






Defending Against Poisoning Attacks in Federated Learning With Blockchain

Nanqing Dong , Zhipeng Wang , Jiahao Sun , Michael Kampffmeyer ,
William Knottenbelt , and Eric Xing, *Fellow, IEEE*

Abstract—In the era of deep learning, federated learning (FL) presents a promising approach that allows multiinstitutional data owners, or clients, to collaboratively train machine learning models without compromising data privacy. However, most existing FL approaches rely on a centralized server for global model aggregation, leading to a single point of failure. This makes the system vulnerable to malicious attacks when dealing with dishonest clients. In this work, we address this problem by proposing a secure and reliable FL system based on blockchain and distributed ledger technology. Our system incorporates a peer-to-peer voting mechanism and a reward-and-slash mechanism, which are powered by on-chain smart contracts, to detect and deter malicious behaviors. Both theoretical and empirical analyses are presented to demonstrate the effectiveness of the proposed approach, showing that our framework is robust against malicious client-side behaviors.

Impact Statement—FL has been a promising solution to utilize multisite data while preserving users' privacy. Despite the success of integrating blockchain with FL to decentralize global model aggregation, the protection of this integration from clients with malicious intent in federated scenarios remains unclear. This article presents the first formulation of this problem, and the proposed stake-based aggregation mechanism shows robustness in detecting malicious behaviors. The results in this work not only pose a new research direction in FL but can also benefit a wide variety of applications such as finance and healthcare.

Index Terms—Blockchain, deep learning, federated learning (FL), trustworthy machine learning.

Manuscript received 7 October 2023; revised 20 December 2023 and 10 February 2024; accepted 8 March 2024. Date of publication 18 March 2024; date of current version 9 July 2024. This work was supported in part by FLock.io under the FLock Research Grant. This article was recommended for publication by Associate Editor Kaitai Liang upon evaluation of the reviewers' comments. (*Corresponding author: Nanqing Dong.*)

Nanqing Dong is with Shanghai Artificial Intelligence Laboratory, Shanghai 200232, China (e-mail: dongnanqing@pjlab.org.cn).

Zhipeng Wang and William Knottenbelt are with the Department of Computing, Imperial College London, SW7 2AZ London, U.K. (e-mail: zhipeng.wang20@imperial.ac.uk; w.knottenbelt@imperial.ac.uk).

Jiahao Sun is with FLock.io, WC2H 9JQ London, U.K. (e-mail: sun@flock.io).

Michael Kampffmeyer is with the Department of Physics and Technology, UiT The Arctic University of Norway, 9019 Tromsø, Norway (e-mail: michael.c.kampffmeyer@uit.no).

Eric Xing is with the Machine Learning Department, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA, and also with Mohamed bin Zayed University of Artificial Intelligence, Masdar City, Abu Dhabi, UAE (e-mail: epxing@cs.cmu.edu).

Digital Object Identifier 10.1109/TAI.2024.3376651

I. INTRODUCTION

NOWADAYS, machine learning (ML), or more specifically, deep learning, has transformed a broad spectrum of industries, ranging from finance to healthcare. In current ML paradigms, training data are first collected and curated, and then ML models are optimized by minimizing certain loss criteria on the training data. A common underlying assumption in the learning environment is that the training data can be instantly accessed or easily distributed across computing nodes without communication constraints, i.e., data are *centralized*.

However, in a system with multiple *clients* (i.e., data holders), to ensure data centralization, clients have to upload local data to a centralized device (e.g., a central server) to conduct the centralized training described above. Despite the success of centralized training in various deep learning applications [1], [2], [3], there is growing concern about data privacy and security, especially when the local data held by the clients are private or contain sensitive information. Especially, to ensure data governance, strict data regulations have been established [4], [5].

To address the aforementioned concern, federated learning (FL) has been proposed [6]. In a typical FL system, a central server [7] is responsible for aggregating and synchronizing model weights, while a set of clients manipulate multisite data. This facilitates data governance, as clients only exchange model weights or gradients with a central server instead of uploading local data to the central server, and has led to FL becoming a standardized solution to utilize multisite data while preserving privacy.

Although FL perfectly implements *data decentralization*, a trustworthy central server is required in the system. In such a system design, the central server in fact has privileges over clients, as the central server determines the global aggregation and synchronization. If the central server is compromised or manipulated by a malicious party, the clients are vulnerable if the central server intentionally distributes problematic model updates. This can potentially increase the cost of system management and maintenance. Toward avoiding this single point of failure, many efforts have been made to decentralize the central server, and one particularly promising solution is to use a blockchain as decentralized storage [8].

