Enhancing Security in Parallel Federated Learning With Sharded Blockchain for Internet of Vehicles

Lei Tian, Feilong Lin, *Member, IEEE*, Jiahao Gan, Jiahao Qi, *Graduate Student Member, IEEE*, Riheng Jia, *Member, IEEE*, Zhonglong Zheng, *Member, IEEE*, Minglu Li, *Fellow, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—The Internet of Vehicles (IoV) has experienced rapid growth, generating extensive data that can be leveraged for intelligentapplications. Federated learning provides a privacy-preserving approach to utilize this data but faces challenges in performing multi-task learning and ensuring security. To address these issues, we propose a secure Federated Learning framework with Sharded Blockchain (FLSB), where the blockchain design aligns with FL requirements to enhance security, robustness, and scalability. FLSB partitions the blockchain into sharded chains to support parallel FL tasks, while a main chain governs task management and security enforcement. To prevent malicious behaviors such as creating fake identities or disrupting consensus, we develop the Proof of List Update (PoLU) consensus on the main chain. For secure FL, we design a committee-based adaptive weighted aggregation algorithm, whcih dynamically adjusts the model weights during aggregation to defend against poisoning attacks. In addition, we introduce the Proof of Task Participation (PoTP) consensus on sharded chains to ensure trustworthy model aggregation and defend against adversarial manipulations. Experimental results on public datasets demonstrate that FLSB effectively balances FL task performance, security, and privacy, making it well-suited for IoV environments.**

***Index Terms*—Internet of vehicles (IoV), parallel federated learning, consensus mechanism, sharded blockchain, security.**

I. Introduction

IN RECENT years, the Internet of Vehicles (IoV) paradigm has advanced rapidly, establishing seamless connectivity among vehicles, infrastructure, individuals, and the Internet. By interlinking intelligent vehicles with other relevant entities, IoV has demonstrated significant potential in areas such as traffic flow prediction, vehicle route optimization, and accident detection [1], [2]. This rapid development has been accompanied by an exponential increase in the generation of vast amounts of data, including sensor readings, location information, and environmental monitoring data. Harnessing these massive datasets through machine learning-based analysis can yield high-quality models, providing critical support for a wide range of intelligent IoV applications.

To utilize these valuable data distributed among vehicles, a direct approach is to collect and centrally store them on a centralized server for model training. However, this approach can lead to severe privacy issues, as vehicle data may contain sensitive user information. To address this contradiction of leveraging data through machine learning, Federated Learning (FL) [3] has recently emerged as a highly attractive paradigm. The core feature of FL is that model training primarily occurs on the local devices of data owners. Compared to traditional machine learning methods, FL constructs a global model through distributed parameter exchange, thus avoiding direct access to or exposure of raw data.

Despite its promising prospects, applying FL in IoV environments faces two challenges. The first challenge is the management and execution of multi-task FL. The massive collected data needs to be decomposed into multiple tasks for training, as training a single large model to support all applications is impractical [4]. For instance, urban traffic management systems require distributed learning tasks for different types of applications, such as traffic flow prediction, vehicle route optimization, and accident detection. Existing studies primarily adopt a continual learning approach, where new data or tasks are incrementally added to a single model to enhance its capabilities [4], [5], [6]. However, this training method is highly susceptible to catastrophic forgetting, where the model learns new knowledge at the cost of forgetting previously acquired information. Consequently, it is essential to train distinct models for different tasks while ensuring a unified management framework for task training and allocation coordination.

The second challenge involves the various security issues present in IoV environments [7], [8]. During the transmission of vehicle models, malicious vehicles may infer private information from the models, such as vehicle locations and driving trajectories. Additionally, malicious vehicles may conduct data poisoning or model poisoning during training, compromising

Received 26 December 2024; revised 8 April 2025 and 3 June 2025; accepted 22 July 2025. Date of publication 29 July 2025; date of current version 19 January 2026. This work was supported by the National Science Foundation of China under Grant 62273310, Grant 62320106006, Grant 62272419 and Grant 62272417. The review of this article was coordinated by Prof. Yi Qian. *(Corresponding author: Feilong Lin.)*

Lei Tian, Feilong Lin, Jiahao Gan, Riheng Jia, Zhonglong Zheng, and Minglu Li are with the Zhejiang Key Laboratory of Intelligent Education Technology and Application, Zhejiang Normal University, Jinhua 321004, China, and also with the College of Computer Science and Technology, Zhejiang Normal University, Jinhua 321004, China (e-mail: soyelt@zjnu.edu.cn; bruce_lin@zjnu.edu.cn; wise_coach@zjnu.edu.cn; rihengjia@zjnu.edu.cn; zhonglong@zjnu.edu.cn; mlli@zjnu.edu.cn).

Jiahao Qi is with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: qjh2023@sjtu.edu.cn).

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

Digital Object Identifier 10.1109/TVT.2025.3593493

the accuracy of the global model. At the same time, the central server cannot be assumed to be fully trustworthy, as it may also infer private information from models or even manipulate the aggregation process to generate a poisoned global model. Existing studies often focus on addressing specific security threats in IoV, such as inference attacks or poisoning attacks, while assuming an honest server [9], [10], [11], [12]. However, few approaches consider multiple attack types simultaneously. Therefore, achieving secure training under the premise of mutual distrust among all parties remains a challenge.

To address the first challenge, we propose a sharded blockchain architecture consisting of a main chain and multiple sharded chains to manage and execute different FL tasks in parallel. The main chain serves task publishers such as urban traffic management systems and is responsible for global task allocation, cross-region/multi-task coordination, and network-wide task progress monitoring. Specifically, we disign a dedicated main chain consensus mechanism, Proof of List Update (PoLU), which enables participant allocation while defending against malicious attacks. Each sharded chain corresponds to an independent FL task (e.g., traffic flow prediction or traffic sign recognition) and focuses on managing collaborative learning among local vehicles. Data sharing and model updates are facilitated through On-Board Units (OBUs) and Road-Side Units (RSUs), ensuring efficient and secure communication.

For the second challenge, we integrate Differential Privacy (DP) techniques [13] to introduce controlled noise into the model, mitigating inference attacks. Additionally, we design a committee mechanism to validate local training models, preventing data poisoning and model poisoning attacks. Furthermore, we propose an adaptive model weight adjustment algorithm to counteract collusion among participants attempting to manipulate the learning outcome. To ensure the integrity of the aggregation process and prevent malicious server behavior, we introduce a sharded chain consensus mechanism, Proof of Task Participation (PoTP), which enables consensus on the authenticity of the aggregation process.

Our main contributions are summarized as follows:

- We propose a secure Federated Learning framework with Sharded Blockchain (FLSB) for IoV that leverages sharded blockchain technology to address the challenges of executing parallel FL tasks while enhancing security in FL environments.
- We design a Proof of List Update (PoLU) consensus mechanism for the main chain to manage FL tasks and allocate participants. PoLU ensures the fairness and reliability of task allocation while resisting Sybil attacks and other malicious attacks, supporting the secure execution of multiple parallel tasks.
- In the FL training process, we design a committee-based adaptive weighted aggregation algorithm that dynamically adjusts the model weights during aggregation to defend against poisoning attacks. Additionally, we develop a customized Proof of Task Participation (PoTP) consensus mechanism tailored for FL training. PoTP effectively mitigates collusion attacks while maintaining trustless model aggregation.
- Experimental results demonstrate that FLSB exhibits high parallel processing capabilities in executing FL tasks and provides a transparent and verifiable training process while protecting model privacy. Moreover, the framework effectively addresses various types of poisoning attacks.

The remaining parts of this paper are as follows: Section II reviews the related work. Section III elaborates on the system framework of FLSB. Section IV presents the privacy preservation mechanism for FLSB. Section V shows the experimental evaluations and results. Finally, the paper is concluded in Section VI.

II. Related Works

This section reviews the literature on the integration of blockchain and FL in IoV and the privacy preservation of FL, which are closely related to this work.

A. Federated Learning Based on Blockchain

In recent years, the integration of blockchain and FL has gained significant attention [14], [15], [16]. An et al. [17] proposed a blockchain-based FL reputation evaluation mechanism that prevents malicious participants from compromising data privacy or uploading low-quality data by incorporating DP noise and reputation scoring. Ali et al. [18] combined FL with a blockchain-based distributed database to enhance data privacy and security in IoV. Houda et al. [19] leveraged FL and blockchain to defend against evolving attacks while ensuring the reliability of FL in Intelligent Transportation Systems (ITS). Meese et al. [20] proposed a blockchain-enhanced FL framework for online traffic prediction, enabling dynamic urban traffic management in ITS.

To address scalability and interactivity in FL, researchers have explored sharded blockchain solutions. Madill et al. [21] introduced ScaleSFL, which enhances FL scalability by decoupling off-chain FL components while ensuring model verification and interoperability. Lin et al. [22] developed a parallel sharded blockchain FL framework, where training occurs within shards, and model updates from multiple shards are aggregated on-chain for global optimization.

Existing FL-based blockchain research primarily focuses on single-task learning, making it ineffective in multi-task scenarios, which results in inefficient resource allocation and model management. Additionally, these approaches lack a consensus mechanism specifically coupled with FL to defend against various attacks, including Sybil attacks and collusion attacks.

B. Privacy Security for Federated Learning

Privacy security is a critical challenge in FL. Various frameworks have been proposed to enhance privacy and robustness against adversarial threats. Liu et al. [9] introduced PEFL, leveraging Homomorphic Encryption (HE) and adaptive aggregation to resist model poisoning. Ma et al. [10] designed DPBFL, integrating Local DP (LDP) with an intermediate shuffler and a Byzantine-robust random aggregation algorithm for enhanced security. Raja et al. [28] enhanced privacy protection

and communication efficiency in FL by enabling multiple UAVs to process user data locally. Tong et al. [29] integrated FL with UAVs to improve communication efficiency and resistance to eavesdropping, achieving a balance between privacy security and training performance.

To further strengthen privacy protection, Li et al. [30] developed chain-PPFL, a Secure Multi-party Computation (SMC)-based framework with shielding and chain communication mechanisms. Yang et al. [23] proposed a Blockchain-based FL architecture (B-FL), which defends against malicious devices by designing a secure model aggregation algorithm, and deploys a practical Byzantine fault-tolerant consensus protocol among edge servers to prevent model tampering by malicious servers. Li et al. [24] proposed a decentralized FL framework based on a Committee mechanism using Blockchain (BFLC), and designed a committee consensus protocol to reduce the impact of malicious attacks. Wang et al. [26] integrated blockchain with FL to enhance security by designing a novel block structure and transaction types that resist cheating and Sybil attacks. Moreover, a differential privacy mechanism was employed for miners to prevent privacy leakage from trained models. In the IoV domain, Zhou et al. [11] proposed RoHFL, a hierarchical FL framework using logarithmic normalization for robust model aggregation against malicious vehicles. Similarly, He et al. [12] introduced RSAM, employing a divide-and-conquer strategy to approximate the median of local models, effectively mitigating Byzantine attacks.

Current approaches mainly address specific security threats, such as Byzantine faults, inference attacks, or model poisoning. Few studies provide a comprehensive security framework that simultaneously considers multiple attack vectors during model transmission, including untrusted servers, malicious participants, and inference threats.

