitly computed as

$$\mathbb{E}[\rho_i^2] = \frac{C(C-1)}{\alpha(\alpha C+1)}\bigg|_{\alpha=0.5} = \frac{2C(C-1)}{C+2},$$

where C is the number of classes. Incorporating this into the client drift term refines the bound to

$$D_1 \leq \frac{\sigma^2 + \mathbb{E}[\rho_i^2]}{n_{\text{eff}}}.$$

Finally, the trust scoring mechanism remains effective in distinguishing malicious updates from benign non-IID variations provided that

$$\epsilon_{\text{trust}} \leq \frac{\mu\gamma}{2G} - \sqrt{\frac{\mathbb{E}[\rho_i^2]}{n}}.$$

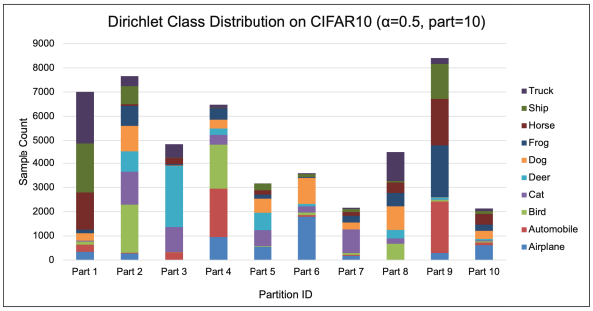


Fig. 4. Example of class distribution by dirichlet partition.

This condition guarantees that the trust scores separate malicious behavior, completing the argument for the theorem.

Taking the square root of the final bound concludes the proof of Theorem 3.

The theorem highlights several intrinsic trade-offs inherent to the system design. First, the term involving ϵ_{trust} reflects a balance between safety and computational resources: increasing the size of the attack dataset reduces this term but incurs additional training overhead. Second, the reinforcement learning mechanism introduces a transient error that decays with the number of training rounds, balancing adaptivity and stability. Finally, the linear dependence on the fraction of malicious clients f indicates that the system degrades gracefully rather than catastrophically as attack intensity increases.

VII. Experiments

A. Experimental Setup

1) Datasets: We evaluate our framework on six benchmark datasets spanning visual recognition and biomedical domains: **MNIST** [13], **Fashion-MNIST** [14], **CIFAR10** [15], **Human Activity Recognition (HAR)** [16], and **Colorectal Histology MNIST (CH-MNIST)** [17]. All vision datasets are loaded through PyTorch's datasets API, preserving original splits. The HAR dataset utilizes predefined training (7,352 samples) and test (2,947 samples) partitions containing triaxial smartphone sensor time-series. MNIST offers 60,000 training and 10,000 test grayscale digits (28×28), structurally mirrored by Fashion-MNIST's apparel images. CIFAR10 provides 50,000 training and 10,000 test RGB images (32×32) spanning 10 object categories. CH-MNIST provides biomedical data with 5,000 histology tile samples (150×150 RGB) across 8 tissue morphology classes, while HAR captures human motion patterns extracted from accelerometer and gyroscope readings, representing 6 activity types.

To simulate realistic federated scenarios, we apply *Dirichlet distribution-based* partitioning [27] with concentration parameter $\alpha = 0.5$. This stratified partitioning mechanism assigns shards of varying sizes to clients, where smaller α values induce higher inter-client data imbalance while preserving label distribution characteristics within each shard. As illustrated in Fig. 4, chromatic differentiation encodes distinct classes, while the cumulative bar chart's vertical stacking quantifies inter-client data distribution heterogeneity across categories. By leveraging the Dirichlet distribution, we can effectively simulate real-world non-IID data distributions where distinct partitions exhibit skewed class distributions with varying degrees of quantity deviations across categories. Our implementation strictly adheres to the original

dataset specifications without additional data augmentation or resampling.

2) Parameters: We systematically evaluate 11 attack methods and one composite attack. The composite attack method randomly selects one of the 11 methods for each malicious client to simulate dynamic adversarial behavior. Each malicious client implements one of the following strategies:

- **No Attack (NA)**: The client is benign, there is no attack method during local training.
- **Label Flipping Attack (LF)**: We use the same label flipping attack setting as [9]: Transforms original label l into $M-l-1$, where M denotes the total class count.
- **Sample Poisoning Attack (SP)**: Sample Poisoning Attack perturbs the input data by adding random noise to the original samples, thereby corrupting the training data distribution. We injects Gaussian noise $\mathcal{N}(0,1)$ into all training samples, with pixel values clipped to $[-1,1]$.
- **p-Norm Attack (PN)**: Computes layer-wise mean gradients from benign clients, then perturbs a randomly selected coordinate by $\gamma = 0.1$ through unit scaling.
- **Cosine-Constrained Substitution Attack (CCS)**: Executes parameter substitution through an iterative process: (1) Regular local training to obtain initial weights $\theta_i^{(t)}$; (2) Gradually replaces the smallest-magnitude weights with zeros while maintaining convex optimization constraints that jointly minimize l_1 and l_2 distances from benign parameters; (3) Terminates substitution when the cosine similarity between modified weights and original weights drops below threshold $\theta = 0.9$.
- **Gradient Ascent Attack (GA)**: During local training, attacker implement gradient inversion completely reversing the optimization direction while maintaining the original magnitude of updates.
- **Median Shift (MS) & Krum Median Shift Attacks (KMS)**: These collusion attacks strategically crafts malicious updates by generating parameters from a Gaussian distribution. The mean shift magnitude automatically adapts to maintain the statistical boundary of benign parameter distribution, thereby distorting aggregation geometry without manual hyper-parameter tuning.
- **Basic Backdoor Attack (BB)**: For image datasets, embeds a 3×3 special square (grayscale=0) at the top-left corner of poisoned samples. For HAR (Human Activity Recognition) dataset, implements temporal backdoor by zeroing the first 9 timesteps across all sensor channels.
- **Blend Backdoor Attack (BLB)**: Combines clean and backdoored models with equal weights.
- **Constrain-and-Scale Backdoor Attack (CSB)**: Balances cross-entropy loss and l_2-distance between clean/backdoor models using equal loss weights.
- **Distributed Backdoor Attack (DBA)**: Distributes complementary trigger patterns through uniform task allocation - each malicious client receives one distinct pixel position within the original 3×3 trigger region. The global backdoor emerges through trigger union, with excess positions beyond the attacker count $M^{(t)}$ being automatically discarded. Formally, for $M^{(t)} < 9$ attackers, the composite trigger becomes $\bigcup_{m=1}^{M^{(t)}}(x_m, y_m)$ where (x_m, y_m) denotes unique pixel coordinates.