Originally proposed for cryptocurrencies, a blockchain is a distributed ledger that can record the state transition information among multiple parties [9], [10], without relying on a

centralized server. Blockchain technology has gained widespread attention for its potential to revolutionize a variety of industries, such as finance [9], healthcare [11], and supply chain management [12]. By leveraging the decentralized nature of the blockchain, FL can benefit from increased security, privacy, and efficiency, as well as reduced reliance on centralized servers [13]. Concretely, in FL with blockchain, each client participating in the learning process uploads their local model updates to the blockchain, where they are stored in *blocks*, the metadata of a blockchain system. These blocks are then used to aggregate the local model updates into a global model, which can be downloaded by the clients. The use of blockchain *smart contracts* [9], which are computer programs triggered by blockchain events, ensures that the global aggregation process is performed automatically and transparently, without the need for human intervention or centralized control.

Although integrating blockchain with existing FL systems can partially solve the threat to the central server, it cannot guarantee the quality of uploaded model updates from the clients. That is to say, blockchain-enabled FL systems are still vulnerable to client-side malicious attacks [14]. In this work, we define malicious behaviors as actions that intentionally decrease the learning performance (e.g., accuracy and convergence) of the global model via poisoning attacks (such as data poisoning [15] or model poisoning [14]). Instead of hacking the central server, the attackers can sabotage the FL systems by manipulating the clients. This work focuses on defending against client-side poisoning attacks. One solution is to combine blockchain-enabled FL with cryptographic protocols, such as fully homomorphic encryption (FHE) [16] and secure multiparty computation (SMPC) [17], to mitigate malicious behaviors from the client side. However, the adoption of these intricate cryptographic protocols introduces significant computational overhead for FL participants, thus impairing the system performance. Besides, the malicious clients can still attack the system without breaching the protocols. It is challenging to address malicious behaviors without substantially compromising the efficiency of a blockchain-based FL system.

We propose a generic framework that can integrate an FL system with a blockchain system and can defend against poisoning attacks without adopting complex cryptographic protocols. The proposed defense mechanism is motivated by *proof-of-stake* (PoS) [18], a *consensus mechanism* in blockchain, and *The Resistance* [19], a role-playing board game. PoS has an incentive mechanism that encourages honest behaviors by *rewarding* it and punishes dishonest behaviors via *slashing*. *The Resistance*, on the other hand, has two mismatched competing parties, where the party with a larger size is denoted as the resistance force and the other party is denoted as the spies. In *The Resistance*, there is a voting mechanism where, in each round, each player conducts independent reasoning and votes for a player, and the player with the highest votes will be deemed as a “spy” and kicked out of the game. The goal of the resistance force is to vote out all the spies while the spies aim to impersonate the resistance force and survive until the end. Based on these two concepts, this work proposes a novel majority-voting mechanism for global aggregation where each

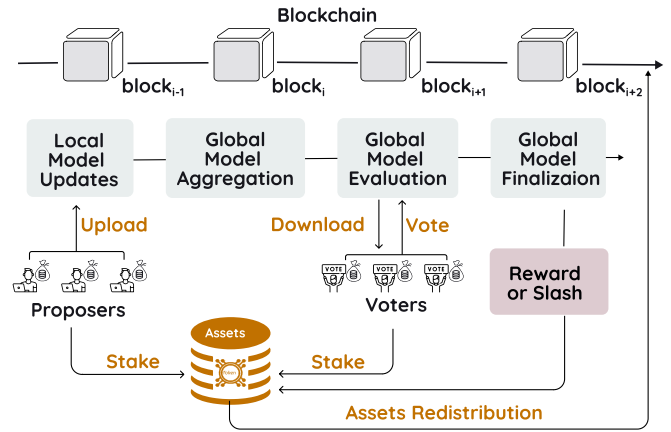


Fig. 1. Stake-based aggregation mechanism for FL with blockchain. In each round, the proposers are randomly selected from the participating clients to perform local training and upload local updates to the blockchain. Then, voters download the aggregated local updates from the blockchain, perform local validation, and vote for acceptance or rejection. If the majority of voters vote for accepting the global aggregation, the global model will be updated, and the proposers and the voters who vote for acceptance will be rewarded. Conversely, if the majority of voters vote for rejection, the global model will not be updated, and the proposers and the voters who vote for acceptance will be slashed.

participating client independently validates the quality of aggregated local updates and votes for acceptance of the global update. The aggregation mechanism is stake-based where participating clients stake assets¹ or *tokens* (a quantitative measurement of the asset, which can be used to indicate the trustworthiness of the client in our system) for their own actions. There are two types of actions, proposing (uploading local updates) and voting. If the majority vote is to accept the global aggregation, a proposer will be refunded with its staked tokens and a voter who votes for acceptance will not only be refunded but also be rewarded with the staked tokens from the voters who vote for rejection and vice versa. The overall procedure of the stake-based aggregation mechanism is illustrated in Fig. 1. To the best of our knowledge, this is the first work that integrates the majority voting and incentive mechanisms in the FL and blockchain literature.