C. Machine Learning-Based Methodologies to Counter FL-Based Attacks

Machine Learning (ML)-based defenses against attacks on FL are typically implemented at the aggregation server [31], [32]. At the data level, statistical-based aggregation methods, such as Krum [31], which relies on Euclidean distance, and GeoMed [33], based on geometric median, enable robust aggregation by filtering out anomalous updates. At the model level, anomaly detection techniques involve distributing sub-models to data owners for validation. Zhao et al. [34] introduced client cross-validation, where updates are evaluated using other clients' local data, allowing the server to adjust weights accordingly.

Additionally, mapping disruptions between feature space and target labels have been explored to detect poisoned models. Wang et al. [35] employed input filtering, neuron pruning, and unlearning techniques to identify backdoor behaviors by analyzing subtle input perturbations that trigger continuous model output changes. However, these methods are limited to defending against specific attack types and are typically effective only when the proportion of attackers remains below 30%.

We comprehensively consider the management and execution of multi-task FL while addressing privacy and security issues. A visual comparison between FLSB and state-of-the-art blockchain-based FL methods is presented in Table I.

III. Parallel Federated Learning Framework With Sharded Blockchain for IoV

This section presents the FLSB system framework, as shown in Fig. 1. First, we introduce the motivation behind this work. Then, we define the threat model. Next, we provide an overview of the FLSB framework for IoV. Following this, we detail the design and consensus mechanism of the main chain. Finally, we describe the block structure of the main chain.

A. Motivation

We outline the limitations of existing FL security mechanisms in multi-tasking environments, the necessity of sharded blockchain over alternative security frameworks, and real-world security vulnerabilities that motivate this work.

1) Limitations of Existing FL Security Mechanisms in Multi-Tasking Environments: Most FL security mechanisms are designed for single-task learning, facing challenges in multi-task, multi-model scenarios. Existing frameworks assume a unified global model, leading to inefficient resource utilization and increased vulnerability due to task competition [4], [5], [6]. Additionally, they focus on isolated threats like inference or poisoning attacks while assuming an honest central server, which is unrealistic in decentralized IoV settings [11], [12], [30].

2) Necessity of Sharded Blockchain Over Alternative Security Frameworks: Alternative security approaches like Trusted Execution Environments (TEE) and HE have limitations. TEE ensures secure training but struggles with scalability and side-channel attacks [27]. HE enables encrypted computation but is computationally expensive for IoV. Traditional blockchains enhance trust but cannot efficiently manage multiple tasks.

3) Real-World Security Vulnerabilities and Their Impact on FL Model Performance: Security threats in FL highlight the need for robust protection. Poisoning attacks in autonomous vehicles can cause traffic sign misclassification, leading to safety risks [36]. Sybil attacks in blockchain-based FL allow adversaries to manipulate consensus and disrupt aggregation [37]. Inference attacks on model updates can leak location data, exposing vehicle trajectories [38].

Our approach integrates sharded blockchain with federated learning, decomposing multi-task learning into independent models managed through a main-chain consensus mechanism. By employing a hierarchical consensus mechanism, we enhance task management, trust verification, and security threat mitigation. This secure framework ensures reliable model transmission, aggregation, and scalable execution, strengthening the security and trustworthiness of intelligent IoV applications.

B. Threat Model

1) Sybil Attack: In a multi-task FL environment, task publishers may overwhelm the system by creating multiple fake identities to submit task requests or by submitting multiple task requests simultaneously.

TABLE I
COMPARISON WITH STATE-OF-THE-ART BLOCKCHAIN-BASED FL METHODS

Scheme	Privacy Protection	Secure Aggregation	Percentage of Attackers	Malicious Server Behavior	Number of Tasks	Blockchain Architecture
Meese et al. [20]	×	×	×	×	Single Task	Single Blockchain
PLC [18]	×	✓	×	×	Single Task	Single Blockchain
Houda et al. [19]	Encryption	×	×	×	Single Task	Single Blockchain
B-FL [23]	×	✓	< 30%	✓	Single Task	Single Blockchain
BFLC [24]	×	✓	< 50%	✓	Single Task	Single Blockchain
Lu et al. [16]	DP	✓	< 30%	✓	Single Task	Single Blockchain
FREB [17]	DP	✓	< 30%	×	Single Task	Single Blockchain
Biscotti [25]	DP	✓	< 30%	✓	Single Task	Single Blockchain
PF-PoFL [26]	DP	×	×	×	Single Task	Single Blockchain
Kalapaaking et al. [27]	TEE	×	×	✓	Single Task	Single Blockchain
ScaleSFL [21]	×	✓	×	✓	Single Task	Sharded Blockchain
Lin et al. [22]	×	✓	×	×	Single Task	Sharded Blockchain
FLSB	DP	✓	< 50%	✓	Multi Task	Sharded Blockchain

Note: Malicious Server Behavior: Whether the issue of a dishonest or adversarial server is addressed. The notation Secure Aggregation: ✓, Percentage of Attackers: × indicates that while the scheme is designed to defend against malicious models to some extent, it lacks specific implementation, analysis, or experimental results regarding the impact of different percentages of attackers.

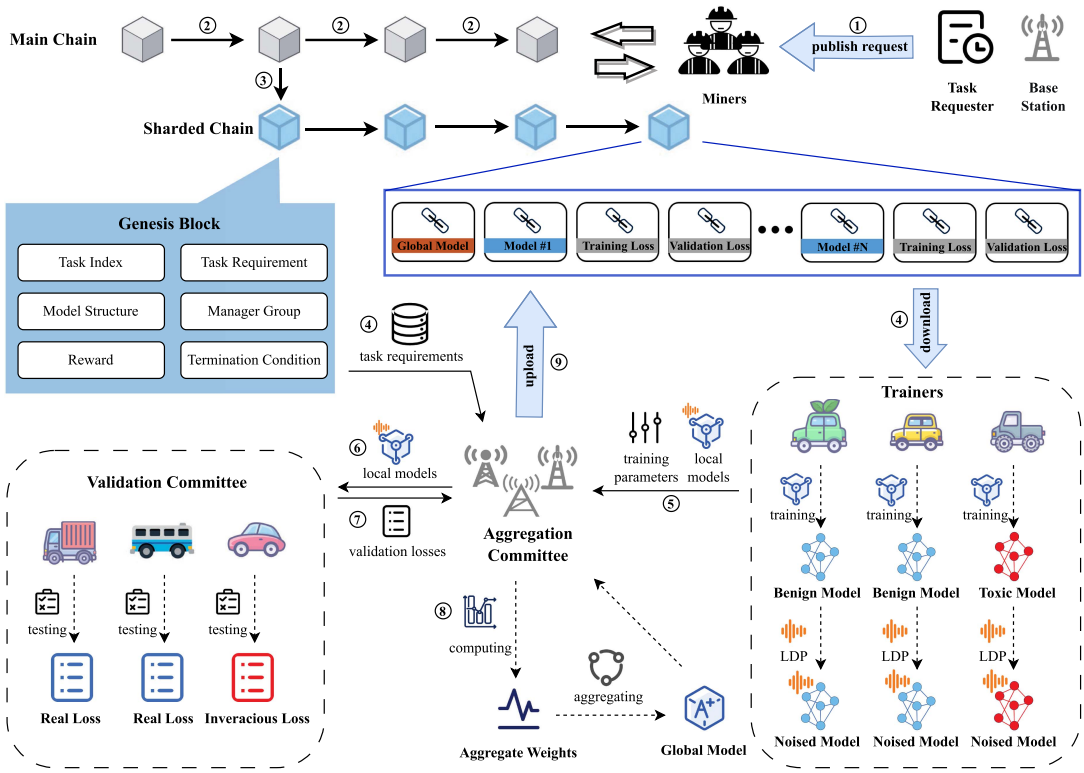

Fig. 1. The overall framework of FLSB for IoV.

2) Inference Attack: In the FL environment, the model weights may contain enough information to be a potential target for inference attacks. Specifically, an attacker might analyze the model weights to infer whether a particular data point was involved in the training of the model, or to identify certain sensitive properties in the training data set.

3) Poisoning Attack: During the weight aggregation phase, there is a risk of poisoning attacks carried out by malicious participants. This paper assumes that the goal of an attacker is to manipulate or destroy the learned global model. Therefore, the attack means are divided into targeted attacks [39], [40] and untargeted attacks [41]. Given the variety of attack possibilities in learning scenarios, we assume that the attacker knows the code on the compromised working device, the local training dataset, the local model, and the aggregation rules in various scenarios.

4) Collusion Attack: Considering the possibility of multiple malicious participants launching a collusive attack, in this attack mode, two or more malicious participants may cooperate privately in an attempt to evade the system's security mechanisms or gain unauthorized information. To ensure the practicality and

availability of the model, this paper makes an assumption about the upper limit on the number of malicious participants, i.e., their number should not exceed 50% of all participants. In addition, the proposed FLSB does not have the knowledge of the exact percentage of malicious clients in the FL system, while some other typical solutions need this information to select benign models, e.g., Biscotti [25].

C. FLSB Architecture Design and Process Analysis

To address the concurrent demands of multiple tasks in IoV, FLSB employs a sharded blockchain architecture consisting of a Main Chain (MC) and multiple Sharded Chains (SCs). As illustrated in Fig. 1, in an IoV scenario, (1) task requesters, such as urban traffic management centers, publish FL task requests on the network. The urban traffic management centers specify the task parameters, including data requirements, model structures, and budget constraints. (2) Miner nodes, represented by cloud servers, collect these task requests along with the information of available base stations. This information is then packaged into a block using the MC consensus mechanism, which will be detailed in Section III-D. (3) Upon receiving a new FL task, cloud servers configure an appropriate set of base stations for execution. Once the longest chain in the MC is confirmed, the genesis block of the corresponding SC is created. The base stations assigned to the SC act as an aggregation committee. (4) Before training begins, base stations extract task-related information from the SC's genesis block, including reward details and task termination conditions. (5) For example, in a traffic flow prediction task, the urban traffic management center issues a request for real-time traffic modeling. Vehicles equipped with OBUs collect local traffic data (e.g., speed, vehicle count, congestion level) and download the model required for the current training round from the SC. These trained local models are then periodically transmitted to nearby base stations. (6) Upon receiving local models, base stations forward them to a verification committee. The setup of this verification committee will be detailed in Section IV-B. (7) The verification committee evaluates the received models and returns the validation loss results to the base stations. (8) Base stations aggregate the verified local models to generate the global model. (9) The aggregated global model is packaged and uploaded to the SC for the next training round.

To ensure smooth operation, different participants require specific hardware and network capabilities: Vehicles (Participants) should be equipped with OBUs featuring computing capabilities (e.g., embedded GPUs) and stable wireless connectivity (5G/V2X). Base Stations (Aggregation Nodes) need edge computing servers with multi-core processors, moderate GPU resources, and high-bandwidth internet access. Cloud Servers (Miners) must have high-performance computing clusters to manage MC operations and task scheduling.

D. MC List Update Proof

For MC, we propose a consensus algorithm based on list updates called Proof of List Update (PoLU). PoLU is implemented by Algorithm 1, and described as follows.

Algorithm 1: Proof of Lists Update Consensus.

Input: Tasks set $\mathcal{A}_{on,m}^{t-1}$, base stations set $\mathcal{B}_{on,m}^{t-1}$.
Output: $\mathcal{A}_m^t$, $\mathcal{B}_m^t$, $nonce$.
Initialization: $\mathcal{A}_{new,m}^t = \{\}$, $\mathcal{B}_{new,m}^t = \{\}$, $nonce = 0$.