To simulate real federated scenarios, we establish a federated learning system comprising $N = 100$ client nodes (except for HAR and CHMNIST, which follow the settings in [9] with $N = 30$ and $N = 40$ due to their limited sample sizes). While our experiments systematically investigated varying proportions of malicious clients, we observed that when malicious clients exceed two-thirds of participants under non-IID data distributions (Dirichlet partitions with $\alpha = 0.5$), even complete exclusion of adversarial models fails to enable the global model to capture comprehensive data patterns—thereby significantly diminishing the utility of federated learning. This empirical finding motivates our selection of three representative configurations for detailed analysis: $f = 0$ (baseline without attacks), $f = 30$ (sub-$\frac{1}{3}$ maliciousness), and $f = 60$ (sub-$\frac{2}{3}$ maliciousness), which respectively correspond to scenarios of no security threats, moderate adversarial presence, and critical system vulnerability thresholds. The global iteration count R_g is set to 50 with local iteration count $R_l = 1$, employing stochastic gradient descent with learning rate $\eta = 0.1$ and mini-batch size $|\mathcal{B}| = 32$. Parameters containing NaN values are automatically excluded from aggregation through $\mathbb{I}_{\text{valid}}(\theta_i) = \begin{cases} 1 & \text{if } \theta_i \in \mathbb{R}^d \\ 0 & \text{otherwise} \end{cases}$. To evaluate attack resilience, backdoor triggers $\{\tau_k\}_{k=1}^K$ are embedded in all validation samples $\mathcal{D}_{\text{val}}$. The global model architecture comprises 2-3 convolutional layers $\{\text{Conv2D}(c_{\text{in}}, c_{\text{out}}, k)\}$ followed by 2-3 fully connected layers $\{\text{FC}(d_{\text{in}}, d_{\text{out}})\}$, with precise layer configurations adapted per dataset through $\arg\min_\phi \mathcal{L}(\phi; \mathcal{D}_{\text{train}})$.

3) Metrics: We evaluate our method using two metrics aligned with our objectives: the accuracy of detecting malicious participants and the global model accuracy. The malicious client detection rate measures the method's **Adaptability**, defined as its capability to identify malicious clients across diverse and evolving attack scenarios. To test the defense effectiveness against backdoor attacks, all data in the validation set is embedded with the same backdoor patterns as those in the training set. Meanwhile, the global model accuracy reflects both **Fidelity** and **Effectiveness**, assessing the defense mechanism's ability to preserve the model's accuracy and convergence while effectively mitigating poisoning attacks.

4) Baselines: We compare our method with **FedAvg** [1], **Krum** [6], **Trimmed Mean** [7], and **FLTrust** [9]. Assuming the number of malicious clients (f) is known, Krum selects gradients based on the smallest L_1 distance among $N-f-2$ gradients. For Trimmed Mean, $2f$ gradients are trimmed (removing the f largest and f smallest). Similar to FLTrust, our method uses a small root dataset generated via the Dirichlet method to detect malicious clients. Unlike FLTrust, however, our root dataset does not aim to represent the complete data distribution of all clients.

5) Hardware and Software Environment: All experiments were conducted on a PC equipped with an RTX 4090 GPU, an Intel i5-13600KF processor, 32GB of RAM, and a 1TB SSD. The system ran on Windows 11 with CUDA 12.4 installed. The primary software environment included Python 3.9, PyTorch 2.4.1+cu118 with GPU acceleration enabled. We used virtual environments to manage dependencies and ensure consistency across experiments.

