






Robust and Secure Federated Learning With Verifiable Differential Privacy

Chushan Zhang , Jian Weng , *Member, IEEE*, Jiasi Weng , *Member, IEEE*, Yijian Zhong ,
Jia-Nan Liu , *Member, IEEE*, and Cunle Deng

Abstract—Federated Learning (FL) has emerged as a popular paradigm for training machine learning models on data from multiple sources without the need for data to leave their respective locations. Although FL is widely used, recent research indicates its vulnerability to data privacy breaches and Byzantine attacks. Addressing both threats simultaneously remains challenging, as previous defense mechanisms struggle to balance Byzantine resilience, data privacy, and training accuracy. Although combining Differential Privacy (DP) with Byzantine-resilience in FL offers potential solutions, it presents challenges due to theoretical incompatibilities that can impact the performance of large-scale models. Addressing this underexplored challenge, we propose a novel approach that enhances privacy and Byzantine-resilient in FL while maintaining training accuracy. Our method integrates DP with Byzantine-resilience techniques, overcoming the limitations of existing methods that often sacrifice accuracy or even Byzantine resistance. Our experiments carefully analyze the impact of DP noise on training accuracy and Byzantine-resilience, guiding appropriate parameter selection. The experimental results demonstrate that our approach achieves superior global model accuracy compared

to previous competitive methods, even with strong privacy constraints.

Index Terms—Secure federated learning (FL), Byzantine-resilient, differential privacy (DP).

I. INTRODUCTION

AS AN emerging distributed machine learning paradigm, federated learning (FL) has garnered significant attention from both academia and industry. In FL, users independently train their own local models using their private data, and subsequently submit these local models to a central server. Next, the server aggregates these local models to form a new global model, which is then sent back to the users for further local training. This iterative process continues until the global model reaches convergence [1]. Importantly, throughout this process, the users are not required to share their private data with the central server.

However, recent studies have highlighted several privacy and security threats faced by FL, including gradient-based data inference attacks [2], [3], [4], [5], model poisoning and backdoor attacks [6], [7], [8], [9], [10]. First, adversaries can exploit membership inference attacks (MIA) to infer sensitive local information, thereby compromising data privacy [2], [3]. Second, the inherently non-independent and identically distributed (non-IID) nature of local datasets in FL often leads to significant bias variations in local models. Given that these biased local models directly influence the global model, malicious actors may leverage this vulnerability to launch Byzantine attacks, such as poisoning the global model [8], [11] or injecting backdoors into it [6].

To address privacy leakage, prior works [12], [13], [14], [15], [16] have focused on implementing secure aggregation (SA) algorithms using secure multiparty computation (MPC) techniques to preserve the privacy of local models and ensure lossless aggregation; however, continuous exposure to global models still poses risks of local privacy leakage [4], [5], [17], [18]. Thus, there is a critical need to preserve the global model's privacy throughout each round, employing methods like Homomorphic Encryption (HE) [19], [20], [21] and Differential Privacy (DP) [5], [19], [22], [23], [24], [25], [26]. Although promising, HE remains nascent when it comes to preserving the global model. DP, in particular, is gaining favor due to its ability to add noise to local updates, making it challenging for adversaries to extract sensitive information. Despite its effectiveness, DP's noise injection can potentially reduce model accuracy.

Received 16 July 2024; revised 29 March 2025; accepted 25 May 2025. Date of publication 29 May 2025; date of current version 4 September 2025. The work of Jian Weng was supported in part by the National Natural Science Foundation of China under Grant 62332007, Grant U22B2028, in part by the Science and Technology Major Project of Tibetan Autonomous Region of China under Grant XZ202201ZD0006G, in part by the Open Research Fund of Machine Learning and Cyber Security Interdiscipline Research Engineering Center of Jiangsu Province under Grant SDGC2131, and in part by the National Joint Engineering Research Center of Network Security Detection and Protection Technology, Guangdong Key Laboratory of Data Security and Privacy Preserving, Guangdong Hong Kong Joint Laboratory for Data Security and Privacy Protection, and Engineering Research Center of Trustworthy AI, Ministry of Education. The work of Jiasi Weng was supported in part by the National Natural Science Foundation of China Youth Project under Grant 62302192, in part by the Joint Project of the National Natural Science Foundation of China under Grant U23A20303, in part by the General Project of the Guangdong Provincial Natural Science Foundation under Grant 2024A1515010086, in part by the Special Funding Project of the 17th Batch of the China Postdoctoral Science Foundation under Grant 2024T170348, and in part by the Guangzhou Science and Technology Plan Project under Grant 2024A04J3691. The work of Jia-Nan Liu was supported in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2024A1515011341 and in part by the Dongguan Social Development Technology Project under Grant 20231800940342. (*Corresponding authors: Jian Weng; Jiasi Weng.*)

Chushan Zhang, Jiasi Weng, Yijian Zhong, and Cunle Deng are with the College of Cyber Security, the National Joint Engineering Research Center of Network Security Detection and Protection Technology, the Guangdong Key Laboratory of Data Security and Privacy Preserving, and the Guangdong-Hong Kong-Macao Joint Laboratory of Data Security and Privacy Preserving, Guangzhou 510632, China (e-mail: zhangchushan839@gmail.com; wengjiashi@gmail.com; czs_zhong@163.com; 239826557@qq.com).

Jian Weng and Jia-Nan Liu are with the School of Computer Science and Technology, Dongguan University of Technology, Dongguan 523808, China (e-mail: cryptjweng@gmail.com; j.n.liu@foxmail.com).

Digital Object Identifier 10.1109/TDSC.2025.3574745

The combination of DP and SA offers a promising approach to balancing model utility and privacy, as demonstrated in recent works [22], [25], [27].

In parallel, defenses against Byzantine attacks have primarily focused on Byzantine-resilient (BR) methods, which aim to tolerate and mitigate the impact of maliciously altered model updates [28], [29], [30]. To simultaneously address the threats of Byzantine attacks and privacy inference attacks in FL, many works combine BR methods with privacy protection methods [16], [30], [31]. As mentioned above, combining DP with SA is a comparatively effective method for protecting privacy. Combining DP with BR in FL offers a potential solution. However, the previous approaches for achieving DP and BR are incompatible, as demonstrated by Guerraoui et al. [32]. Specifically, the theoretical conflict arises from the interference between DP noise and BR detection mechanisms. For example, directly superimposing DP noise blurs gradient directions (e.g., invalidating cosine similarity-based detection), which could cause BR methods to misclassify benign updates as malicious (false positives) or fail to detect carefully crafted Byzantine gradients (false negatives). They further found that directly combining DP and BR techniques causes the guarantees of the distributed SGD algorithm to depend unfavorably on model size. This makes the training of large-scale models practically infeasible. Our work addresses the challenge: *how to combine the properties of DP and BR while maintaining the global model quality as much as possible?*

To address the incompatibility between privacy and robustness in FL, we introduce secondary servers between users and the central aggregator. Unlike prior methods that burden users or the aggregator with both tasks, our design decouples responsibilities: secondary servers first validate model updates via Byzantine detection using MPC, then aggregate the validated updates with verifiable DP noise. This sequential approach ensures robust aggregation is performed on clean data before noise injection, which is crucial because BR mechanisms depend on precise gradient statistics. DP noise, if injected prematurely, could distort these statistics. Additionally, noise levels can be minimized once malicious updates are filtered out.

Building on this architecture, our scheme separates robustness evaluation from privacy preservation by employing secret sharing-based secure aggregation to protect local model privacy while applying DP mechanisms only to the global model. This separation reduces overall noise and enhances accuracy, overcoming limitations of methods that directly combine DP and BR.

We also propose a Noise-Immune Shamir Secret Sharing protocol that enables verifiable noise addition while ensuring correct secret reconstruction—a feat traditional Shamir schemes cannot achieve under noisy conditions. By isolating and integrating DP and BR components, our framework optimally balances privacy, robustness, and model utility. Experimental results show that our approach effectively defends against Byzantine and privacy inference attacks while improving convergence and overall performance, making it a promising solution for secure and scalable federated learning.

We highlight our contributions as follows:

- A novel FL architecture with intermediary secondary servers that decouple Byzantine resilience from differential

privacy. This is the first framework enabling both verifiable DP and BR without mutual interference.

- A Noise-Immune Shamir Secret Sharing protocol that injects DP noise into secret-shared model updates while ensuring correct aggregation. This solves the noise amplification problem in traditional secret sharing (Section VII-A).
- Extensive experiments on MNIST, FMNIST, and CIFAR-10 demonstrate that our design achieves 98.03% accuracy on MNIST under $\epsilon = 0.1$ and tolerates up to 60% malicious users with only a 0.76% accuracy drop, outperforming prior DP-BR integrations by 0.57-4.16% in high-adversary scenarios.

II. RELATED WORKS

A. Byzantine Resilient FL

To defend against poisoning attacks, which encompass both data poisoning and model poisoning, several BR schemes have been proposed [27], [28], [29], [33], [34], [35]. Yin et al. proposed an assumption based on the statistical differentiation between honest and Byzantine models, which led to the design of a robust federated aggregation algorithm known as Krum [28]. Dong et al. proposed the $\Pi_{2\text{BROFL}}$ protocol, which combined the Shuffle protocol with a malicious Top-K strategy. This enhanced the Multi-Krum protocol's resistance to poisoning attacks, limiting the model accuracy drop to a mere 1.05% even when up to 50% of users were malicious [28]. Tang et al. [36] introduced PILE, which utilized zero-knowledge proofs (ZKPs) to verify dummy model gradients without decryption. However, PILE is designed to defend against model poisoning attacks but not data poisoning, as it does not validate local datasets. These works comprehensively analyzed poisoning attacks from diverse perspectives and, recognizing their unique characteristics, devised a variety of potent defenses accordingly. In this work, we adopt norm-based [37] and cosine similarity-based [29] BR methods for easy implementation on MPC circuits.

B. Privacy Preserving FL

Currently, numerous privacy-preserving mechanisms are employed to counter privacy leakage in FL, which can be broadly categorized into cryptographic primitives [12], [14], [21] and DP techniques [23], [25], [38]. Bonawitz et al. [12] devised a practical SA algorithm, SecAgg, based on threshold Shamir secret sharing. This algorithm efficiently safeguards the privacy of local models against honest-but-curious servers. Given that SA cannot ensure the global model remains secret throughout the training process in FL, and considering the immaturity of homomorphic encryption (HE) for this purpose, DP remains the widely recognized method for preserving privacy in FL.

To balance the global model accuracy and privacy utility in DP-based FL, some studies aim to integrate the properties of SA and DP [23], [38]. SA preserves the privacy of local models and ensures lossless aggregation, while DP preserves the privacy of the global model but may decrease accuracy. Kairouz et al. [23] analyzed the attributes of the discrete Gaussian mechanism for differential privacy, highlighting that the sum of multiple

discrete Gaussian noises approximates a discrete Gaussian distribution. Leveraging this characteristic, they have integrated SA with DP, outlining an interpretable and manageable federated secure aggregation scheme that achieves $\frac{1}{2}\epsilon^2$ -concentrated DP for the global model, demonstrating a mean squared error bound of $O(\frac{c^2 d}{\epsilon^2})$. Building on this work, Chen et al. [38] employed the Johnson-Lindenstrauss lemma to implement sparse random projections and counting sketches, which reduced vector dimensions and information redundancy to enhance data privacy protection. However, these works predominantly assumed semi-honest FL users, failing to account for the possibility that, in real-world scenarios, malicious adversaries might tamper with the global model parameters by introducing fraudulent noise. This vulnerability potentially heightens the risk of privacy breaches for honest users. To address this vulnerability, we introduce verifiable differential privacy as a safeguard in our work.

C. Byzantine Resilient and Privacy Preserving FL

In pursuit of combining BR with privacy preservation methods, many works have achieved comprehensive BR technologies employing MPC circuits [16], [30], [31], [39]. Corrigan et al. introduced Prio [39], which employed Shamir secret sharing to implement a generic validation circuit named SNIP. Building upon Prio, Roy et al. proposed EIFFeL [16], enhancing it with homomorphic commitment technology as a defense against malicious adversaries. However, they did not consider the risk of privacy leakage from the global model. Zhou et al. [27] formulated a dynamic differential privacy protocol aiming to strike a balance between privacy preservation and model utility, while also integrating Byzantine-resilient methods to repel malicious poisoning attacks. Nonetheless, it might not have fully optimized model accuracy, given that the dynamic DP protocol prioritized establishing a minimum level of DP noise, potentially sacrificing model performance. Moreover, their work did not account for the possibility that the adversary could have corrupted FL training by falsifying the size of the noise. When evaluating FL in terms of privacy preservation, model utility, and defense against malicious attacks, it becomes clear that current methods often excel in one or two of these areas but have not yet achieved an ideal balance across all three.

