DP2Guard: A Lightweight and Byzantine-Robust Privacy-Preserving Federated Learning Scheme for Industrial IoT

Baofu Han, Bing Li, Yining Qi, *Member, IEEE*, Raja Jurdak, *Senior Member, IEEE*, Kaibin Huang, *Fellow, IEEE*, and Chau Yuen, *Fellow, IEEE*

***Abstract*—Privacy-Preserving Federated Learning (PPFL) has emerged as a secure distributed Machine Learning (ML) paradigm that aggregates locally trained gradients without exposing raw data. To defend against model poisoning threats, several robustness-enhanced PPFL schemes have been proposed by integrating anomaly detection. Nevertheless, they still face two major challenges: (1) the reliance on heavyweight encryption techniques results in substantial communication and computation overhead; and (2) single-strategy defense mechanisms often fail to provide sufficient robustness against adaptive adversaries. To overcome these challenges, we propose DP2Guard, a lightweight PPFL framework that enhances both privacy and robustness. DP2Guard leverages a lightweight gradient masking mechanism to replace costly cryptographic operations while ensuring the privacy of local gradients. A hybrid defense strategy is proposed, which extracts gradient features using singular value decomposition and cosine similarity, and applies a clustering algorithm to effectively identify malicious gradients. Additionally, DP2Guard adopts a trust score–based adaptive aggregation scheme that adjusts client weights according to historical behavior, while blockchain records aggregated results and trust scores to ensure tamper-proof and auditable training. Extensive experiments conducted on two public datasets demonstrate that DP2Guard effectively defends against four advanced poisoning attacks while ensuring privacy with reduced communication and computation costs.**

***Index Terms*—Blockchain, federated learning, poisoning attack, privacy preserving, hybrid-defense strategy.**

I. Introduction

The Industrial Internet of Things (IIoT) connects various industrial devices through networks, enabling data collection, exchange, and analysis, and has played a significant role in advancing the digital transformation of industrial systems [1], [2]. Meanwhile, the integration of machine learning (ML) with IIoT enables industrial terminals to extract valuable insights from massive sensory data and make intelligent decisions in complex scenarios [3], [4]. However, the advancement of industrial intelligence heavily depends on large-scale data collection and sharing, which raises serious concerns regarding data privacy and security [5], [6]. Thus, how to fully realise the potential of ML in IIoT systems while preserving data privacy has become a critical challenge.

Federated Learning (FL) has emerged as a distributed ML paradigm that enables multiple organizations to collaboratively train a global model without sharing raw data, thereby preserving user privacy [7], [8]. However, recent studies indicate that FL still encounters challenges that adversaries can infer sensitive information from shared model updates [9], [10]. To address the threat, some privacy-preserving federated learning (PPFL) based on Differential Privacy (DP) [11], [12], Homomorphic Encryption (HE) [13], [14], and Secure Multi-Party Computation (SMPC) [15], [16] have been proposed. DP achieves privacy guarantees by injecting calibrated noise into local model gradients, but this inevitably introduces a trade-off between privacy and model utility [17]. SMPC allows participants to jointly compute global models without revealing their inputs, but the interaction processes of SMPC incur heavy communication burdens on clients. In contrast, HE enables secure computation on encrypted data and offers a compelling balance between privacy and accuracy.

Another security threat in FL is model poisoning attacks (MPAs), in which malicious clients submit manipulated gradients to interfere with the training process and compromise the integrity of the global model [18]. As illustrated in Fig. 1, such attacks become even more difficult to detect in PPFL due to the invisibility of individual gradients resulting from encryption or perturbation mechanisms. To address this issue, several defense strategies have been proposed, including cosine similarity-based similarity-based [19] and distance-based methods [20], which identify abnormal updates by measuring their deviation from the majority. However, recent studies have shown that, under complex attack scenarios, attackers can craft malicious gradients that closely mimic benign behavior, thereby evading detection. Thus, single-strategy defenses are inadequate for addressing increasingly sophisticated poisoning threats, highlighting the need to develop more robust and hybrid defense mechanisms in PPFL.

To address the aforementioned challenges, we propose DP2Guard, a lightweight PPFL framework that enhances both privacy and robustness. DP2Guard adopts a gradient masking

This work was supported in part by the National Natural Science Foundation of China under Grant 62174150; in part by the Postgraduate Research & Practice Innovation Program of Jiangsu Province under Grant KYCX23_0320; in part by the Shenzhen Science Technology and Innovation Commission (SZSTI) under Grant JCYJ20170817115500476; in part by the China Scholarship Council. (Corresponding author: Bing Li.)

Baofu Han is with the School of Cyber Science and Engineering, Southeast University, Nanjing 211102, China, and also with the School of Computing, National University of Singapore, Singapore (e-mail: hanbaofu@seu.edu.cn, baofu.han@comp.nus.edu.sg).

Bing Li is with the School of Integrated Circuits, Southeast University, Nanjing 211102, China, and also with the Shenzhen Research Institute, Shenzhen (e-mail: bernie_seu@seu.edu.cn).

Yining Qi is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: qiyining@hust.edu.cn).

Raja Jurdak is with the School of Computer Science, Queensland University of Technology, Australia (e-mail: r.jurdak@qut.edu.au).

Kaibin Huang is with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong SAR, China (e-mail: huangkb@hku.hk).

Chau Yuen is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore (email: chau.yuen@ntu.edu.sg).

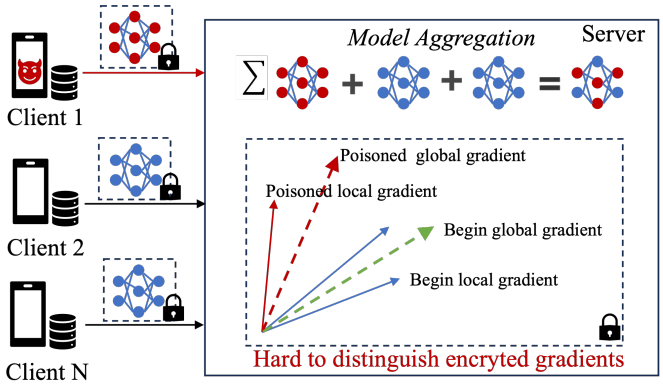


Fig. 1: The example of model poisoning attacks in PPFL. Client 1 uploads a poisoned local mode gradient, guiding the global model towards a predefined direction and impacting overall model performance.

strategy to protect local model updates without relying on computationally expensive cryptographic techniques such as HE or MPC. It also incorporates a hybrid anomaly detection strategy to identify malicious updates. Blockchain is further integrated to secure the training process and provide auditability. The main contributions of this work are summarized as follows:

1) We propose a lightweight PPFL framework that employs efficient gradient masking, where each gradient is divided into two additive shares with opposite random masks sent to two non-colluding servers. The servers collaboratively aggregate the masked shares, ensuring that no individual server can reconstruct the original gradient.
2) We develop a hybrid defense strategy that applies singular value decomposition and cosine similarity to client gradients to construct feature vectors, which are then used by a clustering algorithm to accurately identify malicious updates.
3) We introduce a trust score–based aggregation mechanism that adaptively adjusts each client's aggregation weight according to its historical behavior, thereby improving global model performance. Furthermore, both the aggregated results and trust scores are securely stored on a blockchain, ensuring training integrity and auditability.
4) Extensive experiments on two public datasets demonstrate that DP2Guard provides strong resistance against four advanced poisoning attacks, while achieving privacy protection with reduced communication and computation costs.

The remainder of this paper is organized as follows: Section II reviews related work in areas relevant to our study. Section III introduces the preliminary knowledge. Section IV illustrates the system model and the threat model. The proposed scheme is elaborated in Section V. Section VI provides the security and privacy analysis. Section VII presents performance evaluation. Finally, Section VIII concludes this work.

II. Related Work

PPFL employs advanced techniques such as DP, HE, and SMPC to protect the confidentiality of model updates. Jiang et al. [29] proposed a DP-based PPFL framework with adaptive gradient compression to enhance privacy while reducing communication overhead. Gu et al. [30] introduced FL2DP, which leverages exponential-based noise and gradient shuffling to protect both gradient and identity privacy. Li et al. [31] designed an adaptive noise injection strategy to balance privacy protection with model accuracy. However, DP-based methods inevitably introduce utility loss, making it difficult to optimize both privacy and performance. To improve efficiency, SMPC-based PPFL approaches have been explored. Bonawitz et al. [32] introduced a secure aggregation scheme using double masking with Shamir's Secret Sharing (SSS) to protect client data privacy during aggregation. Fu et al. [33] further optimized this design by replacing the double-masking scheme with a single-masking technique, significantly reducing computational overhead. Nevertheless, such methods still suffer from increasing communication overhead as the number of clients grows. HE-based approaches offer stronger privacy guarantees by enabling computations on encrypted data. Bui et al. [34] proposed a HE-based PPFL framework for intrusion detection in IoV. Li et al. [35] utilized a CKKS cryptosystem to protect model parameters during training. However, HE-based approaches incur substantial ciphertext expansion and computational overhead.

To address the poisoning attacks, numerous Byzantine-robust algorithms have been proposed. For instance, Cao et al. [21] introduced FLTrust, which calculates the cosine similarity between local and root gradients. However, this approach relies heavily on a root dataset. DnC [22] adopts random projection for dimensionality reduction, followed by spectral analysis using SVD to identify and filter out malicious clients. However, DnC require a preset number of malicious clients, leading to limitations in practical applications. To improve privacy, Shayan et al. [24] proposed Biscotti, a robust PPFL scheme that combines the Multi-Krum [20] strategy with differential privacy, detecting anomalous updates based on Euclidean distances. However, these strategies require access to plaintext local models. Li et al. [23] proposed a robust PPFL based on Krum to defend against model poisoning attacks under encrypted gradient aggregation. Mai et al. [25] introduced RFLPA, which integrates cosine similarity detection with verifiable secret sharing to ensure secure aggregation. Ma et al. [26] proposed ShieldFL, utilizing double-trapdoor HE and secure cosine similarity to identify malicious clients. Similarly, Miao et al. [36] used fully HE and cosine similarity to detect malicious gradient. Dong et al. [27] developed FLOD, which applies Hamming distance to identify abnormal gradients and employs secret sharing for privacy-preserving aggregation, though it still requires a trusted root dataset. Feng et al. [28] introduced DPFLA, combining removable masks with SVD-based techniques for malicious gradient filtering, yet it assumes a non-colluding single server and remains vulnerable to collusion attacks. Overall, most existing robust and privacy-preserving FL schemes depend on a single defense

TABLE I: Existing Approaches Based on Malicious Client Detection in PPFL

Solution	Privacy protection	Detection approach	Adaptive aggregation	Test dataset (required)	Blockchain	Byzantine robustness	Distribution setting
FLTrust [21]	No	Cosine similarity	Yes	No	No	Moderate	IID and Non-IID
Multi-Krum [20]	No	Krum	No	No	No	Weak	IID
DnC [22]	No	SVD	Partial	No	No	Moderate	IID
RFLP [23]	HE	Krum	Partial	No	No	Weak	IID and Non-IID
Biscotti [24]	DP	Multi-Krum	Partial	No	Yes	Weak	IID
RFLPA [25]	HE	Cosine similarity	Partial	No	No	Moderate	IID and Non-IID
ShieldFL [26]	Double-trapdoor HE	Cosine similarity	Partial	No	Yes	Moderate	IID and Non-IID
FLOD [27]	SMPC	Hamming distance	No	Yes	No	Weak	IID
DPFLA [28]	Removable mask	SVD	No	No	No	Moderate	IID
DP2Guard	HE + Mask	Hybrid strategy	Yes	No	Yes	Strong	IID and Non-IID

* In the "Adaptive Aggregation" column, "Yes" denotes full support, "No" indicates no support, and "Partial" represents limited or partial support. In the "Byzantine Robustness" column, "Weak" denotes low robustness, "Moderate" indicates moderate robustness, and "Strong" represents strong robustness.

mechanism, and as adversarial tactics become more adaptive and sophisticated, their robustness guarantees remain limited.