B. Experimental Results

As elaborated previously, our framework aims to achieve three core objectives: **Effectiveness** through precise detection and identification of malicious participants followed by systematic reduction of their aggregation weights, ultimately improving final model accuracy; **Fidelity** by maintaining

TABLE II
ACCURACY (%) IN DETECTING MALICIOUS PARTICIPANTS

Attack Category	Attack	MNIST	F-MNIST	CIFAR10	HAR	CH-MNIST
-	No Attack	78.09	79.17	63.12	77.37	81.60
Data Poisoning	Label Flipping Attack	98.41	73.05	78.65	86.93	80.30
	Sample Poisoning Attack	97.80	71.34	73.32	84.43	86.83
	Basic Backdoor Attack	83.26	78.30	74.49	74.40	78.80
	Blending Backdoor Attack	82.14	74.07	60.10	73.56	75.23
	Distributed Backdoor Attack	75.21	74.36	60.78	68.57	75.77
	Constrain-and-Scale Backdoor	87.43	70.71	68.03	62.57	78.93
Model Poisoning	p-Norm Attack	93.43	89.36	87.32	88.87	83.51
	Gradient Ascent Attack	72.15	NaN*	NaN*	NaN*	NaN*
	Median Shift Attack	86.28	89.03	87.24	89.83	83.7
	Krum Median Shift Attack	87.33	89.39	89.18	88.98	87.6
	Cosine-Constrained Substitution Attack	99.71	87.51	88.16	83.80	85.3

* indicates attacks ignored due to invalid model outputs (e.g., None values)

TABLE III
MALICIOUS ATTACK DETECTION ACCURACY (%) BEFORE VS. AFTER FINE-TUNING

Attack	Fashion-MNIST		CIFAR-10		CH-MNIST		HAR	
	Before	After	Before	After	Before	After	Before	After
No Attack	73.14	72.81(↓0.33)	79.61	73.52(↓6.09)	69.12	68.48(↓0.64)	72.30	79.06(↑6.76)
Label Flipping Attack	42.21	75.24(↑33.03)	10.32	61.52(↑51.20)	29.11	63.89(↑34.78)	42.23	71.56(↑29.33)
Sample Poisoning Attack	37.13	76.81(↑39.68)	17.59	51.68(↑34.09)	19.44	58.64(↑39.20)	38.11	72.72(↑34.61)
p-Norm Attack	81.29	89.04(↑7.75)	75.13	84.38(↑9.25)	72.18	81.66(↑9.48)	83.17	88.31(↑5.14)
Median Shift Attack	58.44	84.82(↑26.38)	16.58	81.92(↑65.34)	24.17	73.49(↑49.32)	58.80	74.96(↑16.16)
Krum Median Shift Attack	41.16	87.56(↑46.40)	33.75	86.32(↑52.57)	22.16	71.93(↑49.77)	41.64	74.15(↑32.51)
Basic Backdoor Attack	28.77	56.80(↑28.03)	11.69	49.19(↑37.50)	27.34	61.22(↑33.88)	29.18	52.20(↑23.02)
Blending Backdoor Attack	21.88	61.66(↑39.78)	9.17	53.79(↑44.62)	13.54	52.39(↑38.85)	21.59	67.02(↑45.43)
Distributed Backdoor Attack	19.07	54.43(↑35.36)	9.33	40.54(↑31.21)	20.14	58.17(↑38.03)	12.07	57.58(↑45.51)
Constrain-and-Scale Backdoor	21.34	48.95(↑27.61)	14.34	44.14(↑29.80)	9.88	48.17(↑38.29)	11.33	52.72(↑41.39)
Cosine-Constrained Substitution Attack	41.29	86.95(↑45.66)	14.02	81.28(↑67.26)	24.77	84.13(↑59.36)	44.35	82.01(↑37.66)

baseline-comparable model performance in attack-free scenarios without introducing significant accuracy degradation; **Adaptability** through consistent defensive performance across heterogeneous datasets, diverse learning tasks, and evolving attack strategies. All empirical evaluations are publicly available in our *GitHub* repository.[1] Experimental validation confirms that our method fulfills these design objectives, as quantitatively demonstrated through comprehensive metric analysis:

1) Accuracy in Detecting Malicious Participants: Leveraging the FLMPID prepared simulated attack dataset, we systematically evaluate Fortress's capability to discern malicious patterns under varying adversarial conditions. This table originates from (a) systematic aggregation of client behavioral patterns across 11 attack variants and different adversarial ratios (f=0%, 30% and 60%), as detailed before, and (b) performance benchmarking through standardized 80-20 dataset splits. As shown in Table II, the proposed framework demonstrates reasonable effectiveness and adaptability across various attack scenarios. The non-100% accuracy for benign clients reflects the inherent challenge in completely distinguishing normal updates from adversarial patterns, while maintaining practically effective discrimination capability.

For data poisoning attacks, the detection accuracy remains high in most cases. Notably, the Label Flipping Attack achieves an accuracy of 98.41% on MNIST and 86.93% on HAR, demonstrating the framework's strong capability in identifying this type of attack. However, the accuracy drops for certain attacks and datasets, such as the Blending Backdoor Attack on CIFAR10 (60.10%) and the Distributed Backdoor Attack on HAR (68.57%), suggesting that these attacks are more challenging to detect in some scenarios. Regarding model poisoning attacks, the detection accuracy is generally robust, with most values exceeding 80%. The p-Norm Attack and Krum Median Shift Attack are detected with high accuracy across all datasets. The Gradient Ascent Attack may induces gradient explosion during federated aggregation, ultimately causing invalid model outputs (NaN). As shown in the table, this attack inherently produces catastrophic numerical instability, making explicit detection unnecessary since NaN values directly indicate attack occurrence. The Cosine-Constrained Substitution Attack achieves near-perfect detection on MNIST (99.71%), further highlighting the effectiveness of the detection approach against sophisticated model poisoning strategies. These results suggest that the dual-domain comparison of weights and gradients provides a useful approach for identifying malicious behaviors across diverse datasets and attack types, though further improvements may be needed to handle more subtle or distributed attacks.