We evaluate the proposed framework on a practical financial problem, namely loan default prediction. We simulate the FL and blockchain environment for the Lending Club Kaggle challenge dataset and ChestX-ray14 dataset [20] to conduct experiments in a controllable setting and to provide insights into the problem of interest. We empirically show that an FL system can maintain robust performance under malicious attacks by introducing the proposed stake-based aggregation mechanism.

The contributions of this work are summarized as follows:

- 1) We formulate the problem of decentralized FL with blockchain in the presence of poisoning attacks.
- 2) For the first time, we introduce a novel stake-based aggregation mechanism designed to fortify FL systems against poisoning attacks. In contrast to prior solutions, our mechanism boasts the distinct advantage of seamless

¹In practice, the staked assets can be linked with cryptocurrency or real currency to increase the financial cost of malicious attacks.

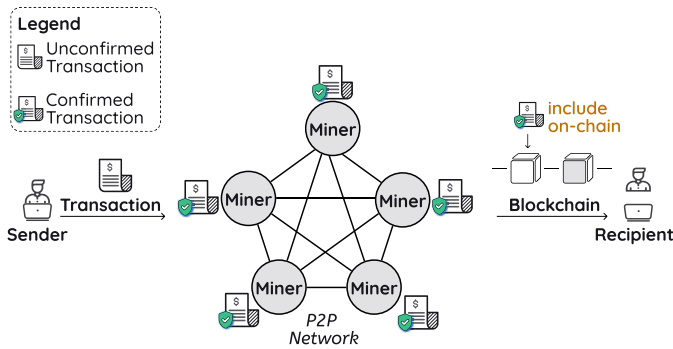


Fig. 2. Blockchain workflow overview. The sender broadcasts the issued transaction to the P2P network, which will be confirmed by the miners. The confirmed transaction will be stored on a public blockchain and can be read by the recipient. Blockchain miners typically adopt a consensus mechanism to achieve an agreement on the state of the blockchain.

integration into any blockchain enabled with smart contracts, all without necessitating alterations to the foundational consensus structure of the underlying blockchain. This approach not only enhances security but also simplifies accessibility, rendering it a more user-friendly option for FL participants.

- 3) We evaluate the robustness of the proposed framework in a simulated environment and provide initial empirical insights into the problem of interest. The findings show evidence that stake-based FL is an under-explored research problem with potential advantages compared with existing FL paradigms in terms of defending against poisoning attacks.

The rest of the article is organized as follows. Section II reviews the related work on FL and blockchain computing. Section III formulates the problem of interest, defines the key concepts, and lists the necessary assumptions. Section IV presents the method and theoretical result. Sections V and VI provide the experimental details on two different setups. Section VII concludes this work.

II. RELATED WORK

In this section, we review the recent progress on blockchain-based FL, and highlight the difference between the proposed method and existing studies.

A. Blockchain

Blockchains refer to distributed ledgers that operate on a global peer-to-peer (P2P) network, as exemplified by popular cryptocurrencies such as Bitcoin [27] and Ethereum [9]. Users can freely join or leave the blockchain system, without a central authority in place to ensure common agreement on the distributed ledgers. Instead, users rely on consensus protocols [18], [28], such as proof-of-work (PoW) or PoS, to achieve agreement in a distributed setting.

As shown in Fig. 2, a blockchain transaction typically involves a sender who transfers digital assets, such as

cryptocurrencies, to a recipient. The sender authorizes the transaction with a digital signature combining transaction details and their private key. The transaction is then broadcasted over a P2P network to *miners*, who are participants in the network responsible for verifying and adding new blocks of transactions to the blockchain. Miners validate and confirm the transaction using consensus protocols, to ensure that the transaction is legitimate and not a duplicate or fraudulent transaction. Once confirmed, the transaction is added to a block, which is then linked to the previous block using cryptographically hash functions [29], forming a chain of blocks (i.e., blockchain). The block is then propagated to all the participants in the network, creating a decentralized, immutable record of the transaction. The combination of cryptography and consensus protocols enhances the security, transparency, and decentralization of transactions, underscoring blockchain’s potential across various applications [11], [12], [30], [31], [32].