Cloud server m collects new task requests and packs them into $\mathcal{A}_{new,m}^t$.
for Task $a_i^{t-1} \in \mathcal{A}_{on,m}^{t-1}$ **do**
  if $a_i^{t-1}.Requirement$ is satisfied **then**
    $\mathcal{A}_{on,m}^{t-1} \leftarrow \mathcal{A}_{on,m}^{t-1} - a_i^{t-1}$
  end if
end for
Updates $\mathcal{A}_{on,m}^t \leftarrow \mathcal{A}_{on,m}^{t-1}$.
Cloud server m creates the new task list as follows:
  $\mathcal{A}_m^t = \mathcal{A}_{new,m}^t \cup \mathcal{A}_{on,m}^t$
Cloud server m collects new base stations and packs them into $\mathcal{B}_{new,m}^t$.
for Base station $b_i^{t-1} \in \mathcal{B}_{on,m}^{t-1}$ **do**
  if Base station b_i^{t-1} is malfunctioning or malicious **then**
    $\mathcal{B}_{on,m}^{t-1} \leftarrow \mathcal{B}_{on,m}^{t-1} - b_i^{t-1}$
  end if
end for
Updates $\mathcal{B}_{on,m}^t \leftarrow \mathcal{B}_{on,m}^{t-1}$.
Cloud server m creates the new base station list as follows:
  $\mathcal{B}_m^t = \mathcal{B}_{new,m}^t \cup \mathcal{B}_{on,m}^t$
$L = |\mathcal{A}_{on,m}^{t-1}| - |\mathcal{A}_{on,m}^t| + |\mathcal{A}_{new,m}^t| + |\mathcal{B}_{on,m}^{t-1}| - |\mathcal{B}_{on,m}^t| + |\mathcal{B}_{new,m}^t|$
for New block is added to blockchain **do**
  $h_{current} = hash(< nonce, data_{current}, h_{prev} >)$
  if $h_{current} \leq \lfloor \frac{L}{\xi} \rfloor \cdot h_{default}$ **then**
    Mining is successful and makes a new block.
    Break.
  end if
end for
Cloud server m broadcasts the new block to the network and publishes it to the MC.
Return $\mathcal{A}_m^t$, $\mathcal{B}_m^t$, $nonce$.

Suppose that there are M cloud servers in the network in the i^{th} block cycle, denoted by $\mathcal{M} = \{1, 2, \ldots M\}$. The urban traffic management center posts task requests in the network along with the rewards to be distributed upon task completion. The distribution of rewards is not the focus of this paper. Define a task set $\mathcal{A}^t = \{a_1^t, a_2^t, \ldots a_N^t\}$ of size N, where a_i^t represents the i^{th} FL task at mining round t. Similarly, the base stations participating in task a_i^t form a set $\mathcal{B}^t = \{b_1^t, b_2^t, \ldots, b_K^t\}$ of size K. At the i^{th} round, cloud server m performs consensus in the following ways:

1) *Consensus preparation:* In order to ensure that the mining process is based on the latest and consistent network status information, cloud server m first acquires the already existing task set $\mathcal{A}_{on,m}^{t-1}$ and base station set $\mathcal{B}_{on,m}^{t-1}$.
2) *Tasks list update:* Cloud server m updates the tasks list by $\mathcal{A}_m^t = \mathcal{A}_{new,m}^t \cup \mathcal{A}_{on,m}^t$, where $\mathcal{A}_{new,m}^t$ represents the newly collected tasks set and $\mathcal{A}_{on,m}^t$ denotes the set of

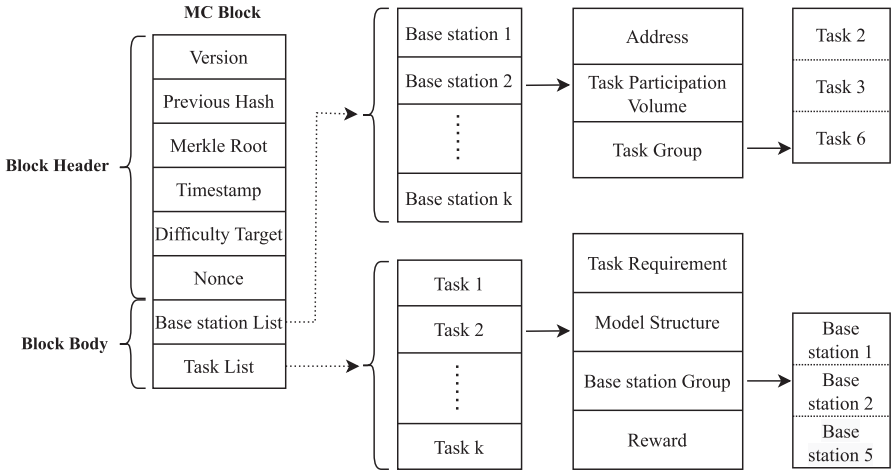

Fig. 2. MC block structure.

ongoing tasks. Meantime, tasks that meet the completion criteria are removed from the task list.

3) *Base stations list update:* The cloud servers collect idle base stations capable of participating in FL tasks and update the list $\mathcal{B}_m^t = \mathcal{B}_{new,m}^t \cup \mathcal{B}_{on,m}^t$, where $\mathcal{B}_{new,m}^t$ represents the newly collected base station set, and $\mathcal{B}_{on,m}^t$ represents the set of currently occupied base stations. Similarly, malfunctioning or idle base stations are removed from the base station set. During the training process, base stations that engage in dishonest aggregation or maliciously disrupt the consensus process will also be removed from the base station set. The detailed detection process will be described in Section IV-C.
4) *Mining Process:* Cloud servers compete to compute the hash value of the current block

$$h_{current} = hash(< nonce, data_{current}, h_{prev} >), \quad (1)$$

where $nonce$ is a random number, $data_{current}$ is the hashing output of the current block data, and h_{prev} is the hash value of the previous block. Set $h_{default}$ as the default difficulty value for mining. Define $L = |\mathcal{A}_{on,m}^{t-1}| - |\mathcal{A}_{on,m}^t| + |\mathcal{A}_{new,m}^t| + |\mathcal{B}_{on,m}^{t-1}| - |\mathcal{B}_{on,m}^t| + |\mathcal{B}_{new,m}^t|$ as the list update quantity, where $|\cdot|$ is the set size. Then, for mining control, the block is successfully generated once a cloud server obtains the hash value meeting

$$h_{current} \leq \left\lfloor \frac{L}{\xi} \right\rfloor \cdot h_{default}, \quad (2)$$

where ξ is an adjustable normalized parameter, used to represent the baseline relationship between the mining difficulty and the number of transactions.
5) *Block publishing:* Cloud server m broadcasts the generated block that meets the conditions to the entire network. When the other cloud servers have received and verified the legitimacy of the block, they add it to their local MC and start mining the next block.

E. MC Block Structure

The details of MC block construction are as follows:

1) *Block header:* The block header primarily contains essential metadata about the block, including the hash value of the current block, the hash value of the previous block, the Merkle root of the transactions, a timestamp, a difficulty target, and an integer value called Nonce. Similar to the Bitcoin architecture, the difficulty target and Nonce work together as part of the PoLU mechanism to validate the legitimacy of the block.
2) *Block body:* The block body is made up of two main parts, one is used to maintain the list of base stations, and the other is used to record all FL tasks published in the network. These two lists are packaged into a transaction and stored in the block body.

As shown in Fig. 2, the base station list maintains a dynamically adjustable set of base stations that records information about each base station, including address, task participation volume and task group. The address is used to identify the base station. The task participation volume is recorded to be used to design the consensus mechanism of the SC. The task group represents the FL tasks that the base station has participated in.

The task list maintains the set of tasks currently in progress, including task requirements, model structure, base station group, and reward. The task list adjusts dynamically as old tasks are completed and new ones appear. The task requirement, reward and model structure are provided by the task requester. Task requirement establishes the conditions under which the task terminates. The model structure informs all vehicles of the details of the model training. The base station group consists of individuals currently applying to become the base stations of the task. Reward refers to the compensation earned for completing the training task.

IV. Federal Learning Privacy Protection Mechanism Based on DP

SC is a blockchain structure designed for FL, implementing an FL framework with DP. This section introduces a committee-based FL training process and proposes a model aggregation algorithm with adaptive weight adjustment to enhance security.

A. DP Introduction

DP [13] introduces noise into machine learning algorithms to ensure that the generated model satisfies ϵ-DP, thereby preventing the inference of individual data records from the model. The formal definition is as follows:

Definition 1 (ϵ-DP): For any two adjacent datasets D and D', a randomized mechanism f satisfies ϵ-DP if for all subsets $S \subseteq Range(f)$, the following holds:

$$Pr\left[f(D) \in S\right] \leq e^{\epsilon} \cdot Pr\left[f(D') \in S\right], \tag{3}$$

where $Pr[f(D) \in S]$ represents the probability that $f(D)$ falls within S, and S denotes an arbitrary possible output range of the mechanism f. The privacy budget ϵ controls the privacy-utility trade-off: a smaller ϵ provides stronger privacy protection but reduces accuracy.

A key concept in DP is sensitivity [13], which quantifies the maximum change in the output of a function due to the modification of a single data record.

Definition 2 (Sensitivity): For a d-dimensional function $f : D \to \mathbb{R}^d$, its sensitivity is defined as:

$$\Delta f = \max_{D,D'} \left\| f(D) - f(D') \right\|, \tag{4}$$

where $\|\cdot\|$ denotes the norm. Specifically, l_1-sensitivity and l_2-sensitivity correspond to the l_1-norm and l_2-norm, respectively.

Local DP (LDP) [42] extends DP by allowing clients to randomize data before sharing. The Gaussian mechanism is a common method for ensuring DP, where noise $\theta \sim \mathcal{N}(0, \sigma^2)$ is added to the output. Given a failure probability $\delta \in (0, 1)$, the noise scale σ must satisfy:

$$\sigma \geq \frac{\Delta f \cdot \sqrt{2\ln(1.25/\delta)}}{\epsilon}. \tag{5}$$

This ensures (ϵ, δ)-DP, meaning that the mechanism satisfies DP with an additional probability of δ for exceeding the bound.

B. Federated Learning Scheme Design With DP Protection

As shown in Fig. 1, the FL communication framework comprises an aggregation committee, trainers (vehicles), and a validation committee. The aggregation committee, formed by base stations, collects and aggregates local models. Trainers (vehicles) perform local training, encrypt models, and send them to the aggregation committee. The validation committee, periodically reselected, tests local models and provides feedback to the aggregation committee.

The initial state of the model and the expected number of iterations (or predetermined precision targets) are derived from the genesis block. In iteration round t, trainers download the model required for the current round of training from the SC, and train the model using their local data. Gaussian noise is added to the local model to perturb it. Given the privacy budget ϵ and the failure probability δ, each trainer i locally computes the standard deviation of the Gaussian noise as:

$$\sigma = \frac{\Delta f \cdot \sqrt{2\ln(1.25/\delta)}}{\epsilon}, \tag{6}$$

where Δf represents the sensitivity of its local dataset. Then, the trainer adds noise ε sampled from a Gaussian distribution with standard deviation σ to the model parameters:

$$\tilde{\omega}_i^t \leftarrow \omega_i^t + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2 I). \tag{7}$$

In this scheme, trainers are required to submit additional training parameters, such as the dataset size and the model's loss function value, along with their local models to the corresponding base station. These parameters are used to determine the aggregation weights. The base station records the received local models and training parameters locally and forwards them to other base stations involved in the task. Simultaneously, the models are distributed to the validation committee. The validation committee evaluates the models' performance using their own local data and reports the computed validation loss to the aggregation committee. The aggregation committee consolidates all reported validation losses, calculates the local loss, and records each validator's evaluation results.

