BIT-FL: Blockchain-Enabled Incentivized and Secure Federated Learning Framework

Chenhao Ying, Fuyuan Xia, David S. L. Wei, *Life Senior Member, IEEE*, Xinchun Yu, *Member, IEEE*, Yibin Xu, Weiting Zhang, *Member, IEEE*, Xikun Jiang, Haiming Jin, *Member, IEEE*, Yuan Luo, *Member, IEEE*, Tao Zhang, *Fellow, IEEE*, and Dacheng Tao, *Fellow, IEEE*

***Abstract*—Harnessing the benefits of blockchain, such as decentralization, immutability, and transparency, to bolster the credibility and security attributes of federated learning (FL) has garnered increasing attention. However, blockchain-enabled FL (BFL) still faces several challenges. The primary and most significant issue arises from its essential but slow validation procedure, which selects high-quality local models by recruiting distributed validators. The second issue stems from its incentive mechanism under the transparent nature of blockchain, increasing the risk of privacy breaches regarding workers' cost information. The final challenge involves data eavesdropping from shared local models. To address these significant obstacles, this paper proposes a Blockchain-enabled Incentivized and Secure Federated Learning (BIT-FL) framework. BIT-FL leverages a novel loop-based sharded consensus algorithm to accelerate the validation procedure, ensuring the same security as non-sharded consensus protocols. It consistently outputs the correct local model selection when the fraction of adversaries among validators is less than $1/2$ with synchronous communication. Furthermore, BIT-FL integrates a randomized incentive procedure, attracting more participants while guaranteeing the privacy of their cost information through meticulous worker selection probability design. Finally, by adding artificial Gaussian noise to local models, it ensures the privacy of trainers' local models. With the careful design of Gaussian noise, the excess empirical risk of BIT-FL is upper-bounded by $\mathcal{O}(\frac{\ln n_{\min}}{n_{\min}^{3/2}} + \frac{\ln n}{n})$, where n represents the size of the union dataset, and $n_{\min}$ represents the size of the smallest dataset. Our extensive experiments demonstrate that BIT-FL exhibits efficiency, robustness, and high accuracy for both classification and regression tasks.**

***Index Terms*—Federated learning, blockchain, incentive mechanism, consensus protocol, differential privacy.**

Received 9 April 2024; revised 17 August 2024; accepted 2 October 2024. Date of publication 10 October 2024; date of current version 9 January 2025. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62402313, Grant 62372288, and Grant U21A20519, in part by Shanghai Science and Technology Innovation Action Plan under Grant 23511100400. Recommended for acceptance by L. Guo. *(Corresponding authors: Xikun Jiang; Yuan Luo; Weiting Zhang.)*

Chenhao Ying, Fuyuan Xia, Haiming Jin, and Yuan Luo are with the Department of Computer Science, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: yingchenhao@sjtu.edu.cn; fuyuanxia@sjtu.edu.cn; jinhaiming@sjtu.edu.cn; yuanluo@sjtu.edu.cn).

David S. L. Wei is with the Department of Computer and Information Science, Fordham University, Bronx, NY 10458 USA (e-mail: wei@cis.fordham.edu).

Xinchun Yu is with the Institute of Data and Informtion, Shenzhen International Graduate School of Tsinghua University, Shenzhen 518071, China (e-mail: yuxinchun@sz.tsinghua.edu.cn).

Yibin Xu and Xikun Jiang are with the Department of Computer Science, University of Copenhagen, 1172 København, Denmark (e-mail: yx@di.ku.dk; xikun@di.ku.dk).

Weiting Zhang is with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100082, China (e-mail: wtzhang@bjtu.edu.cn).

Tao Zhang is with the U.S. National Institute of Standards and Technology (NIST), Gaithersburg, MD 20899 USA (e-mail: tao.zhang@nist.gov).

Dacheng Tao is with the School of Computer Science, Faculty of Engineering, The University of Sydney, Sydney, NSW 2050, Australia (e-mail: dacheng.tao@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TMC.2024.3477616, provided by the authors.

Digital Object Identifier 10.1109/TMC.2024.3477616

I. Introduction

With the dramatic development of machine learning (ML), it has been widely applied to various areas in our daily life. However, training a centralized ML model is difficult due to rapidly increasing model scale and required data volume. Therefore, federated learning (FL), a distributed learning paradigm that can also preserve the privacy of local data, is proposed [1]. In a typical FL framework, the ML model (e.g., Multilayer Perceptron (MLP), Convolutional Neural Network (CNN)) is distributed to a plethora of workers, who train this model with their local data and then update the model weights to a central server without divulging the private data, a setup referred to as centralized FL, or update the weights to other distributed workers, known as decentralized FL.

While FL offers significant advantages in terms of privacy preservation, several challenges continue to hinder its widespread adoption [2], [3], [4], [5], [6], [7], [8]. With regard to centralized FL, the biggest challenge is brought by the central server. Since the global aggregation relies on a central server, it is usually exposed to many cyber attacks, such as the single point failure. In contrast, the biggest challenge in decentralized FL is that without the trustworthy central server, no one validates the quality of local models resulting in many potential attacks on learning model brought by malicious distributed workers, such as the random-gradient attack and sign-flipping attack.

As shown in Fig. 1, in order to alleviate these aforementioned drawbacks of traditional FL, some new blockchain-enabled federated learning (BFL) frameworks [9], [10] have been proposed. In such frameworks, with the help of blockchain, the central server is substituted by a set of distributed miner nodes. In fact, blockchain works as a decentralized and distributed digital ledger that records the training information across multiple workers in a way ensuring the security, transparency, and immutability of recorded information.

A. Motivation

Although blockchain enhances the security of FL, it also introduces many new challenges.

The primary and most significant challenge stems from the necessary yet sluggish validation procedure. To mitigate the risk of potential attacks on the learning model introduced by malicious workers during model training, blockchain implementations rely on validators to conduct local assessments of models provided by distributed trainers. Subsequently, validators consistently select certain local models for global aggregation using consensus algorithms within the blockchain network. A prevalent approach involves each validator validating all local models, followed by collective determination of selected models using consensus algorithms like the Practical Byzantine Fault Tolerance (PBFT) algorithm [11]. However, the validation process incurs linear increases in time with the growing number of validated local models, resulting in significant time consumption, particularly when dealing with large volumes of data, extensive learning model scales, and numerous trainers. Additionally, achieving consensus among validators is necessary due to potential adversaries aiming to disrupt the model selection process. However, existing consensus algorithms in BFL typically involve all validators, leading to substantial time overhead for achieving consensus. To address these challenges, an alternative method is to partition validators into smaller shards, enabling parallel validation of different models. However, this method presents challenges in ensuring accurate model selection, as the presence of adversaries in smaller shards may compromise the validity of the selection process.

The second challenge arises from the incentive procedure within the transparent nature of blockchain. In order to incentivize more participants to engage in blockchain for executing training tasks, incentives play a crucial role. Consequently, various auction-based incentive mechanisms have been proposed, where workers are required to report their actual costs as bidding prices to maximize rewards. Based on the reported bidding prices, a smart contract selects some workers to execute the training tasks. However, due to the transparent nature of blockchain, the list of selected workers must be published, allowing every participant in the blockchain to access this information. Nonetheless, this operation may lead to the leakage of cost information in BFL [17]. For instance, consider a price-based incentive mechanism involving three workers, w_1, w_2, w_3, with corresponding bids $b_1 = 1, b_2 = 2, b_3 = 3$. In this mechanism, at the start of each training round, two workers with the lowest bidding prices are selected to execute the training task. However, in scenarios involving multiple training rounds, a worker like w_3 could repetitively decrease their bidding price until selected. This process enables w_3 to infer the bid of another worker, such as w_2. Since this mechanism ensures bids equal actual costs, the cost information of w_2 is inadvertently disclosed, resulting in what is known as an inference attack. Although this attack can be mitigated using anonymity approaches based on cryptography technologies, such as zero-knowledge proofs, these approaches typically entail significant time overhead.

The final challenge stems from privacy preservation in the training procedure. While FL can safeguard trainers' privacy by transmitting local models instead of local data, studies have revealed that sensitive information can still be inferred by analyzing these models [18]. As a result, some existing approaches employ homomorphic encryption on local models. However, computational operations involving ciphertexts incur significant time overhead, particularly in blockchain environments where models need validation by numerous validators. Another alternative involves introducing artificial noise to local models to achieve differential privacy (DP). Nonetheless, this introduces accuracy loss in the learning model due to the added noise, necessitating the design of appropriate noise levels and the selection of suitable trainers to mitigate performance degradation. Despite existing research on FL with differential privacy, direct application to our scenario is not feasible due to the integration of multiple procedures, notably the incentive mechanism. Existing works typically assume uniform random selection of workers to ensure the accuracy of their proposed frameworks theoretically. However, BIT-FL requires worker selection based on a specific probability distribution due to the randomized incentive procedure, rendering existing approaches inadequate for ensuring performance in our scenario.

Therefore, in order to build a secure, efficient, credible, and accurate BFL framework, it requires us to consider all of the above challenges together. To address the first challenge, the validators are divided into small shards so that they can validate the model quality in parallel. Furthermore, to prevent incorrect model selection, a novel loop-based sharded consensus algorithm is proposed. The proposed consensus algorithm not only improves validation efficiency but also achieves the same security guarantee as non-sharded consensus protocols. It can always output the correct decision when the ratio of adversaries among validators is less than $\frac{1}{2}$ under synchronous communication. To tackle the second challenge, a randomized incentive mechanism is proposed, where each worker is selected according to a probability distribution. By meticulously designing the selection probability, it can achieve differential privacy (DP) on workers' reported bids, preventing malicious workers from inferring cost information from the published winner list. Finally, to address the third challenge, the privacy of workers' local models is protected by adding artificial Gaussian noise. After meticulously designing the shape of the noise distribution, we can guarantee the accuracy of FL. For convenience, we summarise the performance comparison of existing works in Table I.

B. Contributions

To address these aforementioned challenges, we propose a novel framework, called BIT-FL. The main contributions of this paper are as follows.

- *Framework Design:* Unlike existing works, we propose a novel framework for BFL, namely, BIT-FL, to provide secure, accurate, and efficient FL training on blockchain. It consists of three procedures: Incentive Procedure, Training Procedure, and Validation Procedure. The Incentive Procedure utilizes a randomized incentive mechanism to attract more workers while protecting the privacy of their cost

TABLE I
PERFORMANCE COMPARISON OF EXISTING BLOCKCHAIN-ENABLED FEDERATED LEARNING

Metric (s)	BIT-FL (Ours)	SPDL	Refiner	BEFL	BCFL	BAFL	PBFL
Incentive Guarantee	✓	✗	✓	✗	✗	✓	✓
Poisoning Security	✓	✓	✓	✓	✓	✓	✓
Privacy Protection	✓	✓	✗	✗	✓	✗	✓
Sharding Construction	✓	✗	✗	✗	✗	✗	✗
Byzantine Tolerance	$\lfloor(\bar{\kappa}-1)/2\rfloor$	$\lfloor(\bar{\kappa}-1)/3\rfloor$	$\lfloor(\bar{\kappa}-1)/3\rfloor$	$\lfloor(\bar{\kappa}-1)/2\rfloor$	$\lfloor(\bar{\kappa}-1)/2\rfloor$	$\lfloor(\bar{\kappa}-1)/3\rfloor$	$\lfloor(\bar{\kappa}-1)/2\rfloor$
Runtime	Short	Moderate	Long	Moderate	Long	Long	Moderate

SPDL: a blockchain-enabled secure and privacy-preserving decentralized learning framework [11], refiner: a reliable and efficient incentive-driven federated learning framework powered by blockchain [12], BEFL: a lightweight blockchain-empowered secure and efficient federated learning framework [13], BCFL: a fast blockchain-based communication-efficient federated learning framework [14], BAFL: a blockchain-based asynchronous federated learning framework [15], PBFL: a privacy-preserving byzantine-robust federated Learning framework based on blockchain [16].

information. The Training Procedure applies a disturbance-based training approach by adding artificial noise to guarantee the privacy preservation of workers' local models. Additionally, the Validation Procedure employs a loop-based sharded consensus algorithm to improve the efficiency and security of model validation.

- *Validation Procedure:* To improve the efficiency of model validation, BIT-FL integrates a novel validation procedure, where validators are divided into small shards, allowing validators in different shards to validate models in parallel. Additionally, it employs a loop-based sharded consensus algorithm to ensure the correctness of model selection with $\lfloor(\bar{\kappa}-1)/2\rfloor$ malicious validators under synchronous communication, where $\bar{\kappa}$ is the number of validators. This achieves the same security guarantee as the non-sharded consensus protocols. To the best of our knowledge, this is the first sharded consensus algorithm that achieves the same security as the non-sharded consensus algorithm while maintaining reasonable performance, greatly improving the efficiency of local validation in BIT-FL.
- *Incentive Procedure:* In order to attract more workers to join the model training, BIT-FL integrates an incentive procedure. It achieves DP on workers' reported bids by implementing a randomized incentive mechanism with meticulous design of selection probability. This not only attracts workers' participation but also defends against false reporting and inference attacks. To the best of our knowledge, this is the first incentive mechanism that considers the defense against interference attacks caused by the transparent nature of blockchain.
- *Training Procedure:* To address privacy concerns regarding workers' sensitive information, BIT-FL integrates a training procedure. This involves adding artificial Gaussian noise to local models to achieve DP. Furthermore, by meticulously designing the Gaussian noise, it limits the excess empirical risk of BIT-FL to $\mathcal{O}(\frac{\ln n_{\min}}{n_{\min}^{3/2}} + \frac{\ln n}{n})$, where n is the size of the union dataset and $n_{\min}$ is the size of the minimum dataset. As far as we know, this represents the initial endeavor to ensure the accuracy of FL from a theoretical perspective under non-uniform worker selection.