In Table I, we provide a comprehensive summary of the state-of-the-art robust PPFL approaches and compare them with our proposed scheme. The comparison highlights the advantages of our design in terms of privacy preservation, Byzantine robustness, and computational efficiency.

III. Preliminaries

In this section, we provide background on federated learning and briefly review state-of-the-art poisoning attacks.

A. Federated Learning

Assume there are N clients, each of them owns a private dataset D_i, where $i \in [1, N]$. The objective of FL is to learn a global model ω that minimizes the overall global loss across all clients [37]:

$$\min_{\omega} \sum_{i=1}^{N} \frac{|D_i|}{\sum_{j=1}^{N} |D_j|} \cdot \mathcal{L}_i(\omega), \tag{1}$$

where $\mathcal{L}_i(\omega) = \frac{1}{|D_i|} \sum_{x \in D_i} \ell(\omega, x)$, and $\ell(\omega, x)$ denotes the loss function evaluated on sample x using model ω.

At the t-th iteration, the server distributes the current global model ω_t to all clients. Each client i performs local training by minimizing its local loss $\mathcal{L}_i(\omega_t)$ using a local optimizer (e.g., stochastic gradient descent), and computes the gradient:

$$g_i^{(t)} = \nabla_{\omega} \mathcal{L}_i(\omega_t). \tag{2}$$

The client then updates its local model as $\omega_i^t = \omega_t - \eta g_i^{(t)}$, where η is the learning rate. The resulting model update $\Delta_i^{(t)} = \omega_i^t - \omega_t$ is sent back to the server. The server aggregates the received updates using FedAvg algorithm [37]:

$$\omega_{t+1} = \omega_t + \frac{1}{|S_t|} \sum_{i \in S_t} \Delta_i^{(t)} \tag{3}$$

This process is repeated until the global model converges or satisfies a predefined stopping criterion.

B. Poisoning Attacks on FL

Due to the decentralized nature of FL and the inability of the server to access raw client data or verify local updates, adversaries may interfere with the model aggregation process by injecting mislabeled data or constructing malicious model updates. These attacks are typically classified as non-adaptive or adaptive, depending on whether the perturbation is adjusted during training.

1) Non-Adaptive Attack: Non-adaptive attacks generate malicious updates using fixed strategies, regardless of their actual effect on the model. A common method is the label-flipping attack [38], where the attacker replaces a sample's truth label y with an adversarial label $\hat{y}$, computed as:

$$\hat{y} = y + l_{\text{flip}} \mod L, \tag{4}$$

where L is the total number of classes and l_{flip} is a fixed offset. The poisoned gradients are then uploaded for aggregation.

2) Adaptive Attack: Adaptive attacks dynamically adjust the perturbation direction or magnitude to improve both the effectiveness and stealth of the attack. Three representative adaptive attack strategies [39], [22] are outlined below.

- Fang attack [39]: In this attack, the attacker simulates the aggregation rule using known benign gradients and inserts perturbations in the opposite direction:

$$\hat{g}_m = \frac{\sum_{i=1}^{k} g_i}{k} - \lambda * \text{Sign}\left(\frac{\sum_{i=1}^{k} g_i}{k}\right), \tag{5}$$

  where λ is the perturbation factor, and g_i represents benign gradients. The attacker iteratively tunes λ to ensure that the crafted gradients are accepted by the target aggregation rule.
- Min-Max [22]: The Min-Max attack aims to maximize the separation between the malicious gradient and benign

gradients, while keeping the distance within the natural variation among benign clients. The formulation is:

$$\begin{aligned} &\arg\max_{\gamma} \max_{i\in[1,n]} \|\hat{g}_m - g_i\|_2 \leq \max_{i,j\in[1,n]} \|g_i - g_j\|_2 , \\ &\hat{g}_m = \frac{\sum_{i=1}^{k} g_i}{k} + \gamma \nabla_p, \end{aligned} \tag{6}$$

where n is the total number of clients, ∇_p is a predefined perturbation direction, and γ is a scaling factor optimized by the adversary.

- Min-Sum [22]: In the Min-Sum attack, the goal is to minimize the cumulative distance between the malicious update and all benign gradients, thereby increasing its similarity to the majority and reducing its detectability. The optimization objective is:

$$\hat{g}_m = \arg\min_{\hat{g}} \sum_{i=1}^{n} \|\hat{g} - g_i\|_2 + \gamma \cdot \nabla_p \tag{7}$$

where $\gamma \cdot \nabla_p$ introduces controlled perturbation in the direction determined by the attacker.

IV. System and Threat Model

In this section, we will describe the system model, threat model, and design goals, respectively.

A. System Model

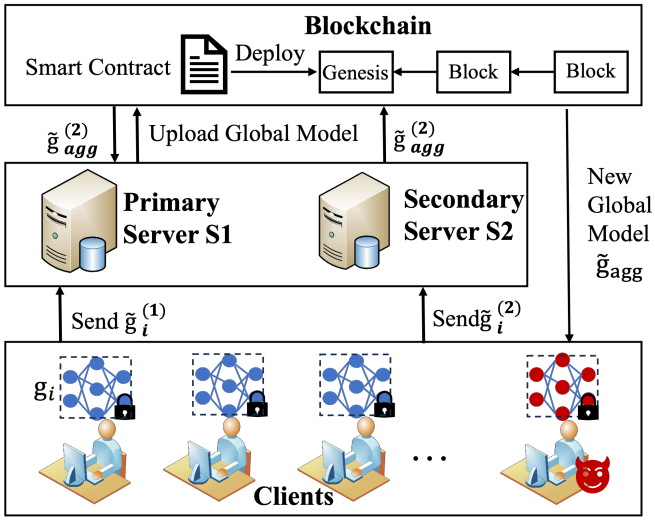


Fig. 2: System architecture of the DP2Guard.

As illustrated in Fig. 2, DP2Guard is a privacy-preserving and Byzantine-robust FL framework designed for IIoT environments. It adopts a dual-server structure integrated with a blockchain to ensure secure, reliable, and auditable aggregation.

During the local training phase, each client i computes a gradient vector $\mathbf{g}_i$ based on its private data. The client then splits g_i into two additive shares, $\mathbf{g}_i^{(1)}$ and $\mathbf{g}_i^{(2)}$, such that $\mathbf{g}_i = \mathbf{g}_i^{(1)} + \mathbf{g}_i^{(2)}$. Each share is independently masked with a random vector m_i, forming $\tilde{\mathbf{g}}_i^{(1)} = \mathbf{g}_i^{(1)} + r_i$ and $\tilde{\mathbf{g}}_i^{(2)} = \mathbf{g}_i^{(2)} - r_i$, which are transmitted to S1 and S2, respectively.

Upon receiving the masked shares from all clients, S1 performs mean-centering on the collected $\tilde{\mathbf{g}}_i^{(1)}$ vectors and forwards the processed results to S2. Leveraging both the centered gradients and the locally held $\tilde{\mathbf{g}}_i^{(2)}$, S2 executes a hybrid anomaly detection mechanism to identify malicious behaviors, calculates trust scores for each client, and conducts trust-weighted aggregation. The resulting global update and trust weights are recorded on the blockchain to ensure transparency and tamper resistance. Finally, S1 downloads the aggregated gradients and trust weights from the blockchain and completes the final aggregation using its own masked shares. The updated model is subsequently distributed to all devices via the blockchain, enabling efficient and trustworthy collaboration in IIoT network.

B. Threat Model

We consider two types of clients: (1) malicious clients; and (2) curious-but-honest clients. Malicious clients deliberately compromise the training process by submitting poisoned gradients aimed at degrading the global model's accuracy. We assume that the maximum proportion of malicious clients does not exceed half of the total participating devices. In contrast, curious-but-honest clients follow the protocol but may attempt to infer private information from shared local updates, and potentially collaborate with others to improve inference success. The aggregation process is jointly managed by two servers, S1 and S2, which are assumed to be non-colluding and curious-but-honest—that is, they faithfully execute the protocol but may analyze collected data to recover individual client updates. For the blockchain, we follow a standard Byzantine fault tolerance (BFT) model commonly employed in permissioned blockchain systems [40], [41], where at least $\zeta > 2/3$ of trustee nodes are trustworthy and consistently return correct computation results. For simplicity, we assume all trustees have uniform computational power.

C. Design Goal

Given the aforementioned security and privacy threats in FL, DP2Guard is designed to meet four key goals:

1) *Accuracy:* The scheme should preserve the accuracy of the global model when no attacks are present. Specifically, the integration of privacy-preserving and anomaly detection mechanisms should not negatively impact the model's ability to learn from legitimate client updates.
2) *Robustness:* The scheme should exhibit strong resilience against various types of poisoning attacks, including more complex and adaptive ones. It must be capable of detecting or mitigating malicious updates effectively, while preserving the privacy of benign clients.
3) *Privacy:* The scheme must guarantee the privacy of each client's local updates, ensuring that no adversary (e.g., the servers or other clients) can infer sensitive information from the shared local updates. Only the originating client should have access to the underlying private data.
4) *Efficiency:* Given the limited computational and communication resources of edge devices, the scheme should

be designed with minimal overhead under privacy-preserving requirements, making it suitable for deployment on resource-limited clients.

V. DP2GUARD SCHEME

To address both privacy leakage and gradient poisoning threats in FL within IIoT environments, we propose DP2Guard, a lightweight PPFL framework that enhances both privacy and robustness. As illustrated in Fig. 3, DP2Guard consists of five main phases: Task Initialization, Local Model Training, Privacy-Preserving Hybrid Defense Strategy, Trust Score-based Weight Calculation and Adaptive Global Aggregation. The main notations used are shown in TABLE II.