At the t^{th} round of the training process, the aggregation committee selects a specific agent to calculate the weighted difference of validation losses for each model:

$$WD_i^t = \sum_{j=1}^{J} \frac{\left|loss_{i,j}^t - loss_i^t\right|}{e^{\left|loss_{i,j}^t - loss_{i,med}^t\right|}}, \tag{8}$$

where J is the number of verification committee members, $loss_i^t$ is the loss of model i uploaded by the trainer (referred to as training loss), $loss_{i,j}^t$ is the verification loss of model i by the verifier j. $loss_{i,med}^t$ is the median of the validation loss of model i, calculated as follows:

$$loss_{i,med}^t \leftarrow median(loss_{i,1}^t, loss_{i,2}^t, \ldots, loss_{i,J}^t). \tag{9}$$

In (8), the differences between validation losses and training losses are considered, and these differences are quantified by weighted average. In particular, a weight coefficient based on the median loss is introduced to mitigate the impact of abnormal loss values on the overall results. After calculating the weighted differences for all models, the validation score D_i^t for model i can be defined as follows:

$$D_i^t = \begin{cases} \lambda_d^t, & |WD_i^t - WD_{med}^t| \leq c_d \cdot MAD_d^t, \\ 0, & |WD_i^t - WD_{med}^t| > c_d \cdot MAD_d^t, \end{cases} \tag{10}$$

where $\lambda_d^t = c_d \cdot MAD_d^t - |WD_i^t - WD_{med}^t|$, c_d is a predefined constant that adjusts the sensitivity of the validation score. MAD_d^t is the median absolute deviation of the weighted difference, which is calculated as follows :

$$MAD_d^t \leftarrow median(\left|WD_1^t - WD_{med}^t\right|, \left|WD_2^t - WD_{med}^t\right|, \ldots, \left|WD_T^t - WD_{med}^t\right|), \tag{11}$$

where T is the number of trainers, the median weighted difference WD_{med}^t is calculated as follows:

$$WD_{med}^t \leftarrow median(WD_1^t, WD_2^t, \ldots, WD_T^t). \tag{12}$$

Similar to the validation score D_i^t, the training score of model i, L_i^t, is a measure used to quantify the model's performance on the training data set. This metric uses a similar framework to D_i^t but differs in that it is based on the model's training loss $loss_i^t$

and the corresponding median absolute deviation MAD_l^t. The calculation of the training score L_i^t can be expressed as:

$$L_i^t = \begin{cases} \lambda_l^t, & |loss_i^t - loss_{med}^t| \le c_l \cdot MAD_l^t, \\ 0, & |loss_i^t - loss_{med}^t| > c_l \cdot MAD_l^t, \end{cases} \tag{13}$$

where $\lambda_l^t = c_l \cdot MAD_l^t - |loss_i^t - loss_{med}^t|$, c_l is a default constant, similar to c_d. The median absolute deviation MAD_l^t is calculated as follows:

$$MAD_l^t \leftarrow median(\left|loss_1^t - loss_{med}^t\right|, \left|loss_2^t - loss_{med}^t\right|, \ldots, \left|loss_T^t - loss_{med}^t\right|), \tag{14}$$

c_l is a default constant, similar to c_d. The median absolute deviation MAD_l^t is calculated as follows:

$$loss_{med}^t \leftarrow median(loss_1^t, loss_2^t, \ldots, loss_T^t). \tag{15}$$

The design of the training score L_i^t also emphasizes the robustness of the model on the training data set and also considers the loss of the model during the training process. Through L_i^t and D_i^t, the framework quantifies the performance of the model in the training and validation phases, resulting in a comprehensive score S_i^t, calculated as follows:

$$S_i^t = L_i^t D_i^t. \tag{16}$$

Taking into account the honesty of both the trainer and the validation committee, the aggregate weight for each model consists of the model score and the dataset size, i.e.,

$$q_i^t = \frac{N_i S_i^t}{\sum_{i=1}^T N_i S_i^t}, \tag{17}$$

where N_i is the dataset size of trainer i. Finally, the global model ω^{t+1} is obtained by weighted aggregation of the local model $\tilde{\omega}_i^t$. The calculation is as follows:

$$\omega^{t+1} \leftarrow \sum_{i=1}^{T} q_i^t \tilde{\omega}_i^t. \tag{18}$$

After obtaining the global model, the agent uploads the global model, local model, and validation results to the blockchain. Trainers download the global model from the blockchain and commence the next round of training.

C. SC Task Participation Proof

We introduce the Proof of Task Participation (PoTP) consensus mechanism for SC, as detailed in Algorithm 2.

The set of trainers capable of participating in the task a in the network is defined as $\mathcal{T} = \{1, 2, \ldots, T\}$, and their local models in the i^{th} round are defined as $\mathcal{W}_a^t = \{\omega_1^t, \omega_2^t, \ldots, \omega_T^t\}$. The base stations (aggregation committee) will perform consensus by:

1) *Consensus preparation:* In the preparation phase, members of the aggregation committee are sorted in descending order based on their historical task participation to obtain the list $\mathcal{H}_a'$. The aggregators in $\mathcal{H}_a'$ are in turn responsible for block generation from highest to lowest, and the aggregator responsible for block generation in this round is called the agent.

Algorithm 2: Proof of Task Participation Consensus.

Input: FL task a, base station group $\mathcal{H}_a$, trainers set $\mathcal{T} = \{1, 2, \ldots, T\}$.
Output: A new block.

Trainers obtain the model through the SC block.
The aggregation committee ranks its members by the task participation volume:
  $\mathcal{H}_a' \leftarrow Sort(\mathcal{H}_a)$
Agent $\leftarrow Top(\mathcal{H}_a')$
Agent selects a subset of trainers as verifiers randomly, defined as $\mathcal{J} = \{1, 2, \ldots, J\}$.
for each trainer$_i$, $i \in \{\mathcal{T} - \mathcal{J}\}$ **do**
  Performs local training.
  Gets local model ω_i^t and training loss $loss_i^t$.
  Adds Gaussian noise to ω_i^t: $\widetilde{\omega}_i^t \leftarrow \omega_i^t + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2 I)$.
  Sends $\widetilde{\omega}_i^t$, $loss_i^t$ and dataset size N_i to the aggregation committee.
end for
Distributes the local model $\widetilde{\omega}_i^t$ to validation committee.
for each verifier$_j$, $j \in \mathcal{J}$ **do**
  Uses private dataset to test $\widetilde{\omega}_i^t$ and obtain the verified loss $loss_{i,j}^t$.
  Uploads $loss_{i,j}^t$ to the agent.
end for
Agent Calculates aggregation weight q_i^t according to (8)–(17).
Agent aggregates global model ω^{t+1} according to (18).
Agent stores ω^{t+1}, $\widetilde{\omega}_i^t$, $loss_i^t$ and $loss_{i,j}^t$ in a new block and adds his signature in the block header.
Agent broadcasts the block to other aggregation committee members.
↻ [Agent waits in this step for signatures of other members until all signatures reach.]
Agent publishes the new block to the network with these signatures.
Append agent to the end of $\mathcal{H}_a'$.
Return the new block.

2) *Local model collection:* The agent collects the encryption model $\tilde{\omega}_i^t$, dataset size N_i and training loss $loss_i^t$ from the trainers, and the local models form the set $\mathcal{W}_a^t$. At the same time, the agent records these training parameters locally.
3) *Weight calculation:* The agent distributes $\mathcal{W}_a^t$ to the validation committee, which evaluates the model using their respective datasets and returns the calculated validation losses to the aggregation committee. After the agent receives all verification losses, the aggregate weight of each model is calculated according to (8)–(17).
4) *Block generation:* The agent computes a new round of global model ω^{t+1} according to (18) and writes it to a new block together with $\mathcal{W}_a^t$. Due to the limitations on block memory, it is impractical to store all local models within a block. Therefore, the hash values of the local models are stored in the block for verification purposes.

5) *Block publishing:* In a blockchain network, when the agent builds a new block, it needs to add his signature to the block header and broadcast it to other aggregation committee members. Each member will conduct an independent audit to verify the legitimacy of the new block. A new block will only be added to the blockchain network if most members agree that it complies with the rules.

When a member receives a new block, it goes through a series of verification procedures to ensure compliance and integrity of the block. This includes: a. Calculation and verification of data consistency; b. The signatures of the agent and other base stations. After completing all verification processes, the member marks the new block as compliant.

If the majority of members fail to verify data consistency, it indicates that the agent has computational errors or exhibits malicious behavior. In this case, the agent is marked as a malicious base station and is prohibited from participating in subsequent consensus processes. Additionally, if a base station repeatedly fails to respond during multiple rounds of consensus or consistently votes against the final consensus result, it will be marked as a problematic base station and similarly banned from future participation. Base stations exhibiting either of these behaviors will be removed from the base station set by the cloud server during the PoLU consensus process.

D. SC Block Design

As shown in Fig. 1, SC is built by linking its genesis block to MC. When a block containing task a exists in the MC and satisfies the longest chain principle, the base station of the task will create its genesis block. The task index, task requirements, model structure, base station group, reward and termination conditions are packaged into a transaction and stored in the block body. The block header is similar to that of an MC block.

In SC, each block except the genesis block is mainly used to store information about each round of FL training, including a new round of global model ω^{t+1}, the hash values of the local models $\tilde{\omega}_i^t$, training losses and validation losses. In addition, honest trainers and verifiers can verify whether their results have been tampered with by looking at the stored data in the block. It can be verified that the aggregation committee performs honest aggregation according to the following steps:

1) Using the losses in the block, calculate the aggregate weight $\bar{q}_i^t$ for each model according to (8)–(17).
2) Using the local models in the block, calculate the global model $\bar{\omega}^{t+1}$ according to (18).

If the following equation is true, then the polymerization process is honestly calculated:

$$\bar{\omega}^{t+1} = \omega^{t+1}. \tag{19}$$

E. Security Analysis

1) Fork Problem: In the MC, we employ a PoLU consensus mechanism similar to PoW. Consequently, it is possible for multiple cloud servers to successfully mine blocks simultaneously, resulting in forks in the MC and rendering FL training conflicting. To address this, we adopt the longest chain rule. After a block for task a is mined, we wait for six subsequent to ensure the longest chain is established before creating the SC and initiating training for the task, rather than starting immediately. In the PoTP consensus of the SC, base stations take turns acting as the agent to produce blocks, ensuring no forking occurs.

2) Sybil Attack: Whenever a task requester publishes a task, a reward is submitted as compensation for collaborative training. Therefore, if the requester initiates an attack by creating multiple fake identities or submitting multiple task requests, they must also submit the corresponding amount of currency, making such attacks unprofitable.

3) Single Point of Failure and Inference Attack: Base stations take turns as the agent to aggregate and store training data in the blockchain. This design not only enhances the credibility of the system, but also effectively eliminates the risk of single points of failure. To guard against inference attacks, trainers add DP noise for encryption before uploading their local models. Below, we provide a formal analysis demonstrating how adding DP noise to model parameters can defend against inference attacks.