III. PRELIMINARIES

A. Notations

In this section, we provide a summary of the key symbols used throughout this paper. TABLE I lists the important notations, along with their definitions, to facilitate understanding of the formulas and concepts presented in the subsequent sections.

B. Federated Learning

Federated Learning (FL) is distinguished into two primary categories: Cross-silo FL and Cross-device FL [40]. In this work, we particularly focus on the Cross-device FL. The workflow of FL generally adheres to the following steps:

- 1) *Initialization*: A central server initializes a global model and distributes it to devices.
- 2) *Local Training*: Each device uses its local data to train the received model, optimizing its parameters based on

TABLE I
SYMBOLS AND THEIR DEFINITIONS

Symbol	Definition
u	A single user
\mathcal{U}	Set of all users in federated learning system: $u \in \mathcal{U}$
$\mathcal{U}_{\mathcal{M}}$	Set of malicious users controlled by adversary
$\mathcal{U}_{\mathcal{H}}$	Subset of users passing Byzantine validation
s	A single secondary server
\mathcal{S}	Set of secondary servers: $s \in \mathcal{S}$
$\mathcal{S}_{\mathcal{M}}$	Set of malicious secondary servers ($ \mathcal{S}_{\mathcal{M}} < t$)
\mathcal{S}_t	Threshold subset of secondary servers ($ \mathcal{S}_t = t$)
\mathcal{A}	Central aggregator server
w	Global model parameter vector ($w \in \mathbb{R}^d$)
w_u	Local model parameters of user u
w'	Noisy global model: $w' = w + \sum_{s \in \mathcal{S}_t} v_s$
q_u	Weight of user u (proportional to $ D_u $)
D_u	Private training dataset of user u
n_b	Number of binomial trials per server for DP noise
$b_{s,j}$	Private coin bit sampled by secondary server s ($b_{s,j} \in \{0, 1\}$)
$v_{s,j}$	Public coin bit from oracle $\mathcal{O}_{\text{Morra}}$
v_s	Verifiable noise
r_s	Randomness for noise commitment
$[w]_s$	Shamir share of parameter w at server s
$[w]_s'$	Noisy share: $[w]_s' = [w]_s + v_s$
n	Dimension of model parameters ($w_u \in \mathbb{R}^n$)
t	Threshold for secret reconstruction
λ	Cryptographic security parameter
p, q	Primes in Pedersen's VSS ($ q = \lambda, q \mid p - 1$)
\mathbb{G}_q	Cyclic subgroup of \mathbb{Z}_p^* with order q
g, h	Generators of \mathbb{G}_q ($h = g^a \prod a_s$)
ϵ, δ	Differential privacy parameters ((ϵ, δ) -DP)
$\Delta_{\mathcal{S}_t}^s$	Lagrange coefficient $\Delta_{\mathcal{S}_t}^s = \prod_{s' \in \mathcal{S}_t \setminus \{s\}} \frac{x_s}{x_s - x_{s'}}$
$\text{Valid}(\cdot)$	Byzantine robustness evaluation function
Π_{OR}	Zero-knowledge proof for bit commitments
cs_u	Commitment to user u 's model: $\text{Ped.Com}(w_u, r_u)$
$b_{u,s}$	Verification result bit for user u at server s
$\text{PRF}(\cdot)$	Pseudorandom function for pairwise masking
\mathbb{Z}_q^*	Multiplicative group modulo q
\oplus	Bitwise XOR operation
$\mathcal{O}_{\text{Morra}}$	Unbiased public coin generation oracle

local gradients or loss functions. This process occurs independently and without the data leaving the device.

- 3) *Model Update Aggregation*: The updated models are then sent back to the central server.
- 4) *Global Model Update*: The server aggregates these updates, typically through weighted averaging, to create an updated global model that reflects the collective knowledge of all contributing devices as follows:

$$w = \sum_{u \in \mathcal{U}} q_u \cdot w_u, \quad (1)$$

where w_u is the vector of the local model parameters of the user u , and q_u is their weight, determined by the size of the user's dataset. And q_u is published to all the participants.

- 5) *Round until Convergence*: Steps 2 through 4 are repeated over multiple communication rounds until the global model converges to a satisfactory performance level or until predefined stopping criteria are met.

C. Pedersen's Verifiable Secret Sharing

Pedersen's Verifiable Secret Sharing (VSS) [41] enhances Shamir's threshold secret sharing by adding verification steps that ensure the correctness and authenticity of the distributed secret. Informally, Π_{pvss} is the Pedersen's VSS protocol involving five algorithms.

- **Ped.Setup** $(n, \lambda) \rightarrow pp = (p, q, g, h, t)$: Given the number of participants n and the security parameter λ , this algorithm produces the public parameters pp , which its general process is as follows:
 - Generates public parameters including two large primes p and q (with q dividing $p - 1$, forming the finite field \mathbb{Z}_p).
 - Chooses two generators g and h for the subgroup \mathbb{G}_q .
 - Sets the threshold t which defines the minimum number of shares required to reconstruct the secret.
- **Ped.Com** $(v, r) \rightarrow c$: This algorithm creates a commitment c to a message $v \in \mathbb{Z}_q$ and randomness $r \in \mathbb{Z}_q$. The commitment scheme used is based on the discrete logarithm problem, ensuring that the commitment is binding and hiding.
- **Ped.Share** $(v, r, n) \rightarrow (([v]_s, [r]_s)_{s \in S}, cs)$: Given a secret v and randomness r , this algorithm constructs random polynomials $F(x)$ and $G(x)$ of degree $t - 1$. The secret v and randomness r are used as the constant terms of these polynomials. The algorithm then distributes shares $([v]_s, [r]_s)$ to each participant s , where $[v]_s$ and $[r]_s$ are the values of the polynomials evaluated at the participant's identifier. Additionally, it computes commitments cs to the polynomial coefficients and the original pair (v, r) .
- **Ped.Verify** $([v]_s, [r]_s, cs, pp) \rightarrow b$: Participants use this algorithm to verify whether the shares they have received are correct. The algorithm takes the shares $[v]_s$ and $[r]_s$, the commitments cs , and the public parameters pp as input, and outputs a boolean value $b \in \{0, 1\}$ indicating whether the shares are consistent with the commitments.
- **Ped.Recon** $(([v]_s, [r]_s)_{s \in S_t}, cs, pp) \rightarrow (v, r, b)$: This algorithm allows the reconstruction of the secret (v, r) using at least t correct shares from the set of participants S_t . It also verifies the correctness of the reconstructed secret through the commitments cs and outputs a boolean value $b \in \{0, 1\}$ to confirm the integrity of the secret.

Pedersen's VSS has an additive homomorphic property. This property allows it to integrate seamlessly with tools like Beaver triples for MPC. Pedersen's VSS also provides robust protection against passive adversaries and defends against active adversaries who try to steal or disrupt the secret. This ensures both the confidentiality and integrity of the secret. Typically, we set the threshold t as $n/3 \leq t \leq n/2$ [42] to maintain system robustness even when some participants are dishonest.

D. Verifiable Differential Privacy

Verifiable differential privacy ensures the randomness and unbiasedness of noise in untrusted environments. Approaches like VerDP [43], which uses the Fuzz query language, and DPrio [44], which integrates differential privacy into share distribution based on Prio, address this need. Biswas et al. [45]

enhanced the approach to prevent active adversaries using the Σ -Protocol, and proved the impossibility of perfect verification with unlimited computational power. We adopt and revise the approach of Biswas et al. [45] in this work. The protocol requires the prover s and verifier \mathcal{A} to negotiate and sum verifiable random coins to achieve verifiable Binomial DP through the following steps.

- **Setup**: The provers and verifier generate public parameters. The client sending an additive share of its secret value to the prover and the commitment of the share to the verifier.
- **Generate private coins**: The prover s samples private bits $b_{s,j}, j \in [n_b]$ and sends their commitment $c_{s,j} = \text{Ped.Com}(b_{s,j}, r_{s,j})$ to the verifier.
- **Check the update from the prover**: The verifier checks the commitments $c_{s,j}$ from the prover using an oracle \mathcal{O}_{OR} .
- **Generate public coins**: If the checks pass, both parties obtain public random and unbiased bits $v_{s,j}$ using an oracle \mathcal{O}_{Morra} .
- **Obtain the verifiable noise**: The prover XORs the private and public coins to obtain the verifiable noise v_s :

$$v_s = \sum_{j=1}^{n_b} (b_{s,j} \oplus v_{s,j}), \quad r_s = \sum_{j=1}^{n_b} (-1)^{v_{s,j}} r_{s,j}. \quad (2)$$

And the verifier calculates the commitment c_s of the noise:

$$c_s = \prod_{j=1}^{n_b} \begin{cases} \text{Ped.Com}(1, 0) * c_{s,j}^{-1}, & \text{If } v_{s,j} = 1; \\ c_{s,j}, & \text{Otherwise.} \end{cases} \quad (3)$$

In the above protocol, the verifier ensures the integrity of the noise using commitment c_s , while the prover keeps the privacy of the noise with random number r_s . This protocol involves two oracles. First, the oracle \mathcal{O}_{OR} confirms that $c_{s,j}$ indeed commits to the secret $b_{s,j}$, which must be either 0 or 1. However, it suffers from a security vulnerability due to its use of two incompatible Σ -protocols in verification, potentially allowing adversaries to steal the secret $b_{s,j}$ by analyzing verification messages (details can be found in [45]). Second, the oracle \mathcal{O}_{Morra} is responsible for generating unbiased public coins but incurs a high communication cost.

To address these concerns, we improve the above two oracles as follows:

- Verification employs the secure protocol Π_{OR} [46] rather than the vulnerable \mathcal{O}_{OR} , which is based on the property that $x(x - 1) = 0 \iff x \in \{0, 1\}$.
- Under the semi-honest assumption about the aggregator, public coins can be directly generated by the aggregator and distributed to secondary servers, thereby reducing communication costs.

IV. SYSTEM MODEL

Fig. 1 illustrates our DPSFL scheme with three entities: Users, Secondary Servers, and an Aggregator. Unlike traditional FL systems with only users and a central aggregator, the introduction of secondary servers is essential for ensuring both privacy and Byzantine resilience. We conduct a theoretical analysis of the necessity of secondary servers in Section VI-A.

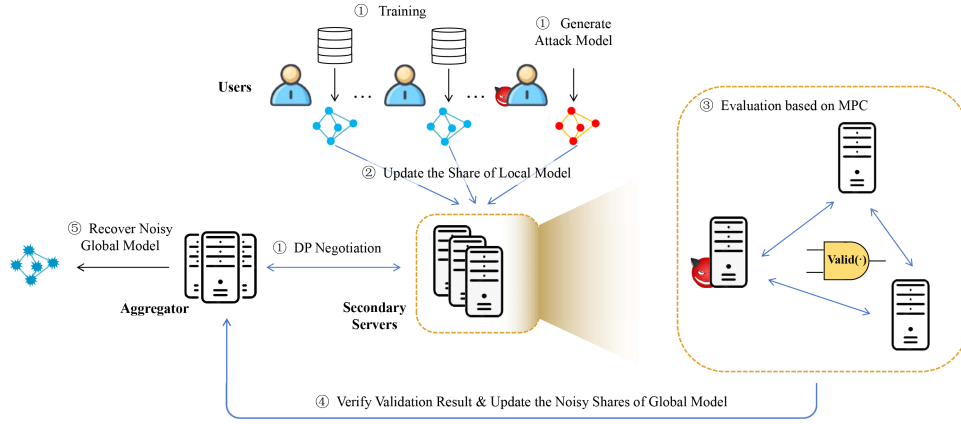


Fig. 1. The figure illustrates our DPSFL scheme.

A. Entity Descriptions

Users (\mathcal{U}): Each user $u \in \mathcal{U}$ performs the following tasks:

- 1) *Local Model Training:* Trains a local model w_u using its private dataset D_u .
- 2) *Sharing Updates:* Distributes local model updates to secondary servers using a secret sharing mechanism.
- 3) *Receiving Global Model:* Downloads the global model from the aggregator for subsequent rounds.