During the task initialization phase, the task publisher initializes the FL process by defining an initial global model, denoted as $\mathbf{w}^{(0)}$, which is then published to the blockchain to guarantee verifiability, integrity, and consistency across all participating devices. A subset of eligible mobile edge devices is then selected to participate in the training. Each selected device downloads the initial global model parameters $\mathbf{w}^{(0)}$ from the blockchain, which serve as the starting point for local training.

TABLE II: Notations and Descriptions

Notations	Descriptions	Notations	Descriptions
E_i	i clients	D_i	Local dataset of E_i
$\mathbf{g}_i$	Local gradient	$\tilde{\mathbf{g}}_i^{(1)}/\tilde{\mathbf{g}}_i^{(2)}$	Masked gradient
ω_t	t-th round global model	ω_t^i	t-th round local model
$\mathbf{r}_i$	Random mask value	$\hat{\mathbf{g}}_i^{(1)}/\hat{\mathbf{g}}_i^{(2)}$	Mean-centering gradient
η	Learning rate	$\mathbf{G}$	Matrix of centered gradients
s_i	Spectral deviation score	$\mathbf{f}_i$	Feature vector of client E_i
c_i	Median cosine similarity score	$\text{Trust}_i^{(t)}$	Trust score
$\tau_i^{(t)}$	Aggregation weight	$\tilde{\mathbf{g}}_{\text{agg}}^{(1)}/\tilde{\mathbf{g}}_{\text{agg}}^{(2)}$	Aggregated masked gradients on $\mathcal{S}_1$ and $\mathcal{S}_2$

A. Local Model Training

In each communication round t, each edge device E_i updates its local model by training on its private dataset using the global model parameters $\mathbf{w}^{(t-1)}$ retrieved from the blockchain. After completing local training, the device computes its local gradient update $\mathbf{g}_i^{(t)}$. To protect the privacy of this gradient, DP2Guard designs a gradient splitting and masking strategy. Specifically, edge device E_i first splits its gradient $\mathbf{g}_i$ into two additive shares $\mathbf{g}_i^{(1)}$ and $\mathbf{g}_i^{(2)}$, such that $\mathbf{g}_i = \mathbf{g}_i^{(1)} + \mathbf{g}_i^{(2)}$. To obfuscate the true gradient values, the device generates a random mask vector $\mathbf{r}_i \in \mathbb{R}^{m\times n}$, and creates two masked gradient shares:

$$\tilde{\mathbf{g}}_i^{(1)} = \mathbf{g}_i^{(1)} + \mathbf{r}_i, \quad \tilde{\mathbf{g}}_i^{(2)} = \mathbf{g}_i^{(2)} - \mathbf{r}_i. \tag{8}$$

These masked gradients are transmitted separately to Servers S1 and S2. This mechanism effectively safeguards against direct gradient leakage, no single server can infer the original gradient or any intermediate component without collaboration. The detailed procedure for this gradient splitting and masking process is presented in Algorithm 1.

Algorithm 1: Local Model Training

Input : Global model $\boldsymbol{w}^{(t-1)}$, Dataset D_i, Local gradient matrix $\mathbf{g}_i^{(t)} \in \mathbb{R}^{m\times n}$
Output: Masked gradients $\tilde{\mathbf{g}}_i^{(1)}, \tilde{\mathbf{g}}_i^{(2)}$

Initialization: $\boldsymbol{w}_i^r = \boldsymbol{w}^{r-1}$;
/* Training Model and Updating Gradient */
for *each edge device* i **do**
  Compute local gradient: $\mathbf{g}_i^{(t)} \leftarrow \nabla \ell\left(\boldsymbol{w}^{(t-1)}; D_i\right)$;
  Update local model: $\boldsymbol{w}_i^{(t)} \leftarrow \boldsymbol{w}^{(t-1)} - \eta \cdot \mathbf{g}_i^{(t)}$;
  /* Gradient Splitting and Masking */
  Generate a random matrix $\mathbf{r}_i \in \mathbb{R}^{m\times n}$;
  Generate two gradient shares $\mathbf{g}_i^{(1)} \in \mathbb{R}^{m\times n}$ and $\mathbf{g}_i^{(2)} \in \mathbb{R}^{m\times n}$;
  Compute masked share $\tilde{\mathbf{g}}_i^{(1)} \leftarrow \mathbf{g}_i^{(1)} + \mathbf{r}_i$;
  Compute masked share $\tilde{\mathbf{g}}_i^{(2)} \leftarrow \mathbf{g}_i^{(2)} - \mathbf{r}_i$;
  /* Sending the Masked Gradient to S1 and S2 */
  Send $\tilde{\mathbf{g}}_i^{(1)}$ to server S1 ;
  Send $\tilde{\mathbf{g}}_i^{(2)}$ to server S2 ;
end

B. Privacy-Preserving Hybrid Defense Strategy

During each training round, an honest edge device E_i randomly samples a mini-batch d_i from its local dataset D_i, and computes a benign gradient g_i accordingly. In contrast, malicious clients may submit adversarial gradients deliberately crafted to impair the performance of the global model. This behavior can be formally described as:

$$\mathbf{g}_i^t = \begin{cases} \nabla \mathcal{L}\left(\boldsymbol{w}_i^{t-1}, d_i\right), & \text{if device } i \text{ is honest} \\ \mathcal{A}_{\text{attack}}\ \left(t, \mathcal{G}_{\text{honest}}, \Theta\right), & \text{if device } i \text{ is malicious} \end{cases} \tag{9}$$

where $\mathcal{A}\text{attack}(\cdot)$ denotes an attack-generation function that may utilize the current training round t, the set of gradients from honest clients $\mathcal{G}\text{honest}$, and auxiliary parameters Θ.

While traditional defense schemes (e.g., Euclidean distance [21], cosine similarity [20]) are effective in simple scenarios, they may struggle under adaptive poisoning attacks. Furthermore, such methods usually require access to raw gradient data, introducing significant privacy risks. To mitigate this, DP2Guard troduces a privacy-preserving hybrid defense mechanism that enables malicious gradient detection without compromising data privacy. The entire process is presented in Algorithm 3, which consists of the following key steps:

1) Mean-Centering Gradient: Both servers independently perform mean-centering on the masked gradient shares they receive. Specifically, Server S_1 computes the centered vectors $\hat{\mathbf{g}}_i^{(1)}$ by subtracting the mean of all received masked shares $\tilde{\mathbf{g}}_i^{(1)}$. Likewise, S_2 processes $\tilde{\mathbf{g}}_i^{(2)}$:

$$\hat{\mathbf{g}}_i^{(1)} = \tilde{\mathbf{g}}_i^{(1)} - \overline{\tilde{\mathbf{g}}}^{(1)}, \quad \text{where} \quad \overline{\tilde{\mathbf{g}}}^{(1)} = \frac{1}{N}\sum_{j=1}^{N} \tilde{\mathbf{g}}_j^{(1)} \tag{10}$$

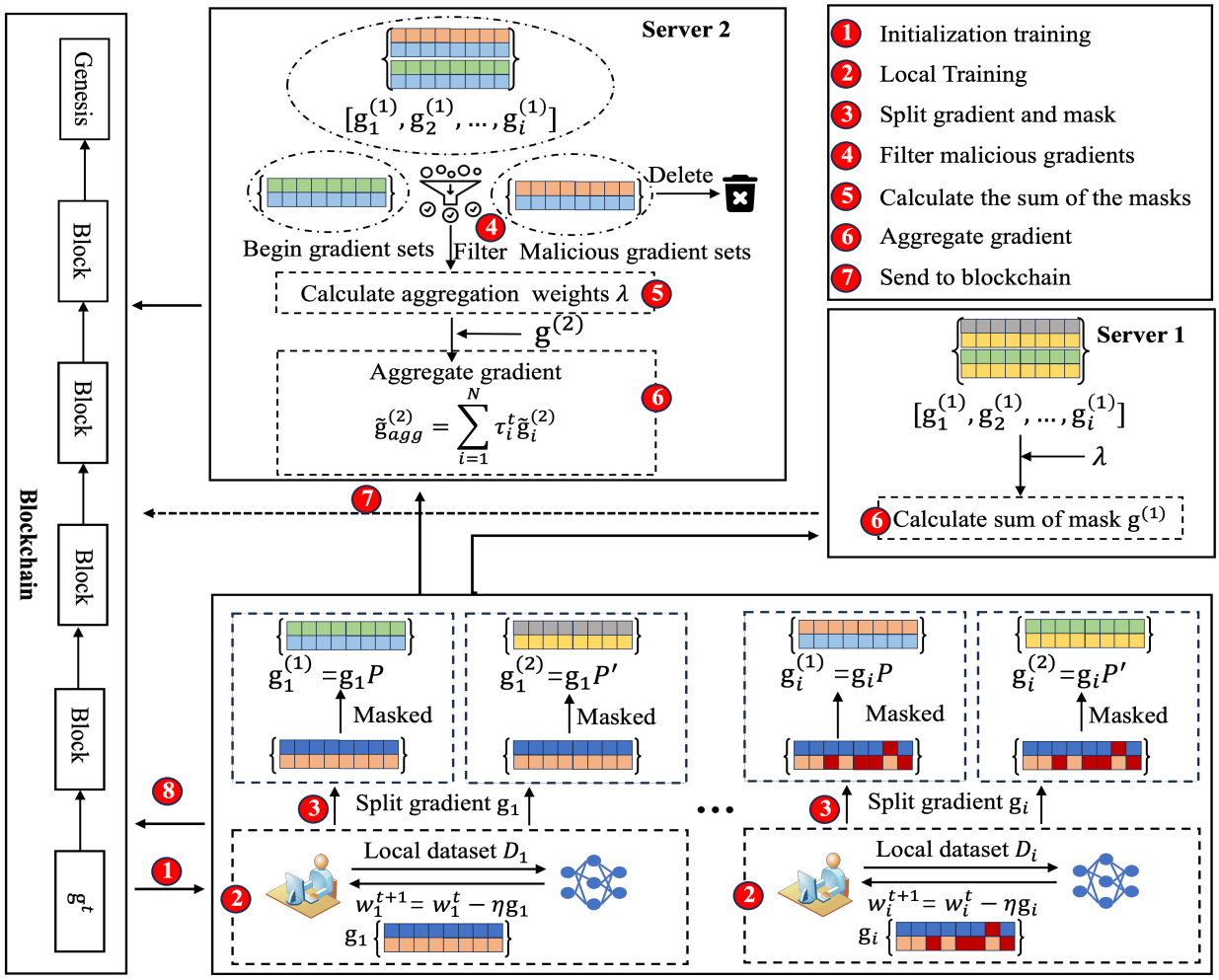


Fig. 3: A workflow of the DP2Guard.

$$\hat{g}_i^{(2)} = \tilde{g}_i^{(2)} - \overline{\tilde{g}}^{(2)}, \quad \text{where} \quad \overline{\tilde{g}}^{(2)} = \frac{1}{N}\sum_{j=1}^{N} \tilde{g}_j^{(2)} \tag{11}$$

Server S_1 then forwards the centered results $\hat{g}_i^{(1)}$ to S_2, which conducts hybrid detection without gaining access to any individual client's raw updates.