Assumption 1 Given a function $f: D \to \mathbb{R}^d$ with L_2-sensitivity $\Delta f = \max_{D,D'} \|f(D) - f(D')\|_2$, the Gaussian mechanism is defined as:

$$\mathcal{M}(D) = f(D) + \varepsilon, \tag{20}$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ denotes a random variable that follows a multivariate Gaussian distribution with mean zero and covariance $\sigma^2 I$.

Corollary 1 (Resistance to Inference Attacks): Suppose an adversary attempts to determine the presence of a specific record in the dataset based on the observed output of the Gaussian mechanism. The maximum probability that the adversary correctly identifies the presence or absence of the target data point is:

$$Pr[\text{Success}] \leq \frac{1}{2} + \frac{1}{2} e^{-\epsilon^2/(2\ln(1.25/\delta))}, \tag{21}$$

where ϵ and δ are privacy parameters.

Proof: Let D and D' be neighboring datasets differing by one entry. Under the Gaussian mechanism:

$$\begin{aligned} \mathcal{M}(D) &= f(D) + \varepsilon, \\ \mathcal{M}(D') &= f(D') + \varepsilon, \end{aligned} \tag{22}$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$. The adversary's optimal strategy is to apply a likelihood ratio test:

$$\frac{p(x|D)}{p(x|D')} = \exp\left(\frac{\|x - f(D')\|_2^2 - \|x - f(D)\|_2^2}{2\sigma^2}\right). \tag{23}$$

Expanding the quadratic terms and applying the Cauchy-Schwarz inequality yields:

$$\frac{p(x|D)}{p(x|D')} \leq \exp\left(\frac{\Delta f^2}{2\sigma^2} + \frac{\|x - f(D)\|_2 \cdot \Delta f}{\sigma^2}\right). \tag{24}$$

By Chernoff Bound, the probability that $\|x - f(D)\|_2$ deviates significantly is upper-bounded by:

$$\Pr\left(\|x - f(D)\|_2 \geq k\sigma\right) \leq 2e^{-k^2/2}. \tag{25}$$

Setting $k = \epsilon/\sqrt{2\ln(1.25/\delta)}$ leads to (21). This result demonstrates that the Gaussian mechanism effectively limits the adversary's success rate to near-random guessing for sufficiently small ϵ.

4) Collusion Attack: In a collaborative FL environment, collusion attacks pose a major security threat. In this mode of attack, multiple participants may secretly band together to gain access to information they should not have. Specific collusion attack situations may exist as follows:

1. *Trainer-Verifier Collusion Attack:* Assume a malicious trainer uploads a toxic model ω_{mal}, and a colluding verifier manipulates their reported validation loss to reduce the perceived error for the toxic model and increase the loss values for other benign models.
   Analysis: In the presence of a malicious trainer-verifier collusion, when calculating the model's loss difference, defined as (8): $WD_i^t = \sum_{j=1}^{J}(|loss_{i,j}^t - loss_i^t|)/(e^{|loss_{i,j}^t - loss_{i,med}^t|})$. The loss difference of the malicious model WD_{mal}^t will deviate significantly from the benign model's difference WD_{ben}^t and the median difference WD_{med}^t, since most verifiers are honest, the following relationship holds:
   $$WD_{mal}^t \gg WD_{ben}^t \quad , \quad WD_{mal}^t \gg WD_{med}^t. \tag{26}$$
   By (10), the malicious model's loss difference exceeds the threshold, and its validation score D_{mal}^t is set to zero:
   $$D_{mal}^t = 0, \quad \text{if} \quad |WD_{mal}^t - WD_{med}^t| > c_d \cdot MAD_d^t. \tag{27}$$
   Combining (16) and (17): $q_{mal}^t = (N_{mal}L_{mal}^t D_{mal}^t)/(\sum_{i=1}^{T} N_i L_i^t D_i^t)$. Since $D_{mal}^t = 0$, the aggregation weight $q_{mal}^t = 0$. This ensures that the malicious model is entirely excluded from the global model aggregation.
2. *Trainer-Agent Collusion Attack:* Assume a malicious trainer submits a toxic model and colludes with the agent to increase its aggregate weight by modifying the weighted difference or final aggregate weight.
   Analysis: Since verification losses are recorded on the blockchain, they are immutable and traceable. The aggregation process is expressed by (8)–(18): $\omega^{t+1} \leftarrow \sum_{i=1}^{T}(N_i L_i^t D_i^t / \sum_{i=1}^{T} N_i L_i^t D_i^t)\tilde{\omega}_i^t$.
   The calculated training loss $loss_i^t$ and verification loss $loss_{i,j}^t$ are verifiable against the immutable records on the blockchain. Any manipulation by the agent is detectable when recalculating the loss differences from recorded data. Thus, discrepancies between the chain-stored values and agent-calculated values signal malicious behavior.
3. *Verifier-Agent Collusion Attack:* Assume a verifier colludes with the agent by selectively modifying validation losses to either amplify or suppress the weighted difference.
   Analysis: Suppose the verifier reports a manipulated loss value $loss_{i,man}^t$ near the trainer's original training loss $loss_i^t$ to reduce the weighted difference. By (8), this change is detected since the majority of honest verifiers maintain correct loss values, keeping the median $loss_{i,med}^t$ stable, the following relationship holds:
   $$WD_{man}^t \gg WD_{ben}^t \quad , \quad WD_{man}^t \gg WD_{med}^t. \tag{28}$$
   Consequently, the manipulated model is penalized by (10), the manipulated model's loss difference exceeds the threshold, and its validation score D_{man}^t is set to zero: $D_{man}^t = 0, \quad \text{if} \quad |WD_{man}^t - WD_{med}^t| > c_d \cdot MAD_d^t$. Furthermore, if the agent manipulates verification losses to influence the final model aggregation, discrepancies in the blockchain records will expose the inconsistencies.
4. *Trainer-Verifier-Agent Collusion Attack:* Assume a malicious trainer, verifier, and agent collaborate to manipulate the system by increasing the toxic model's weight while reducing its observed loss.
   Analysis: Even if the malicious verifier reduces the validation loss, (8) ensures that honest verifiers amplify the loss difference, i.e., $WD_{mal}^t \gg WD_{ben}^t \quad , \quad WD_{mal}^t \gg WD_{med}^t$. Additionally, DP noise (added as $\mathcal{N}(0, \sigma_i^2)$) ensures that any compromised local model cannot be effectively exploited in inference attacks, i.e., maximum probability of attack success: $Pr[\text{Success}] \leq \frac{1}{2} + \frac{1}{2}e^{-\epsilon^2/(2\ln(1.25/\delta))}$.

Suppose the malicious agent attempts to alter the final model weights: $\omega^{t+1} = \sum_{i=1}^{T} q_i^t \tilde{\omega}_i^t$. Since q_i^t is determined from immutable blockchain-stored values, tampering with model weights is detectable by verifying ω^{t+1} against recalculated values from trusted records.

V. Experimental Evaluation

This paper implements FLSB based on Go 1.20 and Python 3.10.11, where Go is used to build shared blockchains, and PyTorch 2.0.1 is used to train the model and generate noise. The sharded blockchain nodes use Go's native RPC framework to communicate with each other and interact with the Python-based Tornado HTTP server through Go's native HTTP client, handing the model over to the training server for training, validation, and aggregation. The experiment was run on a computer with an Intel i5-13400 CPU (2.50 GHz, 10 cores), 32 GB DRAM, and NVIDIA RTX 4070 Ti GPU.

A. Experimental Parameter Settings

1) Blockchain Settings: All nodes run in a local server environment. In the PoLU consensus mechanism of MC, set the default hash difficulty $h_{default}$ to 2^{60} and the parameter ξ to 1. The transaction capacity of each block is capped at a maximum of three transactions. Five cloud server nodes on the MC are responsible for the maintenance of the chain. Five base stations are appointed to form an aggregation committee for each FL task on the SC.

2) Federated Learning Settings: The performance of the FLSB scheme is assessed using the handwritten digit dataset MNIST, the image classification dataset CIFAR10, and the traffic sign recognition dataset GTSRB.

- *MNIST:* The MNIST dataset contains 60,000 training samples and 10,000 test samples. Each sample is a 28x28 grayscale image containing digits from 0 to 9. A model

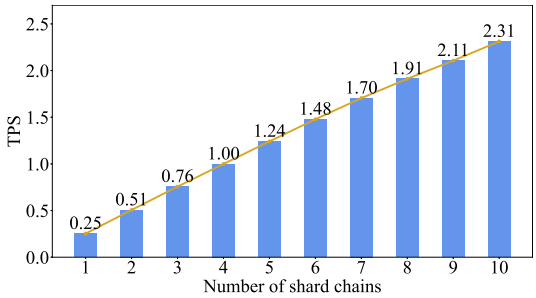

Fig. 3. Number of FL task iteration rounds executed per second for FLSB with different numbers of sharded chains.

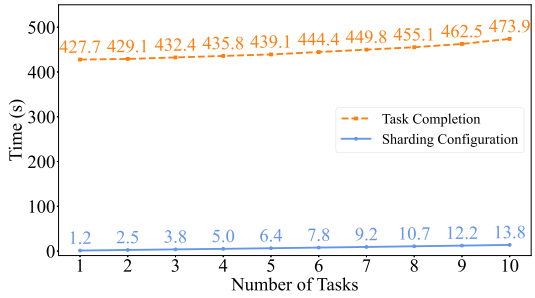

Fig. 4. Task completion time and sharding configuration time under different numbers of FL tasks.

with 2 convolutional layers and 2 fully connected layers is used for training.

- *CIFAR10:* The CIFAR10 dataset includes 50,000 training samples and 10,000 test samples. Each sample is a 32x32 color image categorized into 10 classes. The VGG11 model is used for training.
- *GTSRB:* The GTSRB dataset contains approximately 39,000 training samples and 12,000 test samples. Each sample is a color image with varying resolutions, typically around 30x30 to 50x50 pixels, depicting traffic signs from 43 different classes. The ResNet18 model is used for training.

Considering practical scenarios, we distribute the dataset to 20 trainers using a Dirichlet distribution with a heterogeneity factor of $\alpha = 1.0$, and select 10 trainers to form a validation committee. Each trainer performs SGD updates locally, with a learning rate of 0.01, momentum of 0.5, and a batch size of 64. For the setup of DP noise, the parameters are $(0.25,10^{-5})$-DP for MNIST, $(1,10^{-5})$-DP for CIFAR10, and $(1.5,10^{-5})$-DP for GTSRB.

B. Baseline FL Algorithms and Attacks

We compares the FLSB with the following FL schemes:

1) *FedAvg [3]:* FedAvg calculates the average of all the local models uploaded as the global model.
2) *Median [33]:* Median divides local models into k batches, computes the mean of each batch, and takes the geometric median of these means as the global model.
3) *FL-RAEC [41]:* FL-RAEC uses an autoencoder to filter malicious models after the models are added with DP noise before T^{trust} (set to 40) and evaluates model scores through validator rounds afterward, assigning aggregation weights based on both for global model calculation.
4) *Biscotti [25]:* Biscotti uses a noise committee to add DP noise to training models, a validation committee with the Multi-Krum algorithm to detect poisoned models, and an aggregation committee with Shamir's Secret Sharing for secure global model aggregation. Additionally, the DP noise and models are stored on a blockchain.
5) *RoHFL [11]:* RoHFL uses a logarithmic normalization mechanism to handle scaled gradients from malicious vehicles.
6) *RSAM [12]:* RSAM securely aggregates the global model by calculating the approximate median of local models from vehicles using a divide-and-conquer approach.