C. Organization

The remaining part of the paper proceeds as follows. Section II summarizes related works, and Section III introduces necessary preliminaries. Then, the system model and threat model are introuduced in Section IV, after which, the detailed framework illustration of BIT-FL is provided in Section V, with the corresponding theoretical analysis elaborated in Section VI. Next, we conduct extensive experiments and show the experimental results in Section VII. Finally, this paper is concluded in Section VIII.

II. RELATED WORKS

In this section, some related works are introduced, mainly in terms of incentive-based and DP-based BFL frameworks.

Incentive-Based BFL Frameworks: To attract more participants for executing BFL training tasks, numerous incentive mechanisms have been proposed, employing game-theoretic, contribution-based, or other models, while considering various factors such as the quantity of data [19], [20], [21], [22], [23].

Xu et al. [19] proposed an incentive mechanism based on the local model's contribution, facilitating the quantification of each worker's impact using diverse clustering algorithms. Gao et al. [20] designed an incentive governor aiming to fairly and reliably distribute profits to workers by considering both the contribution and reputation of workers. Wang et al. [21] investigated resource allocation in the BFL framework and subsequently designed an incentive mechanism to offer rewards for both training and mining activities. Wang et al. [22] established economic incentives in blockchain to prevent disturbance by untrustworthy participants and enhance data privacy measures in hierarchical FL.

Differential Privacy-Based BFL Frameworks: There have been many existing works [11], [24], [25], [26] in BFL frameworks, focusing on privacy-preserving issues, typically leveraging DP methods for their excellent efficiency compared to homomorphic encryption and secure multi-party computation [27].

Xu et al. [11] proposed a privacy-preserving and robust BFL system adeptly orchestrating the decentralized learning procedure, exploiting a DP-based approach along with model aggregation techniques drawn from [28]. Rückel et al. [25] proposed an FL framework that incorporates blockchain, local DP, and zero-knowledge proofs to achieve fairness, integrity, and privacy preservation for all participating workers. In terms of DP, Wei et al. [29] introduced the concept of user-level DP, incorporating various scales of noise into the model updates of individual workers. Furthermore, due to the potential presence of Byzantine nodes among the workers in the BFL system, these nodes may maliciously submit fabricated local models in an attempt to undermine the integrity of the training process. In response to this challenge, numerous studies have focused on designing Byzantine-resilient FL systems, with the primary goal of effectively safeguarding the system against attacks from such Byzantine nodes [30], [31], [32], [33], [34], [35], [36], [37].

III. Preliminaries

The preliminaries introduced in this section include blockchain, consensus algorithm, federated learning, and differential privacy.

A. Blockchain and Smart Contract

Blockchain, initially proposed by Nakamoto as an append-only distributed database, serves as a foundational technology for Bitcoin's operation. Functioning as a decentralized ledger, blockchain is maintained and replicated by all participants in the network. When data is written onto the blockchain, it triggers a state transition, which is recorded as a transaction. Furthermore, transactions are collected in the form of blocks, which are then broadcast, synchronized, and confirmed across the blockchain network. Utilizing a peer-to-peer (P2P) network, blockchain technology ensures the secure, transparent, and decentralized recording of transactions.

Smart contracts, initially implemented by Ethereum in 2014, enable the execution of complex transactions within the blockchain. In smart contract-supported blockchain systems, a transaction can carry arbitrary computation. Participants send transactions containing designated program codes to miners for deploying or invoking smart contracts. Each miner that receives the transaction will carry out predefined program codes and verify the legality of the transaction based on the execution results.

B. Vote-Based BFT Consensus Algorithm

Voting to accept or reject a given input is the most common vote-based BFT consensus algorithm, which delivers a binary outcome. Such algorithm requires all participants to vote on only one input. For example, in a leader-based consensus algorithm, participants vote on the value put forward by a leader who is elected according to certain rules. Vote-based BFT consensus algorithms have been applied to blockchains. However, in such algorithms, the next leader cannot be elected and propose a new block until the last block has been voted to accept or reject by all participants. This limitation slows down the state progression of a blockchain and increases transaction latency. To improve the scalability of blockchains, the technique of blockchain sharding is utilized, while addressing issues such as data availability and centralization risks. It partitions the network into smaller units, known as shards, to increase parallelism and reduce the consensus overhead within each shard. Such blockchains are referred to as sharded blockchains.

C. Federated Learning

The FL problem, for the union dataset $\mathcal{D} = \cup_{i=1}^{m} \mathcal{D}_i$ with a total size of n, where $\mathcal{D}_i = \{(\mathbf{x}_1^{(i)}, y_1^{(i)}), \ldots, (\mathbf{x}_{n_i}^{(i)}, y_{n_i}^{(i)})\}$ with a size n_i and $n = \sum_{i=1}^{m} n_i$, aims to find the model parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$ that minimizes the following empirical risk function with a regularization coefficient φ, where $\varphi > 0$:

$$\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta}, \mathcal{D}), \tag{1}$$

for

$$F(\boldsymbol{\theta}, \mathcal{D}) = \sum_{i=1}^{m} \frac{n_i}{n} F_i(\boldsymbol{\theta}, \mathcal{D}_i), \tag{2}$$

where

$$F_i(\boldsymbol{\theta}, \mathcal{D}_i) = \frac{1}{n_i} \sum_{j=1}^{n_i} f(\boldsymbol{\theta}, \mathbf{x}_j^{(i)}, y_j^{(i)}) + \frac{\varphi}{2} ||\boldsymbol{\theta}||^2. \tag{3}$$

D. Differential Privacy

This subsection introduces DP applied to preserve the privacy of the reported bids and the local models. It is defined using a specific concept, i.e., the adjacent sets. Informally, two sets $\mathcal{B}$ and $\mathcal{B}'$ are adjacent if they differ by only one element. Based on the adjacent sets, the (ϵ, δ)-DP is defined as follows, where the privacy budget ϵ and the failure probability δ satisfy $0 < \epsilon < 1$ and $0 < \delta < 1$.

Definition 1 ((ϵ, δ)-Differential Privacy [38]). A randomized mechanism $\mathcal{M} : \mathcal{A} \rightarrow \mathcal{R}$ is (ϵ, δ)-differentially private if for any two adjacent sets $\mathcal{B}, \mathcal{B}' \subseteq \mathcal{A}$, the inequality

$$\mathbf{Pr}[\mathcal{M}(\mathcal{B}) \in \mathcal{K}] \leq e^{\epsilon} \mathbf{Pr}[\mathcal{M}(\mathcal{B}') \in \mathcal{K}] + \delta, \tag{4}$$

holds, where $0 < \epsilon < 1$ and $0 < \delta < 1$. Furthermore, $\mathcal{A}$ and $\mathcal{R}$ are the domain and the output range of the mechanism $\mathcal{M}$, respectively, and $\mathcal{K} \subseteq \mathcal{R}$ is a set of outputs of $\mathcal{M}$.

Definition 2 (Exponential Mechanism [38]). Given a score function $\mu : \mathcal{A} \times \mathcal{R} \rightarrow \mathbb{R}$ and an input set $\mathcal{B} \subseteq \mathcal{A}$, $\mathcal{M}_E : \mathcal{A} \rightarrow \mathcal{R}$ is an exponential mechanism if an output $r \in \mathcal{R}$ is selected by the probability that is proportional to $\exp(\epsilon_E \mu(\mathcal{B}, r))$, i.e.,

$$\mathbf{Pr}[\mathcal{M}(\mathcal{B}) = r] \propto \exp\left(\epsilon_E \mu(\mathcal{B}, r)\right), \tag{5}$$

where the privacy budget ϵ_E satisfies $0 < \epsilon_E < 1$.

Definition 3 (Gaussian Mechanism [38]). Given any function $f : \mathcal{A} \rightarrow \mathbb{R}^d$, the Gaussian mechanism is defined as

$$\mathcal{M}_G(\mathcal{B}) = f(\mathcal{B}) + Z, \tag{6}$$

for any two adjacent sets $\mathcal{B}, \mathcal{B}' \subseteq \mathcal{A}$ with the privacy budget $0 < \epsilon_G < 1$ and the failure probability $0 < \delta_G < 1$, where Z is the additive noise sampled from the Gaussian distribution $\mathcal{N}(0, \sigma^2 I_d)$ with $\sigma \geq \frac{\sqrt{2\ln(1.25/\delta_G)}\Delta_G}{\epsilon_G}$ and $\Delta_G = \sup_{\mathcal{B},\mathcal{B}'} ||f(\mathcal{B}) - f(\mathcal{B}')||$.

IV. System Model and Threat Model

In this section, we introduce the system overview, threat model, and desirable goals.

A. System Overview

We consider a blockchain system with a smart contract C, which comprises a set $\mathcal{W} = \{w_1, \ldots, w_m\}$ of m workers. Each worker w_i holds a dataset denoted as $\mathcal{D}_i = \{(\mathbf{x}_1^{(i)}, y_1^{(i)}), \ldots, (\mathbf{x}_{n_i}^{(i)}, y_{n_i}^{(i)})\}$ with size n_i, where $\mathbf{x}_j^{(i)} \in \mathbb{R}^d$

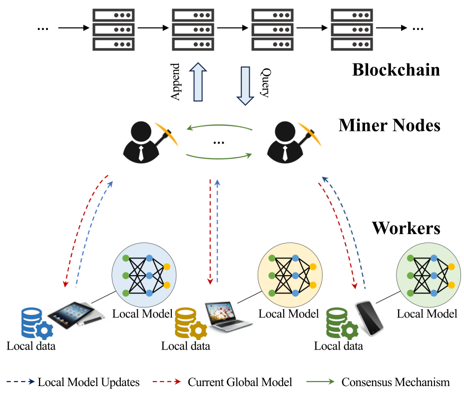


Fig. 1. Illustration of blockchain-based federated learning system.

represents the input in the denotation of a d-dimensional sample vector and $y_j^{(i)}$ represents the output. Let $\mathcal{D} = \cup_{w_i \in \mathcal{W}} \mathcal{D}_i = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ be the total dataset with a size of n, where the superscript of each data is omitted for convenience. At the tth training round, the workflow of BIT-FL is shown in Fig. 2 and described as follows.

- *Incentive Procedure:* The workers in $\mathcal{W}$ obtain the training tasks from blockchain (Step ①). Then each worker w_i submits a bid b_i to the smart contract C (Step ②), which is her bidding price for executing the task in this training round. The actual cost for task execution is denoted as c_i and is not known to any other workers. After receiving worker bids, the smart contract C determines the set $\mathcal{K}_t$ of K winning workers for the tth training round (Step ③). The winner set $\mathcal{K}_t$ is then divided into a set $\mathcal{T}_t$ of κ trainers and a set $\mathcal{V}_t$ of $\bar{\kappa}$ validators at random (Step ④). According to their contributions to model training, smart contract C further determines the payment p_i paid to every trainer in $\mathcal{T}_t$ and validator in $\mathcal{V}_t$ (Step ⑤).
- *Training Procedure:* The winning workers in $\mathcal{K}_t$ download the current global model from the blockchain. Based on the received global model, each trainer $t_i \in \mathcal{T}_t$ conducts the training task locally, and then, submits the local model updates to every validator $v_i \in \mathcal{V}_t$ (Step ⑥). The validators select q models through Validation Procedure to obtain $\boldsymbol{\theta}^{t+1}$ in global aggregation, where the corresponding selected trainers form a selected trainer set $\mathcal{S}_t$ (Step ⑧).
- *Validation Procedure:* The validators form a set $\mathcal{H}_t$ consisting of H shards, each comprising ς validators, where ς is determined as $\varsigma = \lfloor \frac{\bar{\kappa}}{H} \rfloor$. Each shard selects $\lfloor \frac{\kappa}{H} \rfloor$ local models to verify their quality in parallel, utilizing a public dataset $\mathcal{PD}$ that can be accessed by anyone in the system. Subsequently, a leader within each shard proposes a voting procedure, known as a block, and broadcasts it to the other members within the shard (Step ⑦). This block aims to determine whether the validated local model meets the required criteria. Following this, all validators within the shard submit their votes, denoted as 1 or 0 representing "agree" or "disagree". Finally, the validators select q trainers and utilize their local models for global aggregation.

B. Threat Model

Within the framework of BIT-FL, workers are selected from a broad group of participants (e.g., mobile edge computing devices), which may result in malicious workers entering this system. In fact, all of three procedures of BIT-FL are exposed to security risks. We consider two common attacks in the Incentive Procedure:

- *False Reporting Attack [39]:* Workers are required to report their bidding price b_i to compete for the training tasks. Due to their selfish nature, they may provide a false price unequal to their actual cost c_i to maximize utility.
- *Inference Attack [17]:* Due to the transparent nature of blockchain, the smart contract requires the publication of the selected workers. Therefore, when workers report their actual cost as the bidding price, they are exposed to inference attacks on their cost information.

During the Training Procedure of BIT-FL, we consider five potential attacks brought by malicious trainers:

- *Information Eavesdropping Attack [17]:* Although FL can protect the privacy of sensitive information by allowing the submission of local models rather than the private data, it is shown that analyzing local models can also lead to information leakage.
- *Random-gradient Attack [30]:* In this context, malicious trainers generate a random value as the model update instead of conducting the training process. Specifically, the trainer samples values from a Gaussian distribution with mean μ_m and standard deviation σ_m to form their local models.
- *Sign-flipping Attack [30]:* Instead of sending the original model update $G_i(\boldsymbol{\theta}^{t-1})$, malicious trainers set the value $s_m \cdot G_i(\boldsymbol{\theta}^{t-1})$ as the model update, where s_m is the scaling factor of this attack, satisfying $s_m < 0$. Furthermore, as the value of s_m decreases, the magnitude of the sign-flipping attack increases.
- *Adversarial Poisoning Attack [40]:* Unlike the aforementioned untargeted attacks, the adversarial poisoning attack aims to mislead the ML model into incorrect classification behavior on specified tasks. Malicious trainers can corrupt the model in a predetermined manner (e.g., misclassify cats as dogs) by manipulating the training data. As a result, for the specified tasks, the model always outputs an incorrect decision, while for the other tasks, the output is correct.
- *Backdoor Attack [41]:* The malicious trainers conduct the training process using contaminated data to create a hidden trigger within the ML model, introducing a subtle vulnerability. When this trigger condition is met, the model's behavior is maliciously altered, resulting in a predefined incorrect prediction.