Secondary servers (\mathcal{S}): Each secondary server $s \in \mathcal{S}$ is responsible for:

- 1) *DP Noise Negotiation:* Securely negotiating verifiable DP noise with the aggregator.
- 2) *Robustness Evaluation:* Collaboratively evaluating the robustness of user updates using secure MPC techniques and applying a robustness function $\text{Valid}(\cdot)$.
- 3) *Aggregation of Noisy Shares:* Aggregating the user shares, adding the negotiated DP noise, and forwarding the resulting noisy aggregated share to the aggregator.

Aggregator (\mathcal{A}): Its tasks are as follows:

- 1) *Verification:* Checks the correctness of both the verifiable DP noise and the robustness evaluation carried out by the secondary servers.
- 2) *Global Model Reconstruction:* Verifies the received noisy shares (using commitments for both the shares and the noise) and reconstruct the noisy aggregate, which is used as the updated global model.

We consider a scenario where a malicious adversary may control a subset of users and a limited number of secondary servers, while the aggregator is assumed to be honest-but-curious. Specifically, we assume that:

- 1) *Malicious Users:* Users may attempt to poison the model, manipulate updates, or collude with Secondary Servers.
- 2) *Compromised Secondary Servers:* A fraction of secondary servers (fewer than t in quantity) may attempt to distort the robustness evaluation or reconstruct user model updates.
- 3) *Curious Aggregator:* The aggregator is honest-but-curious and follows the protocol but may attempt to infer private information. As the core coordinating party, the aggregator can reduce the risk of malicious behavior by leveraging

trusted hardware (such as Intel SGX [47]) or regulatory constraints.

These assumptions reflect realistic scenarios in which critical infrastructure (e.g., edge hubs or cloud secondary servers) is better protected than end-user devices.

V. PROPOSED SCHEME

In this section, we formally introduce our scheme. Firstly, we introduce the design overview of our scheme in Section V-A. Secondly, we propose the protocol Π_{ln} for the noise-immune Shamir secret sharing in Section V-B. Finally, we then propose the details of our DPSFL scheme in Section V-C and the correctness analyzing in Section V-D.

A. Design Overview

Current FL architectures struggle to harmonize DP and BR due to inherent statistical conflicts and centralized bottlenecks. While prior works [27], [32] attempt to combine DP noise injection with Byzantine-robust aggregation rules, they face two critical limitations: (1) DP noise distorts gradient statistics essential for BR detection mechanisms (e.g., invalidating cosine similarity metrics), and (2) centralized aggregators become single points of failure that undermine both privacy and robustness guarantees. Our DPSFL framework introduces three interconnected innovations to overcome these challenges.

1) *Sequential Validation-Noise Pipeline:* We decouple BR validation from DP noise injection via intermediary secondary servers. Specifically, secondary servers first perform MPC-based Byzantine detection on raw updates of users (Phase 3) before applying verifiable DP noise to aggregated shares (Phase 4). This sequential approach preserves gradient statistics for accurate BR detection while confining DP perturbations to the final aggregation step.

2) *Noise-Immune Shamir Secret Sharing:* Traditional Shamir schemes fail under noisy conditions due to Lagrange interpolation errors (Section VII-A). Our protocol embeds binomial noise directly into secret shares using pairwise PRF masking, enabling correct aggregation while satisfying (ϵ, δ) -DP. This

Algorithm 1: Noise-Immune Shamir Secret Sharing Π_{ln} .

Public Parameters: $t, p, \{x_s, s \in \mathcal{S}\}$
 Private Parameters: **User** u has w_u ;
Secondary server s has $(\{m_{s,s'}, s' \in \mathcal{S} \setminus \{s\}\}, v_s)$.
User u Sharing:
 distributes w_u to $[w_u]_s$ using Shamir secret sharing.
 Sends to secondary servers $s \in \mathcal{S}$: $[w_u]_s$.
Secondary server s executes:
 $[w_u]'_s \leftarrow [w_u]_s + v_s$; and sends to Aggregator: $[w_u]'_s$.
Aggregator \mathcal{A} executes:
 Receives $[w_u]'_s, s \in \mathcal{S}$.
 Selects $\mathcal{S}_t \subseteq \mathcal{S}, |\mathcal{S}_t| = t$ and sends \mathcal{S}_t to secondary servers $s \in \mathcal{S}_t$.
Secondary server $s \in \mathcal{S}_t$ executes:
 Receives \mathcal{S}_t .
 $V_s \leftarrow v_s - v_s \Delta_{\mathcal{S}_t}^s + \sum_{s' < s, s' \in \mathcal{S}_t} \text{PRF}(m_{s,s'}) - \sum_{s' > s, s' \in \mathcal{S}_t} \text{PRF}(m_{s,s'})$.
 Sends to Aggregator V_s .
Aggregator \mathcal{A} executes:
 Receives $V_s, s \in \mathcal{S}_t$.
 $w'_u \leftarrow \sum_{s \in \mathcal{S}_t} ([w_u]'_s \Delta_{\mathcal{S}_t}^s + V_s)$.

allows verifiable noise addition without exposing raw gradients to the aggregator.

3) *Distributed Workload Allocation:* We offload computational heavy operations (Pedersen commitment verification, SNIP validation circuits) to secondary servers through parallelizable MPC protocols while users only perform lightweight operations.

The protocol progresses through four phases:

Phase 1: Users split local models into verifiable Shamir shares using Pedersen VSS, ensuring privacy against compromised secondary servers (Section V-C).

Phase 2: Secondary servers negotiate binomial noise with the aggregator.

Phase 3: Collaborative MPC validation filters malicious updates using a BR method (such a norm-based or cosine similarity checks) before any noise injection, addressing the statistical conflict of DP and BR.

Phase 4: The aggregator reconstructs the global model from verified noisy shares using Lagrange interpolation. To prevent excessive noise amplification, DPSFL leverages controlled noise injection and adaptive thresholding techniques.

DPSFL balances model accuracy, privacy, and robustness by integrating noise into secret sharing and leveraging hierarchical validation. This design supports scalability and adversarial resilience in large-scale edge deployments.

B. Noise-Immune Shamir Secret Sharing

In Shamir's secret sharing scheme, Lagrange interpolation is conventionally employed for secret reconstruction. Consider a perturbed share defined as $[w]'_s = [w]_s + v_s$ for each secondary server $s \in \mathcal{S}$, where $[w]_s$ denotes the original share and v_s represents additive perturbation introduced by secondary server

s . Through Lagrange interpolation, the reconstructed secret exhibits the following error propagation characteristic:

$$w'_{org} = \sum_{s \in \mathcal{S}_t} [w]'_s \Delta_{\mathcal{S}_t}^s = w + \sum_{s \in \mathcal{S}_t} v_s \Delta_{\mathcal{S}_t}^s.$$

Where the Lagrange coefficient $\Delta_{\mathcal{S}_t}^s = \prod_{s' \in \mathcal{S}_t \setminus \{s\}} \frac{x_s - x_{s'}}{x_s - x_{s'}}$ demonstrates factorial scaling behavior. Specifically, $\Delta_{\mathcal{S}_t}^s = \frac{t^{t-1}}{(t-1)!} \propto \frac{e^t}{\sqrt{t}}$ via Stirling approximation, implying that the aggregated perturbation grows super-exponentially with threshold parameter t . Empirical validation of this error amplification phenomenon is provided in Section VII-A.

To address this t -dependent error amplification, we propose an optimized reconstruction operator:

$$w'_{opt} = \sum_{s \in \mathcal{S}_t} ([w]'_s \Delta_{\mathcal{S}_t}^s) + \sum_{s \in \mathcal{S}_t} (v_s - v_s \Delta_{\mathcal{S}_t}^s) = w + \sum_{s \in \mathcal{S}_t} v_s. \quad (4)$$

This construction effectively decouples the final reconstruction error from the threshold parameter t . The formal specification of our Noise-Immune Shamir Secret Sharing protocol Π_{ln} appears in Algorithm 1. We then prove the correctness in Section V-D.

C. Design of DPSFL

The full protocol is presented in Fig. 2. Our scheme involves a setup phase and the following four phases.

Setup Phase: Given a security parameter λ and the set of secondary servers \mathcal{S} , the aggregator runs **Ped.Setup** $(|\mathcal{S}|, \lambda) \rightarrow (p, q, g, h, t)$, and generates two empty sets $\mathcal{S}_M, \mathcal{U}_M$ to store the malicious party. To keep $\log_g h$ private, and to generate the key for the parties, we split this step as follows:

- The aggregator \mathcal{A} and the secondary servers $s \in \mathcal{S}$ generate their own secret number $a, a_s \in \mathbb{Z}_q$ for $s \in \mathcal{S}$;
- The aggregator \mathcal{A} and the secondary servers $s \in \mathcal{S}$ calculate $h = g^a \prod_{s \in \mathcal{S}} a_s$ together.
- The aggregator \mathcal{A} generates an index for all of the secondary servers $s \in \mathcal{S}$: $x_s \in [|\mathcal{S}|]$.
- The aggregator \mathcal{A} then generates the parameter of DP noise n_b according to λ .

The aggregator \mathcal{A} publishes the system parameters $pp = (p, q, g, h, t, \{x_s | s \in \mathcal{S}\}, n_b)$ to $u \in \mathcal{U}$ and $s \in \mathcal{S}$. To mask the noise in the protocol Π_{ln} , secondary servers $s \in \mathcal{S}$ and $s' \in \mathcal{S} \setminus \{s\}$ interactively generate a secret key $m_{s,s'}$ via the Diffie-Hellman key agreement protocol.

Phase 1. Local Training and Parameters Sharing: Each user $u \in \mathcal{U}$ trains its local model using their local private dataset D_u and obtains the parameters of the local model w_u , which is a n dimensional vector. Next, the user u distributes w_u and shares to $s \in \mathcal{S}$ as follow:

- The user u executes **Ped.Share** $(w_u, r_u, |\mathcal{S}|) \rightarrow (([w_u]_s, [r_u]_s)_{s \in \mathcal{S}}, cs_u)$, where $r_u \in \mathbb{Z}_q^*$ is a random vector chosen by the user u .
- The user u sends $([w_u]_s, [r_u]_s)$ to the secondary servers s and publishes the Pedersen commitments of the share cs_u .

DPSFL Protocol

• Setup Phase

Aggregator:

- Initialize system parameters $pp = (p, q, g, t, \{x_s\}_{s \in \mathcal{S}}, n_b, \mathcal{S}_{\mathcal{M}}, \mathcal{U}_{\mathcal{M}}, \text{PRF})$, where p, q, g, t are the parameters of Pedersen's VSS, $\{x_s\}_{s \in \mathcal{S}}$ are the indices of the secondary servers $s \in \mathcal{S}$, n_b is the parameter of DP noise, $\mathcal{S}_{\mathcal{M}}, \mathcal{U}_{\mathcal{M}}$ are two empty sets to store the malicious party, and PRF is a pseudorandom function.
- Sends pp to secondary servers and Users, and chooses $a \xleftarrow{R} \mathbb{Z}_q^*$ as its secret key.

secondary servers:

- Chooses $a_s \xleftarrow{R} \mathbb{Z}_q^*$ as their own secret key.
- Interactively calculate $h = g^{a \prod_{s \in \mathcal{S}} a_s} \bmod p$ with the aggregator \mathcal{A} and publish it.
- Each two secondary servers $s, s' \in \mathcal{S}$ interactively generate a secret key $m_{s,s'}$ as the mask.

• Phase 1. Local Training and Parameters Sharing.

User $u \in \mathcal{U}$:

- Trains the local model w_u using its local dataset, and chooses a random vector r_u .
- Calculates $\text{Ped.Share}(w_u, r_u, |\mathcal{S}|) \rightarrow ([w_u]_s, [r_u]_s)_{s \in \mathcal{S}}, cs_u$.
- Sends $([w_u]_s, [r_u]_s)$ to secondary server $s \in \mathcal{S}$.

• Phase 2. Verifiable Noise Negotiation.

secondary servers $s \in \mathcal{S}$:

- Chooses $\{v_{j,s} \xleftarrow{R} \{0, 1\}, r_{j,s} \xleftarrow{R} \mathbb{Z}_q^* \mid j \in [1, n_b]\}$ and calculates $cs_{s,j} = \text{Ped.Com}(v_{s,j}, r_{s,j})$.
- Sends $\{cs_{s,j} \mid j \in [1, n_b]\}$ to Aggregator.