We evaluates FLSB against the state-of-the-art attacks:

1) *Poisoning attack based on Generative Adversarial Nets (PGAN) [43]:* Initially, the attacker enhances the accuracy of the global model through standard training. At predetermined intervals (set to 70 rounds for MNIST and 150 rounds for CIFAR-10), the attacker employs Generative Adversarial Networks to craft adversarial samples for training the local model.
2) *Static Optimization (STAT-OPT) [39]:* The attacker constrains the poisoned model w_i' as $w_i' = w_{Re} - \lambda s$, where

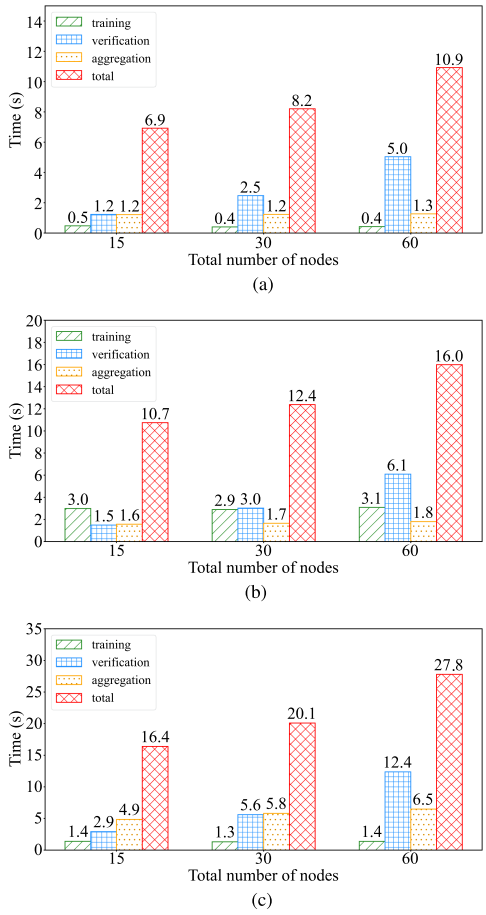

Fig. 5. The time spent in each phase of deploying different numbers of nodes. (a) MNIST. (b) CIFAR10. (c) GTSRB.

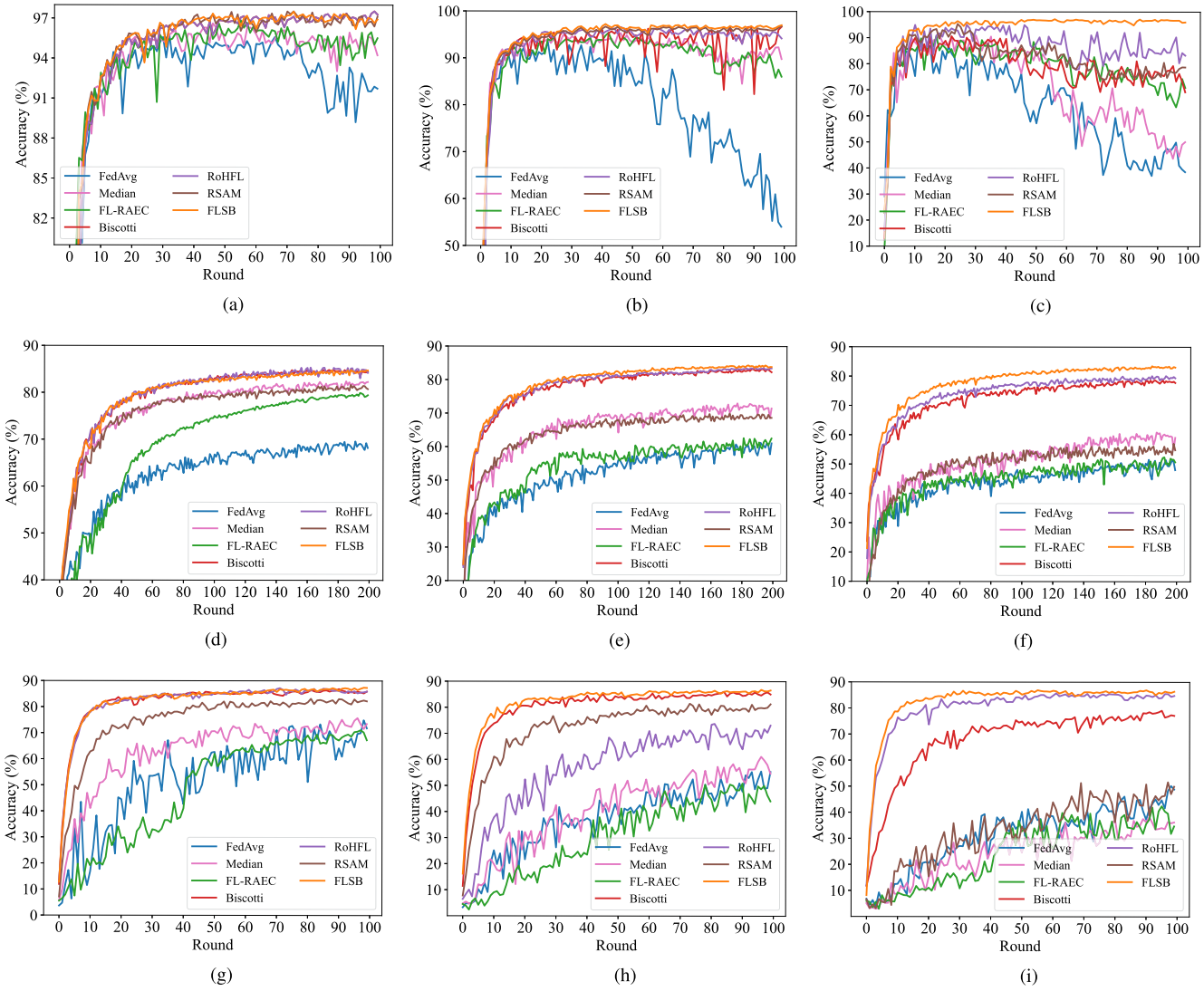

Fig. 6. Model accuracy under different proportions of malicious nodes with standard poisoning attack. (a) 15% malicious nodes on MNIST. (b) 30% malicious nodes on MNIST. (c) 45% malicious nodes on MNIST. (d) 15% malicious nodes on CIFAR10. (e) 30% malicious nodes on CIFAR10. (f) 45% malicious nodes on CIFAR10. (g) 15% malicious nodes on GTSRB. (h) 30% malicious nodes on GTSRB. (i) 45% malicious nodes on GTSRB.

$\lambda > 0$. w_{Re} is the global model received from the server, and s is the column vector of the direction of change for global model parameters. Additionally, $f-1$ (f indicates the maximum number of attackers) other compromised local models are crafted to approximate w_i', with their distance to w_i' not exceeding ϵ.

3) *Edge-case backdoor attacks (Edge-Case) [40]:* The attacker tests the pre-trained model using a set of special and ordinary samples, collecting output vectors from the last layer to fit a Gaussian mixture model. A backdoor is implanted into the model, causing it to output specific labels for certain inputs.

C. Blockchain Performance Analysis

This subsection assesses the system's parallelism and tests the overhead at various stages of the training process.

1) Throughput: Due to hardware resource limitations, each sharded chain is configured with three trainers and one verifier. To evaluate FLSB's capability in handling multiple tasks, we simultaneously deploy 1 to 10 tasks in the network. To minimize the impact of varying task completion times, we use the same dataset, MNIST, for all tasks.

As shown in Fig. 3, when the number of sharded chains in FLSB increases from 1 to 10, the system's TPS, defined as the number of FL task iteration rounds that the FLSB framework can execute per second, reaches 2.31, demonstrating a linear growth trend. Fig. 4 further illustrates that as the number of FL tasks increases, the configuration time of sharded chains grows linearly, indicating the security and stability of the main chain's consensus mechanism in initializing sharded chains. Meanwhile, the task completion time increases only slightly, primarily due to factors such as heterogeneous computing capabilities among nodes and the increased network load resulting from a larger number of total nodes. This also indicates that once

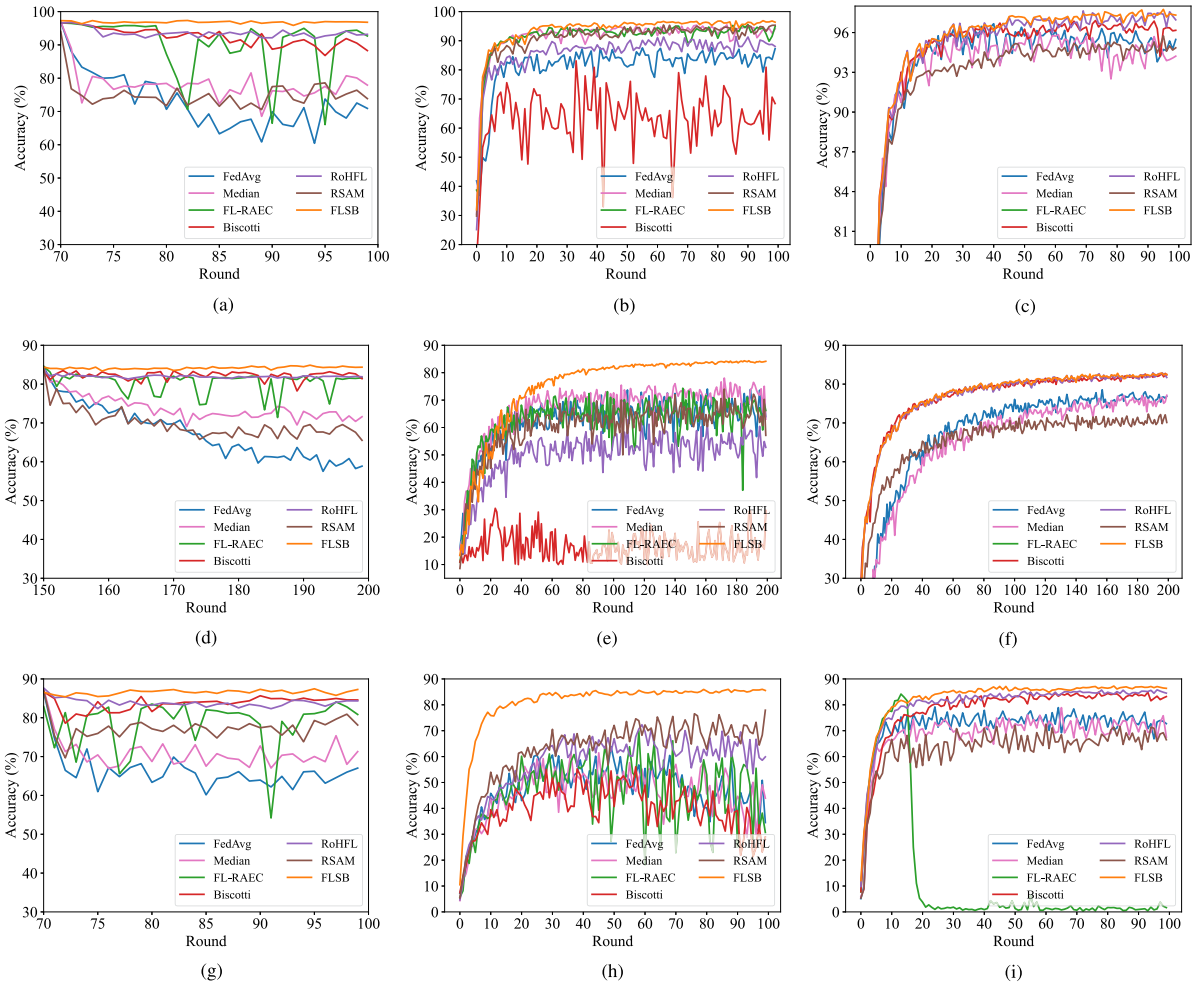

Fig. 7. Model accuracy under the SOTA attack. (a) PGAN attack on MNIST. (b) STAT-OPT attack on MNIST. (c) Edge-Case attack on MNIST. (d) PGAN attack on CIFAR10. (e) STAT-OPT attack on CIFAR10. (f) Edge-Case attack on CIFAR10. (g) PGAN attack on GTSRB. (h) STAT-OPT attack on GTSRB. (i) Edge-Case attack on GTSRB.