During the validation procedure, validators need to determine whether the validated local model can be selected for global aggregation using a vote-based consensus algorithm. In this context, we consider two common attacks in consensus algorithms.

- *Liveness Attack [42]:* It refers to adversaries deviating from the algorithm with invalid operations, hence blocking the progress of the system.

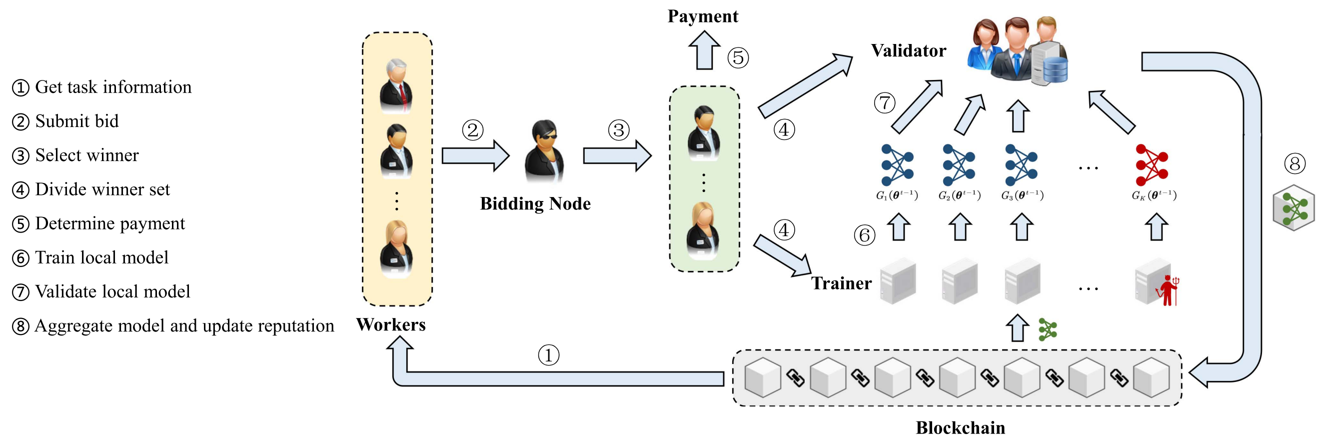


Fig. 2. Illustration of the system overview, where the circled numbers represent the order of events.

- *Safety Attack [42]:* It refers to multiple adversaries colluding and aiming to coerce the system into accepting an incorrect block.

Indeed, consensus algorithms typically utilize liveness and safety thresholds to measure their security.

Definition 4. Liveness threshold I and safety threshold S represent the maximum allowable ratio of adversarial participants within a shard to ensure liveness and safety, respectively, where $S \geq I$.

In fact, the malicious validators implement these two attacks by simply disagreeing the correct validation result or remaining silent during any voting. To illustrate Definition 4 more clearly, let's see an example. In a shard comprising 10 validators with $S = 0.6$, it tolerates six adversarial validators in the shard without compromising safety as long as a decision is reached when at least seven validators concur on the same outcome. However, if four or more validators within the shard disrupt liveness by either disagreeing or remaining silent during voting, the shard becomes unable to reach decisions, thus undermining its liveness. In fact, some works set $S = I = \frac{\tilde{\alpha}}{\varsigma}$, while some set $S = \frac{\tilde{\alpha}}{\varsigma} > I$ where $\tilde{\alpha}$ is the number of adversaries in a shard.

C. Design Objectives

The proposed BIT-FL aims to achieve the following desirable goals:

- *Secure and Fair Incentive:* BIT-FL ensures a fair exchange between task execution and rewards without the need for a central authority, thereby preventing malicious behaviors such as false-reporting attacks. Additionally, it safeguards the privacy of workers' cost information from inference attacks.
- *Secure and Accurate Training:* BIT-FL is capable of defending against attacks in the Training Procedure initiated by malicious trainers. Moreover, as a learning model, it ensures high training accuracy.
- *Secure and Efficient Validation:* BIT-FL ensures security, encompassing liveness and safety, and maintains high efficiency during the validation procedure.

Algorithm 1: BIT-FL $\rightarrow$ Incentive Procedure.

Input: Worker set $\mathcal{W}$, bid b_i of each worker w_i.
Output: Winner set $\mathcal{K}_t$, trainer set $\mathcal{T}_t$, validator set $\mathcal{V}_t$, rewards p_i of winners.

Define a copy set $\widehat{\mathcal{W}} \leftarrow \mathcal{W}$;
$\mathcal{S}_t \leftarrow \emptyset$ (empty set);
while $|\mathcal{K}_t| < K$ **do**
  for *each* $w_i \in \widehat{\mathcal{W}}$ **do**
    `// Winner Selection:`
    Select worker $w_i \in \widehat{\mathcal{W}}$ with probability shown in Eq. (7);
    if w_i *is selected* **then**
      Add w_i to trainer set $\mathcal{T}_i$ with probability P and to validator set $\mathcal{V}_t$ with probability $1 - P$;
      $\widehat{\mathcal{W}} \leftarrow \widehat{\mathcal{W}} \backslash \{w_i\}$;
    Utilizing Validation Procedure to obtain the set $\mathcal{S}_t$ of q selected trainers whose local models will be utilized in global aggregation;
    `// Payment Determination:`
    if $\tau_i \in \mathcal{S}_t$ **then**
      Calculate the payment to τ_i by Eq. (8);
    if v_i *is honest* **then**
      Calculate the payment to v_i by Eq.(9);

V. Framework Illustration

This section presents the design details of BIT-FL and outlines the algorithm. BIT-FL comprises three procedures: the Incentive Procedure, which is used to attract more workers to execute training tasks; the Training Procedure, which aims to train models in a decentralized manner; and the Validation Procedure, which is for validating the quality of local models.

A. Incentive Procedure

As mentioned previously, since other workers may infer the reported bid from the published winner list, it is necessary to protect the privacy of cost information. Although workers can apply zero-knowledge proofs to achieve anonymity, which prevents others from identifying them from the winner list, it consumes a significant amount of time and space to generate corresponding proof. Therefore, here, we will apply the exponential mechanism introduced in Definition 2 in Section III to guarantee the DP of workers' cost.

As shown in Algorithm 1, the Incentive Procedure contains two sub-phases, namely, Winner Selection and Payment Determination, which function as follows.

Winner Selection: To employ this mechanism, the score function is set to $\mu(\widehat{\mathcal{W}}, w_i) = 1 - \frac{b_i}{b_{\max}}$, where $\widehat{\mathcal{W}}$ is a copy set of $\mathcal{W}$. The score function means that for the input $\widehat{\mathcal{W}}$, once worker w_i is selected to execute the training task, the score is $1 - \frac{b_i}{b_{\max}}$. Then, according to this score function, the selection probability of worker w_i is

$$\mathbf{Pr}[\mathcal{M}_E(\widehat{\mathcal{W}}) = w_i] = \begin{cases} \frac{\exp(\epsilon'(1-\frac{b_i}{b_{\max}}))}{\sum_{j:w_j\in\widehat{\mathcal{W}}}\exp\left(\epsilon'\left(1-\frac{b_i}{b_{\max}}\right)\right)} & \text{if } w_i\in\widehat{\mathcal{W}} \\ 0 & \text{otherwise} \end{cases}, \tag{7}$$

where $\epsilon' = \epsilon_E/(e\ln(e/\delta_E))$ for the given privacy budget $0 < \epsilon_E < 1$, failure probability $0 < \delta_E < 1$. Additionally, b_i is w_i's bid. It can be obtained that $\Delta_E = |\frac{1-b_{\min}}{b_{\max}}| \le 1$. For convenient explanation, let's denote $\mathbf{Pr}[\mathcal{M}_E(\widehat{\mathcal{W}}) = w_i]$ as $\mathbf{Pr}(b_i)$.

According to the selection probability determined by specific criteria, the smart contract obtains the winner set $\mathcal{K}_t$ comprising K winning workers. This set is further divided into two subsets uniformly at random: the trainer set $\mathcal{T}_t$ with κ trainers and the validator set $\mathcal{V}_t$ with $\bar{\kappa}$ validators. Each trainer $\tau_i \in \mathcal{T}_t$ is selected from $\mathcal{K}_t$ with a probability P to execute the training tasks, while each validator $v_j \in \mathcal{V}_t$ is selected with a probability $1 - P$ to execute validation tasks.

Payment Determination: The payment p_i to each winner $w_i \in \mathcal{K}_t$ is determined according to her role in model training. In each training round, the smart contract selects a set $\mathcal{S}_t$ of q trainers from the trainer set $\mathcal{T}_t$ based on the workers' model quality validated in the validation procedure. Without loss of generality, let α denote the number of malicious trainers in $\mathcal{T}_t$, and $\bar{\alpha}$ denote the number of malicious validators in $\mathcal{V}_t$. The payments are as follows.

- For the selected trainer τ_i, her payment is

$$p_i = \frac{\kappa - \alpha}{2Pq}\left(b_i + \frac{\int_{b_i}^{b_{\max}}\mathbf{Pr}(b)db}{\mathbf{Pr}(b_i)}\right). \tag{8}$$

- For the honest validator v_j, her payment is

$$p_j = \frac{1}{2(1-P)}\left(b_j + \frac{\int_{b_j}^{b_{\max}}\mathbf{Pr}(b)db}{\mathbf{Pr}(b_j)}\right). \tag{9}$$

Note that the unselected trainers and malicious validators will not obtain the rewards. The details for trainer selection will be illustrated in the validation procedure.

B. Training Procedure

As mentioned previously, since sensitive information can also be divulged by analyzing trainers' submitted local models, it is necessary to protect the privacy of local models. Although trainers can apply homomorphic encryption to guarantee privacy protection, it consumes a lot of time to implement global aggregation with the ciphertexts. Therefore, in this paper, we will apply the Gaussian mechanism, as introduced in Definition 3 in Section III, to ensure DP on workers' local models.

Algorithm 2: BIT-FL → Training Procedure.

Input: Trainer set $\mathcal{T}_t$, validator set $\mathcal{V}_t$, number of training round T, initialized parameter $\boldsymbol{\theta}^0$.
Output: model parameter $\boldsymbol{\theta}^T$.

```
for t=1,...,T do
    // Local Training:
    for each trainer τ_i ∈ T_t do
        Calculate G_i(θ^{t-1}) by Eq. (10), and send it to validators;
    Utilizing Validation Procedure to obtain the set S_t of q selected
      trainers whose local models will be utilized in global
      aggregation;
    // Global Aggregation:
    Update the model parameter θ^t by Eq. (11) utilizing local
      models of selected trainers who are in S_t;
```

As shown in Algorithm 2, this procedure consists of two sub-phases, namely, Local Training, and Global Aggregation, which work as follows.

Local Training: At the tth round, after the incentive procedure, each honest trainer $\tau_i \in \mathcal{T}_t$ calculates the noisy gradient $G_i(\boldsymbol{\theta}^{t-1})$ of the local objective function $F_i(\boldsymbol{\theta}^{t-1}, \mathcal{D}_i)$ according to

$$G_i(\boldsymbol{\theta}^{t-1}) = \nabla F_i(\boldsymbol{\theta}^{t-1}, \mathcal{D}_i) + Z_i^t, \tag{10}$$

where $F_i(\boldsymbol{\theta}^{t-1}, \mathcal{D}_i)$ with trainer τ_i's dataset $\mathcal{D}_i$ is given by (3), and Zi^t are the added noises for $t = 1, \ldots, T$, which are independent and identically distributed (i.i.d) random variables drawn from $\mathcal{N}(0, \sigma^2 I_d)$. Additionally, d represents the dimension of the global model parameter $\boldsymbol{\theta}$. Finally, the noisy gradient $G_i(\boldsymbol{\theta}^{t-1})$ is transmitted to every validator $v_i \in \mathcal{V}_t$ for model validation. In contrast, the malicious trainers in $\mathcal{T}_t$ launch the attacks shown in Section IV by submitting malicious local models.

Global Aggregation: After the validation procedure, it obtains a list $\mathcal{S}_t$ of selected local models. Each validator then calculates the global aggregation and updates the global model $\boldsymbol{\theta}^t$ as

$$\boldsymbol{\theta}^t = \boldsymbol{\theta}^{t-1} - \eta \cdot \sum_{i:w_i\in\mathcal{S}_t}\frac{n_i}{\sum_{j:\tau_j\in\mathcal{S}_t} n_j}G_i(\boldsymbol{\theta}^{t-1}), \tag{11}$$

where $\eta > 0$ is the learning rate, and n_i is the size of trainer τ_i's dataset.

C. Validation Procedure

As mentioned previously, since malicious trainers may launch attacks introduced in Section IV to bring down the model performance, it is necessary to employ validators to assess the quality of local models. However, the time required for model validation increases linearly with the number of models to be validated. If each validator is tasked with validating all local models, it will consume a significant amount of time. Therefore, in the validation procedure, we divide the validators into small shards, where different models are validated in parallel across distinct shards. However, some malicious validators may violate the validation rules. Moreover, due to the division of validators, some shards may contain more malicious validators, making it challenging to accurately validate model quality. Therefore, we propose a loop-based sharded consensus algorithm to assist validators in making accurate decisions and avoiding false positives. In summary, the validation procedure functions as follows: All $\bar{\kappa}$ validators are divided into H shards, each containing $\varsigma = \lfloor \frac{\bar{\kappa}}{H} \rfloor$ validators. At the lth loop, upon receiving the validation task for the local model of trainer τ_i, each validator in shard h_ℓ, where $(i + l - 1) \equiv \ell \mod H$, evaluates the model quality using a public dataset $\mathcal{PD}$. Subsequently, validators in shard h_ℓ exchange their votes with one another. The voting process concludes when the number of validators in validator set $\mathcal{V}_t$ with the same vote surpasses $S_l \times \varsigma \times l$, at which point the validation process is terminated. If this condition is not met, the process proceeds to the next loop, involving a new shard $h_{\ell+1}$. Here, S_l represents the termination threshold, corresponding to the security threshold S discussed in Section IV. Specifically, at the lth loop, S_l is calculated as

$$S_l = R + z\sqrt{\frac{R(1-R)}{\varsigma \times l}}, \tag{12}$$

where $R = \frac{\bar{\alpha}}{\bar{\kappa}}$, with $\bar{\alpha}$ denoting the number of adversaries in the validator set $\mathcal{V}_t$. Additionally, z is a variable derived from the standard Gaussian distribution with a cumulative probability of $1 - \frac{10^{-b}}{H}$, where b represents a security parameter.