2) Gradient Reconstruction: Upon receiving $\hat{g}_i^{(1)}$ from S_1, server S_2 reconstructs the centered gradients as follows:

$$\hat{g}_i = \hat{g}_i^{(1)} + \hat{g}_i^{(2)} \tag{12}$$

where random masks can cancel each other out:

$$\begin{aligned} \hat{g}_i &= \left(\tilde{g}_i^{(1)} - \overline{\tilde{g}}^{(1)}\right) + \left(\tilde{g}_i^{(2)} - \overline{\tilde{g}}^{(2)}\right) \\ &= \left(g_i^{(1)} + r_i - \overline{g}^{(1)} - \overline{r}\right) + \left(g_i^{(2)} - r_i - \overline{g}^{(2)} + \overline{r}\right) \\ &= g_i^{(1)} + g_i^{(2)} - \overline{g}^{(1)} - \overline{g}^{(2)} \\ &= g_i - \overline{g} \end{aligned} \tag{13}$$

Here, $\overline{g} = \frac{1}{N}\sum_{j=1}^{N} g_j$ denotes the global mean of the true (unmasked) gradients.

3) Malicious Gradient Detection: At this stage, server $\mathcal{S}_2$ conducts a hybrid anomaly detection procedure in collaboration with $\mathcal{S}_1$. The process integrates spectral analysis and cosine similarity to the reliability of client gradients.

Algorithm 2: Gradient Mean-Central

Input : Masked shares:$\tilde{g}_i^{(1)}$ and $\tilde{g}_i^{(2)}$
Output: $\hat{g}_i$
/* Mean-Centering of Shares */
Server $\mathcal{S}_1$: Mean-Centering of First Shares;
Receive $\tilde{g}_i^{(1)}$ from all clients;
for *each edge device* i **do**
  Compute:$\hat{g}_i^{(1)} = \tilde{g}_i^{(1)} - \frac{1}{N}\sum_{j=1}^{N} \tilde{g}_j^{(1)}$;
end
Send $\hat{g}_i^{(1)}$ to $\mathcal{S}_2$;
Server $\mathcal{S}_2$: Construction of Centered Gradient
Receive $\hat{g}_i^{(1)}$ from $\mathcal{S}_1$ and $\tilde{g}_i^{(2)}$ from all clients;
for *each edge device* i **do**
  Compute:$\hat{g}_i^{(2)} = \tilde{g}_i^{(2)} - \frac{1}{N}\sum_{j=1}^{N} \tilde{g}_j^{(2)}$;
end
Obtain centered gradient: $\hat{g}_i = \hat{g}_i^{(1)} + \hat{g}_i^{(2)} = g_i - \overline{g}$;
Use $\{\hat{g}_i\}_{i=1}^{N}$ for following anomaly detection;

Spectral Projections: Server $\mathcal{S}_2$ aggregates all centered gradients into a matrix:

$$\mathbf{G} = [\hat{\mathbf{g}}_1, \hat{\mathbf{g}}_2, \ldots, \hat{\mathbf{g}}_N] \in \mathbb{R}^{d \times N} \tag{14}$$

where each column corresponds to a centered gradient vector from a client. Singular value decomposition (SVD) [42] is then applied to extract the principal components:

$$\mathbf{G} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \tag{15}$$

Here, $\mathbf{U} \in \mathbb{R}^{d \times d}$ contains the left singular vectors, $\mathbf{\Sigma} \in \mathbb{R}^{d \times N}$ is a diagonal matrix of singular values sorted in descending order, and $\mathbf{V} \in \mathbb{R}^{N \times N}$ contains the right singular vectors. The top right singular vector $\mathbf{v}_1$ (corresponding to the largest singular value) is selected as the dominant direction. Each client's gradient is projected onto $\mathbf{v}_1$ to compute its spectral score:

$$s_i = \left(\hat{\mathbf{g}}_i^\top \mathbf{v}_1\right)^2 \tag{16}$$

Clients with high s_i values are more likely to be malicious as their gradients deviate significantly from the principal component.

Cosine similarity: To further assess the directional consistency, the cosine similarity between each client's centered gradient and others is calculated as:

$$c_{ij} = \frac{\hat{\mathbf{g}}_i^\top \hat{\mathbf{g}}_j}{\|\hat{\mathbf{g}}_i\| \cdot \|\hat{\mathbf{g}}_j\|}, \quad \forall i \neq j \tag{17}$$

The median cosine similarity for each client is used as a reliability score:

$$c_i = \text{median}\left(\{c_{ij} \mid j \neq i\}\right) \tag{18}$$

Clustering-Based Detection: Next, a two-dimensional feature vector is constructed for each client:

$$\mathbf{f}_i = [s_i, c_i]^\top \tag{19}$$

A K-means clustering algorithm [43] (with a predefined number of clusters $K = 2$) is applied to these features. Clients in the largest cluster are considered benign, while those in smaller clusters are viewed as anomalous and excluded from aggregation. This approach allows efficient detection of malicious behaviors while preserving gradient privacy.

C. Tust Score–based Weight Calculation

The trust scoring mechanism is a critical factor in ensuring that individual client nodes contribute appropriately to the global model aggregation process [44]. Unlike prior approaches that rely solely on per-round behavior, DP2Guard introduces a time-evolving trust evaluation scheme that captures clients' long-term reliability.

In each round t, server $\mathcal{S}_2$ calculates the Euclidean distance between a client's feature vector $\mathbf{f}_i^{(t)}$ and the cluster centroid $\boldsymbol{\mu}^{(t)}$ as:

$$\text{Dis}_i^{(t)} = \left\|\mathbf{f}_i^{(t)} - \boldsymbol{\mu}^{(t)}\right\| \tag{20}$$

Based on this distance, the direct trust score for device E_i is computed as:

$$\gamma_i^{(t)} = 1/\left(1 + \text{Dis}_i^{(t)}\right) \tag{21}$$

Algorithm 3: Privacy-Preserving Hybrid Defense Strategy

Input: Mean-centered masked gradients $\hat{\mathbf{g}}_i^{(1)}$ from $\mathcal{S}_1$, masked gradients $\tilde{\mathbf{g}}_i^{(2)}$ from clients
Output: Trusted gradient set $\mathcal{G}$

Stack all $\hat{\mathbf{g}}_i$ into matrix $G \in \mathbb{R}^{N \times d}$
Perform SVD: $G = U\Sigma V^\top$; let v_1 be the top right singular vector
foreach *client* i **do**
  Compute spectral score: $s_i^{\text{svd}} \leftarrow |\langle \hat{\mathbf{g}}_i, v_1 \rangle|$
  Compute cosine similarity with others:
    $c_i \leftarrow \text{median}(\{\cos(\hat{\mathbf{g}}_i, \hat{\mathbf{g}}_j)\}_{j \neq i})$
  Construct feature vector: $\phi_i \leftarrow [s_i^{\text{svd}}, c_i]$
end
Apply Mean Shift clustering on $\{\phi_i\}_{i=1}^N$
Let $\mathcal{G} \leftarrow$ the cluster with the largest number of clients
return $\mathcal{G}$

To balance historical performance and recent behavior, DP2Guard utilizes a time-based trust mechanism. For each client i, the trust score is updated as:

$$\text{Trust}_i^{(t)} = \beta \cdot \text{Trust}_i^{(t-1)} + (1-\beta) \cdot \gamma_i^{(t)}, \quad \beta \in [0, 1) \tag{22}$$

The normalized trust value is used to compute the aggregation weight:

$$\tau_i^{(t)} = \frac{\text{Trust}_i^{(t)}}{\sum_{j=1}^N \text{Trust}_j^{(t)}}$$

Server $\mathcal{S}_2$ then applies these weights to aggregate the received masked gradients:

$$\tilde{\mathbf{g}}_{\text{agg}}^{(2)} = \sum_{i=1}^N \tau_i^{(t)} \cdot \tilde{\mathbf{g}}_i^{(2)} \tag{23}$$

Then, both $\tilde{\mathbf{g}}_{\text{agg}}^{(2)}$ and trust weights $\left\{\tau_i^{(t)}\right\}_{i=1}^N$ are published to the blockchain.

D. Adaptive Global Aggregation

In the stage, server $\mathcal{S}_1$ downloads the aggregated masked gradient $\tilde{\mathbf{g}}_{\text{agg}}^{(2)}$ and corresponding trust scores from blockchain. Using the same trust weights, $\mathcal{S}_1$ performs weighted aggregation over its local set of masked gradients:

$$\tilde{\mathbf{g}}_{\text{agg}}^{(1)} = \sum_{i=1}^N \tau_i^{(t)} \cdot \tilde{\mathbf{g}}_i^{(1)} \tag{24}$$

The final gradient can be reconstructed by combining the two:

$$\begin{aligned}
\tilde{\mathbf{g}}_{\text{agg}} &= \tilde{\mathbf{g}}_{\text{agg}}^{(1)} + \tilde{\mathbf{g}}_{\text{agg}}^{(2)} \\
&= \sum_{i=1}^N \tau_i^{(t)} \left(\mathbf{g}_i^{(1)} + \mathbf{r}_i + \mathbf{g}_i^{(2)} - \mathbf{r}_i\right) \\
&= \sum_{i=1}^N \tau_i^{(t)} \left(\mathbf{g}_i^{(1)} + \mathbf{g}_i^{(2)}\right) \\
&= \sum_{i=1}^N \tau_i^{(t)} \cdot \mathbf{g}_i
\end{aligned} \tag{25}$$

This aggregated result $\tilde{\mathbf{g}}_{\text{agg}}$ is then submitted to the blockchain to support model updates in the next training round.