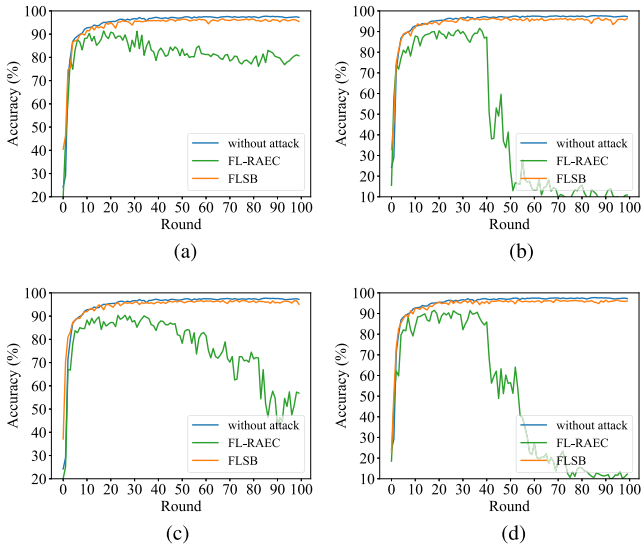

Fig. 8. Model accuracy under different collusive attack scenarios. (a) False loss, honest verification. (b) False loss, false verification. (c) Honest loss, honest verification. (d) Honest loss, false verification.

the sharded chains are established, the system can efficiently execute concurrent FL tasks, aligning with the expected outcomes of this study.

2) SC Performance Overhead: Fig. 5 displays the average execution time per node at various stages under three different deployment scales of node numbers (i.e.,15, 30, and 60 nodes). The experimental results show a slight increase in total overhead as the number of nodes doubles, attributed primarily to additional expenses incurred during the validation phase. The calculation results show that the overhead for nodes to upload training data to the SC and download the global model remains nearly constant, with 4.1 s, 4.8 s, and 7.4 s for the MNIST, CIFAR10, and GTSRB datasets, respectively.

D. Federated Learning Outcomes Analysis

To test the ability of FLSB to resist attacks, we compares FLSB with state-of-the-art (SOTA) robust aggregation methods: FedAvg, Median, FL-RAEC, Biscotti, RoHFL,and RSAM. Both standard and SOTA poisoning attacks are considered in the evaluation. The standard poisoning attacks include the extra

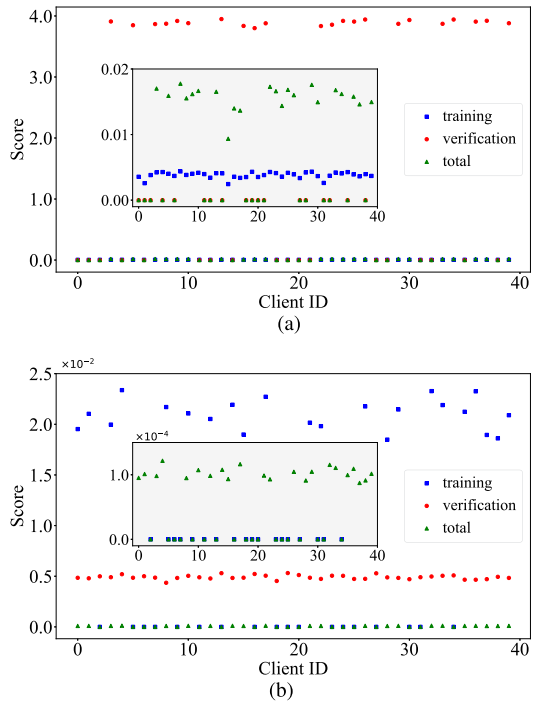

Fig. 9. Model scores under different collusive attack scenarios. (a) False loss, false verification. (b) Honest loss, false verification.

noise attack and random weight attack, which represent common and relatively simple adversarial strategies. In contrast, the SOTA poisoning attacks encompass more advanced and effective methods such as PGAN, STAT-OPT, and Edge-Case, which are designed to bypass traditional defenses and pose greater challenges to FL systems.

1) Standard Poisoning Attack: In this experiment, since RoHFL does not use DP, we employ a similar random weight attack as a substitute, where malicious participants scale the original model weights. In other schemes, malicious participants introduce additional noise into the model to launch attacks. Fig. 6 illustrates the impact of different proportions of malicious nodes on model accuracy across MNIST, CIFAR10, and GTSRB datasets. It can be observed that even when the proportion of malicious nodes reaches 45%, FLSB maintains strong robustness with minimal impact from poisoning attacks. In contrast, other schemes exhibit varying degrees of performance degradation. Notably, on the CIFAR10 dataset, poisoned models show slow convergence rather than a direct accuracy drop. Studies suggest that poisoning initially acts as a form of regularization but ultimately prevents the model from converging to a usable performance.

2) SOTA Attacks: We will further test the robustness of FLSB using SOTA attacks, and the proportion of malicious nodes is set to 20%, as shown in Fig. 7.

In PGAN attack, FLSB demonstrates superior resistance, while FedAvg, Median and RSAM fail, showing rapid accuracy declines, and FL-RAEC suffers from accuracy fluctuations due to autoencoder limitations. Biscotti and RoHFL perform relatively well, resisting most attacks. In STAT-OPT attack, Biscotti exhibits the poorest performance as poisoned models within an ϵ-distance are more likely to be selected. FedAvg, Median, FL-RAEC, RSAM and RoHFL show better accuracy curves but still struggle to withstand the attack, whereas FLSB achieves higher accuracy by detecting attack models based on validation loss. For Edge-Case attack, Biscotti, RoHFL and FLSB display strong robustness due to large norm differences caused by poisoned datasets. Median and RSAM exacerbate the attack process, resulting in accuracy lower than FedAvg. FL-RAEC completely fails to detect Edge-Case models, causing a severe accuracy drop.

3) Collusion Attack: Fig. 8 depicts four scenarios of the "Trainer-Verifier" collusion attack, in which malicious trainers launch extra noise attacks and the proportion of malicious nodes is 45%.

a) As shown in Fig. 8(a), in this scenario, the malicious trainers upload fake low training losses, and the collusion verifiers generate false low validation losses for them, but perform honest validation on the other models. This mode of attack is relatively weak against FL-RAEC and the global model accuracy stabilizes at about 90%. Unlike the scenario in Fig. 8(a), Fig. 8(b) shows that FL-RAEC's accuracy drops dramatically at the 40$^{\text{th}}$ round when the collusion verifiers deliberately amplify the loss values of other models. This is because this behavior disrupts the normal verification process and gives malicious models a disproportionate share of the weight aggregation.

b) In Fig. 8(c), malicious trainers upload true training losses, while collusion verifiers still generate false low validation losses but maintain honest validations of other models. Similar to the attack mode depicted in Fig. 8(a), the accuracy of FL-RAEC still shows a declining trend. Fig. 8(d) further explores a similar but more dangerous attack pattern as Fig. 8(c), where the collusion verifiers amplify other models' losses simultaneously. This is similar to the result in Fig. 8(b), resulting in a sharp decline in the accuracy of FL-RAEC after the 40th iteration.

In contrast, FLSB, by utilizing both training scores and validation scores as metrics, ensures that trainers and validators cannot collude with each other, successfully eliminating malicious models. Thus, it achieves accuracy levels close to the baseline in all four collusive attack scenarios.

E. Ablation Study

1) Model Score: To evaluate the effectiveness of the model score algorithm in FLSB for detecting malicious models, we test the training, validation, and total scores under the "Trainer-Verifier" collusion attack.

In Fig. 9(a), the malicious trainers upload fake low training losses, and the collusion verifiers generate false low validation losses for them. It can be seen that because the malicious trainers upload the fake low training losses, the training scores of all models are roughly the same (the blue squares in the plot inside the plot). During the validation phase, by applying MAD as an anomaly detection mechanism, the validation scores of malicious models are reduced to 0 (in Fig. 9(a), indicated by

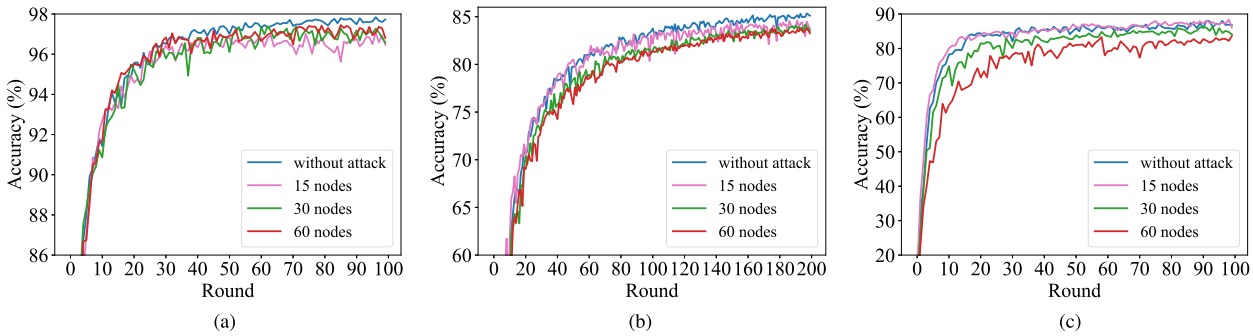

Fig. 10. Robustness of the number of nodes to FLSB. (a) MNIST. (b) CIFAR10. (c) GTSRB.

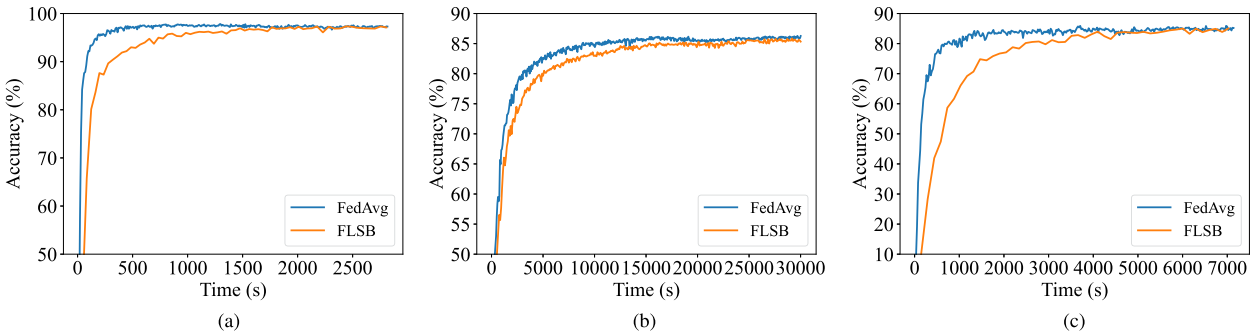

Fig. 11. Comparison of convergence rates between FLSB and FedAvg. (a) MNIST. (b) CIFAR10. (c) GTSRB.

red circles). Correspondingly, the benign models achieve higher total scores (the green triangles in Fig. 9(a)).