Another key feature of blockchain technology is the use of smart contracts [9], which are quasi-Turing-complete programs that can be executed within a virtual machine. When a transaction is initiated, a smart contract is typically used to encode the terms and conditions of the transaction, such as the amount, currency, and time of transfer. The smart contract is then stored on the blockchain network and executed automatically when the predefined conditions are met.

B. Federated Learning With Blockchain

Traditional FL faces challenges [33], [34], [35], such as privacy and security concerns (e.g., poisoning attacks), unreliable communication, and difficulty in reaching a consensus among the parties. Blockchain, on the other hand, provides a decentralized, secure, and transparent platform for data storage and sharing. This makes the use of blockchain for FL a promising direction to potentially address privacy and security concerns by allowing parties to keep their data private while still contributing to the training process. Additionally, blockchain can provide a secure communication channel for FL participants and ensure the integrity of the FL process.

Current blockchain-based FL designs [8], [35], [36], [37], [38] have been broadly used in diverse fields [39], [40], [41]. For example, Ma et al. [25] propose a blockchain-assisted decentralized FL (BLADE-FL) framework, to prevent malicious clients from poisoning the learning process. Li et al. [13] analyze the impact of lazy clients on the learning performance of BLADE-FL and propose optimization for minimizing the loss function. Cui et al. [22] propose a blockchain-based decentralized and asynchronous FL framework for anomaly detection in IoTs by using a model named DP-GAN. Qu et al. [26] introduce a committee-based blockchain consensus algorithm for decentralized FL to prevent the single point of failure and poisoning attacks in FL.

Despite the potential benefits of combining FL with blockchain, several challenges remain. For instance, FL systems are still vulnerable to client-side malicious attacks [14] and lack incentive-compatible mechanisms to motivate FL participants

TABLE I
COMPARISON OF EXISTING BLOCKCHAIN-BASED FL SOLUTIONS

Framework	Technique	Incentive Mechanism	Cryptocurrency Incentive	Blockchain Agnostic	Attack Mitigation
[16]	FHE	✗	✗	-	✓
[17]	SMPC	✗	✗	-	✓
[21]	DP	✗	✗	-	✓
[22]	DP-GAN	✗	✗	-	✓
[23]	Incentive mechanism	✓	✓	-	✗
[24]	Reputation + reverse auction	✓	✓	-	✗
[25]	Blockchain + reward only	✓	✓	✗	✓
[13]	Committee consensus	✓	✓	✗	✓
[26]	Committee consensus	✓	✓	✗	✓
Ours	Majority vote + reward & slash	✓	✓	✓	✓

to behave honestly during the training process. For instance, Miao et al. [16] leverage cosine similarity and blockchain to counteract poisoning attacks and penalize malicious clients. Their designs also rely on cryptographic techniques such as FHE. Kalapaaking et al. [17] adopt SMPC to enhance the security of blockchain-based FL which can mitigate poisoning attacks for healthcare systems. Yan et al. [21] integrate differential privacy (DP) techniques with blockchain to mitigate the issues of a single point of failure or untrusted aggregation caused by a malicious central server in privacy-preserving FL. However, using FHE, SMPC, or DP, instead of incorporating incentive mechanisms, may introduce an additional computation burden for clients or the server.

Several incentive mechanisms [23], [24] have recently been proposed to encourage participants and enhance model accuracy in blockchain-based FL. However, it remains unclear how to effectively utilize the blockchain infrastructure and leverage its inherent incentive mechanism (i.e., cryptocurrencies) to incentivize trustworthy FL behaviors and penalize malicious clients. Furthermore, to thwart potential malicious activities, such as poisoning attacks, existing blockchain-based FL solutions [8], [13], [25], [26], [36] have pointed out that participants can engage in both the FL training process and the block validation in PoS-based blockchains or mining activities in PoW-based blockchains. These design choices not only raise the entry barrier for regular users wishing to partake in blockchain-based FL systems but also add complexity to the fundamental consensus mechanism.

In this work, we introduce a PoS-based reward-and-slash mechanism for the FL system. We compare our solution with existing work in Table I. Our solution can be seamlessly integrated into any smart-contract-enabled blockchain without requiring modifications to the underlying consensus design (i.e., our design is blockchain agnostic). This approach facilitates the participation of any FL users, making it more accessible and user-friendly.