Algorithm 4: Trust-Weighted Adaptive Aggregation

Input: Trust weights $\{\tau_i^{(t)}\}_{i=1}^N$, Masked gradients $\{\tilde{\mathbf{g}}_i^{(1)}\}_{i=1}^N$ at $\mathcal{S}_1$, $\{\tilde{\mathbf{g}}_i^{(2)}\}_{i=1}^N$ at $\mathcal{S}_2$

Output: Global aggregated gradient $\tilde{\mathbf{g}}_{\text{agg}}$

At Server $\mathcal{S}_2$:

Compute weighted aggregation of second shares: $\tilde{\mathbf{g}}_{\text{agg}}^{(2)} \leftarrow \sum_{i=1}^N \tau_i^{(t)} \cdot \tilde{\mathbf{g}}_i^{(2)}$

Send $\tilde{\mathbf{g}}_{\text{agg}}^{(2)}$ and $\{\tau_i^{(t)}\}$ to blockchain

At Server $\mathcal{S}_1$:

Download the $\tilde{\mathbf{g}}_{\text{agg}}^{(2)}$ and $\{\tau_i^{(t)}\}$ from blockchain

Compute weighted aggregation of first shares: $\tilde{\mathbf{g}}_{\text{agg}}^{(1)} \leftarrow \sum_{i=1}^N \tau_i^{(t)} \cdot \tilde{\mathbf{g}}_i^{(1)}$

Reconstruct the global aggregated gradient: $\tilde{\mathbf{g}}_{\text{agg}} \leftarrow \tilde{\mathbf{g}}_{\text{agg}}^{(1)} + \tilde{\mathbf{g}}_{\text{agg}}^{(2)}$

return $\tilde{\mathbf{g}}_{agg}$

VI. Security Analysis

In this section, we prove the privacy and robustness of DP2Gurd, and analyze the computational and communication complexity in DP2Guard.

Theorem 1 (Privacy protection against semi-honest servers)**.** *Under the assumption that servers S_1 and S_2 do not collude, the execution of DP2Guard does not leak any information about the true local gradients $\mathbf{g}_i$ to either server individually.*

Proof. We use a standard hybrid argument [45] to prove the Theorem 1. Let $REAL_{\mathcal{A}}^{\Pi}$ denote the view of an adversary $\mathcal{A}$ during the real execution of protocol Π, and let $IDEAL_{\mathcal{A}^*}^{\mathcal{F}}$ denote the ideal execution where a trusted functionality $\mathcal{F}$ performs all privacy-sensitive operations. We construct a sequence of hybrid experiments, $H_0, H_1, \ldots, H_5$, such that each pair of consecutive hybrids is computationally indistinguishable.

Hyb$_0$: This hybrid corresponds to the actual execution of the DP2Guard protocol (as described in Section V). Each client computes its true local gradient $\mathrm{g}_i^{(t)}$, splits it into two parts $\mathrm{g}_i^{(1)}$ and $\mathrm{g}_i^{(2)}$, and applies a random mask $\mathbf{r}_i$ to obtain $\tilde{\mathrm{g}}_i^{(1)} = \mathrm{g}_i^{(1)} + \mathrm{r}_i$ and $\tilde{\mathrm{g}}_i^{(2)} = \mathrm{g}_i^{(2)} - \mathrm{r}_i$, which are sent to $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively.

Hyb$_1$: In this hybrid, the simulator replaces the masked gradient share $\tilde{\mathbf{g}}_i^{(1)}$ sent to $\mathcal{S}_1$ with a uniformly random vector $\chi_i^{(1)}$ of the same dimension. The second share $\tilde{\mathbf{g}}_i^{(2)}$ remains unchanged. Since $\chi_i^{(1)}$ is independent and indistinguishable from a valid masked share, and $\mathcal{S}_1$ does not access the corresponding second share, this view is computationally indistinguishable from $\mathbf{Hyb}_0$.

Hyb$_2$: Building on $\mathbf{Hyb}_1$, this hybrid replaces the share sent to $\mathcal{S}_2$ as well. Each client sends a uniformly random vector $\chi_i^{(2)}$ to $\mathcal{S}_2$, where $\chi_i^{(1)} + \chi_i^{(2)} = \chi_i$ and χ_i is sampled uniformly at random. As both shares are now random and the servers are non-colluding, this hybrid is indistinguishable from $\mathbf{Hyb}_1$.

Hyb$_3$: In this hybrid, we simulate the internal computations of $\mathcal{S}_1$ that are transmitted to $\mathcal{S}_2$, such as the mean-centered gradients $\hat{\mathbf{g}}_i^{(1)}$. Specifically, for each client i, the simulator replaces $\hat{\mathbf{g}}_i^{(1)}$ with a randomly generated vector $\psi_i^{(1)}$ sampled from a distribution that is statistically consistent with that of the true centered values. Since $\mathcal{S}_2$ does not observe the raw gradients or the corresponding masked shares held by $\mathcal{S}_1$, its view remains computationally indistinguishable from $\mathbf{Hyb}_2$.

Hyb$_4$: Similarly, this hybrid simulates the internal computations of $\mathcal{S}_2$ that are sent to $\mathcal{S}_1$ (e.g., trust scores or aggregation results). Each intermediate value is replaced by a random vector $\psi_i^{(2)}$ drawn from a consistent distribution. As $\mathcal{S}_1$ does not observe the raw gradients or masked shares held by $\mathcal{S}_2$, its view remains indistinguishable from $\mathbf{Hyb}_3$.

Hyb$_5$: Finally, we simulate the messages that $\mathcal{S}_1$ sends back to the clients (e.g., global model updates) using random values that are statistically indistinguishable from actual aggregated updates. This completes the transition from the real-world protocol to an ideal-world simulation.

By the transitivity of computational indistinguishability, we conclude that the output of the simulator SIM is indistinguishable from the real execution:

$$\mathrm{REAL}_{\mathcal{A}}^{\Pi} \approx \mathrm{IDEAL}_{\mathcal{A}^*}^{\mathcal{F}}$$

Hence, DP2Guard preserves the privacy of local gradients under the semi-honest model, ensuring that neither $\mathcal{S}_1$ nor $\mathcal{S}_2$ can learn any sensitive information individually. □

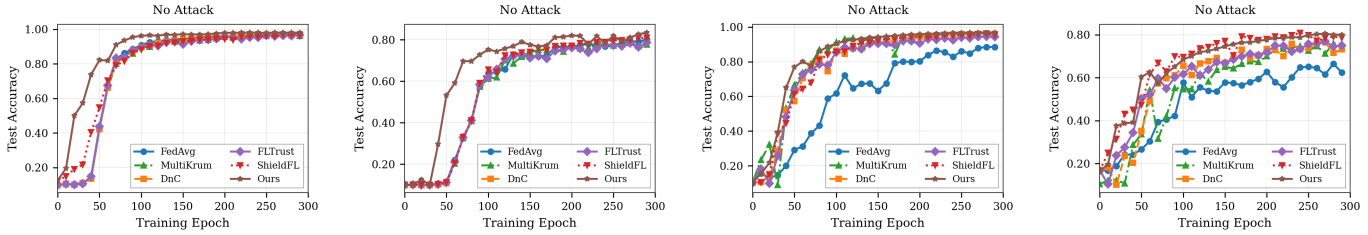
VII. Performance Evaluation

A. Experimental Settings

All experiments were conducted on a high-performance workstation running Ubuntu 20.04 LTS, equipped with an Intel i9 CPU, 64 GB of RAM, and four NVIDIA RTX 4090 GPUs. The FL framework was implemented using PyTorch version 1.6.0 and Python version 3.8.10.

Datasets and Model Architectures: We evaluate our proposed scheme on two widely used benchmark datasets: MNIST and Fashion-MNIST. The MNIST dataset contains 60,000 training and 10,000 testing grayscale images of handwritten digits, each of size 28×28 pixels and classified into 10 categories. The Fashion-MNIST dataset consists of 60,000 training and 10,000 testing grayscale images representing various types of clothing items, also distributed across 10 classes. For both datasets, we employ the LeNet-5 model [46] to perform the training tasks. We adopt stochastic gradient descent (SGD) as the optimizer, with a learning rate $\eta = 0.01$ and a batch size of 32.

Datasets Distribution: We conduct experiments under both independent and identically distributed (IID) and non-IID data settings. In the IID setting, data samples are randomly and uniformly distributed among clients. For the non-IID setting, we adopt the commonly used Dirichlet distribution strategy [47] to partition the datasets among clients. According to the configuration in prior work [48], the data heterogeneity parameter is set to $\alpha = 0.5$, where a smaller α indicates higher



(a) MNIST IID (b) Fashion IID (c) MNIST Non-IID (d) Fashion Non-IID

Fig. 4: Impact of training iterations on the effectiveness of defense strategies on MNIST and Fashion-MNIST datasets under IID and Non-IID settings.

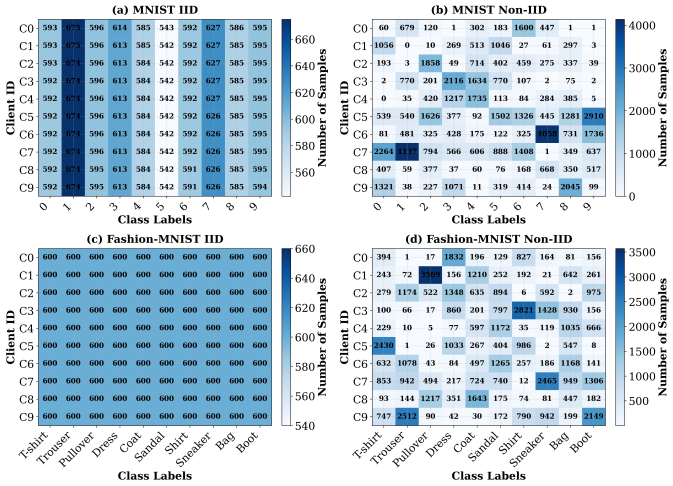


Fig. 5: Class-wise sample distribution heatmap for 10 randomly selected clients in a FL setup with 50 clients. The datasets used are FashionMNIST and MNIST. Data partitioning is performed using a Dirichlet distribution with a heterogeneity parameter of $\alpha = 0.5$ to simulate non-IID conditions.

data skewness and greater class imbalance across clients. Fig. 5 illustrates the class distribution of 10 selected clients out of a total of 50 under this partitioning scheme.

FL System Settings: In our experiments, we set up 50 clients for FL training process. Each client computes local gradients and sends them to the central server, which aggregates these updates using different aggregation rules. The FedAvg [37] is adopted as the default aggregation mechanism. The training is conducted over 300 global communication rounds, with each client performing one local epoch per round. To assess the impact of adversarial behavior, we set the ratio of adversary $Adv_{\text{ratio}} = \{0\%, 10\%, 20\%, 30\%, 40\%\}$.

Baselines: We compare DP2Guard with four state-of-the-art defense methods: Multi-Krum [20], DnC [22], and FLTrust [21], which do not incorporate privacy protection mechanisms, as well as ShieldFL [26], which support cosine similarity detection for ciphertext gradients. These baselines have been discussed in detail in Section II.