The fine-tuning mechanism serves as the cornerstone for maintaining defense efficacy across heterogeneous federated learning scenarios. The experimental framework begins with a model pre-trained on the MNIST dataset. This baseline model undergoes direct evaluation on multiple target datasets including Fashion-MNIST, CIFAR-10, CH-MNIST, and HAR without adaptation. Subsequently, the feature extraction layers are frozen while the classification layers are fine-tuned separately on each target dataset. After domain-specific fine-tuning, the adapted models are evaluated on their respective target datasets to assess performance changes. This two-stage approach examines how transferable the malicious participant detection capabilities remain across varying data distributions. As shown in Table III, substantial performance improvements are observed after fine-tuning. On CIFAR-10, detection accuracy for Label Flipping Attack rises from 10.32% to 61.52%, while Cosine-Constrained Substitution

[1] https://github.com/Aquarids/Flmpid-pub

TABLE IV
ACCURACY (%) OF DETECTING MALICIOUS PARTICIPANT ACROSS DATASETS BEFORE AND AFTER KNOWLEDGE DISTILLATION

Attack	MNIST		Fashion-MNIST		CIFAR-10		HAR		CH-MNIST	
	Before	After	Before	After	Before	After	Before	After	Before	After
No Attack	87.46	73.33	83.12	76.93	68.41	62.65	78.19	73.58	87.73	83.94
Label Flipping Attack	92.08	87.31	85.14	79.69	77.17	69.61	83.82	82.15	80.23	72.50
Sample Poisoning Attack	71.82	87.69	82.04	71.29	69.77	58.51	86.16	83.84	83.07	78.40
p-Norm Attack	86.13	85.23	89.28	78.15	89.15	83.17	89.55	87.19	86.08	87.50
Gradient Ascent Attack	82.87	89.45	NaN*	NaN*	NaN*	NaN*	NaN*	NaN*	NaN*	NaN*
Median Shift Attack	89.83	86.38	92.18	83.18	89.36	85.53	88.14	86.57	86.55	85.93
Krum Median Shift Attack	88.67	70.16	89.17	84.25	91.03	87.50	87.57	84.39	88.32	81.79
Basic Backdoor Attack	89.78	94.02	78.37	73.29	62.47	59.23	76.18	72.16	82.12	82.26
Blending Backdoor Attack	-	78.63	-	69.10	-	66.03	-	74.87	-	84.54
Distributed Backdoor Attack	-	81.86	-	67.45	-	63.80	-	79.44	-	80.24
Constrain-and-Scale Attack	-	73.23	-	68.62	-	69.29	-	73.38	-	61.16
Cosine-Constrained Substitution	-	92.37	-	89.43	-	89.72	-	85.19	-	92.15

* indicates attacks ignored due to invalid model outputs (e.g., None values)
- indicates the model before distillation does not have this category label.

Attack increases from 14.02% to 81.28%. Similar enhancements are observed across Fashion-MNIST, CH-MNIST, and HAR datasets, where attacks such as Krum Median Shift show gains exceeding 40 percentage points. These outcomes confirm that core features from the source domain are preserved, while retrained classification layers rapidly adapt to new data distributions and attack patterns. Although minor accuracy reductions occur in benign ("No Attack") scenarios on some datasets, these are outweighed by significant attack detection improvements, indicating an balance between adaptability and stability. This selective fine-tuning enables reliable generalization of malicious participant recognition to evolving federated learning environments.

Knowledge distillation serves as a pivotal technique for extending malicious participant detection coverage without full model retraining, enabling continuous learning of novel attack signatures while maintaining recognition capability for both existing and emerging threats. As shown in Table IV, we evaluate the detection accuracy across five datasets before and after applying knowledge distillation. The experiment is conducted in two phases. In the first phase (Before Distillation), the defense model is trained using a limited set of attack categories available in the pre-training corpus. During the second phase (After Distillation), knowledge distillation is employed to transfer the teacher model's knowledge to a student model, which is further exposed to new attack variants. The student model is optimized jointly with classification and feature alignment losses, and inherits the robust feature extraction capabilities from the teacher while learning to identify novel attack signatures. Attack types not present in the teacher's training set are indicated by "-". As indicated in the table, the baseline model achieves high detection accuracy on known attacks but completely fails to detect unseen attacks. Notably, after distillation, the student model demonstrates substantial improvements in detecting previously unseen attacks. For instance, achieving 92.37% accuracy on CCS Attack for MNIST where no detection capability existed previously, while for CH-MNIST it reaches 92.15%. Similarly, NaN values represent Gradient Ascent Attack-induced invalid outputs as previously described. This expansion comes with marginal trade-offs: accuracy for certain known attacks like "No Attack" on MNIST decreased from 87.46% to 73.33%, as learning new signatures may temporarily dilute feature focus on existing threats. However, as evidenced by the experimental results, such declines remain within an acceptable range (no more than 15%), and the added capability to recognize new attacks makes this acceptable.

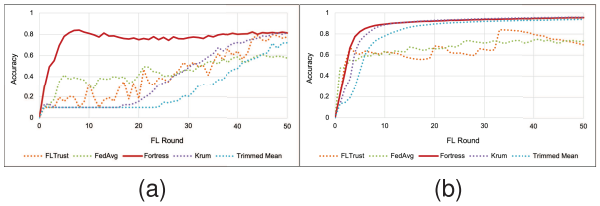


Fig. 5. Accuracy curves of basic backdoor attacks (a) and p-Norm Attacks (b) on MNIST dataset with malicious participants $f = 30\%$.