In detail, as depicted in Algorithm 3, this procedure consists of two sub-phases: Local Validation and Global Consensus. Here is how they operate: Before commencing the local validation, the validators form a set $\mathcal{H}_t = \{h_1, \ldots, h_H\}$ comprising H shards, with each shard accommodating $\varsigma = \lfloor \frac{\bar{\kappa}}{H} \rfloor$ validators. Additionally, each validator v_j is allocated ς assistants randomly, responsible for storing messages such as the signed vote of validator v_j. Subsequently, each validator receives all local models from trainers and computes the accuracy of the global model $\boldsymbol{\theta}^{t-1}$ using the formula

$$F(\boldsymbol{\theta}^{t-1}, \mathcal{PD}) = \frac{1}{\tilde{n}}\sum_{k=1}^{\tilde{n}} f(\boldsymbol{\theta}^{t-1}, \mathbf{x}_k^{(j)}, y_k^{(j)}) + \frac{\varphi}{2}||\boldsymbol{\theta}^{t-1}||^2, \tag{13}$$

where $\mathcal{PD}$ denotes a public dataset stored in the blockchain for model validation, with a size of $\tilde{n}$.

Local Validation: During the lth loop, each trainer τ_i's local model undergoes validation solely by the validators in shard h_ℓ, where $(i + l - 1) \equiv \ell \mod H$. This validation involves computing the model $\boldsymbol{\theta}_{i,j}^t$ using the formula

$$\boldsymbol{\theta}_{i,j}^t = \boldsymbol{\theta}^{t-1} - \eta \cdot G_i(\boldsymbol{\theta}^{t-1}), \tag{14}$$

where η denotes the learning rate. Subsequently, validator v_j evaluates the accuracy of trainer τ_i's model using

$$F_j(\boldsymbol{\theta}_{i,j}^t, \mathcal{PD}) = \frac{1}{\tilde{n}}\sum_{k=1}^{\tilde{n}} f(\boldsymbol{\theta}_{i,j}^t, \mathbf{x}_k^{(j)}, y_k^{(j)}) + \frac{\varphi}{2}||\boldsymbol{\theta}_{i,j}^t||^2. \tag{15}$$

If $F_j(\boldsymbol{\theta}_{i,j}^t, \mathcal{PD}) < F(\boldsymbol{\theta}^{t-1}, \mathcal{PD})$, it indicates that trainer τ_i's local model positively contributes and can be selected for global aggregation; otherwise, it cannot be selected. Notably, all honest validators will arrive at the same decision for each trainer τ_i, as their input of $F_j(\boldsymbol{\theta}_{i,j}^t, \mathcal{PD})$ and $F(\boldsymbol{\theta}^{t-1}, \mathcal{PD})$ are identical. Following this, the procedure proceeds to the Consensus Implementation of the Global Consensus to determine whether this local model will be selected.

Algorithm 3: BIT-FL $\rightarrow$ Validation Procedure.

Input: Trainer set $\mathcal{T}_t$ with updated of local models, validator set $\mathcal{V}_t$, global model parameter $\boldsymbol{\theta}^{t-1}$.
Output: Decision on the selection of each local model.

Each validator v_j calculates the model accuracy $F(\boldsymbol{\theta}^{t-1}, \mathcal{PD})$ under parameter $\boldsymbol{\theta}^{t-1}$ by Eq. (13);
LOOP $l \leftarrow 0$;
while *Selection of all local model has not been determined* **do**
  for *Each local model $G_i(\boldsymbol{\theta}^{t-1})$ of trainer τ_i* **do**
    // **Local Validation**
    for *Each v_j in shard h_ℓ for $(i + l - 1) \equiv \ell \mod H$* **do**
      Validate model quality $F_j(\boldsymbol{\theta}_{i,j}^t, \mathcal{PD})$ by Eq. (15);
      if $F_j(\boldsymbol{\theta}_{i,j}^t, \mathcal{PD}) > F(\boldsymbol{\theta}^{t-1}, \mathcal{PD})$ **then**
        The model $G_i(\boldsymbol{\theta}^{t-1})$ can be selected;
      Go to Consensus Implementation;
    // **Global Consensus**
    // *Consensus Preparation*
    LOOP $l \leftarrow l + 1$;
    for *Each shard h_ℓ for $(i + l - 1) \equiv \ell \mod H$* **do**
      Select a leader;
      if *LOOP* $l = 1$ **then**
        Leader proposes a block containing the question whether $G_i(\boldsymbol{\theta}^{t-1})$ is selected;
      else
        for *Each validator v_j in shard h_ℓ* **do**
          Receive vote summaries $\boldsymbol{U}_{k,\ell-1}$ from all validators v_k in shard $h_{\ell-1}$;
          Check the summaries and record them to her vote summary;
          if *More than $(1 - S_{l-1})$ fraction of validators in $\cup_j h_j$ disagree* **then**
            Synchronize the block and latest states;
            Go to Local Validation;
    // *Consensus Implementation*
    for *Each v_j in shard h_ℓ for $(i + l - 1) \equiv \ell \mod H$* **do**
      Send her vote to all validators in h_ℓ and her assistants;
      Record all votes of validators in h_ℓ in her summary;
      Go to Consensus Verification;
    // *Consensus Verification*
    Any validator v_i check her vote summary;
    if *No more than S_l fraction identical votes in $\cup_j h_j$ for $\forall j : (i \mod H) \leq j \leq \ell$* **then**
      Validator v_i broadcasts a command for going to Consensus Preparation to the all validators in shards $\cup_j h_j$ for $\forall j : (i \mod H) \leq j \leq (\ell + 1)$;
    else
      Validator v_i sends the vote summary to the current shard leader;
      Go to Consensus Determination;
    // *Consensus Determination*
    for *Leader in shard h_ℓ* **do**
      Request all ς validators in shard h_ℓ to submit their vote summaries;
      Check correction of validator v_i's vote summary submitted at the end of Consensus Verification;
      Propose a decision block containing all summaries and determined correct vote summaries;
    All $\bar{\kappa}$ validators in blockchain make consensus on this decision block by any BFT algorithm;
    if *Decision block is correct* **then**
      Terminate the voting process and broadcast the decision on selection of trainer τ_i's local model;
    else
      Go to Consensus Preparation by additionally involving shard $h_{\ell+1}$;

Global Consensus In order to prevent false decisions in the presence of malicious validators, this paper proposes a new loop-based consensus algorithm. The algorithm consists of four sub-phases.

- *Consensus Preparation:* Before reaching consensus, each shard h_ℓ randomly selects a leader. During the first iteration, this leader broadcasts a block containing the question of whether trainer τ_i's local model will be selected.

Conversely, in subsequent iterations (denoted by l), validators within shard h_ℓ, indexed by $(i+l-1) \equiv \ell \mod H$, receive all vote summaries $\boldsymbol{U}_{k,\ell-1}$ from all ς validators v_k. These summaries are obtained exclusively from the last involved shard $h_{\ell-1}$, for which $(i+l-2) \equiv (\ell-1) \mod H$, where $\forall k: v_k \in h_{\ell-1}$. The vote summary $\boldsymbol{U}_{k,\ell-1}$, a matrix authored by validator v_k, contains $\bar{\kappa}$ rows and two columns representing different voting statuses (Acceptance (00), Rejection (01), No-voting (10), Double Voting (11)). Each row stores the voting status of validator $v_j \in \cup_z h_z$, where $(i \mod H) \leq z \leq (\ell-1)$, corresponding to the jth row. Any honest validator $v_e \in h_\ell$ diligently checks for conflicting information in all received $\boldsymbol{U}_{k,\ell-1}$. In case of discrepancies, such as the vote status of validator v_b recorded in $\boldsymbol{U}_{k,\ell-1}$ differing from other summaries, validator v_e resolves the issue by querying the vote in question from the assistants of validator v_b and drawing her own conclusion. Subsequently, she records all these votes in her vote summary. If the combination of $\boldsymbol{U}_{k,\ell-1}$ for all $k: v_k \in h_{\ell-1}$ reveals that at least $(1-S_{l-1})$ fraction of all voted validators, i.e., the fraction of all validators in shards $\cup_j h_j$ where $\forall j: (i \mod H) \leq j \leq (\ell-1)$, either disagree with the majority voting or remain silent, each validator $v_j \in h_\ell$ synchronizes the block and proceeds to Local Validation to implement the $(l+1)$th local validation.

- *Consensus Implementation:* Each validator $v_j \in h_\ell$ sends their vote to all validators in shard h_ℓ. Simultaneously, they transmit the vote to their assistants, including a block hash, their decision, and a digital signature. Subsequently, each validator v_j records all received votes from validators in shard h_ℓ in their vote summary. Then, the validators proceed to Consensus Verification.
- *Consensus Verification:* Any validator $v_i \in h_\ell$ checks their vote summary. If no more than S_l fraction of validators cast identical votes and the pre-defined time bound Δ_{loop} is reached, validator v_i broadcasts a command for all validators in shards $\cup_j h_j$, where $\forall j: (i \mod H) \leq j \leq (\ell+1)$, to proceed to Consensus Preparation. Otherwise, if more than S_l fraction of validators in shards $\cup_j h_j$, where $\forall j: (i \mod H) \leq j \leq \ell$, cast identical votes, validator v_i sends the vote summary to the current shard leader.
- *Consensus Determination:* Upon receiving the summary from validator v_i, the leader requests all ς validators in shard h_ℓ to submit their vote summaries to prevent adversary manipulation. In fact, the leader checks the correctness of validator v_i's submitted vote summary by comparing it with the others. If there are discrepancies among the received vote summaries, the leader determines the correct votes from the assistants of validators whose votes are distinct. Finally, the leader proposes a decision block containing all summaries as well as the correct vote summary determined. Following this, all $\bar{\kappa}$ validators in the network reach a consensus on this decision block, which can be facilitated by any BFT algorithm after simply checking the contained vote summaries. If the decision is deemed correct, the leader terminates the voting process in shard h_ℓ and broadcasts the decision on the selection of trainer τ_i's local model. Otherwise, the voting process proceeds to Consensus Preparation for the $(l+1)$th loop.

Remark 1. In the Validation Procedure, several key considerations arise:

- During Global Consensus, malicious behaviors by adversaries may result in a delay in accepting or rejecting a block (decision of model selection), requiring multiple iterations involving additional shards. To prevent adversaries from manipulating the voting process by submitting malicious votes, validators whose votes (including those opposing the final decision or remaining silent) are contrary to the consensus will be expelled from the system.
- Since adversaries may attempt to prematurely terminate the voting process by sending an incorrect vote summary, it is necessary for a randomly selected leader to verify their summary to prevent such malicious behavior. However, given the possibility that the leader may also act dishonestly, the leader is required to propose a decision block for other validators to verify her actions. To ensure the integrity of this process, a Byzantine Fault Tolerance (BFT) algorithm is employed in Consensus Determination. It's important to note that this BFT algorithm is solely utilized to validate the leader's vote checking, which can be efficiently completed by verifying the contained vote summaries in the decision block without requiring additional complex operations such as model validation.

VI. Theoretical Analysis

In this section, it will be shown that BIT-FL satisfies the properties introduced in Section IV. In brief, it preserves the DP of reported bids (Theorem 1) and local models (Theorem 2) while maintaining security against Byzantine attacks when validators validate models (Theorem 4). Tight bounds on the excess empirical risk are derived when the empirical risk function is φ-strongly convex (Theorem 3).

A. Analysis of Incentive Procedure

Due to the incentive mechanism employed, BIT-FL demonstrates the ability to thwart the attacks discussed in Section IV.

Lemma 1. BIT-FL enables all workers to truthfully report their bids.

Proof. Please refer to the proof in Appendix A, available online. □

In addition to defending against false reporting attacks, an effective incentive mechanism should also provide workers with incentives to participate and benefit from task execution.

Lemma 2. When there are α malicious trainers in the trainer set $\mathcal{T}_t$, with $\alpha \leq \kappa - q$, BIT-FL ensures that both validators and selected trainers satisfy individual rationality in Payment Determination. Here, q represents the number of selected trainers in the trainer set $\mathcal{T}_t$, which has a size of κ.

Proof. The conclusion follows from (8) and (9) since workers will truthfully report their bids, implying $b_i = c_i$ for each worker w_i. □

Furthermore, BIT-FL's meticulous design of worker selection probability helps prevent inference attacks on workers' cost information."

Theorem 1 (Differential Privacy of Bids). For any privacy budget $0 < \epsilon_E < 1$ and failure probability $\delta_E \in (0, \frac{1}{2}]$, BIT-FL is $(\epsilon_E(e-1)/e, \delta_E)$-differentially private on the reported bids, where e is the base of the natural logarithm.

Proof. Please refer to the details in Appendix B, available online. □

B. Analysis of Training Procedure

With the help of the Validation Procedure, BIT-FL can defend against attacks by malicious trainers, including information eavesdropping attack, random-gradient attack, sign-flipping attack, adversarial poisoning attack, and backdoor attack. Before presenting the theoretical analysis, it requires making some assumptions, denoting the ℓ_2-norm and the inner product as $||\cdot||$ and $< \cdot, \cdot >$, respectively.