Attack Type: To comprehensively evaluate the robustness of DP2Guard, we consider four advanced poisoning attack scenarios:

1) **Label-Flipping attack:** Label-flipping attacks [49] manipulate the labels of selected training samples on the malicious client, causing the local model to learn incorrect class boundaries. For example, samples with the true label 2 may be maliciously relabeled as 7 to cause targeted misclassification. In our experiments, each malicious client flips 30% of its local labels.
2) **Fang Attack:** In this adaptive attack, the adversary is assumed to have full knowledge of all benign gradients and the aggregation rule. Following the setting in [39], we set the perturbation threshold $\gamma_{\min} = 1e-5$, and the optimization step size to $\alpha = 0.5$. The unit perturbation direction is determined by the sign of the pre-aggregation gradient computed locally by the adversary.
3) **Min-Max [22]:** In this adaptive attack, the adversary has access to all benign client gradients but does not know the aggregation rule. An efficient binary search algorithm is used to determine the optimal perturbation coefficient λ. The initial perturbation factor is set to $\gamma = 10$, with an initial step size of $\gamma/2 = 5$, and the search stops when the perturbation threshold $\gamma_{\min} = 1 \times 10^{-5}$ is reached. The direction of the perturbation is defined as the unit vector of the mean of the known benign gradients.
4) **Min-Sum Attack [22]:** Min-Sum follows the same procedure as Min-Max but changes the objective: instead of maximizing the largest deviation, it minimizes the sum of squared distances to all benign gradients. A binary search is used with $\gamma = 10$, step size $= 5$, and stopping threshold $\gamma_{\min} = 1 \times 10^{-5}$. The perturbation direction is the unit vector of the mean benign gradient.

B. Comparison Analysis

1) **Impact of Different Number of Iterations:** To evaluate the convergence efficiency of different aggregation algorithms, we conducted experiments over 300 global communication rounds on the MNIST and Fashion-MNIST datasets under a no-attack scenario, using both IID and non-IID data settings. As shown in Fig. 4, all methods achieve high final accuracy under the IID setting. However, DP2Guard stabilizes within the first 50 rounds, whereas FedAvg and MultiKrum typically require over 100 rounds to converge. Under the non-IID setting, the performance gap widens. In particular,

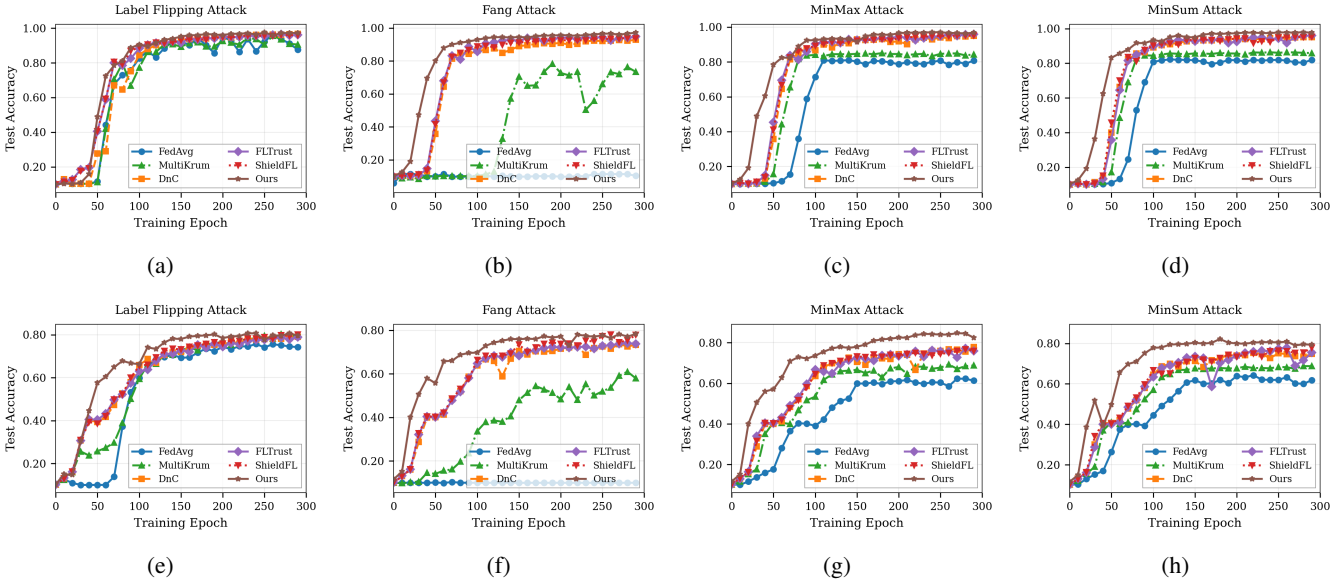


Fig. 6: impact of defense strategies against four types poisoning attacks on MNIST(a-d) and Fashion-MNIST(e-f) datasets under IID setting.

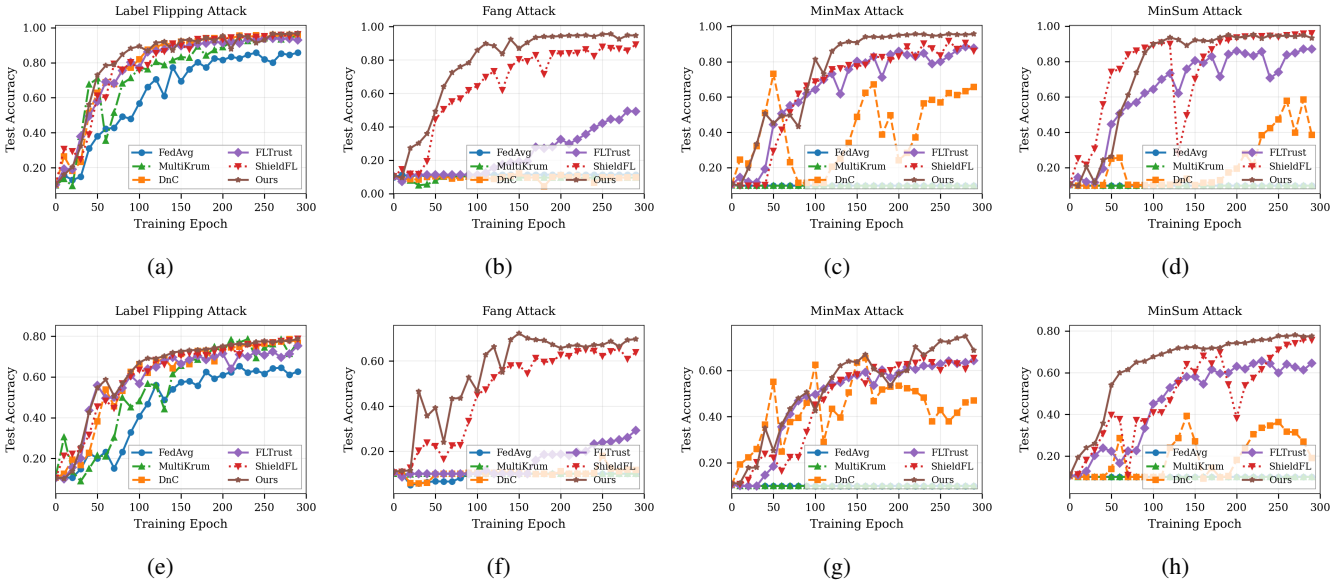


Fig. 7: Impact of training iterations on the effectiveness of defense strategies against four poisoning attacks on MNIST(a-d) and Fashion-MNIST(e-f) datasets under Non-IID setting.

on the Fashion-MNIST dataset, FedAvg and MultiKrum show noticeable fluctuations during training, indicating sensitivity to data heterogeneity. In contrast, ShieldFL and DP2Guard exhibit smoother convergence curves. Overall, DP2Guard achieves faster convergence and better training stability under both IID and non-IID conditions.

2) **Impact of Different Poisoning Attack in IID:** To evaluate the robustness of various defense mechanisms under IID data settings, we examine the performance of all methods against four representative poisoning attacks: Label Flipping, Fang, Min-Max, and Min-Sum, each conducted with a 40% adversary ratio. Results are illustrated in Fig. 6 for both MNIST and Fashion-MNIST datasets.

Under the Label Flipping attack, most defense methods on the MNIST dataset remain effective and achieve high accuracy. As illustrated in Fig. 6a, MultiKrum exhibits noticeable fluctuations during training, indicating limited stability. In contrast, DP2Guard maintains a smooth and stable learning trajectory, demonstrating strong robustness to poisoned label information. On the more challenging Fashion-MNIST dataset, as shown in Fig. 6e, both FedAvg and MultiKrum exhibit slow convergence, while DP2Guard achieves stable convergence and reaches a final accuracy of 78.4%, outperforming

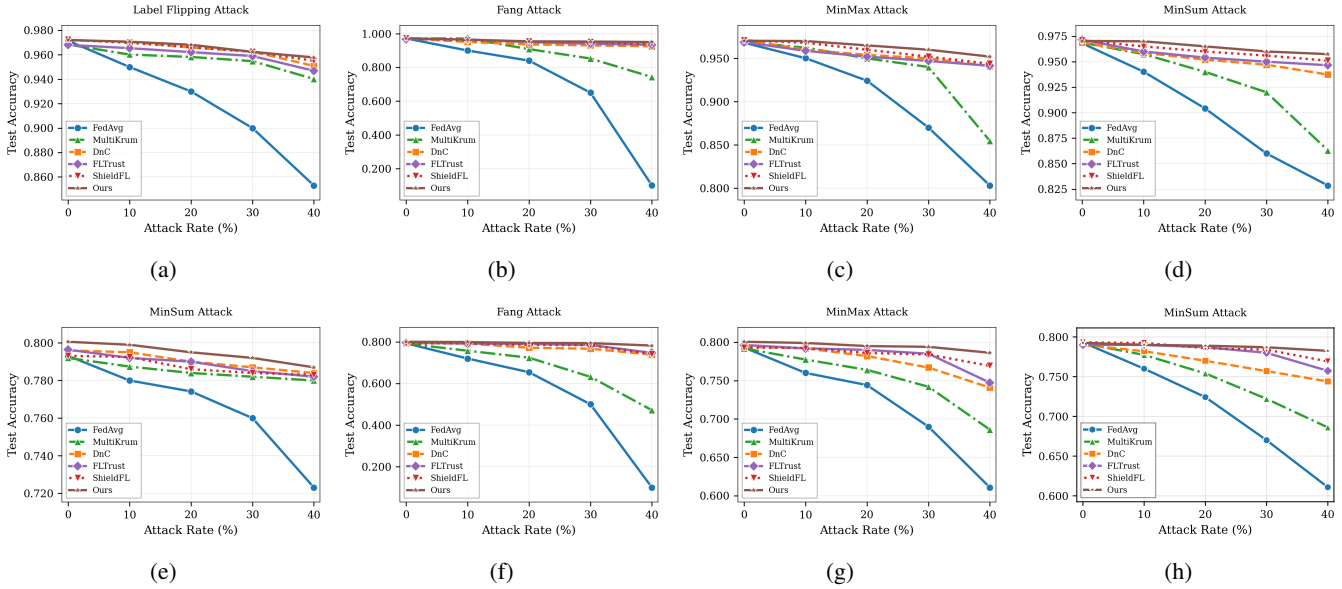


Fig. 8: Impact of varying malicious clients ratios (0%–40%) on the performance of different defense strategies against four poisoning attacks on the MNIST (a-d) and Fashion-MNIST (e-f) datasets in the IID setting.

existing baselines in both robustness and learning efficiency.