III. PROBLEM FORMULATION

This section introduces the problem of interest, the definition of the malicious behaviors considered, and the underlying assumptions in this work. The main definitions and notations adopted in this work are summarized in Table II.

TABLE II
SUMMARY OF MAIN DEFINITIONS AND NOTATIONS

Notation	Description
Blockchain-based FL	FL framework with a blockchain architecture
FedAVG	Federated Averaging [6], a FL algorithm
Non-IID	Nonindependently and identically distributed
\mathcal{K}	Set of participating clients at round t
K_v^t	The selected set of voters
\mathcal{K}_p^t	Set of proposers at round t
a^t	Majority voting decision at round t
M_k	Asset of client k
γ_p	Staked tokens for proposing
$pool_p$	Pool for storing proposers' stake

A. Setup

There are $K > 1$ clients in a federated system. Let $\mathcal{K} = \{1, 2, \dots, K\}$ denote the set of all clients. Let \mathcal{D}_k denote the local data stored in client k , we have $\mathcal{D}_k \cap \mathcal{D}_l = \emptyset$ for $k \neq l$ and $k, l \in \mathcal{K}$. Each local dataset \mathcal{D}_k can be randomly split into a training set and a test set, which are both private to client k . In addition to K clients, a blockchain plays the role of a parameter server [7] for global aggregation. Let f_θ be the model of interest. In the parameter server, the parameter set θ_0^0 is randomly initialized at round 0 and K clients download θ_0^0 from the blockchain as K local copies $\{\theta_k^0\}_{k=1}^K$ for full synchronization. During the federated optimization phase, a set of \mathcal{K}_p^t clients is randomly selected for round t . For each $k \in \mathcal{K}_p^t$, the client k updates θ_k^{t-1} by training on the training set of \mathcal{D}_k independently for a number of local epochs. Then, the blockchain aggregates updated $\{\theta_k^t\}_{k \in \mathcal{K}}$ collected from all the K clients to update θ_0^t . The K clients then synchronize with the parameter server, i.e., $\theta_k^t \leftarrow \theta_0^t$. To facilitate data governance, as required in among others the medical domain [4], [5], we assume that the patient's data (either raw data or encoded data) in a client can not be uploaded to the blockchain or other clients, i.e., only parameters $\{\theta_k\}_{k=0}^K$ and *metadata* (e.g., the statistics of data) [42], [43] can be exchanged between the blockchain and the clients. It is worth mentioning that this work focuses on the interactions between FL and blockchain, where blockchain computing (or *mining*, in a more fashionable sense) and the application of additional

privacy-preserving techniques [44] are considered orthogonal research directions and thus beyond the scope of this work.

B. Malicious Behaviors

The definition of malicious behavior in this work is an action that intentionally decreases the global model performance. There are two types of actions for each client that interact with the federated system, i.e., a client can propose (i.e., be a *proposer*) and vote (i.e., be a *voter*). Proposing is to upload local model or gradient updates to the parameter server, while voting is a peer-review process to validate the “virtually” aggregated model updates. The technical details of the two actions are described in Section IV. There are thus two corresponding malicious behaviors. The first malicious behavior is to propose harmful local model updates and the second one is to vote dishonestly. More specifically, in the second case, a client votes for approval when it is aware that the proposed model updates are poisoned and votes for rejection when there is no evidence that indicates that the proposed model updates are poisoned. It is worth mentioning that the clients themselves might not intentionally attack the FL system as they can be compromised by attackers. For simplicity, we define the clients that have malicious behaviors as *malicious clients* in this work, denoted as \mathcal{K}_m . We use η to denote the ratio of malicious clients among all clients, i.e., $\eta = (|\mathcal{K}_m|/|\mathcal{K}|)$, where $|\cdot|$ is the cardinality of a set.

C. Assumptions

There are six important assumptions in this work.

- 1) A1: The goal of malicious behaviors is to decrease the global model performance. This is also reflected in Section III-B. Under these assumptions, behaviors that are harmful to the system but do not influence the global model performance are beyond the scope of discussion in this work. An example is eavesdropping, i.e., cloning the model specifications.
- 2) A2: All clients are rational. This means that both honest and malicious clients expect to maximize their gain or minimize their loss while achieving their goals.
- 3) A3: Following previous studies on blockchain [13], we assume that η is strictly smaller than 50%. This means there are always more honest clients than malicious clients in a federated system.
- 4) A4: There is no capacity constraint on the hardware, including computing, communication, and storage, allowing us to solely focus on the algorithmic side of the problem.
- 5) A5: The underlying blockchain of the FL system of interest is running securely with a consensus protocol that ensures the validity and integrity of transactions and blocks. While the security of the blockchain is crucial for the overall security of the FL system, addressing the malicious miners falls outside the scope of this study.