Assumption 1 (G-Lipschitz). A loss function $f : \mathcal{C} \times \mathcal{D} \rightarrow \mathbb{R}$ is said to be G-Lipschitz over $\boldsymbol{\theta}$ if, for any jth data point $(\mathbf{x}_j, y_j) \in \mathcal{D}$ and parameters $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathcal{C}$, where $0 < j \leq n$ and $G > 0$, the inequality $|f(\boldsymbol{\theta}_1, \mathbf{x}_j, y_j) - f(\boldsymbol{\theta}_2, \mathbf{x}_j, y_j)| \leq G||\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2||$ holds.

Assumption 2 (L-Smooth). A differentiable loss function $f : \mathcal{C} \times \mathcal{D} \rightarrow \mathbb{R}$ is L-smooth over $\boldsymbol{\theta}$, if for any jth data $(\mathbf{x}_j, y_j) \in \mathcal{D}$ and parameters $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathcal{C}$, where $0 < j \leq n$ and $L > 0$, the inequality $f(\boldsymbol{\theta}_1, \mathbf{x}_j, y_j) \leq f(\boldsymbol{\theta}_2, \mathbf{x}_j, y_j) + < f(\boldsymbol{\theta}_2, \mathbf{x}_j, y_j), \boldsymbol{\theta}_1 - \boldsymbol{\theta}_2 > + \frac{L}{2}||\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2||^2$ holds.

Assumption 3 (φ-Strongly Convex). A differentiable empirical risk function $F : \mathcal{C} \times \mathcal{D} \rightarrow \mathbb{R}$ is φ-strongly convex over $\boldsymbol{\theta}$ if for all $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathcal{C}$ and a constant $\varphi > 0$, it satisfies

$$F(\boldsymbol{\theta}_2, \mathcal{D}) \geq F(\boldsymbol{\theta}_1, \mathcal{D}) + < \nabla F(\boldsymbol{\theta}_1, \mathcal{D}), \boldsymbol{\theta}_2 - \boldsymbol{\theta}_1 > + \frac{\varphi}{2}||\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1||^2. \tag{16}$$

Lemma 3. BIT-FL can defend against the random-gradient attack, sign-flipping attack, adversarial poisoning attack, and backdoor attack from malicious trainers.

Proof. These attacks can be defended by implementing the Validation Procedure. In this procedure, the local model of each trainer needs to be validated by distributed validators. According to the selection rules, only the models that positively contribute to global aggregation can be selected. □

To defend against information eavesdropping attacks, BIT-FL allows different trainers to add artificial Gaussian noise. In fact, by meticulously designing the shape of the noise, BIT-FL can achieve DP on trainers' local models.

Theorem 2 (Differential Privacy of Local Models). For any privacy budget $0 < \epsilon_G < 1$ and failure probability $0 < \delta_G < 1$, if the loss function $f(\boldsymbol{\theta}, \cdot, \cdot)$ is G-Lipschitz over $\boldsymbol{\theta}$, with q selected trainers, BIT-FL is (ϵ_G, δ_G)-differentially private on the local models by setting the variance σ^2 of the additive Gaussian noise according to

$$\sigma^2 = c \cdot \frac{\ln(1/\delta_G)}{\epsilon_G^2} \cdot \frac{G^2 T}{q^2 n_{\min}^2}, \tag{17}$$

where $c > 0$ is a constant, T is the number of training rounds, and $n_{\min}$ is the size of minimum dataset.

Proof. Please refer to the details in Appendix C, available online. □

Remark 2. To ensure the differential privacy of local trainers in federated learning (FL), many studies add Gaussian noise to local models by carefully designing the Gaussian distribution's shape (i.e., the distribution's variance). To demonstrate that their mechanisms protect the differential privacy of local trainers, these studies typically use the moment accountant method proposed in [18]. Moreover, most existing FL frameworks are based on FedAvg, proposed in [1], where each trainer is selected uniformly at random. When incentive mechanisms do not affect the selection probability, the aggregated global model also follows a Gaussian distribution, allowing the existing analysis to ensure differential privacy. However, unlike these approaches, BIT-FL applies a randomized incentive mechanism, meaning that workers are not selected uniformly at random. As a result, the aggregated global model follows a mixture of Gaussian distributions rather than a pure Gaussian distribution, leading to a more complex analysis. Therefore, it is necessary to design a new shape for the added Gaussian distribution to generate artificial noise that guarantees the differential privacy of local models.

By denoting the random sequences $\mathcal{S}_{[t]} = \mathcal{S}_1 \ldots \mathcal{S}_t$ and $\mathcal{Z}_{[t]} = \mathcal{Z}_1 \ldots \mathcal{Z}_t$, the following theorem is established, where $Z_i^t \sim \mathcal{N}(0, \sigma^2 I_d)$ and $\mathcal{Z}_t = \sum_{i:w_i \in \mathcal{S}_t} \frac{n_i}{\sum_{j:w_j \in \mathcal{S}_t}^{n_j}} Z_i^t$ for $t = 1, 2, \ldots, T$. Furthermore, $\mathcal{S}_t$ is the selected trainer set at the tth training round, where each selected trainer $w_i \in \mathcal{S}_t$ has a dataset with the size n_i.

Theorem 3 (Training Accuracy). Without loss of generality, assume that the loss function $f(\boldsymbol{\theta}, \cdot, \cdot)$ is G-Lipschitz and L-smooth over $\boldsymbol{\theta}$, and that $F(\boldsymbol{\theta}, \mathcal{D})$ is differentiable. Furthermore, assume that $F(\boldsymbol{\theta}, \mathcal{D})$ is φ-strongly convex. For the privacy budget $0 < \epsilon_G < 1$ and the failure probability $0 < \delta_G < 1$, when selecting q honest trainers in each training round and setting the variance σ^2 of the additive Gaussian noise according to (17) with the positive learning rate $\eta = \frac{1}{L}$, BIT-FL satisfies

$$0 \leq \mathbb{E}_{\mathcal{S}_{[T]}, \mathcal{Z}_{[T]}}[F(\boldsymbol{\theta}^T, \mathcal{D})] - F^* \leq c_1 \frac{\ln(1/\delta_G)\ln(\epsilon_G q n_{\min})}{(\epsilon_G q n_{\min})^2} + c_2 \frac{\ln(1/\delta_G)\ln(\epsilon_G q n_{\min})}{(\epsilon_G q n_{\min})^{3/2}} + c_3 \frac{\ln n}{n}, \tag{18}$$

where $T = c_4 \cdot \ln\left(\frac{\epsilon_G^2 q^2 n_{\min}^2}{d G^2 \ln(1/\delta_G)}\right)$ is the number of training rounds, $F^* = \min_{\boldsymbol{\theta}} F(\boldsymbol{\theta}, \mathcal{D})$ is the global minimization of $F(\boldsymbol{\theta}, \mathcal{D})$, and d is the dimension of $\boldsymbol{\theta}$. Furthermore, the minimum data set has the size $n_{\min}$ while the union set of all data has the size n. Additionally, the parameters c_1, c_2 and c_3 are some positive constants.

Proof. Please refer to the detailed proof in Appendix D, available online. □

Remark 3. The artificial noise added to the local models, aimed at preserving individual privacy, often results in a loss of accuracy. In Definition 1, ϵ_G represents the privacy parameter, with higher values indicating a greater risk of disclosing private information. According to Theorem 2, an increase in ϵ_G corresponds to a decrease in the variance σ^2 of artificial Gaussian noise. This suggests that smaller noise, typically resulting in higher training accuracy, is more likely to be drawn and added to the local models. Conversely, a decrease in ϵ_G leads to lower training accuracy. Furthermore, Theorem 3 demonstrates that a larger ϵ_G results in a smaller gap between the training loss and the optimal loss, typically implying higher training accuracy. Therefore, by considering Theorems 2 and 3, we can design the Gaussian noise distribution by selecting an appropriate ϵ_G to strike a balance between privacy preservation and training accuracy.

C. Analysis of Validation Procedure

It is known that there are $\bar{\alpha}$ adversaries in the validator set $\mathcal{V}_t$. In order to improve the efficiency of the validation procedure with a loop-based sharded consensus algorithm, BIT-FL divides the validator set into H shards, which may reduce the security of the algorithm since there may be more adversaries in some shards. Therefore, to guarantee the security of the consensus algorithm, it is necessary to determine the size of the shard.

Lemma 4. Given a validator set $\mathcal{V}_t$ with size $\bar{\kappa}$ and $\bar{\alpha}$ adversaries such that $R = \frac{\bar{\alpha}}{\bar{\kappa}} < \frac{1}{2}$, for any $R < S_l < 1$, to guarantee that the probability of having at least $S_l \times \varsigma$ adversaries in each of H randomly selected shards is greater than or equal to $1 - \frac{10^{-b}}{H}$, the shard size ς must satisfy $\varsigma \geq \frac{R(1-R)}{(\frac{S_l - R}{z})^2}$, where z is a Gaussian variable with a cumulative distribution probability of $1 - \frac{10^{-b}}{H}$ for a security parameter b.

Proof. Please refer to the detailed proof in Appendix E, available online. □

Corollary 1. With the same condition as in Lemma 4, to guarantee that the probability of having at least one honest validator in each of H randomly selected shards is greater than or equal to $1 - \frac{10^{-b}}{H}$, the shard size ς must satisfy: $\varsigma \geq \frac{R(1-R)}{(\frac{S_1 - R}{z})^2}$ for $S_1 = 1 - \frac{1}{\varsigma}$.

Lemma 5. The adversary cannot compel the activation of a new loop if the voting process has terminated in the previous loop.

Proof. Please refer to the detailed proof in Appendix F, available online. □

Lemma 6. An adversarial shard leader can be identified through additional measures.

Proof. Please refer to the detailed proof in Appendix G, available online. □

Theorem 4. The proposed loop-based sharded consensus algorithm achieves the same security guarantee as that of a non-sharded consensus algorithm. Specifically, when there are $\bar{\alpha} \leq \lfloor(\bar{\kappa} - 1)/2\rfloor$ adversaries in the validator set, validators can always make the correct decision on the selection of local models.

Proof. Please refer to the detailed proof in Appendix H, available online. □

VII. Performance Evaluation

We perform the extensive experiments and discuss the experimental results in this section.

A. Baseline Methods

As previously introduced, BIT-FL consists of an incentive procedure (IncPro), a training procedure (TraPro), and a validation procedure (ValPro). Therefore, its performance will be evaluated from these three aspects.

1) Incentive Procedure: Since no existing works can be applied to the BFL system examined in this paper, two baseline methods are considered.

The first method is a modification of the incentive mechanism proposed in [43], known as the Enabling Privacy-Preserving Incentive Mechanism (EPPIM). Originally designed for collecting high-quality sensing data, it has been adapted for our purposes. In this modified version, when recruiting a fixed number of workers to execute the training task, the platform randomly selects a worker set from several possible sets. Each set contains at least the fixed number of workers. Once a set is selected, each participant carries out the task at a certain cost. Afterwards, the platform pays all participants, but only the fixed number of local models are selected for global aggregation.

The second method is a modification of the IncPro of BIT-FL, referred to as the Modified Randomized Incentive Mechanism (MRIM). In contrast to the original Winner Selection, in MRIM, the selection probability of each w_i is proportional to her bid (i.e., $\frac{b_i}{b_{\max}}$). The Payment Determination process of MRIM remains the same as that of BIT-FL.

2) Training Procedure: To evaluate the performance of TraPro, we compare it with various existing distributed learning frameworks.

The first distributed learning framework was proposed by McMahan in [1], known as FedAvg. In FedAvg, trainers submit local models for aggregation without incorporating any form of noise. For a fair comparison, we adapt the approach by introducing noise at the same magnitude used in TraPro.

The second framework, proposed by Blanchard in [28], is called Multi-Krum. This method aims to enhance the resilience of distributed implementations of the SGD algorithm against model attacks, particularly in scenarios where no noise is initially incorporated. We adopt a similar approach to that used in the FedAvg framework by introducing noise to ensure the preservation of DP.

The third framework, proposed by Wei in [29], involves adding Gaussian noise and is known as User-Level Differential Private Federated Learning (UDP-FL). This baseline inherently includes a privacy-preserving mechanism based on DP, eliminating the need for additional interventions.

Note that all of the above baseline methods are fully distributed learning frameworks, where all workers are recruited to train the model. To compare them with BIT-FL, we modify them as follows: In each training round, only some workers are selected uniformly at random to execute the training task.

3) Validation Procedure: In the design of BFL, many existing works [11] utilize all workers to validate the local models and integrate a PBFT consensus algorithm for model selection. Therefore, we initially compare the ValPro of BIT-FL with similar approaches, such as the scheme integrated by a blockchain-enabled Secure and Privacy-preserving Decentralized Learning system (SPDL) proposed in [11]. Furthermore, since BIT-FL

TABLE II
SETTINGS OF PARAMETER INTERVALS FOR EVALUATION

Parameters	Values
Environment	Intel(R) Xeon(R) Gold 5120 CPU, Ubuntu 18.04, Python 3.7, PyTorch 1.12 library
Blockchain architecture	Substrate with Rust 1.66 programming language
Worker quantity m	$[10, 90]$ in IncPro, 100 in TraPro, $[100, 1000]$ in ValPro
Dataset $\mathcal{D}$	MNIST, for classification CT Slices for regression
Size of public dataset	$30KB$
Worker cost c_i	$[5, 10]$
Learning rate η	0.005
Number of Training rounds T	200
Regularization coefficient ϕ	0.0005
Lischitz constant G	1
Smooth constant L	1
Privacy bound ϵ_E	0.1
Failure probability δ_E	0.25
Failure probability δ_G	0.001
Privacy bound ϵ_G	0.01 in regression, 0.001 in classification
Shard quantity H	10 in ValPro 40 in an additional experiment for loop-based sharded consensus algorithm
Security parameter b	2
Vote Size	513 bits compromising a 256-bit SHA-256 block hash, a 256-bit digital signature, a 1-bit indicator of acceptance or rejection

integrates a loop-based sharded consensus algorithm, we will further compare its performance with existing sharded consensus algorithms.