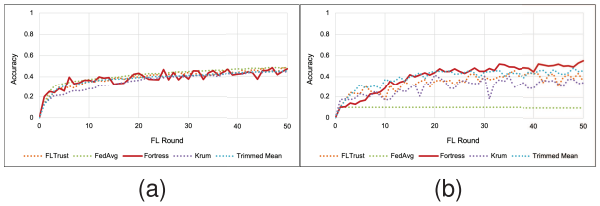


Fig. 6. Accuracy curves of sample poisoning attacks (a) and Gradient ascent attacks (b) on CIFAR10 dataset with malicious participants $f = 60\%$.

In summary, fine-tuning adapts models trained on one FL task to another FL task within consistent threat scenarios without full retraining which reducing training time. Concurrently, knowledge distillation expands threat coverage by assimilating new attack signatures on the same FL dataset while avoiding complete model retraining. Both approaches substantially decrease deployment overhead, enhancing practical viability in real-world federated learning systems across diverse scenarios.

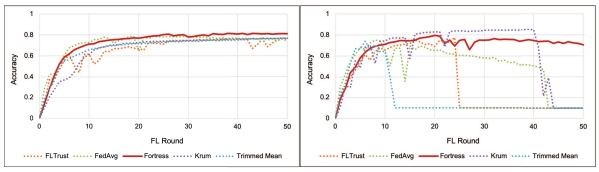
2) Accuracy of Global Model: The results in Table V demonstrate the final-round accuracy of defenses under various attacks. Under no attack (NA), FORTRESS achieves competitive accuracy across all datasets: 95.5% on MNIST (slightly below FedAvg's 96.7%), 77.2% on F-MNIST, and 55.2% on CIFAR10. This indicates minimal degradation of model performance when no malicious clients are present, validating FORTRESS's design for fidelity preservation during federated training. Fig. 5 illustrates Fortress performance against Basic Backdoor Attacks (a) and p-Norm Attacks (b) on MNIST dataset with $f = 30\%$ malicious participants, while Fig. 6 demonstrates its resilience against Sample Poisoning Attacks (a) and Gradient Ascent Attacks (b) on CIFAR10 dataset under $f = 60\%$ corruption. Collectively, these results demonstrate Fortress's capability to effectively identify malicious clients at an early stage and maintain persistent defense against attacks through its reputation-based framework.

TABLE V
ACCURACY (%) OF DIFFERENT DEFENSES AGAINST VARIOUS ATTACKS ACROSS DATASETS (FL TASK PERFORMANCE)