IV. METHOD

In this section, we first introduce the basics of federated aggregation in Section IV-A, and describe the local

validation and majority voting in Sections IV-B and IV-C. We then propose a novel incentive mechanism in Section IV-D. A theoretical analysis on malicious voting is presented in Section IV-E. We then describe the whole training pipeline in Section IV-F. Finally, the analysis of computational cost is provided in Section IV-G.

A. Federated Aggregation

In this work, we illustrate the proposed framework in the context of the seminal FL method, FedAVG [6]. At the end of round t , the local models $\{\theta_k^t\}_{k=1}^K$ are uploaded and aggregated as a weighted average

$$\theta_0^t = \sum_{k=1}^K a_k \theta_k^t \quad (1)$$

where $a_k = (n_k/N)$. The metadata $n_k = |\mathcal{D}_k|$ is the number of local training examples stored in client k , and $N = \sum_{k=1}^K n_k$ is the total number of training examples in the K clients.

B. Local Validation

In contrast to standard FL algorithms, the aggregated global model is not recorded in a block directly. Instead, θ_0^t , a copy of θ_0^t is downloaded by a randomly selected set of clients, denoted as *voters*, \mathcal{K}_v^t . A voter k runs a local inference with θ_0^t on its local test set and outputs a local validation score. The local validation score s_k^t is a scalar, which can be linked with common metrics of ML tasks.² If s_k^t is not lower than a threshold, the voter votes for accepting this aggregated model; otherwise, the voter votes against it. The threshold can be based on a validation score s_k^{t-1} acquired in the previous round. In the training of ML tasks, the scores can be volatile due to the characteristics of the tasks. Thus, a hyperparameter $\epsilon \in (0, 1)$ is introduced to control the tolerance of performance decrease in a single round. Mathematically, the k th voter has the following score:

$$v_k^t = \begin{cases} 1, & s_k^t \geq (1 - \epsilon)s_k^{t-1} \\ -1, & s_k^t < (1 - \epsilon)s_k^{t-1}. \end{cases} \quad (2)$$

It is worth mentioning that the likelihood of the attackers consistently manipulating the scores by fooling all the randomly selected voters (e.g., via adversarial attacks [45]) diminishes quickly toward zero as the number of epochs increases. According to A4, the majority of voters are honest. It is thus difficult to attack (either via data poisoning or model poisoning) as the validation set of each client is private.

C. Majority Voting

The majority voting process for whether to apply the global aggregation operation at round t can be described as follows. Here, we use a binary variable a^t to denote the decision

$$a^t = \begin{cases} 1, & \sum_{k \in \mathcal{K}_v} v_k > 0 \\ -1, & \sum_{k \in \mathcal{K}_v} v_k \leq 0. \end{cases} \quad (3)$$

²For example, common evaluation metrics include accuracy for classification, mean intersection over union (mIOU) for semantic segmentation, and mean average precision (mAP) for object detection.

Algorithm 1 Reward-and-slash design for a set of randomly selected proposers.

a^t : Majority voting decision at round t
 \mathcal{K} : Set of participating clients at round t
 \mathcal{K}_p^t : Set of proposers at round t
 M_k : Asset of client k
 γ_p : Staked tokens for proposing
 $pool_p$: Pool for storing proposers' stake
1: **if** $a^t == -1$ **then**
2: **for** $k \in \mathcal{K}_p^t$ **do**
3: **if** $M_k \geq \gamma_p$ **then**
4: $M_k \leftarrow M_k - \gamma_p$
5: $pool_p \leftarrow pool_p + \gamma_p$
6: **else**
7: $pool_p \leftarrow pool_p + M_k$
8: $M_k \leftarrow 0$
9: $\mathcal{K}^t \leftarrow \mathcal{K}^t \setminus \{k\}$
10: **else**
11: **if** $pool_p > 0$ **then**
12: **for** $k \in \mathcal{K}_p^t$ **do**
13: $M_k \leftarrow M_k + \frac{pool_p}{|\mathcal{K}_p^t|}$
14: $pool_p \leftarrow 0$

If $a^t = 1$, the global aggregation will be finalized and recorded in the block; otherwise, the global aggregation will be discarded.