The first consensus algorithm is the trivial PBFT, where all validators require to vote. This algorithm can output correct decision under $\lfloor (\bar{\kappa}-1)/3 \rfloor$ adversaries.

The second is called Gearbox and is proposed in [44]. It is also a sharded consensus algorithm. It changes I and S to allow the use of small shards when the ratio of adversarial validators intending to attack liveness is low. This algorithm can output correct decision under $\lfloor (\bar{\kappa}-1)/2 \rfloor$ adversaries.

The third algorithm is called Reticulum proposed in [42]. It relies on a two-layer sharding design that does not adjust shard memberships to restore liveness. However, this algorithm can only tolerate $\lfloor (\bar{\kappa}-1)/3 \rfloor$ adversaries.

B. Experimental Settings

Our experiments are conducted on a machine running the Ubuntu 18.04 operating system, equipped with an Intel(R) Xeon(R) Gold 5120 CPU operating at 2.20 GHz, featuring a total of 56 cores and a memory capacity of 512 GB. The blockchain application is developed using the Rust 1.66 programming language in conjunction with the Substrate[1] framework to simulate a blockchain network. The machine learning procedures were executed using the Python 3.7 programming language along with the PyTorch 1.12 library. For convenience, the experimental settings are shown in Table II.

1) Incentive Procedure: In the evaluation of BIT-FL's IncPro, we have established specific parameters: $T = 1500$ training rounds, $K = 80$ recruited workers, a privacy budget of $\epsilon_E = 0.1$, and a corresponding failure probability of $\delta_E = 0.25$. Our investigation focuses on understanding the social cost and total payment achieved by BIT-FL in a single training round. We vary the number of workers, denoted as m, across a range from 10 to 90. Additionally, we conduct the evaluation under varying scenarios of training rounds by adjusting T within the range of 100 to 1500.

2) Training Procedure: It can be seen that both regularized logistic regression and ridge regression satisfy the conditions outlined in our corresponding theorems and propositions. For convenience, we summarized their details in Appendix J, available online. However, it should be pointed out that the G-Lipschitz property of the loss function $f(\boldsymbol{\theta}, \mathrm{x}_i, y_i)$ for the ridge regression holds only when the values of (x_i, y_i) are finite, i.e., $||\mathbf{x}_i|| + |y_i| < \infty$.

The following two popular datasets are utilized to evaluate BIT-FL for classification and regression tasks.

- *Classification Tasks:* For classification, a regularized logistic regression model is applied over the MNIST data set. The MNIST dataset comprises 70,000 samples, with 60,000 samples designated for training and 10,000 samples for testing. Each sample represents a handwritten digit in the form of a grayscale image with dimensions of 28×28 pixels. The objective of this task is to predict the digit represented by each of these images.
- *Regression Tasks:* For regression, a ridge regression model is trained over the CT Slices [45] dataset, which contains 53,500 samples of CT slices described by two histograms in polar space. The task is to predict the relative location of the image along the axial axis based on the histograms that describe bone structures and air inclusions. We randomly select 42,800 samples as the training set, while the remaining 10,700 samples are designated as the testing set.

In our experiments, the training set is randomly divided into 100 subsets, where each participant selects one subset, i.e., there are 100 workers in total. The dataset is not evenly divided, as the largest subset contains 2,000 samples, while the smallest has only 200. Furthermore, the experiments set the Lipschitz constant $G = 1$, the regularization coefficient $\varphi = 0.0005$, the learning rate $\eta = 0.005$, and the total number of training rounds $T = 200$. Besides, in the context of both Multi-Krum and BIT-FL experiments, only a part of trainers' local models are selected for the purpose of global aggregation, while all local models are selected in UDP-FL and FedAvg.

In regression tasks, the privacy bound ϵ_G and the failure probability δ_E are set to 0.01 and 0.001, respectively, and both Multi-Krum and BIT-FL involve 50% of local models for global aggregation. In contrast, in classification tasks, the privacy bound ϵ_G is set to 0.001, and both Multi-Krum and BIT-FL only involve 24 local models for global aggregation. To ensure fairness, the number of honest trainers in BIT-FL is equivalent to the number of nodes conducting local model training in the other baselines for both tasks.

Furthermore, the experiments also involve four training attacks mentioned in Section IV-B. Specifically, in the random-gradient attack, the mean μ_m and standard deviation σ_m are set to 0 and 1, respectively. Meanwhile, in the sign-flipping attack, the scaling factor s_m is set to 4. We further conduct mixed untargeted

[1]https://github.com/paritytech/substrate

attack by allowing adversaries to employ the above two attacks with a probability of 50% each. For the adversarial poisoning attack and the backdoor attack, we adopt label flipping strategy mentioned in [40] and pattern backdoor strategy mentioned in [46], respectively.

3) Validation Procedure: In order to validate the model quality, it requires the utilization of a public dataset $\mathcal{PD}$. In the evaluation, the size of the public dataset $\mathcal{PD}$ is set to $30KB$, sampled from MNIST and CT slices datasets, respectively. Additionally, we assume that there are $\bar{\alpha} = \lfloor(\bar{\kappa}-1)/2\rfloor$ or $\bar{\alpha} = \lfloor(\bar{\kappa}-1)/3\rfloor$ adversaries among validators, depending on the tolerance capability of different baselines. When evaluating the performance of ValPro, we further divide the validators into 10 shards. To guarantee that each shard has a limited number of adversaries under the security parameter $b = 2$, there are at least 10 validators in each shard. The size of each shard increases from 10 to 100 in the evaluation of ValPro. Due to the proposed loop-based sharded algorithm, we further evaluate its performance with the number of validators set to 4000 and the number of shards set to 50, resulting in each shard containing 80 validators. Furthermore, a vote comprises a 256-bit SHA-256 block hash, a 1-bit indicator of acceptance or rejection of the block, and a 256-bit digital signature from validators.

C. Experimental Metrics

We measure the performance of BIT-FL by comparing it with other baselines, using appropriate metrics in the performance evaluation.

1) Incentive Procedure: We will focus on its financial properties since IncPro aims to attract more participation. Therefore, we evaluate the social cost $\sum_{i:w_i\in\mathcal{S}_t} c_i$ and total payment $\sum_{i:w_i\in\mathcal{S}_t} p_i$. Lower social costs and total payments indicate better performance.

2) Training Procedure: For classification tasks, we measure the performance using the average cross-entropy loss of the global model across all local samples and the prediction accuracy on the test dataset. For regression tasks, we use the mean squared error (MSE) as the metric.

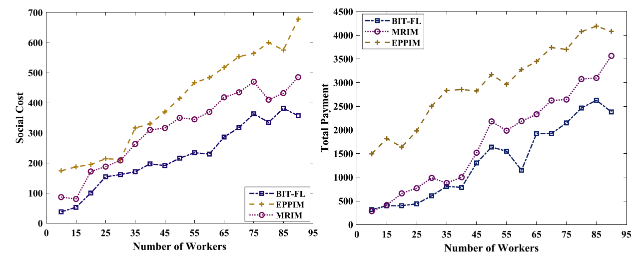
3) Validation Procedure: To evaluate the performance of the Validation Procedure (ValPro) in BIT-FL, we measure the time consumed for both local validation and global consensus. For local validation time, we calculate the average time over 1000 repeated experiments. Regarding global consensus, we analyze the average (Avg), minimum (Min), and maximum (Max) times over 1000 experiments. Lower times indicate better performance.

D. Experimental Results

In this subsection, we present the experimental results for IncPro, TraPro and ValPro, respectively.

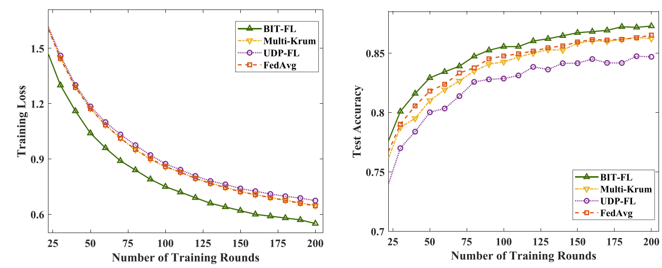
1) Incentive Procedure: In this subsection, we present the evaluation results of IncPro.

Fig. 3 displays the social cost and total payment in one training round for different numbers of workers. An increasing number of workers leads to higher total payment and social cost since more trainers and validators incur costs and receive payments. As explained in Section VII-A, MRIM's selection probability is directly proportional to bids, favoring higher-cost workers. This leads to increased social costs and total payments. Furthermore, as explained in Section VII-A, unlike BIT-FL, EPPIM selects a set to perform the training tasks with a size larger than the required number of workers, leading to higher social costs and total payments.



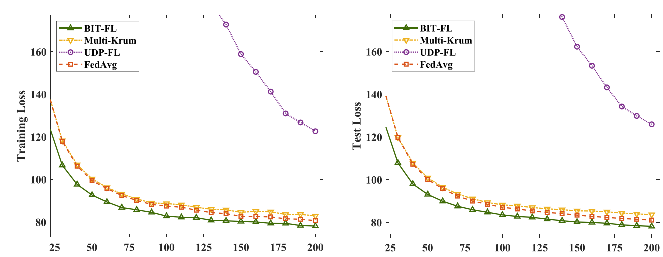
(a) (b)

Fig. 3. (a) Social cost versus different numbers of recruited workers. (b) Total payment versus different numbers of recruited workers.



(a) (b)

Fig. 4. Training loss (a) and test accuracy (b) on MNIST dataset versus different numbers of training rounds, where the selection probability is 0.8 in baselines.

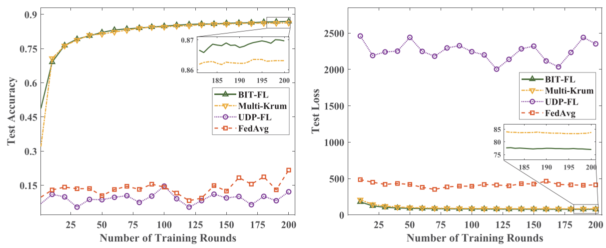


(a) (b)

Fig. 5. Training loss (a) and test loss (b) on CT Slices dataset versus different numbers of training rounds, where the selection probability is 0.8 in baselines.

2) Training Procedure: In this section, we present the evaluation results of TraPro compared to the baseline methods. As mentioned earlier, in TraPro, winners are selected by the IncPro, whereas in the baseline methods, winners are chosen uniformly at random.

In the absence of model attacks, Figs. 4 and 5 show the experimental results of BIT-FL and other baseline methods for different numbers of training rounds on the MNIST and CT Slices datasets, respectively. In Figs. 4 and 5, the selection probability in the baselines is set to 0.8. From the above results, it is evident that when the selection probability is 0.8, BIT-FL outperforms the other baselines in terms of performance on both the training



(a) (b)

Fig. 6. Test accuracy on MNIST data set (a) and test loss on CT Slices data set (b) versus different numbers of training rounds in mixed untargeted attack scenarios, with a selection probability of 0.8 in the baselines.

TABLE III
PREDICTION ACCURACY OF DIFFERENT METHODS VERSUS THREE SOURCE → TARGET CLASS SETTINGS

Methods	$4 \to 9$	$8 \to 2$	$0 \to 1$
FedAvg	82.12	84.60	84.07
UDP-FL	81.72	83.78	82.86
Multi-Krum	83.15	85.26	84.42
BIT-FL	**86.52**	**86.57**	**86.57**

and testing datasets. This observation highlights BIT-FL's ability to select superior local models for aggregation, thus enhancing the performance of the global model. Furthermore, compared to UDP-FL, the noise added in BIT-FL to preserve model privacy incurs minimal accuracy loss, while the noise added in UDP-FL leads to a noticeable degradation in performance.

In the presence of model attacks, it is demonstrated that both BIT-FL and Multi-Krum have capability to achieve secure and reliable model aggregation. Consistent with the assumptions in [1], the local data of workers in classification and regression tasks are independently and identically distributed. Therefore, in BIT-FL, for any honest trainer τ_i, her local models are computed based on (3), i.e.,

$$\boldsymbol{\theta}_i^t = \arg\min_{\boldsymbol{\theta}} F_i(\boldsymbol{\theta}, \mathcal{D}_i). \tag{19}$$

Therefore, for any malicious trainer $\tau_{i'}$ and her local model $\boldsymbol{\theta}_{i'}$, the condition

$$\mathbb{E}\left[e_i\right] = \mathbb{E}\left[F_i(\boldsymbol{\theta}_i^t, \mathcal{D}_i)\right] < \mathbb{E}\left[F_i(\boldsymbol{\theta}_{i'}, \mathcal{D}_i)\right] = \mathbb{E}\left[e_{i'}\right], \tag{20}$$

is satisfied, where e_i and e_i' are the scores of honest trainer τ_i and malicious trainer τ_i', respectively. This signifies that BIT-FL does not incorporate $\boldsymbol{\theta}_{\tau_{i'}}$ into global aggregation. Therefore, BIT-FL achieves robustness against both aforementioned model attacks.

Fig. 6 shows the experimental results in the mixed untargeted attack compromising random-gradient attack and sign-flipping attack, where the selection probability is 0.8 in the baselines. We set the proportion of malicious trainers to 15%. It is evident that, in the presence of model attacks, both FedAvg and UDP-FL fail to complete the training process on the MNIST dataset. On the CT Slices dataset, they exhibit significant performance degradation compared to BIT-FL. Experimental results demonstrate that, on both regression and classification tasks, BIT-FL exhibits robustness against untargeted model attacks, achieving higher accuracy and lower loss on test set compared to Multi-Krum.