Dataset	Defense	NA	LF		SP		PN		GA		MS		KMS		BB		BLB		DBA		CSB		CCS	
		0%	30%	60%	30%	60%	30%	60%	30%	60%	30%	60%	30%	60%	30%	60%	30%	60%	30%	60%	30%	60%	30%	60%
MNIST	FedAvg	96.7	53.4	1.2	69.5	71.3	73.5	61.1	38.4	10.5	79.6	26.6	64.7	20.7	57.1	35.7	11.4	9.8	62.9	10.0	82.2	65.4	10.0	10.0
	FLTrust	90.6	**96.7**	**87.9**	93.6	93.7	69.7	75.1	88.7	89.3	95.9	91.9	95.0	83.9	77.2	**91.3**	10.1	20.7	**80.4**	72.8	77.1	75.3	10.9	10.3
	FORTRESS	95.5	96.1	83.1	**95.3**	**95.4**	95.5	**95.7**	**96.3**	**96.2**	96.2	95.7	95.6	**94.7**	81.0	61.3	**81.8**	74.2	76.8	**79.5**	**84.5**	**68.9**	**95.8**	**96.2**
	Krum	95.8	96.1	10.3	95.1	94.3	**96.0**	10.1	9.8	9.8	95.9	**95.9**	95.3	19.1	**81.3**	9.8	85.9	**81.2**	79.7	59.3	79.4	42.3	86.1	90.5
	Trim	95.6	95.8	10.2	94.8	94.6	93.9	11.2	95.6	96.0	**96.5**	95.1	**95.9**	93.4	71.8	9.8	80.3	28.2	77.1	68.9	80.2	50.8	85.2	81.1
F-MNIST	FedAvg	76.6	53.4	26.1	76.5	73.1	75.3	72.1	10.0	10.0	75.6	54.9	64.7	64.9	76.0	71.2	10.0	10.0	**76.2**	72.9	62.7	48.2	74.4	10.0
	FLTrust	70.8	73.0	**65.8**	73.6	72.1	73.9	54.4	55.1	63.2	68.6	75.6	65.5	74.7	72.0	75.6	10.0	10.0	73.3	67.8	73.8	71.6	75.9	10.0
	FORTRESS	**77.2**	73.6	51.0	**77.3**	**76.0**	**75.5**	72.1	74.3	**74.7**	**76.3**	**76.2**	**76.4**	**74.8**	**76.8**	**77.3**	62.4	10.0	75.6	**74.1**	**78.1**	**75.5**	**81.1**	**70.5**
	Krum	60.4	**76.7**	10.6	75.9	74.2	**76.3**	75.5	10.0	12.5	74.4	10.0	76.0	10.0	76.2	73.7	**76.7**	**75.2**	75.4	69.1	75.6	70.7	76.9	10.0
	Trim	76.8	75.3	12.8	75.6	75.2	73.1	12.5	**75.2**	73.5	72.7	10.0	74.7	10.0	75.1	71.7	75.0	55.8	75.2	71.5	75.1	71.7	76.9	10.0
CIFAR10	FedAvg	53.4	49.0	10.6	50.6	**48.1**	49.9	44.0	10.0	10.0	**52.3**	10.0	51.2	29.4	52.5	52.0	10.0	10.0	**54.6**	**51.6**	53.5	49.2	10.0	10.0
	FLTrust	51.7	45.6	**47.6**	49.3	47.1	47.9	47.8	**56.6**	34.9	45.0	**24.2**	51.4	**47.8**	52.7	**52.2**	16.7	10.0	49.1	50.8	**56.3**	**50.8**	31.2	32.6
	FORTRESS	**55.2**	47.9	21.7	**51.6**	47.0	45.5	47.1	52.9	**54.7**	46.9	14.4	53.8	33.5	**53.7**	44.2	56.1	10.0	51.3	41.4	31.6	33.7	**66.1**	**44.3**
	Krum	50.9	**51.4**	11.5	49.4	44.2	**53.3**	**52.7**	40.6	33.7	46.5	10.0	48.3	10.0	47.9	42.3	53.8	**47.6**	50.5	43.7	43.9	36.9	54.7	30.3
	Trim	49.2	44.5	13.2	49.1	45.0	39.8	17.2	49.1	45.8	34.9	10.0	46.7	10.0	48.1	44.9	43.2	31.6	48.4	43.9	50.1	47.6	52.7	10.0
HAR	FedAvg	65.9	62.9	60.5	57.0	**65.7**	**85.9**	**80.4**	16.8	16.8	**72.2**	82.6	79.4	16.8	**72.9**	75.4	16.8	41.9	16.8	58.1	30.6	76.3	16.8	16.8
	FLTrust	69.7	24.1	70.8	51.2	53.4	18.2	74.6	16.8	51.0	54.2	**83.7**	16.8	**80.7**	44.9	**75.8**	16.8	40.8	18.5	72.9	40.8	**69.8**	16.8	16.8
	FORTRESS	66.1	**66.7**	**68.5**	**68.4**	65.3	67.1	63.7	**62.0**	68.6	67.6	65.7	**68.3**	68.5	61.6	65.0	17.0	33.1	**67.3**	64.7	**69.3**	64.7	**67.3**	**68.1**
	Krum	**67.4**	10.4	81.2	38.4	49.7	79.2	81.5	14.3	**69.7**	45.3	79.6	16.8	77.3	16.8	67.7	**74.6**	**82.1**	16.8	**78.0**	16.8	63.7	16.8	16.8
	Trim	62.7	10.7	83.3	46.8	53.8	16.8	72.6	16.8	58.5	27.7	82.4	52.8	80.1	16.8	16.8	64.3	75.2	16.8	72.9	16.8	70.7	27.6	16.8
CHMNIST	FedAvg	62.4	53.2	10.6	50.4	**53.8**	55.8	50.9	19.8	50.2	**62.9**	20.5	57.1	13.2	22.9	11.9	20.5	12.3	40.5	**13.4**	38.6	13.7	13.1	12.3
	FLTrust	49.1	12.9	11.7	13.2	24.5	14.2	14.0	12.8	13.3	13.4	14.2	17.3	13.9	12.6	**14.1**	12.6	12.4	13.8	13.2	12.6	**13.9**	12.4	10.0
	FORTRESS	61.7	**61.8**	**48.3**	52.6	52.1	**58.3**	46.8	**57.6**	**54.5**	56.8	**26.4**	55.5	13.1	31.3	12.1	27.4	12.3	33.4	13.2	36.4	14.1	**58.5**	**60.2**
	Krum	**62.7**	52.8	12.1	**58.6**	14.1	56.7	**52.1**	50.5	38.2	53.9	12.9	51.1	11.5	**48.2**	11.9	53.7	**52.1**	57.8	13.2	**53.1**	13.7	52.8	12.3
	Trim	61.8	60.4	11.9	56.0	23.2	47.6	14.6	55.2	47.1	52.1	14.2	53.8	**13.2**	41.6	11.9	**55.7**	31.7	47.3	13.2	50.3	13.7	56.6	12.3

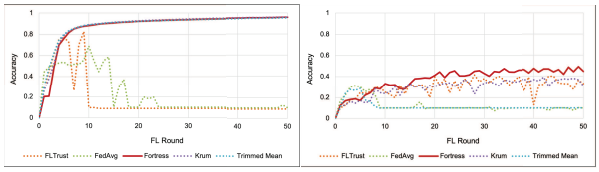
While not universally optimal, FORTRESS maintains certain accuracy across multiple threat scenarios where other defenses may fail. For example, under 60% malicious clients on MNIST, FORTRESS sustains 95.7% accuracy against p-norm attacks, outperforming FedAvg (61.1%) and FLTrust (75.1%). Similarly, against Gradient Ascent attacks on MNIST at 60% corruption, it achieves 96.2% accuracy, far exceeding FedAvg's 10.5% and Krum's 9.8%. On F-MNIST, FORTRESS retains 76.0% accuracy against sample poisoning attacks at 60% malicious clients, surpassing alternatives like Trim (75.2%). These results highlight its resilience even under extreme adversarial conditions. Additionally, FORTRESS partially mitigates backdoor threats but shows limitations at high corruption rates. Against Basic Backdoor attacks on MNIST with 30% malicious clients, FORTRESS achieves 81.0% accuracy, higher than FedAvg (57.1%) though below FLTrust (91.3%). For Blend Backdoor attacks on CIFAR10 at 30% corruption, it attains 56.1% accuracy. However, effectiveness diminishes at 60% malicious clients: under Blend Backdoor attacks the accuracy drops to 10.0% on CIFAR10. Notably, high corruption rates inherently degrade model performance (e.g., FedAvg drops to 35.7% for Basic Backdoor on MNIST at 60%), suggesting backdoor attacks may be less pronounced when global model integrity is already compromised.