Aggregator:

- Checks the integrity of $\{cs_{s,j} \mid j \in [1, n_b]\}_{s \in \mathcal{S}}$ using the protocol Π_{OR} . If fail, $s \rightarrow \mathcal{S}_{\mathcal{M}}$.
- Chooses $\{b_{s,j} \xleftarrow{R} \{0, 1\} \mid j \in [1, n_b]\}$ and sends to secondary servers s .
- Calculates c_s with Equation (3)

secondary servers $s \in \mathcal{S}$:

- Calculates v_s and r_s using Equation (2)

• Phase 3. Evaluating the Integrity and Byzantine Robustness of Local Models.

secondary servers $s \in \mathcal{S}$:

- Check the share from the users $u \in \mathcal{U}$: $\text{Ped.Verify}([w_u]_s, [r_u]_s, cs_u, pp) \rightarrow b_{u,s}$.
- Sends the result $b_{u,s}$ to the aggregator \mathcal{A} .

Aggregator:

- Checks if $\sum_{s \in \mathcal{S}} b_{u,s} \geq t$, if not, then $u \rightarrow \mathcal{U}_{\mathcal{M}}$.
- Sends the command to continue checking to the secondary servers $s \in \mathcal{S}$ with $b_{u,s} = 1$.

secondary servers $s \in \mathcal{S}$:

- Interactively evaluates $\text{Valid}(w_u)$, meanwhile checks the integrity of the other secondary servers $s' \neq s$ computation using the commitment cs_u .
- If the intermediate process $\pi_{s'}$ and its commitment $cs_{s'}$ is not correspondent, then reports $(\pi_{s'}, cs_{s'})$ to the aggregator.

Aggregator:

- If the secondary servers s reports the exception of s' , the aggregator checks again the integrity of $(\pi_{s'}, cs_{s'})$, if fails, then $s' \rightarrow \mathcal{S}_{\mathcal{M}}$, otherwise, then $s \rightarrow \mathcal{S}_{\mathcal{M}}$.

secondary servers $s \in \mathcal{S}$:

- Checks $\text{Valid}(w_u)$, places u with $\text{Valid}(w_u) = 1$ in the set $\mathcal{U}_{\mathcal{H}}$, and sends $\mathcal{U}_{\mathcal{H}}$ to the aggregator.

• Phase 4. Aggregation and Verification.

secondary servers $s \in \mathcal{S}$:

- Calculates $([w]'_s = \sum_{u \in \mathcal{U}_{\mathcal{H}}} q_u \cdot [w_u]_s + v_s, [r]'_s = \sum_{u \in \mathcal{U}_{\mathcal{H}}} q_u \cdot [r_u]_s + r_s)$ and sends to the aggregator.

Aggregator:

- Checks $([w]'_s, [r]'_s)$ using Equation (6). If fail, $s \rightarrow \mathcal{S}_{\mathcal{M}}$. If $|\mathcal{S}| < t$, then training abort.
- Chooses t secondary servers as $\mathcal{S}_t \subseteq \mathcal{S}$ and sends \mathcal{S}_t to $s \in \mathcal{S}_t$.

Aggregator and secondary servers $s \in \mathcal{S}_t$ jointly execute the **protocol** Π_{In} to recover w' and r'

Aggregator:

- Checks (w', r') using Equation (7). If fail, then training abort.

Fig. 2. Detailed description of the DPSFL.

Phase 2. Verifiable Noise Negotiation: In this phase, the aggregator \mathcal{A} and the secondary servers $s \in \mathcal{S}$ negotiate the verifiable noise as follows:

- The secondary server s samples n_b private coins $b_{s,j}$, $j \in [1, n_b]$, and sends their commitments $cs_{s,j}$ to the aggregator \mathcal{A} .

- The aggregator \mathcal{A} checks the integrity of the private coins using the protocol Π_{OR} [46].
- If checks pass, the aggregator \mathcal{A} sends the public coins $v_{s,j}$ to the secondary server s . Otherwise, the aggregator regards the secondary server s as the malicious party and $s \rightarrow \mathcal{S}_{\mathcal{M}}$.

- The secondary server s calculates the verifiable noise (v_s, r_s) using the (2), where v_s is the raw noise. The aggregator \mathcal{A} calculates the commitment of the noise $c_s = \text{Ped.Com}(v_s, r_s)$ using the (3).

Phase 3. Evaluating the Integrity and Byzantine Robustness of Local Models: In this phase, the secondary server $s \in \mathcal{S}$ first checks the commitment and the share from users $u \in \mathcal{U}$ using $\text{Ped.Verify}(\cdot) \rightarrow b$.

- If there is more than $|\mathcal{S}| - t$ secondary servers to obtain the result $b = 0$, we then regard the user u as a malicious adversary and $u \rightarrow \mathcal{U}_M$.

Next, to evaluate the robustness of the local models, we first construct an interactive evaluation of the function $\text{Valid}(\cdot)$ using the MPC methods, as in SNIP [39]. Because of the homomorphism of Pedersen's VSS, we can further design a verifiable $\text{Valid}(\cdot)$ by synchronously calculating the commitments of the intermediate and final values of the function $\text{Valid}(\cdot)$ and verify the integrity of the evaluation.

- If a secondary server s identifies another secondary server s' as the abnormal party during the evaluation, the aggregator \mathcal{A} then verifies the integrity of the evaluation. If the verification fails, we classify s' as the malicious party; otherwise, we classify s as the malicious party.

Based on $\text{Valid}(w_u)$, we decide whether the local model w_u participates in the current round of aggregation and the user u involves the set \mathcal{U}_H .

Phase 4. Aggregation and Verification: In this phase, the secondary servers $s \in \mathcal{S}$ first aggregate the share and add the noise as follows:

$$[w]'_s = \sum_{u \in \mathcal{U}_H} (q_u \cdot [w_u]_s) + v_s; \quad [r]'_s = \sum_{u \in \mathcal{U}_H} (q_u \cdot [r_u]_s) + r_s. \quad (5)$$

Where q_u is the weight of user u determined by the size of the user's dataset and $\sum_{u \in \mathcal{U}_H} q_u = 1$.

Secondly, the secondary servers s sends $([w]'_s, [r]'_s)$ to the aggregator \mathcal{A} , \mathcal{A} checks these share of noisy parameters of the global model using the commitment as follows:

$$\text{Ped.Com}([w]'_s, [r]'_s) \stackrel{?}{=} \prod_{u \in \mathcal{U}_H} \left(\prod_{k=1}^{t-1} c_{u,k}^{x_k^s} \times c_u \right)^{q_u} \times c_s. \quad (6)$$

Next, the aggregator \mathcal{A} chooses t secondary servers $s \in \mathcal{S}_t$, where $s \in \mathcal{S}_t$ is selected from the set of secondary servers that can be executed normally in the previous steps. The aggregator \mathcal{A} and the set of secondary servers \mathcal{S}_t interactively reconstruct the noisy parameters of the global model (w', r') using the protocol Π_{ln} . (Since the protocol Π_{ln} requires two passes of execution, we label the intermediate variables V_s and R_s to distinguish between the two passes. Note that V_s represents the intermediate variable at the first execution of the protocol, while R_s is the intermediate variable at the second execution.)

Finally, the aggregator \mathcal{A} then checks the correctness of the noisy parameters of the global model (w', r') using the commitment as follows:

$$\text{Ped.Com}(w', r') \stackrel{?}{=} \prod_{u \in \mathcal{U}_H} c_u^{q_u} \times \prod_{s \in \mathcal{S}_t} c_s. \quad (7)$$

D. Correctness of DPSFL

1) Aggregation Correctness With Noise: The correctness of the aggregation directly follows from the correctness of the protocol Π_{ln} . We prove the correctness of the protocol Π_{ln} as follows:

$$\begin{aligned} w' &= \sum_{s \in \mathcal{S}_t} ([w]'_s \Delta_{\mathcal{S}_t}^s) + \sum_{s \in \mathcal{S}_t} (v_s - v_s \Delta_{\mathcal{S}_t}^s) \\ &= \sum_{s \in \mathcal{S}_t} (([w]_s + v_s) \Delta_{\mathcal{S}_t}^s + v_s - v_s \Delta_{\mathcal{S}_t}^s) \\ &= w + \sum_{s \in \mathcal{S}_t} v_s, \end{aligned} \quad (8)$$

where $[w]_s = \sum_{u \in \mathcal{U}_H} (q_u \cdot [w_u]_s)$ is the share of the raw global model, and $w = \sum_{u \in \mathcal{U}_H} q_u \cdot w_u$ is the raw global model.

Furthermore, the noise $\sum_{s \in \mathcal{S}_t} v_s$ is the sum of verifiable binomial noises, each of which follows a $\text{Binomial}(n_b, \frac{1}{2})$ distribution. Consequently, $\sum_{s \in \mathcal{S}_t} v_s$ follows a $\text{Binomial}(n_b \cdot t, \frac{1}{2})$ distribution.

Therefore, w' is the noisy global model that satisfies our goals and is correct.

Similarly, we have:

$$r' = \sum_{u \in \mathcal{U}_H} q_u \cdot r_u + \sum_{s \in \mathcal{S}_t} r_s. \quad (9)$$

2) Verification Correctness: We now discuss the correctness of the (6) and (7).

For the (6), because $\prod_{k=1}^{t-1} c_{u,k}^{x_k^s} \times c_u = \text{Ped.Com}([w_u]_s, [r_u]_s)$ according to the algorithm $\text{Ped.Verify}(\cdot)$, and $c_s = \text{Ped.Com}(v_s, r_s)$ according to the protocol of verifiable DP, we can get that:

$$\begin{aligned} \text{Ped.Com}([w]'_s, [r]'_s) &= \text{Ped.Com} \left(\sum_{u \in \mathcal{U}_H} q_u \cdot [w_u]_s + v_s, \right. \\ &\quad \left. \sum_{u \in \mathcal{U}_H} q_u \cdot [r_u]_s + r_s \right) \\ &= \prod_{u \in \mathcal{U}_H} \text{Ped.Com}([w_u]_s, [r_u]_s)^{q_u} \times c_s \\ &= \prod_{u \in \mathcal{U}_H} \left(\prod_{k=1}^{t-1} c_{u,k}^{x_k^s} \times c_u \right)^{q_u} \times c_s. \end{aligned}$$

Based on (8) and (9), the left hand side of (7) is that:

$$\begin{aligned} \text{Ped.Com}(w', r') &= \text{Ped.Com} \left(w + \sum_{s \in \mathcal{S}_t} v_s, \sum_{u \in \mathcal{U}_H} q_u \cdot r_u + \sum_{s \in \mathcal{S}_t} r_s \right) \\ &= \text{Ped.Com} \left(\sum_{u \in \mathcal{U}_H} q_u \cdot w_u, \sum_{u \in \mathcal{U}_H} q_u \cdot r_u \right) \times \prod_{s \in \mathcal{S}_t} c_s \\ &= \prod_{u \in \mathcal{U}_H} c_u^{q_u} \times \prod_{s \in \mathcal{S}_t} c_s. \end{aligned}$$

Therefore, the correctness of (6) and (7) is guaranteed.

VI. THEORETICAL ANALYSIS

A. Theoretical Analysis of the Necessity of Secondary Servers

In conventional FL architectures (user-aggregator systems), three methods exist to achieve both DP and BR. We analyze their fundamental limitations to demonstrate the necessity of introducing secondary servers.

Case 1. Local DP with BR: In this case, users inject DP noise into their local updates before transmitting them to the aggregator, which subsequently applies BR mechanisms. While theoretically appealing, this approach suffers from a critical statistical incompatibility as demonstrated in recent studies [32], [48]. Specifically, the introduction of DP noise significantly increases the variance of updates (such as gradients), severely disrupting the statistical properties used to identify malicious (Byzantine) updates.

Case 2. BR with Global DP: This approach employs SA to protect individual updates during transmission, followed by BR filtering at the aggregator and global DP on the final model. While SA prevents direct access to individual gradients, critical privacy risks persist:

First, even with SA protecting individual updates, the exact global update (e.g. gradient) becomes accessible to the aggregator after decryption. Through iterative observations of unprotected global updates across rounds, curious aggregator can reconstruct individual contributions through inference attacks, such as membership inference [18] or privacy reconstruct attack [4]. This risk escalates in long-running training processes, as cumulative information leakage violates the compositional privacy principle—a flaw unaddressed by SA alone.