Under the Fang Attack, which involves highly adaptive gradient manipulation, the results shown in Fig. 6b and 6f illustrate that DP2Guard consistently outperforms all baselines. On MNIST, it converges early and achieves 94.26% accuracy, whereas FedAvg completely fails, dropping to 10.24%. Similar trends are observed on Fashion-MNIST, where DP2Guard achieves 79.4%, while ShieldFL, FLTrust, and DnC converge more slowly and to lower accuracies. Although MultiKrum offers some robustness, its accuracy fluctuates heavily and remains notably lower than other defenses.

Under the Min-Max attack, which disrupts the aggregation process by maximizing the global loss while minimizing local objectives, DP2Guard continues to exhibit strong robustness, as illustrated in Fig. 6c and 6g. On the MNIST dataset, it achieves rapid convergence in the early stages and reaches a final accuracy of 95.5%, outperforming FedAvg (79.85%) and MultiKrum (85%), the latter of which suffers from instability during training. ShieldFL also performs well, achieving 95.32% accuracy, though slightly lower than DP2Guard. On Fashion-MNIST dataset, DP2Guard maintains its advantage, reaching a final accuracy of 79.97%, while FLTrust, ShieldFL, and DnC lag behind both in convergence speed and final performance under this type of optimization-based poisoning attack.

For Min-Sum Attack, which manipulates model updates by minimizing the sum of gradients to mislead the global model direction, most robust aggregation mechanisms demonstrate stable performance under this subtle adversarial manipulation. As shown in Fig. 6d and 6h, DP2Guard, by effectively detecting and suppressing abnormal gradient patterns, achieves smooth convergence on both MNIST and Fashion-MNIST, reaching final accuracies of 96.24% and 78.3%, respectively. These results surpass those of ShieldFL (95.7%, 77.4%), FLTrust (94.4%, 76.25%), and DnC (93.2%, 74.39%). In contrast, MultiKrum performs considerably worse, with accuracies of 86.5% and 68.73%.

3) **Impact of Different Poisoning Attack in Non-IID:** To further evaluate the robustness of defense methods under Non-IID settings, we compare all approaches against four poisoning attacks, with results shown in Fig.7a-7h. In the Label Flipping attack, most methods show comparable performance across both datasets. However, FLTrust suffers a notable accuracy drop on Fashion-MNIST, falling behind ShieldFL, DnC, and DP2Guard, indicating limited robustness under data heterogeneity. The Fang attack severely disrupts most defenses—FedAvg, MultiKrum, and DnC nearly collapse, with accuracies dropping below 20%. In contrast, DP2Guard and ShieldFL retain consistent performance, with DP2Guard outperforming ShieldFL on both datasets. Under the Min-Max attack, most methods experience unstable convergence, especially DnC. DP2Guard achieves robust and efficient training, reaching 90.24% on MNIST and 68.91% on Fashion-MNIST. In the Min-Sum scenario, FedAvg and MultiKrum fail to make progress, while only DP2Guard maintains high and stable accuracy—91.04% on MNIST and 72.6% on Fashion-MNIST.

Overall, across all four attack types, DP2Guard consistently demonstrates superior robustness, faster convergence, and higher final accuracy across both IID and non-IID data settings. These results validate the method's adaptability and effectiveness in resisting a wide range of poisoning threats in FL.

4) **Impact of Different Poisoning Ratio:** To further assess

the robustness of defense mechanisms against varying intensities of poisoning, we conduct experiments under an independent and identically distributed (IID) data setting, gradually increasing the poisoning ratio from 0% to 40%. The evaluation covers four representative attack types: Label Flipping, Fang, Min-Max, and Min-Sum, with results illustrated in Fig. 8a-8h. Across all attack scenarios, we observe a clear decline in model accuracy for most baseline methods (e.g., FedAvg, MultiKrum) as the poisoning ratio increases. In contrast, the proposed DP2Guard consistently exhibits minimal performance degradation, maintaining stable accuracy even under higher poisoning rates. These results highlight DP2Guard's strong resilience and adaptability against escalating adversarial threats.

5) **Complexity Analysis:** We evaluate the overall computational and communication complexity of DP2Guard from both the client and server perspectives, as shown in table III. Notably, the cost of local model training is excluded from this analysis, as it is standard across all federated learning (FL) protocols. On the client side, each device E_i performs gradient splitting and masking with a computational overhead of $\mathcal{O}(d)$, where d is the gradient dimension. The masked shares are transmitted to two servers, incurring a communication cost of $\mathcal{O}(d)$. On the server side, both $\mathcal{S}_1$ and $\mathcal{S}_2$ execute mean-centering and gradient reconstruction operations across N clients, each with complexity $\mathcal{O}(Nd)$. During hybrid anomaly detection, $\mathcal{S}_2$ applies SVD to the $d \times N$ gradient matrix, which requires $\mathcal{O}(\min Nd^2, dN^2)$ time. Pairwise cosine similarity calculations among N clients incur an additional cost of $\mathcal{O}(N^2 d)$. Trust score updating and normalization impose only $\mathcal{O}(N)$ cost per round. Finally, both servers perform weighted aggregation of masked gradients using trust scores, with a cost of $\mathcal{O}(Nd)$. Unlike HE-based or MPC-based solutions, which often incur substantial computational and communication overhead, DP2Guard achieves efficient privacy-preserving gradient processing with minimal client-side burden, making it well-suited for resource-constrained IIoT environments.

TABLE III: Computation and Communication Complexity

Operation	Computation Cost	Communication Cost
Client: Oper. 1	$\mathcal{O}(d)$	$\mathcal{O}(d)$ to each server
Server: Oper. 2	$\mathcal{O}(Nd)$	$\mathcal{O}(Nd)$
Server: Oper. 3	$\mathcal{O}(Nd^2) + \mathcal{O}(N^2 d)$	–
Server: Oper. 4	$\mathcal{O}(N)$	–
Aggregation	$\mathcal{O}(Nd)$	–

Notes: Oper. 1 = Gradient masking; Oper. 2 = Mean-centering and reconstruction; Oper. 3 = Detection (SVD + similarity); Oper. 4 = Trust score update.

VIII. Conclusion

In this paper, we presented DP2Guard, a lightweight PPFL framework designed to address the dual challenges of privacy leakage and vulnerability to model poisoning. To reduce computational overhead, DP2Guard replaces traditional cryptographic techniques with an efficient gradient masking strategy that protects the privacy of local model updates. To improve robustness, we designed a hybrid anomaly detection mechanism that combines cosine similarity and spectral analysis to effectively identify malicious updates. Furthermore, the blockchain is integrated to provide a secure and auditable training process. Extensive experimental results demonstrate that DP2Guard achieves superior performance compared to state-of-the-art defenses, offering enhanced robustness and lower communication and computation costs. In this work, the current design relies on the assumption of two non-colluding servers, without considering potential collusion risks. Future work will focus on relaxing this assumption by designing more flexible multi-party collaboration protocols, thereby enhancing DP2Guard's applicability and security in practical scenarios.

References

[1] B. Yang, O. Fagbohungbe, X. Cao, C. Yuen, L. Qian, D. Niyato, and Y. Zhang, "A joint energy and latency framework for transfer learning over 5g industrial edge networks," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 531–541, 2021.

[2] S. Huang, S. Wang, R. Wang, M. Wen, and K. Huang, "Reconfigurable intelligent surface assisted mobile edge computing with heterogeneous learning tasks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 2, pp. 369–382, 2021.

[3] A. Arunan, Y. Qin, X. Li, and C. Yuen, "A federated learning-based industrial health prognostics for heterogeneous edge devices using matched feature extraction," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 3, pp. 3065–3079, 2024.

[4] H. Wu, X. Chen, and K. Huang, "Device-edge cooperative fine-tuning of foundation models as a 6g service," *IEEE Wireless Communications*, vol. 31, no. 3, pp. 60–67, 2024.

[5] X. Yuan, H. Tian, X. Zhang, H. Du, N. Zhang, K. Huang, and L. Cai, "Digital twin-driven madrl approaches for communication-computing-control co-optimization," *IEEE Journal on Selected Areas in Communications*, pp. 1–1, 2025.

[6] F. Naeem, M. Ali, G. Kaddoum, C. Huang, and C. Yuen, "Security and privacy for reconfigurable intelligent surface in 6g: A review of prospective applications and challenges," *IEEE Open Journal of the Communications Society*, vol. 4, pp. 1196–1217, 2023.

[7] H. Xie, M. Xia, P. Wu, S. Wang, and K. Huang, "Decentralized federated learning with asynchronous parameter sharing for large-scale iot networks," *IEEE Internet of Things Journal*, vol. 11, no. 21, pp. 34 123–34 139, 2024.

[8] Z. Lin, X. Li, V. K. N. Lau, Y. Gong, and K. Huang, "Deploying federated learning in large-scale cellular networks: Spatial convergence analysis," *IEEE Transactions on Wireless Communications*, vol. 21, no. 3, pp. 1542–1556, 2022.

[9] L. You, Z. Guo, B. Zuo, Y. Chang, and C. Yuen, "Slmfed: A stage-based and layerwise mechanism for incremental federated learning to assist dynamic and ubiquitous iot," *IEEE Internet of Things Journal*, vol. 11, no. 9, pp. 16 364–16 381, 2024.

[10] Z. Lin, G. Zhu, Y. Deng, X. Chen, Y. Gao, K. Huang, and Y. Fang, "Efficient parallel split learning over resource-constrained wireless edge networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 10, pp. 9224–9239, 2024.

[11] K. Wei, J. Li, M. Ding, C. Ma, H. Su, B. Zhang, and H. V. Poor, "User-level privacy-preserving federated learning: Analysis and performance optimization," *IEEE Transactions on Mobile Computing*, vol. 21, no. 9, pp. 3388–3401, 2022.

[12] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Differentially private asynchronous federated learning for mobile edge computing in urban informatics," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2134–2143, 2020.

[13] B. Han, B. Li, R. Jurdak, P. Zhang, H. Zhang, P. Feng, and C. Yuen, "Pbfl: A privacy-preserving blockchain-based federated learning framework with homomorphic encryption and single masking," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[14] W. Yang, X. Wang, Z. Guan, L. Wu, X. Du, and M. Guizani, "Securesl: A privacy-preserving vertical cooperative learning scheme for web 3.0," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 5, pp. 3983–3994, 2024.

[15] R. Bi, J. Xiong, C. Luo, J. Ning, X. Liu, Y. Tian, and Y. Zhang, "Communication-efficient privacy-preserving neural network inference via arithmetic secret sharing," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 6722–6737, 2024.