Fig. 7 and Fig. 8 illustrates the accuracy curves of CCS Attack. These figures demonstrating cross-dataset consistency in CCS attack behaviors. These collective results establish that at lower corruption rates (f = 30%), CCS Attack systematically impedes global model convergence speed across diverse defenses and datasets, while higher corruption rates (f = 60%) induce performance oscillations and non-convergence during late training stages. The Fortress maintains resilience against this spectrum of CCS threats by leveraging the inherent parameter sparsity signature as a core detection feature, enabling robust mitigation throughout the training lifecycle.



(a) (b)

Fig. 7. Accuracy curves of CCS Attack with malicious participants f = 30% (a) and f = 60% (b) on Fashion-MNIST dataset.



(a) (b)

Fig. 8. Accuracy curves of CCS Attack with malicious participants f = 30% (a) and f = 60% (b) on Fashion-MNIST (a) and CIFAR10 (b) datasets.

VIII. CONCLUSION

This paper introduces FORTRESS, a novel defense mechanism for federated learning (FL) that mitigates the impact of malicious participants through a robust encoder-decoder architecture. By dynamically adjusting client update weights and integrating reinforcement learning with trust bootstrapping, FORTRESS effectively detects and suppresses malicious updates while maintaining certain accuracy. Experimental results, based on two metrics—the accuracy of detecting malicious participants and the global model accuracy—demonstrate its effectiveness, robustness, and adaptability across diverse datasets and attack

methods. Moreover, the fine-tuning of the encoder on simulated attack scenarios and the use of knowledge distillation enable FORTRESS to adapt to evolving adversarial behaviors, ensuring long-term reliability in dynamic FL environments. Overall, FORTRESS provides a practical and effective solution for securing FL systems against adversarial threats in real-world scenarios.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist. (AISTATS)*, Jan. 2016, pp. 1273–1282.

[2] Y. Miao, Z. Liu, H. Li, K. R. Choo, and R. H. Deng, "Privacy-preserving Byzantine-robust federated learning via blockchain systems," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 2848–2861, 2022.

[3] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, "Privacy-enhanced federated learning against poisoning adversaries," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4574–4588, 2021.

[4] C. Xie, O. Koyejo, and I. Gupta, "Generalized Byzantine-tolerant SGD," 2018, *arXiv:1802.10116*.

[5] M. Baruch, G. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Jan. 2019, pp. 8635–8645.

[6] 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. (NeurIPS)*, vol. 30, Dec. 2017, pp. 118–128.

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

[8] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Mitigating Sybils in federated learning poisoning," 2018, *arXiv:1808.04866*.

[9] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust federated learning via trust bootstrapping," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–18.

[10] T. D. Nguyen et al., "FLAME: Taming backdoors in federated learning," in *Proc. 31st USENIX Secur. Symp. (USENIX Secur.)*, 2022, pp. 1415–1432.

[11] H. Fereidooni, A. Pegoraro, P. Rieger, A. Dmitrienko, and A.-R. Sadeghi, "FreqFed: A frequency analysis-based approach for mitigating poisoning attacks in federated learning," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2024, pp. 1–16.

[12] P. Sun, X. Liu, Z. Wang, and B. Liu, "Byzantine-robust decentralized federated learning via dual-domain clustering and trust bootstrapping," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 24756–24765.

[13] Y. LeCun. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[14] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.

[15] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[16] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in *Proc. Eur. Symp. Artif. Neural Netw. (ESANN)*, Jan. 2013, pp. 437–442.

[17] J. N. Kather et al., "Multi-class texture analysis in colorectal cancer histology," *Sci. Rep.*, vol. 6, no. 1, Jun. 2016, Art. no. 27988.

[18] S. Guo et al., "Robust and privacy-preserving collaborative learning: A comprehensive survey," 2021, *arXiv:2112.10183*.

[19] L. Lyu et al., "Privacy and robustness in federated learning: Attacks and defenses," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 7, pp. 1–21, Jun. 2022.

[20] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, *arXiv:1712.05526*.

[21] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.

[22] C. Xie, K. Huang, P. Chen, and B. Li, "DBA: Distributed backdoor attacks against federated learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Apr. 2020, pp. 1–19.

[23] L. Munoz-González et al., "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, Nov. 2017, pp. 27–38.

[24] E. M. El Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 3521–3530.

[25] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[26] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, Aug. 1988.

[27] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-IID data silos: An experimental study," in *Proc. IEEE 38th Int. Conf. Data Eng. (ICDE)*, May 2022, pp. 965–978.

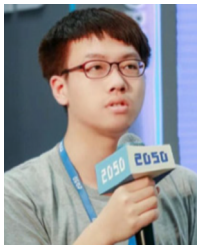
[28] J. Zeng, S. Lin, Y. Wang, and Z. Xu, "$L_{1/2}$ regularization: Convergence of iterative half thresholding algorithm," *IEEE Trans. Signal Process.*, vol. 62, no. 9, pp. 2317–2329, May 2014.