Second, the architecture further introduces contradictory trust assumptions. The aggregator must be trusted to faithfully execute BR faithfully using raw updates (to ensure robustness) while simultaneously being distrusted from exploiting those same updates to infer private information (to ensure privacy). This paradox cannot be efficiently resolved within a conventional FL architecture, as highlighted in [49]. SA alone cannot mitigate this conflict, as it only secures the aggregation process and does not protect the privacy of the decrypted global states. Thus, global DP must directly perturb the aggregated model to break statistical correlations, but doing so after BR operations fails to protect intermediate results.

Case 3. Fully Decentralized MPC Implementation: A theoretically viable alternative employs secure MPC among users to jointly implement BR and DP without centralized aggregators. However, practical implementation faces insurmountable scalability barriers. For n participants, MPC protocols like SPDZ [50] require $\mathcal{O}(n^2)$ communication complexity per training round due to pairwise consistency verifications—a prohibitive overhead for large-scale FL systems with $n > 10^3$ users. Moreover, resource-constrained edge devices cannot sustain the cryptographic computations required for high-dimensional models (e.g., neural networks with $d > 10^6$ parameters), as shown by latency measurements in [51]. These limitations reduce theoretical guarantees to impractical ideals, particularly for time-sensitive applications.

In summary, the three aforementioned approaches for achieving both DP and BR in conventional FL architectures are all impractical. The necessity of secondary servers thus emerges not as an optional optimization but as a foundational requirement for achieving practical, privacy-preserving, and Byzantine-resilient FL systems. No existing conventional FL architecture or fully decentralized protocol can meet these requirements simultaneously without introducing significant trade-offs in security, efficiency, or statistical validity.

B. Cryptographic Assumptions and Definitions

Definition 1 (Discrete Logarithm Assumption (DLA) [52], [53]): Let G be a finite cyclic group of order n with generator α , and let $\beta \in G$. The discrete logarithm of β to the base α , denoted $\log_\alpha \beta$, is the unique integer x such that $\beta = \alpha^x \bmod n$. The discrete logarithm problem is considered hard if for any probabilistic polynomial-time adversary \mathcal{M} , there exists a negligible function $\text{negl}(n)$ such that

$$\Pr[\mathcal{M}(\alpha, \beta) = x] \leq \text{negl}(n).$$

Definition 2 ((ϵ, δ) -Differential Privacy [54]): A randomized mechanism $M : X \rightarrow Y$ satisfies (ϵ, δ) -differential privacy if for any two adjacent inputs $x, x' \in X$ differing in at most one element and any subset $S \subseteq Y$,

$$\Pr[M(x) \in S] \leq e^\epsilon \Pr[M(x') \in S] + \delta.$$

C. Differential Privacy Analysis

Lemma 1 (Binomial Mechanism [45], [55]): Let $X = (x_1, \dots, x_n) \in \mathbb{Z}_q^n$ and define the counting query $Q(X) = \sum_{i=1}^n x_i$. Fix $n_b > 30$, $0 < \delta \leq \mathcal{O}(1/n_b)$, and let $Z \sim \text{Binomial}(n_b, \frac{1}{2})$. Then $Q(X) + Z$ satisfies (ϵ, δ) -differential privacy with $\epsilon = 10\sqrt{\frac{1}{n_b} \ln \frac{2}{\delta}}$.

A detailed derivation follows the proof structure in [23].

Theorem 1 (Global Model Privacy): In DPSFL, the global model update is computed as

$$w' = w + \sum_{s \in \mathcal{S}_t} v_s,$$

where w is the aggregated model update from honest users and v_s is the noise injected by secondary server s . Under the Binomial Mechanism, the global model satisfies (ϵ, δ) -differential privacy with

$$\epsilon = 10\sqrt{\frac{1}{t \cdot n_b} \ln \frac{2}{\delta}},$$

where t is the threshold for secret reconstruction, and n_b is the per-server noise parameter.

Proof: Since each secondary server injects noise $v_s \sim \text{Binomial}(n_b, \frac{1}{2})$, the total noise added to the global model is $\sum_{s \in \mathcal{S}_t} v_s \sim \text{Binomial}(t \cdot n_b, \frac{1}{2})$. Applying Lemma 1 with total noise parameter $t \cdot n_b$, the privacy guarantee follows directly. \square

D. Impact of Secondary Servers on Privacy

Corollary 1 (Impact of Secondary Servers on Privacy): Let the global model satisfy (ε, δ) -DP with noise generated collaboratively by $|\mathcal{S}|$ secondary servers under threshold t . For fixed ε , if $t = \Theta(|\mathcal{S}|)$ (e.g., $t = \lfloor \frac{|\mathcal{S}|}{2} \rfloor + 1$), then the per-server noise parameter n_b scales as $n_b = \mathcal{O}(\frac{1}{|\mathcal{S}|})$. Thus, increasing $|\mathcal{S}|$ reduces individual noise requirements while preserving privacy.

Proof: From Theorem 1, the global model guarantees (ε, δ) -DP with: $\varepsilon = 10\sqrt{\frac{1}{t \cdot n_b}} \ln \frac{2}{\delta}$.

Rearranging for $t \cdot n_b$:

$$t \cdot n_b = \frac{100}{\varepsilon^2} \ln \frac{2}{\delta} \triangleq C(\varepsilon, \delta),$$

where $C(\varepsilon, \delta)$ is a constant for fixed ε, δ .

Under the threshold scheme $t = \Theta(|\mathcal{S}|)$, let $t = k|\mathcal{S}|$ for constant $1/3 \leq k \leq 1/2$ (e.g., $k = 1/2$ for $t = \lfloor |\mathcal{S}|/2 \rfloor + 1$). Substituting $t = k|\mathcal{S}|$ into the equation:

$$k|\mathcal{S}| \cdot n_b = C(\varepsilon, \delta) \Rightarrow n_b = \frac{C(\varepsilon, \delta)}{k|\mathcal{S}|}.$$

Thus, $n_b = \mathcal{O}(\frac{1}{|\mathcal{S}|})$.

This implies: 1. For fixed ε , increasing $|\mathcal{S}|$ linearly reduces the per secondary server noise parameter n_b . 2. The total noise budget $t \cdot n_b = C(\varepsilon, \delta)$ remains constant, preserving the global (ε, δ) -DP guarantee. 3. Distributed noise generation across $|\mathcal{S}|$ secondary servers avoids single-point bottlenecks while maintaining privacy. \square

E. Information-Theoretic Security

Definition 3 (Information-Theoretic Security (IT-Security)) [56]: A cryptosystem is information-theoretically secure if its security does not rely on computational assumptions. That is, even an adversary with unlimited computational power cannot break the system, except by directly accessing the secret key.

Theorem 2 (Security Under Collusion): The colluding malicious users ($u_{\mathcal{M}} \in \mathcal{U}_{\mathcal{M}}$) and a few secondary servers ($s_{\mathcal{M}} \in \mathcal{S}_{\mathcal{M}}$, $|\mathcal{S}_{\mathcal{M}}| < t$) are unknown for any information of another user under IT-Security and the DLA.

Proof: Firstly, the aggregator \mathcal{A} is not under adversarial control and does not collude with malicious parties. Thus, we do not consider the security threat of the aggregator in this proof.

As we known, such as Shamir secret sharing are information-theoretically secure in that having less than the requisite number of shares of the secret provides no information about the secret [57].

The adversary can perform two types of attacks: passive (eavesdropping and analytical inference) and active (protocol manipulation).

Passive listening and analytical attack

In this way, the adversary passively collects information and attempts to analyze the privacy of honest users. It can obtain the following information about user u during training:

- $([w_u]_s, [r_u]_s)$: The share of the local model of the user u ;
- (cs_u) : The commitment of the local model of the user u and the commitment of the coefficient;

- The information leaked during calculating $\text{Valid}(w_u)$.

Based on information-theoretic security, the adversary cannot gain any information about the user u from the share $([w_u]_s, [r_u]_s)$. Furthermore, the adversary cannot gain any information from the commitment: (cs_u) , unless it can solve the discrete logarithm problem. During the process of calculating $\text{Valid}(\cdot)$, the secondary servers interactively perform the multiplication by using the multiplication triple. However, the information in the interaction is random and does not contain any privacy. Therefore, the passive adversary does not gain any privacy about the honest user u .

Active attack against the protocol: In this way, the adversary can attack the protocol to obtain information or even break it. Consider a malicious adversary \mathcal{M} who controls some users $u_{\mathcal{M}} \in \mathcal{U}_{\mathcal{M}}$ and a few secondary servers $s_{\mathcal{M}} \in \mathcal{S}_{\mathcal{M}}$, where $|\mathcal{S}_{\mathcal{M}}| < t$. The adversary \mathcal{M} can perform the following attacks:

- 1) **DoS/DDoS Attack:** The adversary \mathcal{M} can initiate Denial-of-Service (DoS) or Distributed Denial-of-Service (DDoS) attacks leveraging controlled users and secondary servers. These attacks aim to make the system inaccessible by overwhelming it with traffic or requests that exceed its processing capacity.
- 2) **Data tampering:** The adversary \mathcal{M} can alter the data submitted by the users and the secondary servers to distort the results or mislead other parties. This attack can undermine the trustworthiness and availability of the system. The adversary can tamper following data:
 - $([w_{u_{\mathcal{M}}}]_s, [r_{u_{\mathcal{M}}}]_s, cs_{u_{\mathcal{M}}})$: The data from malicious users $u_{\mathcal{M}}$.
 - $\{P_{s_{\mathcal{M}},j}, c_{s_{\mathcal{M}},j} | j \in [n_b]\}$: The data from malicious secondary servers $s_{\mathcal{M}}$ during the phase 2. **Verifiable Noise Negotiation.**
 - The interaction data with other secondary servers in phase 3. **Evaluating the Integrity and Byzantine Robustness of Local Models.**
 - $([w']_{s_{\mathcal{M}}}, [r']_{s_{\mathcal{M}}}, V_{s_{\mathcal{M}}}, R_{s_{\mathcal{M}}})$: The data from malicious secondary servers $s_{\mathcal{M}}$ during aggregation.

For the DoS / DDoS attacks, the DPSFL has a powerful detection and filtering mechanism. If some users drop out, the honest secondary servers report this occurrence to the aggregator and remove the updates related to the dropped users. If the updates from malicious secondary servers do not align with those from most other secondary servers or if they drop out, the aggregator will broadcast the IDs of these secondary servers and exclude them from the training process.

For data tempering, our scheme incorporates a verification mechanism based on Pedersen's Commitment into all enumerated updates to verify their correctness. Therefore, an adversary cannot temper with any updates unless it can solve the discrete logarithm problem.

In summary, Theorem 3. is proved to be correct. \square

F. Trade-Offs Between Differential Privacy and Byzantine Resilience

The DPSFL framework theoretically reconciles DP and BR through three interconnected mechanisms.

First, its hierarchical architecture isolates BR validation from DP noise injection: Byzantine detection is performed on raw gradients using Secure MPC to preserve detection accuracy, while binomial noise is applied solely to the aggregated model. This prevents DP perturbations from distorting BR statistical analysis (Theorem 2).

Second, cryptographic protocols (e.g., Π_{OR} proofs, Shamir sharing) ensure adversarial collusion cannot bypass BR checks or reconstruct raw updates (local and global), securing both phases against adaptive threats (Theorem 2).

Adjustable parameter tuning further harmonizes these guarantees—tightening BR thresholds under attacks while proportionally adjusting n_b to maintain DP—via verifiable MPC protocols. By confining BR to raw data and DP to aggregated outputs, DPSFL eliminates their mutual interference, achieving provable equilibrium between privacy, robustness, and utility.

VII. EXPERIMENTS

In this section, we empirically examine the accuracy of the global model when different noise sizes are added to DPSFL. Next, we evaluate its integrity using common BR methods and a series of attacks. Additionally, we compare the accuracy of the global model with different proportion of adversaries in DPSFL to the work of Zhou et al. [27].

Experiments are conducted on a single NVIDIA A5000-24 G GPU with AMD EPYC 7551P processors. Our scheme is implemented in Python using the PyTorch library, with cryptographic operations handled by the PyCrypto library [58].