D. Asset Redistribution

As there are two independent actions, there are two parallel reward-and-slash designs for proposing and voting. For both actions, the randomly selected proposers and voters are required to stake a fixed sum of tokens before they act. If some of these actors fail to stake (they do not have enough tokens left), they lose their access to the blockchain and are removed from the FL system permanently. Proposers will be rewarded with tokens accumulated in an independent pool (if there are any tokens left in the pool) if the global aggregation is approved and lose their stakes if the global aggregation is rejected. The reward-and-slash design for the proposers is illustrated in Algorithm 1. For the voters, the majority party will not only take back their stakes but also be rewarded with the staked tokens lost by the minority party. The reward-and-slash design for the voters is illustrated in Algorithm 2. In the following section, Section IV-E, we demonstrate that under the proposed design and assumptions in Section III-C, malicious voters have no incentive to make dishonest votes. Note, Section IV-D highlights the key difference between the proposed voting mechanism and traditional majority voting as the voting is directly linked with the incentive mechanism.

E. Theoretical Analysis on Malicious Votes

In this section, we theoretically show that malicious voters in the proposed framework have no incentive to make dishonest votes.

Algorithm 2 Reward-and-slash design for a set of randomly selected voters.

a^t : Majority voting decision at round t
 \mathcal{K} : Set of participating clients at round t
 \mathcal{K}_v^t : Set of voters at round t
 \mathcal{K}_m^t : Set of voters at round t with $v_k^t == a^t$
 M_k : Asset of client k
 γ_v : Staked tokens for voting
 $pool_v$: Pool for storing voters' stake
1: **for** $k \in \mathcal{K}_v^t \setminus \mathcal{K}_m^t$ **do**
2: **if** $M_k \geq \gamma_v$ **then**
3: $M_k \leftarrow M_k - \gamma_v$
4: $pool_v \leftarrow pool_v + \gamma_v$
5: **else**
6: $pool_v \leftarrow pool_v + M_k$
7: $M_k \leftarrow 0$
8: $\mathcal{K}^t \leftarrow \mathcal{K}^t \setminus \{k\}$
9: **for** $k \in \mathcal{K}_m^t$ **do**
10: $M_k \leftarrow M_k + \frac{pool_v}{|\mathcal{K}_m^t|}$
11: $pool_v \leftarrow 0$

Theorem 1 (Honest Voting Hypothesis): When all clients are rational, a malicious client should not make a malicious vote.

Proof: Let \mathcal{K}_v denote a randomly selected set of voters and $n_v = |\mathcal{K}_v|$. For client $k \in \mathcal{K}_v$, let $\gamma_v > 0$ denote the staked tokens for voting, i.e., client k must stake γ_v to participate in the voting, otherwise, it will be removed from the system.

Let us consider a multiagent scenario, where malicious clients can collude. No matter how the malicious clients cooperate, there are two types of malicious clients. The first type behaves maliciously to achieve the goal of sabotaging the FL training, i.e., lowering the global performance. The second type acts honestly to hide themselves and survive to be able to implement the complex policy to sabotage the FL training at the a later stage. If the malicious clients belong to the second type, they are factually "honest" ones.

Let r be the ratio of malicious clients in \mathcal{K}_v , there are $r \cdot n_v$ malicious clients in \mathcal{K}_v and $(1 - r) \cdot n_v$ honest clients. If $r \cdot n_v < (1 - r) \cdot n_v$, i.e., $r < 0.5$, each malicious client will lose γ_v ; if $r \cdot n_v > (1 - r) \cdot n_v$, i.e., $r > 0.5$, each malicious client will gain $((1 - r) \cdot n_v \cdot \gamma_v / r \cdot n_v) = (1 - r/r)\gamma_v$. The expected return \mathcal{R} of a malicious client will be as follows:

$$\begin{aligned}
\mathcal{R} &= \int_0^{0.5} -\gamma_v dr + \int_{0.5}^1 \frac{1-r}{r} \gamma_v dr \\
&= -0.5\gamma_v + ((\ln(1) - 1) - (\ln(0.5) - 0.5))\gamma_v \\
&= -(\ln(0.5) + 1)\gamma_v < 0.
\end{aligned} \tag{4}$$

Under A2, each client is rational. As $\mathcal{R} < 0$, in the long run, a malicious client will lose all tokens and be removed from the system. So, a given client has no reason to make a dishonest vote resulting in honest votes by all clients. ■

Theorem 1 will further be empirically validated in Section V-B1.