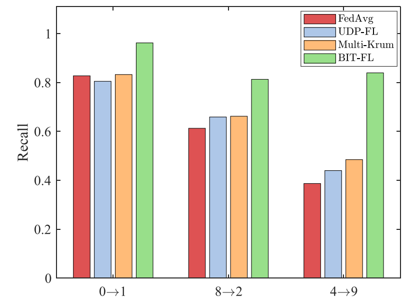


Fig. 7. Source class recall for three different source → target class settings on the MNIST dataset, with the proportion of malicious workers set to 30%.

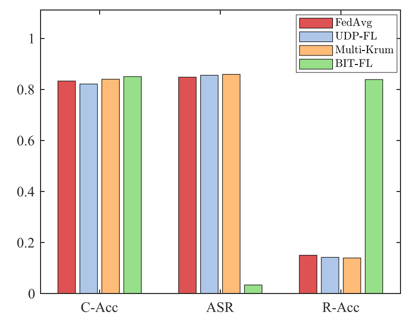
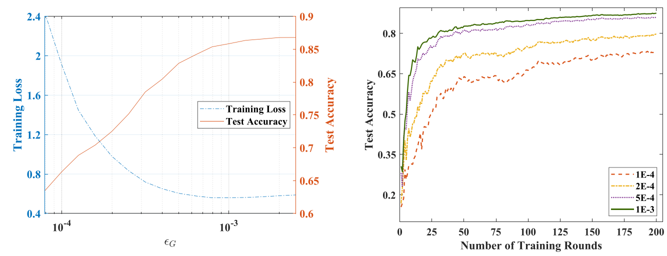


Fig. 8. Experimental results of three evaluation metrics under backdoor attack on MNIST data set.

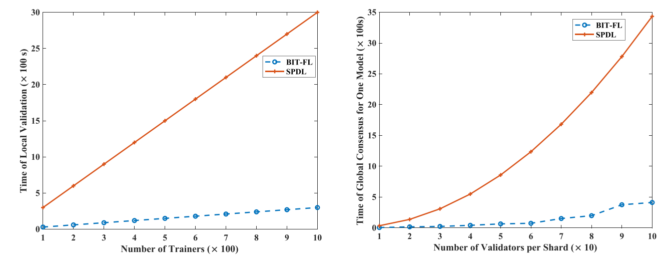
Fig. 7 and Table III show the experimental results in the adversarial poisoning attack scenario. We utilize the method in [40] to select three sets of source → target class pairings (i.e., $0 \to 1$, $8 \to 2$, and $4 \to 9$), and change the source label to the target label of the malicious worker's training data. We maintain all parameters consistent with those in the untargeted attack experiments, except for setting the proportion of malicious workers to 30%. The experimental results demonstrate that BIT-FL exhibits remarkably high source class recall rate and prediction accuracy under all three different settings, indicating that BIT-FL is effective against such model attacks, while three other baselines are significantly affected.

Fig. 8 shows the experimental results in a backdoor attack scenario, where the pattern backdoor strategy from [46] is employed. In comparison to the adversarial poisoning attack, we set the sample poisoned rate of malicious workers to 30%, while keeping other experimental parameters unchanged. We evaluate the prediction accuracy using the *clean accuracy (C-Acc)*, *attack success rate (ASR)*, and *robust accuracy (R-Acc)* metrics as provided in [41]. ASR represents the prediction accuracy of poisoned samples for the target class, while R-Acc represents the prediction accuracy of poisoned samples for the original class. In Fig. 8, from the attacker's perspective, it can be observed that the clean accuracy in all four methods is not affected by the backdoor attack, indicating its stealthiness. However, from the defender's perspective, the high attack success rate, coupled with the low robust accuracy, highlights the vulnerability of the model aggregation approach to the influence of such attacks. The experimental results illustrate that even in the presence of highly stealthy backdoor attacks, BIT-FL still exhibits remarkable robustness against model attacks.



(a) (b)

Fig. 9. (a) Training loss and test accuracy on MNIST data set versus different values of ϵ_G in BIT-FL, where the number of workers is 80. (b) Test accuracy on MNIST data set versus four typical values of ϵ_G in the training process of BIT-FL, where the number of workers is 80.



(a) (b)

Fig. 10. (a) Time delay for local validation with $H = 10$ shards, where the number of trainers increases from 100 to 1000. (b) Time delay of global consensus with 10 shards for selection one local model, where the number in each shard increases from 10 to 100.

To investigate the impact of ϵ_G on TraPro, we vary its values within the range of $[10^{-4}, 10^{-2.5}]$, while keeping the number of trainers and training rounds fixed at 80 and 200, respectively, using the MNIST dataset. As outlined in Definition 1, an increase in ϵ_G heightens the risk of privacy breaches for workers. Conversely, higher values of ϵ_G correlate with reduced training loss and improved test accuracy. These observations are illustrated in Fig. 9. While the addition of artificial noise to local models aims to safeguard individual privacy, it often results in accuracy loss. According to Theorem 2, increasing ϵ_G decreases the variance σ^2 of artificial Gaussian noise, implying that smaller noise levels are more likely to be added to local models, typically leading to higher training accuracy. Conversely, decreasing ϵ_G reduces training accuracy. Furthermore, Theorem 3 indicates that larger ϵ_G values result in a smaller gap between training loss and optimal loss, typically indicating higher training accuracy. Thus, with guidance from Theorems 2 and 3, we can tailor the Gaussian noise distribution by selecting an appropriate ϵ_G to strike a balance between privacy preservation and training accuracy.

E. Validation Procedure

Finally, we demonstrate the time delay of the Validation Procedure (ValPro). In Fig. 10(a), it illustrates the time taken for local validation. It can be observed that with the increasing number of trainers, the validation time increases linearly. This is because different validators take the same amount of time to validate each local model, regardless of variations in computing power among devices. BIT-FL divides the validators in the blockchain network into many small shards so that model validation in different shards can be implemented in parallel. In contrast, the validation of each model in SPDL requires all validators, leading to significant time delays. Similarly, as depicted in Fig. 10(b), the consensus in BIT-FL begins from a small shard and involves one more shard only when they cannot make a decision. However, SPDL requires all validators to vote for the final decision, resulting in serious time delays.

TABLE IV
CONSENSUS TIME OF SELECTING ONE LOCAL MODEL OVER 1000 EPOCHS FOR DIFFERENT METHODS

Metric (s)	PBFT $\lfloor(\bar{\kappa}-1)/3\rfloor$	GearBox $\lfloor(\bar{\kappa}-1)/2\rfloor$	Reticulum $\lfloor(\bar{\kappa}-1)/3\rfloor$	BIT-FL $\lfloor(\bar{\kappa}-1)/2\rfloor$
Avg	12743.852	1986.034	162.363	**52.528**
Max	19602.623	2106.471	**486.261**	11488.738
Min	8712.532	1718.628	82.416	**31.245**

At the conclusion of the evaluation, we further demonstrate the superior performance of the proposed loop-based sharded consensus integrated into the global consensus of ValPro. As depicted in Table IV, BIT-FL exhibits the best performance compared to the baselines. This superiority can be attributed to several factors. First, Gearbox modifies parameters I and S, enabling the utilization of small shards when the proportion of adversarial nodes aiming to disrupt liveness is low. However, when this proportion increases during runtime, shards can be regenerated after losing liveness with larger shard sizes, increased I, and decreased S. Despite this, since Gearbox only enlarges dead shards without altering the number of shards, overlaps between shards may occur. The process of reconstructing a shard is time-consuming, and frequent adjustments triggered by adversaries can halt the system. Second, Reticulum relies on a two-layer sharding design that does not adjust shard memberships to restore liveness. Consequently, it involves more validators in each loop. Additionally, as a synchronous communication protocol, Reticulum can practically withstand only $\bar{\alpha} = \lfloor(\bar{\kappa}-1)/3\rfloor$ adversaries.

Apart from the minimum and average consensus times, the maximum consensus time of BIT-FL is significantly longer than those of Reticulum and Gearbox. Unlike BIT-FL, Reticulum not only has process shards where validators carry out the voting process but also control shards where validators handle the finalization. Similar to the process shards, each finalization in Reticulum involves only one control shard, with each shard comprising 200 validators. In contrast, BIT-FL's finalization process requires the participation of all 4,000 validators, which naturally leads to longer consensus times. Moreover, as outlined in Algorithm 3, each validator in BIT-FL has ς assistants to record their vote, enabling the network to promptly identify and expel malicious validators by cross-referencing the vote summaries from these assistants. However, to explore the maximum consensus time, we relaxed the expulsion rules: if an adversary engages in only one malicious act during τ consensus loops, where τ is a predefined parameter, they are not expelled from the network. This tolerance of malicious behavior necessitates more consensus loops and introduces complex cross-shard communication, ultimately resulting in a longer maximum consensus time for BIT-FL compared to Gearbox.

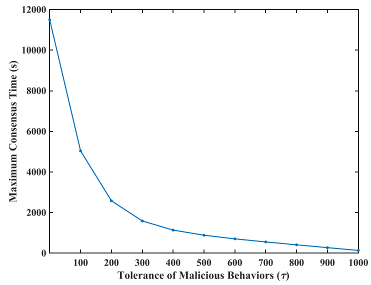


Fig. 11. Maximum consensus time of BIT-FL versus different tolerance τ of malicious behaviors.

Finally, we further investigate the impact of expelling malicious validators. As previously mentioned, each validator has assistants that enable the immediate detection and expulsion of validators engaging in Byzantine behavior. To study the effect of this expulsion process, we allow each malicious validator to perform one Byzantine action within τ voting processes. This means a validator will only be expelled from the network if they commit more than one Byzantine attack during the τ voting processes. In our experiment, we vary the value of τ to observe its effect on the corresponding consensus time.

As shown in Fig. 11, the maximum consensus time decreases as the tolerance τ increases. Tolerance τ represents the number of malicious behaviors that can be overlooked during the consensus process. A higher τ means fewer malicious actions are tolerated. Specifically, when τ approaches infinity, validators who submit dishonest votes are immediately detected and expelled, leading to fewer adversaries and shorter consensus times, which aligns with the requirements of BIT-FL. Conversely, when $\tau = 1$ dishonest validators are not expelled, resulting in more adversaries and longer consensus time.

VIII. Conclusion

In this paper, we proposed a novel incentivized and secure BFL framework called BIT-FL. This framework effectively incentivizes workers to participate in training tasks while guaranteeing the privacy of their cost information. We also established provable privacy guarantees for local models in BIT-FL by incorporating meticulously designed Gaussian noise. To efficiently select high-quality local models for global aggregation, BIT-FL integrates a validation procedure where validators can assess model quality in parallel using a loop-based sharded consensus algorithm. We proved that the proposed consensus algorithm ensures the same level of security as non-sharded consensus protocols with synchronous communication. Furthermore, the designed noise enables tight upper bounds on the excess empirical risk of BIT-FL. Experimental results consistently demonstrate that BIT-FL outperforms existing methods, confirming its superior performance.

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.*, PMLR, 2017, pp. 1273–1282.

[2] Z. Wang, Y. Huang, M. Song, L. Wu, F. Xue, and K. Ren, "Poisoning-assisted property inference attack against federated learning," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 4, pp. 3328–3340, Jul./Aug. 2023.

[3] Q. Chen, Z. Wang, H. Wang, and X. Lin, "FedDual: Pair-wise gossip helps federated learning in large decentralized networks," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 335–350, 2022.

[4] D. Yang, Z. Cheng, W. Zhang, H. Zhang, and X. Shen, "Burst-aware time-triggered flow scheduling with enhanced multi-CQF in time-sensitive networks," *IEEE/ACM Trans. Netw.*, vol. 31, no. 6, pp. 2809–2824, Dec. 2023, doi: 10.1109/TNET.2023.3264583.

[5] D. Yang et al., "DetFed: Dynamic resource scheduling for deterministic federated learning over time-sensitive networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 5162–5178, May 2024, doi: 10.1109/TMC.2023.3303017.

[6] W. Zhang et al., "Optimizing federated learning in distributed industrial IoT: A multi-agent approach," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3688–3703, Dec. 2021.

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

[8] 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.

[9] J. Zhu, J. Cao, D. Saxena, S. Jiang, and H. Ferradi, "Blockchain-empowered federated learning: Challenges, solutions, and future directions," *ACM Comput. Surv.*, vol. 55, no. 11, pp. 1–31, 2023.

[10] Y. Qu, M. P. Uddin, C. Gan, Y. Xiang, L. Gao, and J. Yearwood, "Blockchain-enabled federated learning: A survey," *ACM Comput. Surv.*, vol. 55, no. 4, pp. 1–35, 2022.

[11] M. Xu, Z. Zou, Y. Cheng, Q. Hu, D. Yu, and X. Cheng, "SPDL: A blockchain-enabled secure and privacy-preserving decentralized learning system," *IEEE Trans. Comput.*, vol. 72, no. 2, pp. 548–558, Feb. 2023.

[12] H. Lin, K. Chen, D. Jiang, L. Shou, and G. Chen, "Refiner: A reliable and efficient incentive-driven federated learning system powered by blockchain," *VLDB J.*, vol. 33, no. 3, pp. 807–831, 2024.

[13] R. Jin, J. Hu, G. Min, and J. Mills, "Lightweight blockchain-empowered secure and efficient federated edge learning," *IEEE Trans. Comput.*, vol. 72, no. 11, pp. 3314–3325, Nov. 2023.

[14] L. Cui, X. Su, and Y. Zhou, "A fast blockchain-based federated learning framework with compressed communications," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 12, pp. 3358–3372, Dec. 2022.

[15] L. Feng, Y. Zhao, S. Guo, X. Qiu, W. Li, and P. Yu, "BAFL: A blockchain-based asynchronous federated learning framework," *IEEE Trans. Comput.*, vol. 71, no. 5, pp. 1092–1103, May 2022.

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

[17] C. Ying, H. Jin, X. Wang, and Y. Luo, "Double insurance: Incentivized federated learning with differential privacy in mobile crowdsensing," in *Proc. 2020 Int. Symp. Reliable Distrib. Syst.*, 2020, pp. 81–90.