A. Evaluation on Noise of the Lagrange's Interpolation

To assess the noise amplification effect in the secret recovery phase of noise-immune Shamir secret sharing, we design a comprehensive simulation experiment as outlined here:

The experimental setup involves a single secret holder and n participants (dealers) within a secret sharing framework. The secret holder possesses a confidential value x , which is initially disseminated among all dealers in accordance with an (n, t) -threshold Shamir secret sharing scheme. Subsequently, each dealer introduces noise to their respective share, where the noise is modeled by a Binomial distribution $\text{Binomial}(n_b, \frac{1}{2})$, symbolizing a controlled disturbance. These augmented shares are then transmitted to a central server. The server, employing Lagrange interpolation, attempts to reconstruct the original secret, yielding an estimate x' . A pivotal aspect of our analysis entails comparing x with x' to quantify the magnitude of noise amplification, thereby elucidating how the introduced noise propagates through the reconstruction process.

In exploring the behavior of noise amplification under varying noise profiles and thresholds t , we establish a noise magnitude spectrum defined by $\varepsilon = \{1, 2, 3, 4, 5\}$ and $\delta = 10^{-32}$. The values of $n = t$ both range from 3 to 50. The result of this experiment is as Fig. 3. We found that the noise size grows exponentially with the number of threshold t for ε ranging from 1 to 5. This means that we cannot recover the secret value when the secret share is noisy.

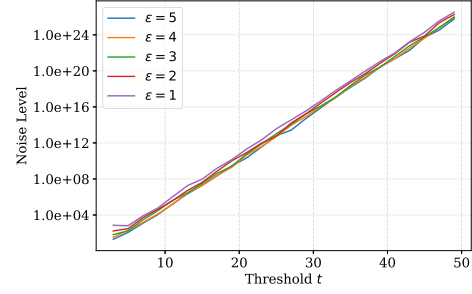


Fig. 3. The figure above shows how the average noise changes with the threshold t under below setting.

B. Evaluation on Accuracy

In the following experiment, we evaluate DPSFL on three datasets (MNIST, FMNIST and CIFAR-10), and divide the datasets by using Dirichlet distribution ($\alpha = 0.5$) to obtain non-IID datasets for FL training. The specific descriptions of the datasets and corresponding models are as follows:

Datasets:

- MNIST [59] is a handwritten digit recognition dataset containing 60,000 training images and 10,000 testing images of size 28x28 pixels, each labeled with one of ten digits from 0 to 9.
- FMNIST (Fashion-MNIST) [60] is a dataset of Zalando's paper images consisting of 60,000 training examples and 10,000 test examples of size 28x28 pixels, each associated with one of ten fashion categories like shoes, shirts, or bags.
- CIFAR-10 [61] is a dataset containing 60,000 32x32 color images in 10 different classes, widely employed for training and evaluating image classification algorithms.

Models:

- Multilayer Perceptron (MLP): We utilize a neural network of two layers and 400K parameters to train on the MNIST datasets.
- LeNet-5: We utilize the LeNet-5, a simple CNN network, with 5 layers and 1,986K parameters for FMNIST.
- We re-designed a small VGG net with 14 layers and 5,621K parameters for CIFAR-10.

To investigate the impact of noise intensity on federated learning performance, we configure a system with $|\mathcal{U}| = 50$ users and $|\mathcal{S}| = 5$ secondary servers, employing a user dropout rate of 0.01. Fig. 4(a)–(c) presents the per-round accuracy trajectories under differential privacy constraints ($\varepsilon \in \{1.0, 0.5, 0.1, 0.05\}$, $\delta = 10^{-32}$) and a noiseless baseline across MNIST, Fashion MNIST, and CIFAR-10 datasets.

The experimental results (Fig. 4) demonstrate the impact of DP noise levels ($\varepsilon = 1.0, 0.5, 0.1, 0.05$) on model accuracy and convergence across three benchmark datasets: MNIST, FMNIST, and CIFAR10. The analysis reveals significant variations in noise sensitivity depending on task complexity and training phases.

Noise Robustness Across Architectures: For MNIST (MLP model, Fig. 4(a)), the accuracy remains at 98.12% under $\varepsilon = 1.0$ (baseline: 98.24%) and 97.59% even at $\varepsilon = 0.05$,

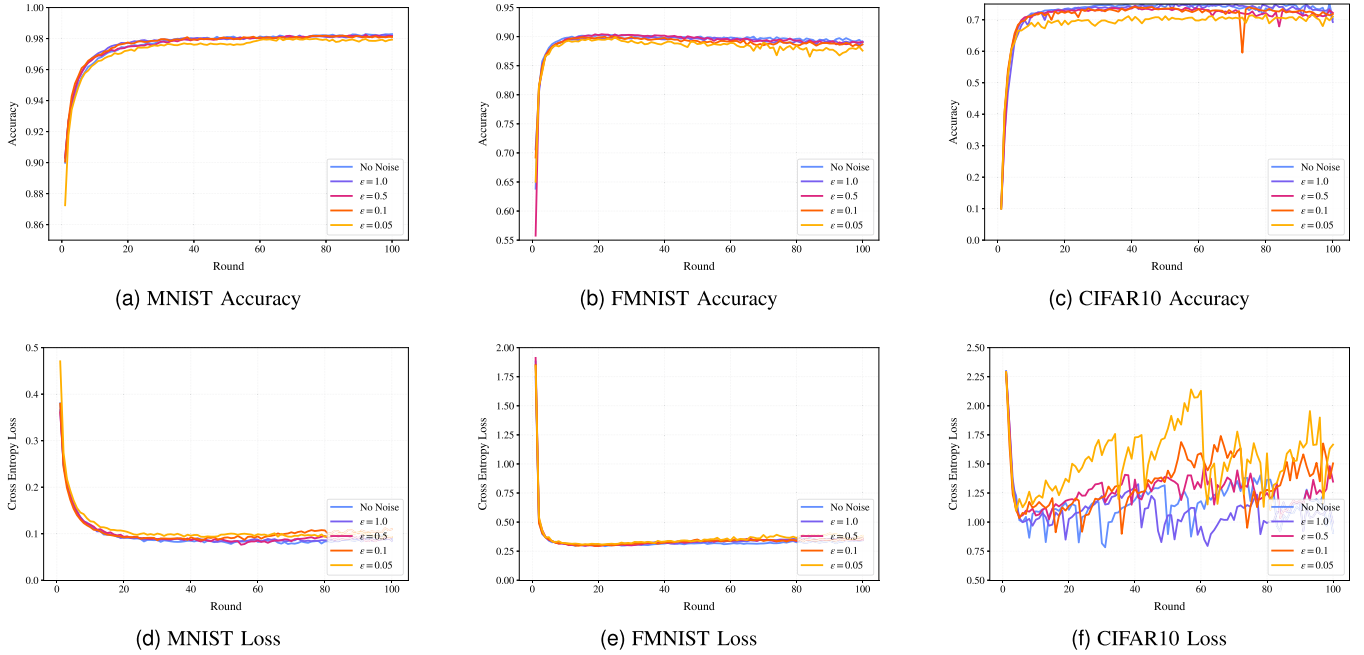


Fig. 4. Training accuracy and loss varies with noise size.

attributed to the global feature modeling capability of fully connected layers on low-dimensional grayscale data. In contrast, Fashion MNIST (LeNet-5 model, Fig. 4(b)) exhibits significant accuracy degradation to 88% at $\varepsilon = 0.05$ (baseline: 91.5%), primarily due to the sensitivity of its shallow convolutional structure (e.g., 5×5 kernels in the first layer) to local texture perturbations. CIFAR-10 (VGG-14 model, Fig. 4(c)) shows unexpected robustness, with only a 2% drop at $\varepsilon = 0.1$ (73% vs 75% baseline), suggesting that hierarchical nonlinear transformations in deep convolutional networks can partially buffer noise propagation, albeit with notable late-training instability.

Convergence Dynamics and Stability: Under identical noise levels, convergence efficiency decreases with task complexity. MNIST/MLP (Fig. 4(d)) stabilizes within 50 rounds (loss 0.1 ± 0.02), while CIFAR-10/VGG-14 (Fig. 4(f)) requires 65 rounds at $\varepsilon = 0.1$ (18% longer than baseline), accompanied by significant late-stage accuracy fluctuations (standard deviation $\pm 1.8\%$ vs baseline $\pm 0.6\%$). Further analysis reveals that DP noise amplifies gradient norms in the classification layers of deep networks, destabilizing parameter updates.

Practical Recommendations: Based on these findings, we propose: (1) For shallow convolutional models (e.g., LeNet-5), set $\varepsilon \geq 0.1$ to prevent feature extraction degradation; (2) Deep networks (e.g., VGG-14) benefit from extended training (20% additional rounds) and progressive noise decay (e.g., linearly reducing to 30% of initial noise in the final 10% of training), which experimentally reduces CIFAR-10 fluctuations to $\pm 1.1\%$; (3) Under high-privacy constraints ($\varepsilon = 0.05$), task viability varies: MNIST/MLP (Fig. 4(a)) maintains $> 97.5\%$ accuracy, whereas CIFAR-10/VGG-14 (Fig. 4(c)) is limited to 69.2% absolute accuracy (5.8% drop from baseline).

C. Evaluation on Integrity

To analyze the integrity of our scheme in the setting of malicious parties, we compare DPSFL with EIFFeL [16]. According to the above conclusion, we set the noise size $\varepsilon = 0.1$.

Additionally, we further select various effective BR methods, such as norm ball [37], cosine similarity validation [29], and FLAME [62]. To implement these defense strategies within MPC circuits, we initially note that the pivotal operations in these techniques all involve computing the dot product of two vectors, which may be held as secret shares among the parties. We recognize that the dot product's result does not compromise the vectors' privacy without prior knowledge of their relationships. Our goal is to create a function that computes the dot product of two secret-shared vectors and then reveals the dot product's result for further processing.

The comparative evaluation of DPSFL and EIFFeL under 30% malicious users reveals nuanced performance trade-offs shaped by their respective defense mechanisms. As shown in Fig. 5(a)–(e), the two frameworks exhibit nearly overlapping accuracy trajectories across additive noise, sign-flipping, and gradient manipulation attacks. This parity suggests that the shared BR methods—norm clipping, cosine similarity validation, and FLAME—dominate robustness outcomes in this threat regime. The marginal performance gap ($< 0.5\%$ across datasets) further indicates that DPSFL's calibrated DP noise neither degrades nor enhances robustness significantly when applied alongside these BR techniques, effectively preserving utility while enforcing privacy.

Divergence emerges in backdoor attack resilience (Fig. 5(g)). While EIFFeL achieves a 1% higher main-task accuracy on CIFAR-10, DPSFL reduces backdoor success rates by 5%—a

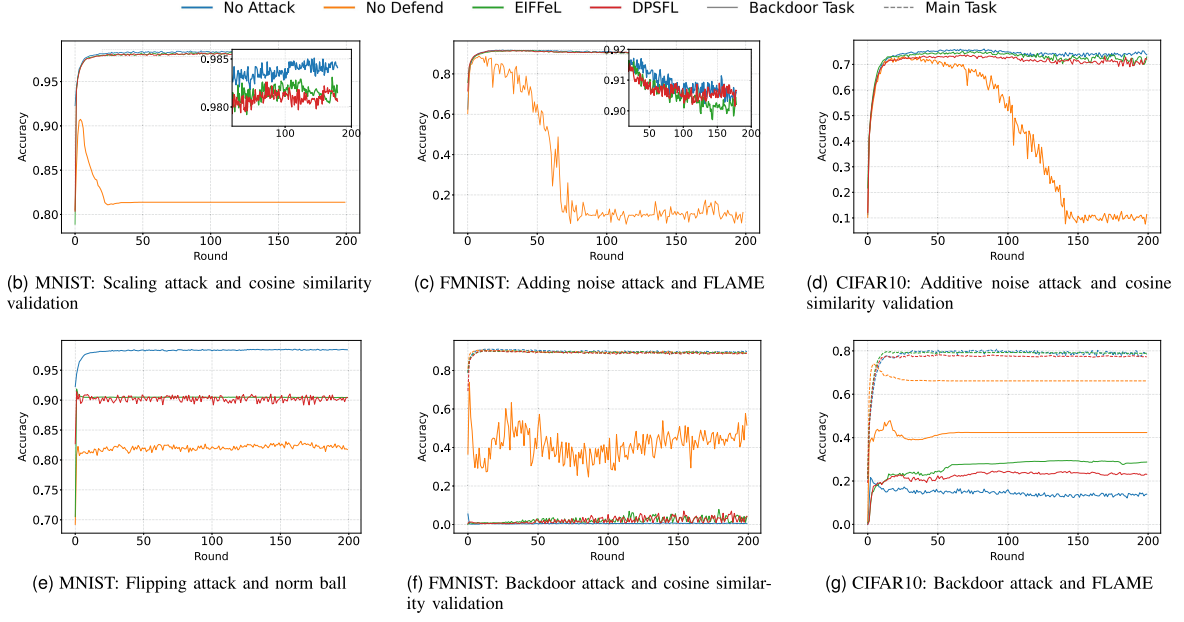


Fig. 5. Accuracy with different attacks and BR methods.

critical advantage for security-sensitive applications. This aligns with the hypothesis that DP noise disrupts the stealthy gradient patterns required for persistent backdoor embedding, whereas EIFFeL’s non differential privacy design remains vulnerable to such latent perturbations. The inverse relationship between main-task accuracy and backdoor suppression underscores a fundamental tension: privacy noise introduces targeted randomness that selectively degrades adversarial features more than legitimate learning signals.