In Fig. 9(b), the observations are in stark contrast to those in Fig. 9(a). In this scenario, the malicious trainers choose to upload actual training losses. This behavior results in all models obtain roughly identical validation scores (the red circles in Fig. 9(b)). However, the training loss values of malicious models are significantly higher than those of benign models, indicating lower training scores (indicated by blue squares in Fig. 9(b)). Thus, the benign models obtain higher total scores (the green triangles in Fig. 9(b)).

2) Robustness of the Number of Nodes to FLSB: We measure the system's robustness by varying the number of participants. The proportion of malicious nodes is set to 30%.

As shown in Fig. 10, on MNIST, CIFAR10 and GTSRB datasets, FLSB demonstrates high accuracy performance similar to scenarios without attacks. The experiments indicate that whether changing the number of trainers (e.g., 10, 20, 40) or the size of the validation committee (e.g., 5, 10, 20), FLSB can effectively identify and resist malicious attacks, proving that the number of participants has little impact on FLSB's robustness and anti-attack capabilities.

3) Convergence Speed: This experiment takes into account both model accuracy and communication overhead, comparing FLSB with FedAvg using 30 nodes.

The convergence of both systems is shown in Fig. 11. On the MNIST dataset (Fig. 11(a)), both systems achieve 97% accuracy at 1,767 s, but FedAvg reaches this accuracy earlier at 372 s. Similarly, on CIFAR10 (Fig. 11(b)), both systems achieve 85.8% accuracy at 24,914 s, while FedAvg reaches it earlier at 14,034 s. On GTSRB (Fig. 11(c)), both systems reach 83.9% accuracy at 4,100 s, but FedAvg achieves it first at 2,330 s. Overall, FLSB requires approximately 4.75×, 1.78×, and 1.76× more time on MNIST, CIFAR10, and GTSRB, respectively. These results indicate that while FLSB incurs higher computational costs, it remains cost-effective, especially for deeper model structures.

VI. Conclusion

In conclusion, this paper presents FLSB, a secure Federated Learning framework based on Sharded Blockchain, specifically designed for IoV environments. By leveraging sharded blockchain technology, FLSB effectively addresses the challenges of executing parallel FL tasks while ensuring enhanced security. The framework incorporates a PoLU consensus mechanism for the main chain to manage FL tasks and allocate participants fairly and reliably. Additionally, the PoTP consensus mechanism, integrated with the FL training process, defends against inference attacks and poisoning attacks while maintaining trustless model aggregation. Experimental results validate the high parallel processing capabilities, transparent and verifiable training processes, and robust defenses against various attacks, highlighting FLSB's potential to significantly advance secure and efficient FL in IoV scenarios. Further research can be conducted to improve training efficiency and reduce latency, thereby enhancing the practicality of the FLSB framework for IoV.

References

[1] W. Y. B. Lim et al., "Towards federated learning in UAV-enabled internet of vehicles: A multi-dimensional contract-matching approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5140–5154, Aug. 2021.

[2] X. Huang, P. Li, R. Yu, Y. Wu, K. Xie, and S. Xie, "FedParking: A federated learning based parking space estimation with parked vehicle assisted edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9355–9368, Sep. 2021.
[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Stat.*, 2017, pp. 1273–1282.
[4] Z. Li, H. Wu, Y. Lu, B. Ai, Z. Zhong, and Y. Zhang, "Matching game for multi-task federated learning in Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 73, no. 2, pp. 1623–1636, Feb. 2024.
[5] Y. Hui et al., "Digital twins enabled on-demand matching for multi-task federated learning in HetVNets," *IEEE Trans. Veh. Technol.*, vol. 72, no. 2, pp. 2352–2364, Feb. 2023.
[6] C. Zhang, G. Shan, and B.-h. Roh, "Fair federated learning for multi-task 6G NWDAF network anomaly detection," *IEEE Trans. Intell. Transp. Syst.*, early access, Sep. 25, 2024, doi: 10.1109/TITS.2024.3461679.
[7] Y. Li, X. Tao, X. Zhang, J. Liu, and J. Xu, "Privacy-preserved federated learning for autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8423–8434, Jul. 2022.
[8] L. Tian, F. Lin, J. Gan, R. Jia, Z. Zheng, and M. Li, "PEFL: Privacy-preserved and efficient federated learning with blockchain," *IEEE Internet Things J.*, vol. 12, no. 3, pp. 3305–3317, Feb. 2025.
[9] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, "Privacy-enhanced federated learning against poisoning adversaries," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 4574–4588, 2021.
[10] X. Ma, X. Sun, Y. Wu, Z. Liu, X. Chen, and C. Dong, "Differentially private Byzantine-robust federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 3690–3701, Dec. 2022.
[11] H. Zhou, Y. Zheng, H. Huang, J. Shu, and X. Jia, "Toward robust hierarchical federated learning in Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 5, pp. 5600–5614, May 2023.
[12] Y. He et al., "RSAM: Byzantine-robust and secure model aggregation in federated learning for internet of vehicles using private approximate median," *IEEE Trans. Veh. Technol.*, vol. 73, no. 5, pp. 6714–6726, May 2024.
[13] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptogr. Conf.*, 2006, pp. 265–284.
[14] M. Zhang, F. Lin, L. Tian, R. Jia, Z. Zheng, and M. Li, "Evolutionary medical data modeling and sharing via federated learning over sharded blockchain," *IEEE Internet Things J.*, vol. 11, no. 13, pp. 24234–24246, Jul. 2024.
[15] J. Qi, F. Lin, Z. Chen, C. Tang, R. Jia, and M. Li, "High-quality model aggregation for blockchain-based federated learning via reputation-motivated task participation," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 18378–18391, Oct. 2022.
[16] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Inform.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020.
[17] J. An, S. Tang, X. Sun, X. Gui, X. He, and F. Wang, "FREB: Participant selection in federated learning with reputation evaluation and blockchain," *IEEE Trans. Serv. Comput.*, vol. 17, no. 6, pp. 3685–3698, Nov./Dec. 2024.
[18] W. Ali, I. U. Din, A. Almogren, and J. J. Rodrigues, "Federated learning-based privacy-aware location prediction model for Internet of vehicular things," *IEEE Trans. Veh. Technol.*, vol. 74, no. 2, pp. 1968–1978, Feb. 2025.
[19] Z. A. E. Houda, H. Moudoud, B. Brik, and L. Khoukhi, "Blockchain-enabled federated learning for enhanced collaborative intrusion detection in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 7, pp. 7661–7672, Jul. 2024.
[20] C. Meese et al., "Adaptive traffic prediction at the its edge with online models and blockchain-based federated learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 9, pp. 10725–10740, Sep. 2024.
[21] E. Madill, B. Nguyen, C. K. Leung, and S. Rouhani, "ScaleSFL: A sharding solution for blockchain-based federated learning," in *Proc. ACM Int. Symp. Blockchain Secure Crit. Infrastructure*, 2022, pp. 95–106.
[22] Y. Lin et al., "DRL-based adaptive sharding for blockchain-based federated learning," *IEEE Trans. Commun.*, vol. 71, no. 10, pp. 5992–6004, Oct. 2023.
[23] Z. Yang, Y. Shi, Y. Zhou, Z. Wang, and K. Yang, "Trustworthy federated learning via blockchain," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 92–109, Jan. 2023.
[24] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Netw.*, vol. 35, no. 1, pp. 234–241, Jan./Feb. 2021.
[25] M. Shayan, C. Fung, C. J. Yoon, and I. Beschastnikh, "Biscotti: A blockchain system for private and secure federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1513–1525, Jul. 2021.
[26] Y. Wang, H. Peng, Z. Su, T. H. Luan, A. Benslimane, and Y. Wu, "A platform-free proof of federated learning consensus mechanism for sustainable blockchains," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 12, pp. 3305–3324, Dec. 2022.
[27] A. P. Kalapaaking, I. Khalil, M. S. Rahman, M. Atiquzzaman, X. Yi, and M. Almashor, "Blockchain-based federated learning with secure aggregation in trusted execution environment for internet-of-things," *IEEE Trans. Ind. Inform.*, vol. 19, no. 2, pp. 1703–1714, Feb. 2023.
[28] K. Raja et al., "An efficient 6G federated learning-enabled energy-efficient scheme for UAV deployment," *IEEE Trans. Veh. Technol.*, vol. 74, no. 2, pp. 2057–2066, Feb. 2025.
[29] Z. Tong, J. Wang, X. Hou, C. Jiang, and J. Liu, "UAV-assisted covert federated learning over mmWave massive MIMO," *IEEE Trans. Wireless Commun.*, vol. 23, no. 9, pp. 11785–11798, Sep. 2024.
[30] Y. Li, Y. Zhou, A. Jolfaei, D. Yu, G. Xu, and X. Zheng, "Privacy-preserving federated learning framework based on chained secure multiparty computing," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6178–6186, Apr. 2021.
[31] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2017, pp. 119–129.
[32] S. Rajput, H. Wang, Z. Charles, and D. Papailiopoulos, "DETOX: A redundancy-based framework for faster and more robust gradient aggregation," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2019, pp. 10320–10330.
[33] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 2, pp. 1–25, 2017.
[34] L. Zhao et al., "Shielding collaborative learning: Mitigating poisoning attacks through client-side detection," *IEEE Trans. Depend. Secure Comput.*, vol. 18, no. 5, pp. 2029–2041, Sep./Oct. 2020.
[35] B. Wang et al., "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 707–723.
[36] Y. Chen, X. Zhu, X. Gong, X. Yi, and S. Li, "Data poisoning attacks in internet-of-vehicle networks: Taxonomy, state-of-the-art, and future directions," *IEEE Trans. Ind. Inform.*, vol. 19, no. 1, pp. 20–28, Jan. 2023.
[37] A. Hafid, A. S. Hafid, and M. Samih, "A tractable probabilistic approach to analyze sybil attacks in sharding-based blockchain protocols," *IEEE Trans. Emerg. Topics Comput.*, vol. 11, no. 1, pp. 126–136, Jan.–Mar. 2023.
[38] G. Zhang, A. Zhang, and P. Zhao, "Locmia: Membership inference attacks against aggregated location data," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11778–11788, Dec. 2020.
[39] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-Robust federated learning," in *Proc. USENIX Secur. Symp.*, 2020, pp. 1605–1622.
[40] H. Wang et al., "Attack of the tails: Yes, you really can backdoor federated learning," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 16070–16084.
[41] W. Liu et al., "Privacy preservation for federated learning with robust aggregation in edge computing," *IEEE Internet Things J.*, vol. 10, no. 8, pp. 7343–7355, Apr. 2023.
[42] M. Yang, T. Guo, T. Zhu, I. Tjuawinata, J. Zhao, and K.-Y. Lam, "Local differential privacy and its applications: A comprehensive survey," *Comput. Stand. Interfaces*, vol. 89, 2024, Art. no. 103827.
[43] J. Zhang, J. Chen, D. Wu, B. Chen, and S. Yu, "Poisoning attack in federated learning using generative adversarial nets," in *Proc. IEEE 18th Int. Conf. Trust Secur. Privacy Comput. Commun. 13th IEEE Int. Conf. Big Data Sci. Eng.*, 2019, pp. 374–380.