[18] M. Abadi et al., "Deep learning with differential privacy," in *Proc. 2016 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.

[19] R. Xu, S. R. Pokhrel, Q. Lan, and G. Li, "FAIR-BFL: Flexible and incentive redesign for blockchain-based federated learning," in *Proc. 51st Int. Conf. Parallel Process.*, 2022, pp. 1–11.

[20] L. Gao, L. Li, Y. Chen, C. Xu, and M. Xu, "FGFL: A blockchain-based fair incentive governor for federated learning," *J. Parallel Distrib. Comput.*, vol. 163, pp. 283–299, 2022.

[21] Z. Wang, Q. Hu, R. Li, M. Xu, and Z. Xiong, "Incentive mechanism design for joint resource allocation in blockchain-based federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 5, pp. 1536–1547, May 2023.

[22] X. Wang, Y. Zhao, C. Qiu, Z. Liu, J. Nie, and V. C. Leung, "InFEDge: A blockchain-based incentive mechanism in hierarchical federated learning for end-edge-cloud communications," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 12, pp. 3325–3342, Dec. 2022.

[23] Y. Xu et al., "BESIFL: Blockchain empowered secure and incentive federated learning paradigm in IoT," *IEEE Internet of Things J.*, vol. 10, no. 8, pp. 6561–6573, Apr. 2023.

[24] K. Wei et al., "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, 2020.

[25] T. Rückel, J. Sedlmeir, and P. Hofmann, "Fairness, integrity, and privacy in a scalable blockchain-based federated learning system," *Comput. Netw.*, vol. 202, 2022, Art. no. 108621.

[26] P. K. Sharma, P. Gope, and D. Puthal, "Blockchain and federated learning-enabled distributed secure and privacy-preserving computing architecture for IoT network," in *Proc. 2022 IEEE Eur. Symp. Secur. Privacy Workshops*, 2022, pp. 1–9.

[27] X. Qu, S. Wang, Q. Hu, and X. Cheng, "Proof of federated learning: A novel energy-recycling consensus algorithm," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 8, pp. 2074–2085, Aug. 2021.

[28] 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.

[29] K. Wei et al., "User-level privacy-preserving federated learning: Analysis and performance optimization," *IEEE Trans. Mobile Comput.*, vol. 21, no. 9, pp. 3388–3401, Sep. 2022.

[30] C. Xie, S. Koyejo, and I. Gupta, "Zeno: Robust fully asynchronous SGD," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2020, pp. 10495–10503.

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

[32] C. Che, X. Li, C. Chen, X. He, and Z. Zheng, "A decentralized federated learning framework via committee mechanism with convergence guarantee," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4783–4800, Dec. 2022.

[33] T. Jiang, G. Shen, C. Guo, Y. Cui, and B. Xie, "BFLS: Blockchain and federated learning for sharing threat detection models as cyber threat intelligence," *Comput. Netw.*, vol. 224, 2023, Art. no. 109604.

[34] A. Ahmad, W. Luo, and A. Robles-Kelly, "Robust federated learning under statistical heterogeneity via Hessian-weighted aggregation," *Mach. Learn.*, vol. 112, no. 2, pp. 633–654, 2023.

[35] S. Li, E. Ngai, and T. Voigt, "Byzantine-robust aggregation in federated learning empowered Industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1165–1175, Feb. 2023.

[36] J. Hong, H. Wang, Z. Wang, and J. Zhou, "Federated robustness propagation: Sharing adversarial robustness in heterogeneous federated learning," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 7893–7901.

[37] H. Lycklama, L. Burkhalter, A. Viand, N. Küchler, and A. Hithnawi, "RoFL: Robustness of secure federated learning," in *Proc. 2023 IEEE Symp. Secur. Privacy*, 2023, pp. 453–476.

[38] C. Dwork et al., "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3/4, pp. 211–407, 2014.

[39] X. Zhang, G. Xue, R. Yu, D. Yang, and J. Tang, "Keep your promise: Mechanism design against free-riding and false-reporting in crowdsourcing," *IEEE Internet of Things J.*, vol. 2, no. 6, pp. 562–572, Dec. 2015.

[40] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Proc. 25th Eur. Symp. Res. Comput. Secur.*, Guildford, U.K., Springer, 2020, pp. 480–501.

[41] B. Wu et al., "BackdoorBench: A comprehensive benchmark of backdoor learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 10546–10559.

[42] Y. Xu, J. Zheng, B. Düdder, T. Slaats, and Y. Zhou, "A two-layer blockchain sharding protocol leveraging safety and liveness for enhanced performance," in *Proc. 2024 Netw. Distrib. Syst. Secur. Symp.*, 2024, pp. 1–20.

[43] H. Jin, L. Su, B. Ding, K. Nahrstedt, and N. Borisov, "Enabling privacy-preserving incentives for mobile crowd sensing systems," in *Proc. IEEE 36th Int. Conf. Distrib. Comput. Syst.*, 2016, pp. 344–353.

[44] B. David, B. Magri, C. Matt, J. B. Nielsen, and D. Tschudi, "GearBox: Optimal-size shard committees by leveraging the safety-liveness dichotomy," in *Proc. 2022 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2022, pp. 683–696.

[45] F. Graf, H. Kriegel, M. Schubert, S. Poelsterl, and A. Cavallaro, "Relative location of CT slices on axial axis data set," *UCI Mach. Learn. Repository*, 2011. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Relative+location+of+CT+slices+on+axial+axis

[46] T. Gu, B. Dolan-Gavitt, and S. Garg, "BadNets: Identifying vulnerabilities in the machine learning model supply chain," 2017, *arXiv: 1708.06733*.

[47] A. Archer and E. Tardos, "Truthful mechanisms for one-parameter agents," in *Proc. 42nd IEEE Symp. Found. Comput. Sci.*, 2001, pp. 482–491.

[48] A. Gupta, K. Ligett, F. McSherry, A. Roth, and K. Talwar, "Differentially private combinatorial optimization," in *Proc. 21st Annu. ACM-SIAM Symp. Discrete Algorithms*, SIAM, 2010, pp. 1106–1125.

[49] T. Van Erven and P. Harremos, "Rényi divergence and Kullback-Leibler divergence," *IEEE Trans. Inf. Theory*, vol. 60, no. 7, pp. 3797–3820, Jul. 2014.

[50] M. Bun and T. Steinke, "Concentrated differential privacy: Simplifications, extensions, and lower bounds," in *Proc. Theory Cryptogr. Conf.*, Springer, 2016, pp. 635–658.

[51] H. Lin, J. Mairal, and Z. Harchaoui, "Catalyst acceleration for first-order convex optimization: From theory to practice," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 7854–7907, 2018.

[52] Z. Allen-Zhu and Y. Yuan, "Improved SVRG for non-strongly-convex or sum-of-non-convex objectives," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2016, pp. 1080–1089.

[53] S. Das, T. Yurek, Z. Xiang, A. Miller, L. Kokoris-Kogias, and L. Ren, "Practical asynchronous distributed key generation," in *Proc. IEEE Symp. Secur. Privacy*, 2022, pp. 2518–2534.

[54] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," in *Proc. 41st Annu. Symp. Found. Comput. Sci.*, 2000, pp. 565–574.

[55] J. Hellings and M. Sadoghi, "ByShard: Sharding in a Byzantine environment," *VLDB J.*, vol. 32, no. 6, pp. 1343–1367, 2023.

Chenhao Ying received the BE degree from the Department of Communication Engineering, Xidian University, China, in 2016, and the PhD degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China, in 2022. He is a research assistant professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His current research interests include mobile crowd sensing, blockchain, and quantum computing.

Fuyuan Xia received the BE degree from the Nanjing University of Posts and Telecommunications, in 2022. He is currently working toward the PhD degree with the Department of Computer Science and Technology, Shanghai Jiao Tong University. His research interests include federated learning, privacy preserving, and blockchain.

David S. L. Wei (Life Senior Member) received the PhD degree in computer and information science from the University of Pennsylvania, in 1991. Presently, he is a full professor with the Computer and Information Science Department, Fordham University. From 1993 to 1997, he worked as an associate professor and then full professor with the Faculty of Computer Science and Engineering, University of Aizu, Japan. He has published more than 150 technical papers in various archival journals and conference proceedings, covering areas such as distributed and parallel processing, wireless networks and mobile computing, optical networks, peer-to-peer communications, cloud and edge computing, cybersecurity, and quantum computing and communications. He has served as a program committee member and a session chair for several well-known international conferences, including Infocom. He was also a lead guest editor of *IEEE Journal on Selected Areas in Communications* for the SI on *Mobile Computing and Networking*, *Networking Challenges in Cloud Computing Systems and Applications*, Leveraging Machine Learning in SDN/NFV-based Networks, a guest editor of *IEEE Journal on Selected Areas in Communications* for the SI on *Peer-to-Peer Communications and Applications*, a lead guest editor of *IEEE Transactions on Cloud Computing* for the SI on Cloud Security, and a lead guest editor of *IEEE Transactions on Big Data* for the SI on Edge Analytics in the Internet of Things. Additionally, he served as an associate editor of *IEEE Transactions on Cloud Computing* from 2014 to 2018, an editor of *IEEE Journal on Selected Areas in Communications* for the Series on Network Softwarization and Enablers from 2018 to 2020, and an associate editor of *Journal of Circuits, Systems and Computers* from 2013 to 2018. He received the IEEE Region 1 Technological Innovation Award (Academic), in 2020 for his contributions to information security in wireless and satellite communications and cyber-physical systems. He is a member of ACM and AAAS and is a life senior member of IEEE Computer Society, and IEEE Communications Society. Currently, His research is focused on cloud and edge computing, cybersecurity, and quantum computing and communications.

Xinchun Yu (Member, IEEE) received the PhD degree from the Department of Computer Science and Engineering, Shanghai JiaoTong University, Shanghai, China. He is currently a research scientist with Tsinghua-Berkeley Shenzhen Institute. His research interests include information theory, coding theory, statistical inference and their applications in communication networks.

Yibin Xu is currently working toward the PhD degree with the Department of Computer Science, University of Copenhagen. His research interests include blockchain.

Weiting Zhang (Member, IEEE) received the PhD degree in communication and information systems with Beijing Jiaotong University, Beijing, China, in 2021. From 2019 to 2020, he was a visiting PhD student with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Starting from December 2021, he works as an associate professor with the School of Electronic and Information Engineering, Beijing Jiaotong University. His research interests include industrial Internet of Things, federated learning, edge intelligence, and machine learning for network optimization.

Xikun Jiang received the PhD degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China, in 2023. Presently, she holds a postdoctoral position with the Department of Computer Science (DIKU), University of Copenhagen (UCPH), Denmark. Her research interests include mobile crowdsensing, online markets, and machine learning.

Haiming Jin (Member, IEEE) received the BS degree from Shanghai Jiao Tong University, Shanghai, China, in 2012, and the PhD degree from the University of Illinois at UrbanaChampaign (UIUC), Urbana, Illinois, in 2017. He is currently a tenure-track associate professor with the John Hopcroft Center for Computer Science and the Department of Electronic Engineering, Shanghai Jiao Tong University. Before this, he was a post-doctoral research associate with the Coordinated Science Laboratory, UIUC. His research interests include crowd and social sensing systems, reinforcement learning, and mobile pervasive and ubiquitous computing.

Yuan Luo (Member, IEEE) received the BS degree in applied mathematics, and the MS and PhD degrees in probability statistics from Nankai University, Tianjin, China, in 1993, 1996, and 1999, respectively. From 1999 to 2001, he was a post-doctoral researcher with the Institute of Systems Science, Chinese Academy of Sciences, Beijing, China. From 2001 to 2003, he was a post-doctoral researcher with the Institute for Experimental Mathematics, University of Duisburg–Essen, Essen, Germany. Since June 2003, he has been with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. Since 2006, he has been a full professor and the vice director of the Department, from 2016 to 2018 and since 2021. His current research interests include coding theory, information theory, and Big Data analysis.

Tao Zhang (Fellow, IEEE) has been leading research, product development, and corporate strategies for more than 30 years. He is now managing the Transformational Networks and Services Research Group with the US National Institute of Standards and Technology (NIST). He was the chief technology officer for the Smart Connected Vehicles Business Unit with Cisco Systems, and the chief scientist for mobile and vehicular networking with Telcordia Technologies (formerly Bell Communications Research). He cofounded three industry consortia, including the OpenFog Consortium. He holds more than 100 patents awarded globally and coauthored two books on vehicular networks and all-IP (Internet Protocol) mobile networks. He has cofounded several successful international conferences. He served as the chief information officer and a board governor of the IEEE Communications Society, a distinguished lecturer of the IEEE Vehicular Technology Society, and an editor or guest editor for numerous journals or their special issues. He is a fellow of the Society for Information Reuse and Integration, and a fellow of the Asia-Pacific Artificial Intelligence Academy.

Dacheng Tao (Fellow, IEEE) received the PhD degree from the University of London, London, U.K., in 2007. He is an Australian Laureate fellow, a professor of computer science, and a Peter Nicol Russell chair with the School of Computer Science; an advisor and chief scientist of the Digital Sciences Initiative; and the founding director of the Sydney AI Centre, Faculty of Engineering, The University of Sydney, Darlington, NSW, Australia. His research is detailed in one monograph and more than 200 publications in prestigious journals and proceedings with prominent conferences, such as *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *IEEE Transactions on Image Processing*, *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Intelligent Transportation Systems*, *International Journal of Computer Vision*, *Journal of Machine Learning Research*, NeurIPS, ICML, ICLR, CVPR, ICCV, ECCV, AAAI, IJCAI, ICDM, and ACM SIGKDD, with several best paper awards. He received the 2015 and 2020 Australian Eureka Prize, the 2018 IEEE ICDM Research Contributions Award, and the 2021 IEEE Computer Society McCluskey Technical Achievement Award. He is a fellow of the Australian Academy of Science, the Royal Society of NSW, TWAS, AAAS, and ACM.