Notably, the cosine similarity validation mechanism (Fig. 5(b), (d), and (f)) demonstrates equivalent efficacy in both frameworks, with MPC-secured computation preserving detection fidelity despite DPSFL’s noised gradients. This consistency confirms that the BR methods operate independently of the privacy layer under moderate attack intensities (30% malicious nodes), challenging the presumed incompatibility between DP and Byzantine resilience. However, DPSFL’s 0.2% accuracy drop on FMNIST under coordinated additive noise attacks (Fig. 5(c)) hints at edge cases where DP perturbations may slightly amplify benign gradient misalignment—a trade-off warranting further theoretical analysis.

These results collectively demonstrate that integrating DP with established BR mechanisms achieves functionally equivalent robustness to non-private alternatives (EIFFeL) within the 30% adversarial threshold, while providing quantifiable privacy benefits. The framework’s backdoor resistance further positions DP as a complementary defense layer against gradient-space attack vectors.

D. Experimental Evaluation of System Overheads

The DPSFL framework was evaluated for end-to-end latency and model performance under varying user scales ($|\mathcal{U}|$) and the number of secondary servers ($|\mathcal{S}|$), with $\varepsilon = 0.1$ and Byzantine

tolerance $t = \lfloor |\mathcal{S}|/2 \rfloor$. Experiments on MNIST and CIFAR-10 (Fig. 6, Table II) reveal critical trade-offs between scalability and efficiency.

For MNIST (Fig. 6(a)), latency scales linearly from 68.4 s ($|\mathcal{U}| = 50$, $|\mathcal{S}| = 10$) to 142.3 s ($|\mathcal{U}| = 500$), a $2.08\times$ increase. However, with $|\mathcal{U}| = 50$, $|\mathcal{S}| = 50$, latency surges to 214.5 s ($3.14\times$ higher), reflecting quadratic complexity growth in MPC verification. CIFAR-10 exhibits similar trends, with $|\mathcal{S}| = 50$ and $|\mathcal{U}| = 500$ incurring 2196.5 s latency, where 68% of time is spent on noise verification due to $10\times$ larger parameter size amplifying Shamir secret sharing and Pedersen commitment overheads. Notably, both tasks achieve sub-minute latency at $|\mathcal{S}| = 10$ (MNIST: 68.4 s, CIFAR-10: 357.6 s), demonstrating real-time feasibility in lightweight configurations.

Communication and accuracy metrics (Table II) further highlight task-specific sensitivities. MNIST maintains stable accuracy (97.12–98.12%) with modest communication growth (5.1 MB/user at $|\mathcal{S}| = 10$ vs. 15.3 MB at $|\mathcal{S}| = 50$), aligning with secret sharing redundancy $k = 2t + 1$ (error $< 2\%$). In contrast, CIFAR-10 shows pronounced trade-offs: communication increases from 10.2MB ($|\mathcal{S}| = 10$) to 30.6MB ($|\mathcal{S}| = 50$), while accuracy declines from 75.2% to 70.1%, indicating high-dimensional models’ vulnerability to noise accumulation and data heterogeneity. Convergence efficiency also diverges: MNIST converges in 42 rounds ($|\mathcal{S}| = 20$), whereas CIFAR-10 requires 230 rounds under the same configuration, escalating to 415 rounds at $|\mathcal{S}| = 50$, as higher server counts reduce global update frequency through latency accumulation.

These results establish practical guidelines: For lightweight tasks (e.g., MNIST), $|\mathcal{S}| = 20$ balances latency ($< 500s$) and accuracy ($> 98\%$). For complex tasks (e.g., CIFAR-10), limiting $|\mathcal{S}| \leq 20$ prevents verification overhead from dominating system latency ($> 65\%$) while compensating convergence loss via dynamic local training adjustments. This empirically

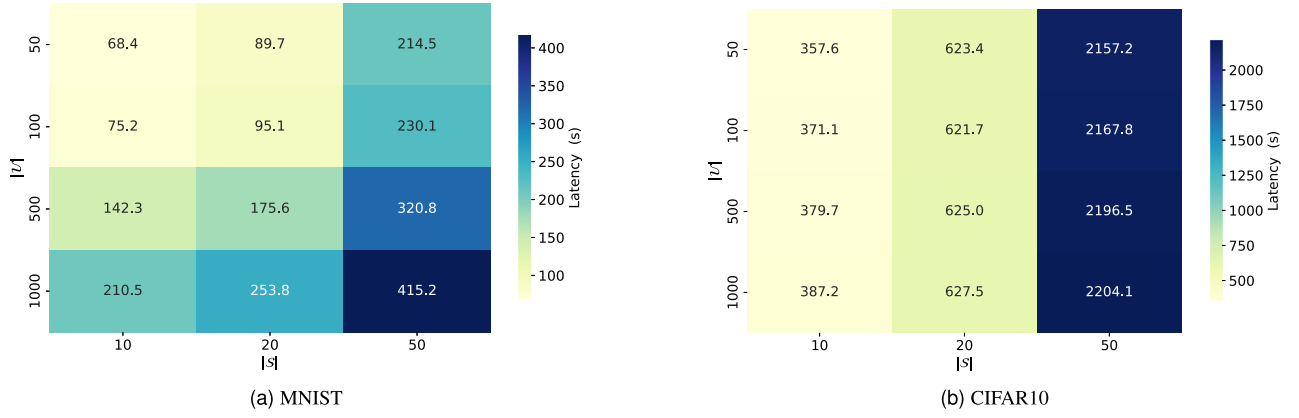


Fig. 6. End to end latency.

TABLE II
ANALYSIS OF DPSFL SYSTEM PERFORMANCE ON MNIST AND CIFAR-10 DATASETS

Dataset	$ \mathcal{U} $	$ \mathcal{S} $	Communication Per User (MB)	Accuracy (%)	Convergence Rounds
MNIST	50	10	5.1	98.03	41
	100	10	5.1	97.51	72
	500	10	5.1	97.12	153
	50	20	10.2	98.12	42
	100	20	10.2	97.62	70
	500	20	10.2	97.01	147
	50	50	25.5	98.24	40
	100	50	25.5	97.55	73
	500	50	25.5	96.98	157
CIFAR-10	50	10	71.5	73.58	52
	100	10	71.5	71.27	127
	500	10	71.5	67.58	253
	50	20	142.8	73.15	51
	100	20	142.8	71.15	132
	500	20	142.8	68.29	265
	50	50	357	73.27	50
	100	50	357	70.82	124
	500	50	357	67.89	267

validates DPSFL's capability to harmonize privacy, efficiency, and robustness in cross-heterogeneous federated learning scenarios.

E. Compare With Others

In Section II-C, we reference a related study introduced by Zhou et al. [27], which likewise proposes a framework integrating DP with BR. Subsequently, we delve deeper into comparing the advantages and disadvantages of our proposed scheme versus that of Zhou et al. [27] through a series of experiments.

To enhance the efficiency of BR amidst noisy conditions, Zhou et al. [27] put forth a novel BR strategy that combine norm-based detection with accuracy-based detection. To elucidate the distinctions between our scheme and that of Zhou et al. [27], we incorporate their BR methodology within a MPC circuit. This integration poses several pivotal challenges:

- i) The circuit must compute the accuracy of local models on a sample dataset, yet the parameters of these models are secret-shared, necessitating intricate handling of confidential information.
- ii) Unlike a straightforward filtering of anomalous models, the output of the circuit is intended to be the updated weight of each local model, further complicating the design.

For experimental purposes, we consider MLP and CNN models as the analysis cases. Recognizing that fully connected layers correspond to vector calculations and that convolutional and pooling layers can be translated into linear operations simplifies the process. Moreover, a compact validation dataset is shared among all secondary servers, simplifying accuracy computations. Regarding the weights of local models, we adopt a two-step approach: initially calculating and subsequently disclosing them, followed by aggregation across all local models.

TABLE III
COMPARE DPSFL WITH ZHOU ET AL.

Malicious Users Ratio	MNIST		FMNIST		CIFAR-10	
	DPSFL	Zhou <i>et al.</i> [27]	DPSFL	Zhou <i>et al.</i> [27]	DPSFL	Zhou <i>et al.</i> [27]
0	98.04%	97.57%	91.90%	91.35%	74.37%	73.98%
20%	97.83%	97.49%	91.45%	90.97%	72.28%	70.03%
40%	97.47%	97.43%	91.40%	90.48%	71.86%	68.70%
60%	97.28%	96.71%	90.41%	89.71%	68.75%	64.59%

To analyze the Byzantine robustness of DPSFL and the scheme proposed by Zhou et al. [27], we first consider a scenario where malicious users conduct label-flipping attacks while secondary servers perform cosine-similarity validation.

This experiment systematically evaluates the robustness of DPSFL and Zhou et al. [27]'s method under varying malicious users ratios (0%, 20%, 40%, 60%) in federated learning. As shown in Table III, DPSFL consistently outperforms Zhou et al.'s method, particularly under high-adversarial conditions. On MNIST, DPSFL achieves 97.28% accuracy at 60% malicious users, surpassing Zhou et al. by 0.57%, while on CIFAR-10, the accuracy gap widens to 4.16% (68.75% vs. 64.59%), highlighting its effectiveness in complex scenarios.

We observe three key aspects emerging:

1. *Progressive Robustness Degradation*: While both methods exhibit declining accuracy with increasing adversarial ratios, DPSFL maintains smaller accuracy drops. For MNIST, DPSFL's accuracy declines by 0.76% (98.04% \rightarrow 97.28%) versus Zhou et al.'s 0.86% (97.57% \rightarrow 96.71%). This trend amplifies for high-dimensional tasks: on CIFAR-10, DPSFL's 5.62% drop contrasts sharply with Zhou et al.'s 9.39% decline, underscoring its resilience to gradient-space attacks.

2. *Task-Specific Superiority*: DPSFL excels in both simple and complex learning tasks. For FMNIST, it sustains $> 90\%$ accuracy even at 60% malicious users (90.41% vs. 89.71%), whereas Zhou et al. falters earlier, reflecting DPSFL's adaptive noise allocation and centralized Byzantine validation.

3. *Backdoor Resistance*: The widening performance gap under higher adversarial ratios (e.g., +4.16% for CIFAR-10 at 60%) suggests DPSFL's fixed-parameter DP noise effectively disrupts adversarial model updates, while Zhou et al.'s dynamic noise adjustment introduces instability that exacerbates accuracy loss.

These results validate DPSFL's architectural advantages: its sequential validation-noise pipeline isolates Byzantine detection from DP perturbations, ensuring precise gradient filtering before noise injection. In contrast, Zhou et al.'s integrated approach struggles with mutual interference between BR and DP, leading to suboptimal equilibrium. The experiments further confirm DPSFL's scalability across heterogeneous tasks, positioning it as a robust solution for privacy-preserving federated learning in adversarial environments.

VIII. CONCLUSION

In this paper, we introduced a novel federated learning framework that balances privacy protection with Byzantine robustness. By using secondary servers, our approach separates robustness checking from differential privacy noise injection. First,

the system detects malicious local updates without interference; then, during aggregation, it adds verifiable DP noise to secure the data and ensure reliable model updates.

The key innovation is our noise-added Shamir secret sharing protocol, which incorporates (ϵ, δ) -DP noise into the sharing process. With cryptographic commitments and secure multi-party computation, this method overcomes the noise amplification issues found in traditional approaches. Experiments on MNIST, FMNIST, and CIFAR-10 show that the framework maintains nearly 98% of baseline accuracy for $\epsilon \geq 0.1$ and tolerates up to 60% malicious users, improving accuracy by 0.57-4.16% compared to existing methods. Moreover, the system scales linearly with the number of participants.