[29] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017, *arXiv:1712.07557*.

[30] R. Labaca-Castro, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Jan. 2023, pp. 73–76.

[31] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 27, Dec. 2014, pp. 3320–3328.

[32] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.



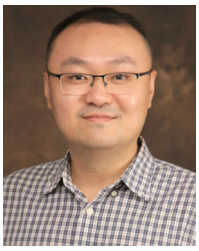
Zhaoqi Wang (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China. His research interests include AI security and federated learning.

Zijian Zhang (Senior Member, IEEE) is currently a Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. His research interests include anonymous communication security, blockchain security, and AI security.

Zhen Li is currently pursuing the Ph.D. degree with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China. His research interests include privacy computing, AI security, and blockchain.

Yan Wu (Graduate Student Member, IEEE) received the B.E. and M.E. degrees in engineering from the University of Electronic Science and Technology of China, in 2017 and 2020, respectively. She is currently pursuing the Ph.D. degree with the School of Cyberspace Science and Technology, Beijing Institute of Technology. Her research interests include blockchain, cryptographic protocols, and network security.

Ye Liu received the B.E. degree from Northeastern University, China, in June 2016, the M.Sc. degree from Beihang University in January 2019, and the Ph.D. degree from the College of Computing and Data Science, Nanyang Technological University (NTU), in June 2023. Later, he worked for half a year as a Research Assistant with the Cybersecurity Laboratory, NTU. He is currently a Research Scientist with Singapore Management University (SMU). His main research interests include smart contract security and program analysis. He has won two Distinguished Paper Awards from top-tier conferences (NDSS'25 and ISSTA'22).



Meng Li (Senior Member, IEEE) received the Ph.D. degree in computer science and technology from the School of Computer Science and Technology, Beijing Institute of Technology (BIT), China, in 2019. He is currently an Associate Professor and the Personnel Secretary of the School of Computer Science and Information Engineering, Hefei University of Technology (HFUT), China. He was a Post-Doctoral Researcher with the Department of Mathematics and the HIT Center, University of Padua, Italy, where he is with the Security and PRIvacy Through Zeal (SPRITZ) Research Group led by Prof. Mauro Conti (IEEE Fellow). He was sponsored by the ERCIM "Alain Bensoussan" Fellowship Program to conduct Post-Doctoral Research Supervised by Prof. Fabio Martinelli at CNR, Italy, from October 2020 to March 2021. He was sponsored by China Scholarship Council (CSC) as a Joint Ph.D. Student supervised by Prof. Xiaodong Lin (IEEE Fellow) with the Broadband Communications Research (BBCR) Laboratory, University of Waterloo, and Wilfrid Laurier University, Canada, from November 2017 to August 2018. He is supported by CSC as a Visiting Scholar collaborating with Prof. Mauro Conti (IEEE Fellow) at the HIT Center, University of Padua, from March 2025 to June 2025. His research interests include security, privacy, applied cryptography, blockchain, TEE, and the Internet of Vehicles. In this area, he has published 115 papers in topmost journals and conferences, including IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, TODS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, IEEE S&P, USENIX Security, ACM MobiCom, and ISSTA. He is a Senior Member of CIE, CIC, and CCF. He was a recipient of the 2024 IEEE HITC Award for Excellence (Early Career Researcher) and the 2025 IEEE TCSVC Rising Star Award. He is an Associate Editor for IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, EURASIP JIS, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, and COMNET. He has served as a TPC Member for conferences, including ICDCS, ICICS, TrustCom, ICC, and Globecom.

Xin Li received the Ph.D. degree in computer science from Hong Kong Baptist University in 2009. She is currently an Associate Professor with the School of Computer Science, Beijing Institute of Technology, China. Her research interests include development of algorithms for representation learning, reasoning under uncertainty, and machine learning with applications to graph data mining, recommender systems, and robotics.

Yong Liu is currently a Research Fellow and a Doctoral Supervisor in cybersecurity technology with Qi An Xin Technology Group Inc. His research interests include cloud computing security, vulnerability mining, and blockchain security evaluation.

Jincheng An is currently an Assistant Research Fellow with Qi An Xin Technology Group Inc. He is an Expert in cybersecurity and technology standardizing document drafting. His research interests include cyber resilience theory and evaluation and data security detection.

Wei Liang received the Ph.D. degree in computer science and technology from Hunan University in 2013. He was a Post-Doctoral Scholar with Lehigh University from 2014 to 2016. He is currently a Professor with the School of Computer Science and Engineering, Hunan University of Science and Technology. He has authored or co-authored more than 140 journals and conference papers, such as IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTION ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, IEEE TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS, and IEEE INTERNET OF THINGS JOURNAL. His research interests include intelligent transportation, security of the IoV, blockchain, embedded systems and hardware IP protection, and security management in wireless sensor networks.

Liehuang Zhu (Senior Member, IEEE) is currently a Professor and the Secretary of the School of Cyberspace Science and Technology, Beijing Institute of Technology. He has published more than 100 peer-reviewed journals or conference papers, including more than 50 IEEE/ACM TRANSACTIONS papers. His research interests include security protocol analysis and design, wireless sensor networks, and cloud computing. He has been granted a number of IEEE Best Paper Awards, including IWQoS 17, TrustCom 18.