[16] Z. Zhang and Y. Li, "Nspfl: A novel secure and privacy-preserving federated learning with data integrity auditing," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 4494–4506, 2024.

[17] M. Chen, D. Gündüz, K. Huang, W. Saad, M. Bennis, A. V. Feljan, and H. V. Poor, "Distributed learning in wireless networks: Recent progress and future challenges," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3579–3605, 2021.

[18] P. Wang, W. Sun, H. Zhang, W. Ma, and Y. Zhang, "Distributed and secure federated learning for wireless computing power networks," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 7, pp. 9381–9393, 2023.

[19] Z. Yuan, Y. Tian, Z. Zhou, T. Li, S. Wang, and J. Xiong, "Trustworthy federated learning against malicious attacks in web 3.0," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 5, pp. 3969–3982, 2024.

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

[21] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," *arXiv preprint arXiv:2012.13995*, 2020.

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

[23] X. Li, M. Wen, S. He, R. Lu, and L. Wang, "A scheme for robust federated learning with privacy-preserving based on krum agr," in *2023 IEEE/CIC International Conference on Communications in China (ICCC)*, 2023, pp. 1–6.

[24] M. Shayan, C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Biscotti: A blockchain system for private and secure federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1513–1525, 2021.

[25] P. Mai, R. Yan, and Y. Pang, "Rflpa: A robust federated learning framework against poisoning attacks with secure aggregation," *Advances in Neural Information Processing Systems*, vol. 37, pp. 104 329–104 356, 2024.

[26] Z. Ma, J. Ma, Y. Miao, Y. Li, and R. H. Deng, "Shieldfl: Mitigating model poisoning attacks in privacy-preserving federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1639–1654, 2022.

[27] Y. Dong, X. Chen, K. Li, D. Wang, and S. Zeng, "Flod: Oblivious defender for private byzantine-robust federated learning with dishonest-majority," in *European Symposium on Research in Computer Security*. Springer, 2021, pp. 497–518.

[28] X. Feng, W. Cheng, C. Cao, L. Wang, and V. S. Sheng, "Dpfla: defending private federated learning against poisoning attacks," *IEEE Transactions on Services Computing*, 2024.

[29] B. Jiang, J. Li, H. Wang, and H. Song, "Privacy-preserving federated learning for industrial edge computing via hybrid differential privacy and adaptive compression," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1136–1144, 2023.

[30] C. Gu, X. Cui, X. Zhu, and D. Hu, "Fl2dp: Privacy-preserving federated learning via differential privacy for artificial iot," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 4, pp. 5100–5111, 2024.

[31] Z. Li, H. Chen, Z. Ni, Y. Gao, and W. Lou, "Towards adaptive privacy protection for interpretable federated learning," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 14 471–14 483, 2024.

[32] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[33] X. Fu, L. Xiong, F. Li, X. Yang, and N. Xiong, "Blockchain-based efficiently privacy-preserving federated learning framework using shamir secret sharing," *IEEE Transactions on Consumer Electronics*, pp. 1–1, 2024.

[34] B. D. Manh, C.-H. Nguyen, D. T. Hoang, and D. N. Nguyen, "Homomorphic encryption-enabled federated learning for privacy-preserving intrusion detection in resource-constrained iov networks," in *2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall)*, 2024, pp. 1–6.

[35] B. Li, Y. Guo, Q. Du, Z. Zhu, X. Li, and R. Lu, "$\boldsymbol{P^3}$: Privacy-preserving prediction of real-time energy demands in ev charging networks," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 3, pp. 3029–3038, 2023.

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

[37] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[38] C. Fung, C. J. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 301–316.

[39] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to {Byzantine-Robust} federated learning," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

[40] D. Reijsbergen, A. Maw, T. T. A. Dinh, W.-T. Li, and C. Yuen, "Securing smart grids through an incentive mechanism for blockchain-based data sharing," in *Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy*, 2022, pp. 191–202.

[41] M. B. Mollah, J. Zhao, D. Niyato, Y. L. Guan, C. Yuen, S. Sun, K.-Y. Lam, and L. H. Koh, "Blockchain for the internet of vehicles towards intelligent transportation systems: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4157–4185, 2020.

[42] A. Hoecker and V. Kartvelishvili, "Svd approach to data unfolding," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 372, no. 3, pp. 469–481, 1996.

[43] H. Liu, J. An, W. Xu, X. Jia, L. Gan, and C. Yuen, "K-means based constellation optimization for index modulated reconfigurable intelligent surfaces," *IEEE Communications Letters*, vol. 27, no. 8, pp. 2152–2156, 2023.

[44] Y. Tong, J. Chen, M. Xu, J. Kang, Z. Xiong, D. Niyato, C. Yuen, and Z. Han, "Multi-attribute auction-based resource allocation for twins migration in vehicular metaverses: A gpt-based drl approach," *IEEE Transactions on Cognitive Communications and Networking*, 2024.

[45] Y. Lindell, "How to simulate it–a tutorial on the simulation proof technique," *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, pp. 277–346, 2017.

[46] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 2002.

[47] G. Chen, X. Li, L. You, A. M. Abdelmoniem, Y. Zhang, and C. Yuen, "A data poisoning resistible and privacy protection federated-learning mechanism for ubiquitous iot," *IEEE Internet of Things Journal*, vol. 12, no. 8, pp. 10 736–10 750, 2025.

[48] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International conference on artificial intelligence and statistics*. PMLR, 2020, pp. 2938–2948.

[49] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389*, 2012.

Baofu Han received the M.S. degree from the School of Information Science and Engineering, Shenyang University of Technology, Shenyang, China, in 2021. He is currently pursuing the Ph.D. degree with the School of Cyber Science and Engineering, Southeast University, Nanjing, China. He is also a joint Ph.D. student at the School of Computing, National University of Singapore, Singapore. His research interests include blockchain, federated learning, game theory, data privacy preservation and their applications in the Internet-of-Things (IoT), Intelligent Transportation Systems (ITS) and Smart Cities.

Bing Li was born in Nanjing, China, in 1968. He received the B.S. degree in electronics science and technology and the Ph.D. degree from Southeast University, Nanjing, China, in 1991 and 2004, respectively. He is currently a Professor and a Tutor of doctoral students with the School of Microelectronics, the School of Cyber Science and Engineering, Southeast University, and the Director of the Joint Research Center for Advanced Cloud System.

His main research is the efficient and secure integrated circuits and system, including data compression, data encryption, physical unclonable functions, the blockchain, Internet of Things, and the area of information security.

Yining Qi (Member, IEEE) received the BS, MS and PhD degrees from Tsinghua University, in 2012, 2015 and 2023, respectively. She is now working as a postdoctoral fellow with the School of Computer Science and Technology, Huazhong University of Science and Technology. Her research interests include data security, privacy computing and federated learning.

Raja Jurdak (Senior Member, IEEE) received the M.S. and Ph.D. degrees from the University of California at Irvine. He is a Professor of distributed systems and the Chair of applied data sciences with Queensland University of Technology, where he is the Head of School of Compouter Science, co-Director of the Energy Transition Centre, and Director of the Trusted Networks Laboratory. Previously, he established and led the Distributed Sensing Systems Group, Data61, CSIRO. He also spent time as a Visiting Academic with MIT and Oxford University in 2011 and 2017, respectively. He has published over 300 peer-reviewed publications, including three authored books on IoT, blockchain, and cyberphysical systems. His publications have attracted over 18,000 citations, with an H-index of 57. His research interests include trust, mobility, and energy efficiency in networks. He serves on the organizing and technical program committees of top international conferences, including Percom, ICBC, IPSN, WoWMoM, and ICDCS. He serves on the editorial boards of Ad Hoc Networks and IEEE Transactions on Network and Service Management. He is an IEEE Computer Society Distinguished Visitor, and IEEE Senior Member, and an Adjunct Professor at University of New South Wales.

Kaibin Huang (Fellow, IEEE) received the B.Eng. and M.Eng. degrees from the National University of Singapore and the Ph.D. degree from The University of Texas at Austin, all in electrical engineering. He is the Philip K H Wong Wilson K L Wong Professor in Electrical Engineering and the Department Head at the Dept. of Electrical and Electronic Engineering, The University of Hong Kong (HKU), Hong Kong. His work was recognized with seven Best Paper awards from the IEEE Communication Society. He is a member of the Engineering Panel of Hong Kong Research Grants Council (RGC) and a RGC Research Fellow (2021 Class). He has served on the editorial boards of five major journals in the area of wireless communications and co-edited ten journal special issues. He has been active in organizing international conferences such as the 2014, 2017, and 2023 editions of IEEE Globecom, a flagship conference in communication. He has been named as a Highly Cited Researcher by Clarivate in the last six years (2019-2024) and an AI 2000 Most Influential Scholar (Top 30 in Internet of Things) in 2023-2024. He was an IEEE Distinguished Lecturer. He is a Fellow of the IEEE and the U.S. National Academy of Inventors.

Chau Yuen (Fellow, IEEE) received the B.Eng. and Ph.D. degrees from Nanyang Technological University, Singapore, in 2000 and 2004, respectively. He was a Post-Doctoral Fellow with the Lucent Technologies Bell Laboratories, Murray Hill, in 2005. From 2006 to 2010, he was with the Institute for Infocomm Research, Singapore. From 2010 to 2023, he was with the Engineering Product Development Pillar, Singapore University of Technology and Design. Since 2023, he has been with the School of Electrical and Electronic Engineering, Nanyang Technological University, where he is currently the Provost's Chair in wireless communications and the Assistant Dean in Graduate College. He has four U.S. patents and published over 400 research papers in international journals or conferences. He received the IEEE Communications Society Leonard G. Abraham Prize (2024), the IEEE Communications Society Best Tutorial Paper Award (2024), the IEEE Communications Society Fred W. Ellersick Prize (2023), the IEEE Marconi Prize Paper Award in Wireless Communications (2021), the IEEE APB Outstanding Paper Award (2023), and the EURASIP Best Paper Award for Journal on Wireless Communications and Networking (2021). He currently serves as the Editor-in-Chief for Computer Science (Springer Nature), and an Editor for IEEE Transactions on Vehicular Technology, IEEE System Journal, and IEEE Transactions on Network Science and Engineering, where he was awarded as IEEE Transactions on Network Science and Engineering Excellent Editor Award and a Top Associate Editor for IEEE Transactions on Vehicular Technology from 2009 to 2015. He also served as a Guest Editor for several special issues, including IEEE JOURNAL on Selected Areas in Communications, IEEE Wireless Communications Magazine, IEEE Communications Magazine, IEEE Vehicular Technology Magazine, IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, and Applied Energy (Elsevier). He is a Distinguished Lecturer of the IEEE Vehicular Technology Society, a top 2% Scientist by Stanford University, and also a Highly Cited Researcher by Clarivate Web of Science.