Our framework's success is driven by three main ideas:

- 1) Using secondary servers to separately handle robustness evaluation and privacy protection.
- 2) Integrating MPC-based verification with Pedersen commitments to secure model updates and noise generation.
- 3) distributing the noise generation across multiple secondary servers, which lowers the noise burden on individual nodes while still meeting DP guarantees.

This approach shows strong resistance against attacks such as label flipping, model poisoning, and backdoor injections, making it promising for applications in privacy-sensitive areas like healthcare and smart grids. Future work could investigate dynamic noise adjustment and hardware-accelerated verification to further enhance system reliability in adversarial environments.

REFERENCES

- [1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
- [2] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 14774–14784.
- [3] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients - How easy is it to break privacy in federated learning," in *Advances in Neural Information Processing Systems*, H.M. Larochelle, R. Ranzato, M. Hadsell, Balcan, and H. Lin, Eds. Red Hook, NY, USA: Curran Associates, Inc., 2020, pp. 16937–16947. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/c4ede56bbd98819ae6112b20ac6bf145-Paper.pdf
- [4] T. Marchand, R. Loeb, U. Marteau-Ferey, J. O. D. Terrail, and A. Pignet, "SRATTA: Sample re-ATtribution attack of secure aggregation in federated learning," 2023, *arXiv:2306.07644*.
- [5] L. Wang, S. Xu, X. Wang, and Q. Zhu, "Eavesdrop the composition proportion of training labels in federated learning," 2019, *arXiv: 1910.06044*.
- [6] 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.
- [7] T. D. Nguyen, P. Rieger, M. Miettinen, and A.-R. Sadeghi, "Poisoning attacks on federated learning-based IoT intrusion detection system," in *Proc. Workshop Decentralized IoT Syst. Secur.*, 2020, pp. 1–7.

- [8] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 118–128.
- [9] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 1605–1622.
- [10] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *Proc. Annu. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–18.
- [11] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 634–643.
- [12] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. 2017 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191.
- [13] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly) logarithmic overhead," in *Proc. 2020 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 1253–1269.
- [14] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran, "Fast-SecAgg: Scalable secure aggregation for privacy-preserving federated learning," 2020, *arXiv: 2009.11248*.
- [15] C.-S. Yang, J. So, C. He, S. Li, Q. Yu, and S. Avestimehr, "LightsSecAgg: Rethinking secure aggregation in federated learning," 2021, *arXiv:2109.14236*.
- [16] A. Roy Chowdhury, C. Guo, S. Jha, and L. van der Maaten, "Eiffel: Ensuring integrity for federated learning," in *Proc. 2022 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2022, pp. 2535–2549.
- [17] D. Pasquini, D. Francati, and G. Ateniese, "Eluding secure aggregation in federated learning via model inconsistency," in *Proc. 2022 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2022, pp. 2429–2443.
- [18] R. Kerkouche, G. Ács, and M. Fritz, "Client-specific property inference against secure aggregation in federated learning," in *Proc. 22nd Workshop Privacy Electron. Soc.*, 2023, pp. 45–60.
- [19] C. Zhou, A. Fu, S. Yu, W. Yang, H. Wang, and Y. Zhang, "Privacy-preserving federated learning in fog computing," *IEEE Internet Things J.*, vol. 7, no. 11, pp. 10782–10793, Nov. 2020.
- [20] S. Hardy et al., "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," 2017, *arXiv: 1711.10677*.
- [21] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. 2020 USENIX Annu. Tech. Conf.*, 2020, pp. 493–506.
- [22] K. Wei et al., "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 3454–3469, 2020.
- [23] P. Kairouz, Z. Liu, and T. Steinke, "The distributed discrete gaussian mechanism for federated learning with secure aggregation," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 5201–5212.
- [24] J. Gao et al., "Secure aggregation is insecure: Category inference attack on federated learning," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 147–160, Jan./Feb. 2023.
- [25] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "LDP-fed: Federated learning with local differential privacy," in *Proc. 3rd ACM Int. Workshop Edge Syst. Analytics Netw.*, 2020, pp. 61–66.
- [26] T. Stevens, C. Skalka, C. Vincent, J. Ring, S. Clark, and J. Near, "Efficient differentially private secure aggregation for federated learning via hardness of learning with errors," in *Proc. 31st USENIX Secur. Symp.*, 2022, pp. 1379–1395.
- [27] J. Zhou et al., "A differentially private federated learning model against poisoning attacks in edge computing," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 3, pp. 1941–1958, May/Jun. 2023.
- [28] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5650–5659.
- [29] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," 2020, *arXiv: 2012.13995*.
- [30] L. Burkhalter, H. Lycklama, A. Viand, N. Küchler, and A. Hithnawi, "RoFL: Attestable robustness for secure federated learning," 2021, *arXiv:2107.03311*.
- [31] M. Rathee, C. Shen, S. Wagh, and R. A. Popa, "ELSA: Secure aggregation for federated learning with malicious actors," in *Proc. 2023 IEEE Symp. Secur. Privacy*, 2023, pp. 1961–1979.
- [32] R. Guerraoui, N. Gupta, R. Pinot, S. Rouault, and J. Stephan, "Differential privacy and Byzantine resilience in SGD: Do they add up?," in *Proc. 2021 ACM Symp. Princ. Distrib. Comput.*, 2021, pp. 391–401.
- [33] C. Xie, S. Koyejo, and I. Gupta, "Zeno: Robust fully asynchronous SGD," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 10495–10503.
- [34] C. Dong et al., "Privacy-preserving and Byzantine-robust federated learning," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 2m, pp. 889–904, Mar./Apr. 2024.
- [35] T. D. Nguyen et al., "Flguard: Secure and private federated learning," *Cryptol. ePrint Arch.*, 2021. [Online]. Available: <https://arxiv.org/abs/2101.02281v2>
- [36] X. Tang, M. Shen, Q. Li, L. Zhu, T. Xue, and Q. Qu, "PILE: Robust privacy-preserving federated learning via verifiable perturbations," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 6, pp. 5005–5023, Nov./Dec. 2023.
- [37] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3520–3532.
- [38] W.-N. Chen, C. A. C. Choo, P. Kairouz, and A. T. Suresh, "The fundamental price of secure aggregation in differentially private federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 3056–3089.
- [39] H. Corrigan-Gibbs and D. Boneh, "Prio: Private, robust, and scalable computation of aggregate statistics," in *Proc. 14th USENIX Symp. Netw. Syst. Des. Implementation*, 2017, pp. 259–282.
- [40] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.
- [41] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. Annu. Int. Cryptol. Conf.*, 1991, pp. 129–140.
- [42] D. Escudero, "Multiparty Computation Over $\mathbb{Z}/2\mathbb{K}$," 2021. [Online]. Available: https://pure.au.dk/portal/files/229201603/Thesis_Daniel_Esteban.pdf
- [43] A. Narayan, A. Feldman, A. Papadimitriou, and A. Haeberlen, "Verifiable differential privacy," in *Proc. 10th Eur. Conf. Comput. Syst.*, 2015, pp. 1–14.
- [44] D. Keeler, C. Komlo, E. Lepert, S. Veitch, and X. He, "DPrio: Efficient differential privacy with high utility for Prio," *Proc. Privacy Enhancing Technol.*, vol. 2023, no. 3, pp. 375–390, 2023.
- [45] A. Biswas and G. Cormode, "Verifiable differential privacy," 2022, *arXiv:2208.09011*.
- [46] J. Groth and M. Kohlweiss, "One-out-of-many proofs: Or how to leak a secret and spend a coin," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2015, pp. 253–280.
- [47] E. Kuznetsov, Y. Chen, and M. Zhao, "Securefl: Privacy preserving federated learning with sgx and trustzone," in *Proc. 2021 IEEE/ACM Symp. Edge Comput.*, 2021, pp. 55–67.
- [48] H. Zhu and Q. Ling, "Bridging differential privacy and Byzantine-robustness via model aggregation," 2022, *arXiv:2205.00107*.
- [49] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2168–2181, Jul. 2021.
- [50] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Proc. Annu. Cryptol. Conf.*, 2012, pp. 643–662.
- [51] E. Hallaji, R. Razavi-Far, M. Saif, B. Wang, and Q. Yang, "Decentralized federated learning: A survey on security and privacy," *IEEE Trans. Big Data*, vol. 10, no. 2, pp. 194–213, Apr. 2024.
- [52] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, 2018.
- [53] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. 2nd Ed. Boca Raton, FL, USA: CRC Press, 2014. [Online]. Available: <https://www.crcpress.com/Introduction-to-Modern-Cryptography-Second-Edition/Katz-Lindell/p/book/9781466570269>
- [54] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. 3rd Theory Cryptogr. Conf. Theory Cryptogr.*, New York, NY, USA, 2006, pp. 265–284.
- [55] B. Ghazi et al., "On the power of multiple anonymous messages," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2020, pp. 463–488. [Online]. Available: <https://eprint.iacr.org/2019/1382>
- [56] U. Maurer, "Information-theoretic cryptography," in *Advances in Cryptology — CRYPTO '99*. Berlin, Germany: Springer, 1999, pp. 47–64.
- [57] R. Ahlswede and I. Csiszár, "Common randomness in information theory and cryptography. I. secret sharing," *IEEE Trans. Inf. Theory*, vol. 39, no. 4, pp. 1121–1132, Jul. 1993.
- [58] 2012. [Online]. Available: <https://www.pycrypto.org/>
- [59] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

- [60] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv: 1708.07747*.
- [61] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," Tech. Rep. TR-2009, Univ. Toronto, Toronto, 2009. [Online]. Available: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>
- [62] T. D. Nguyen et al., "flame: Taming backdoors in federated learning," in *Proc. 31st USENIX Secur. Symp.*, 2022, pp. 1415–1432.



Chushan Zhang received the BS degree from the College of Computer Science and Technology, Han-shan Normal University. He is currently working toward the MS degree with the College of Cyber Security, Jinan University. He researches in the field of federated learning.



Jian Weng (Member, IEEE) received the PhD degree in computer science and engineering from Shanghai Jiao Tong University, Shanghai, China, in 2008. From 2008 to 2010, he was a postdoctoral fellow with the School of Information Systems, Singapore Management University, Singapore. He is currently a professor and the vice-chancellor with the Jinan University, Guangzhou, China. He has published more than 300 papers in cryptography and security conferences and journals, such as CRYPTO, EUROCRYPT, ASIACRYPT, CCS, Usenix Security, NDSS, *IEEE*

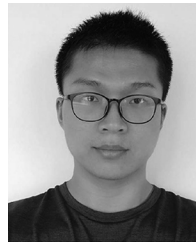
Transactions on Pattern Analysis and Machine Intelligence, *IEEE Transactions on Information Forensics and Security*, and *IEEE Transactions on Dependable and Secure Computing*. His research interests include public key cryptography, cloud security, and blockchain. He served as a PC co-chair or a PC member for more than 30 international conferences. He also serves as an associate editor for *IEEE Transactions on Vehicular Technology*.



Jiasi Weng (Member, IEEE) received the BS degree in software engineering from South China Agriculture University, and the PhD degree from the College of Information Science and Technology & College of Cyber Security, Jinan University. She researches in the field of trustworthy machine learning. She has published over ten papers in peer-reviewed venues and journals.



Yijian Zhong received the BS degree in computer science and technology from Shanghai University, Shanghai, China, in 2017, and the MS degree in computer science from South China Agricultural University, Guangzhou, China, in 2019 respectively. He is currently working toward the PhD degree with Jinan University, Guangzhou, China. His research interests include applied cryptography and security in cloud computing.



Jia-Nan Liu (Member, IEEE) received the BS degree from Zhengzhou University, Zhengzhou, China, in 2013, the MS and PhD degrees from Jinan University, Guangzhou, China, in 2016 and 2020, respectively. He is currently an associate professor with the School of Computer Science and Technology, Dongguan University of Technology. His research interests include cryptography and cloud computing security.



Cunle Deng received the BS degree in computer science and technology from the Guangdong University of Foreign Studies. He is currently working toward the MS degree with the College of Cyber Security, Jinan University. His research interests include zero knowledge proof and blockchain applications.