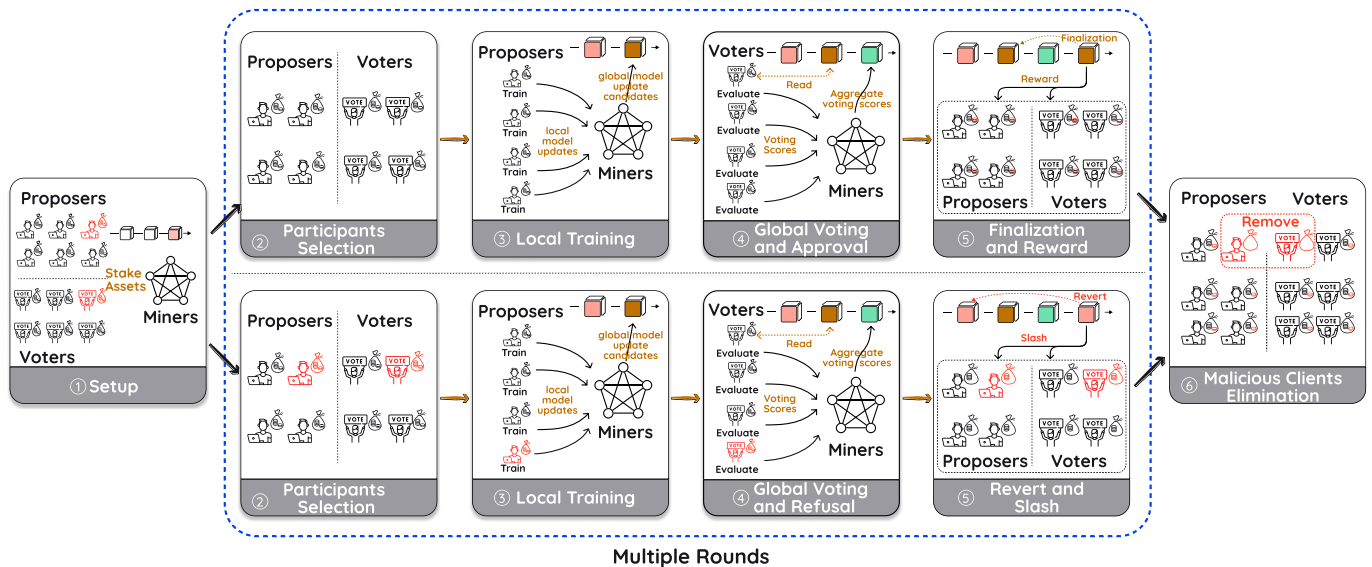


Fig. 3. Round-based training process. In the initial state (indexed as ①), both honest (black) and malicious (red) clients exist in an FL system. In the final state (indexed as ⑥), all malicious clients are expected to be removed from the system. To reach the final state from the initial state, multiple rounds of training are required. Here are two possible scenarios, the proposed aggregation is either approved (the upper branch) or denied (the lower branch) by the voters. In each round (within the dotted blue line), a subset of clients are randomly selected as proposers, and another subset of clients are randomly selected as voters. The proposers and voters interact with the blockchain following the order of orange arrows (from ② to ⑤).

If $A2$ holds, we are certain that the malicious voters will reach a consensus internally before they act to win the majority vote. Intuitively, all malicious voters can be considered as a group together. In this case, this “group” will behave exactly as the single malicious client in Theorem 1 based on the same reasoning. The proof is omitted.

F. Training

Each round consists of the following steps: proposer selection, local training, global aggregation, local validation, majority voting, token redistribution, and block creation (recording *state*³ information). The above steps are repeated in multiple rounds until certain stopping criteria are fulfilled. The complete training process is depicted in Fig. 3. The stopping criteria could be a fixed amount of training epochs, which is commonly adopted in ML.

G. Computation Complexity and Cost

For proposers, the primary computational cost is driven by the local training algorithm, mirroring the structure of traditional FL. For voters, their computation cost stems from evaluating the aggregated global model. In addition to mining blocks, miners engage in further computations by consolidating global model updates, akin to the responsibilities of a centralized aggregator in traditional FL. The overall computational complexity is contingent upon the underlying training network backbone.

The communication and storage costs are outlined as follows. We assume the model size is M . During an epoch, the

communication cost for a client (i.e., a proposer or a voter) is $O(M)$. The storage cost on the blockchain is $O(K \cdot M)$, where K is the number of clients.

V. EXPERIMENTS

A. Experimental Setup

We first evaluate the proposed framework in a simulated environment.

1) *Data and Task*: We consider a standard binary classification task, namely loan default prediction. We use the Kaggle Lending Club dataset⁴ to simulate a realistic financial application scenario. We preprocess the raw dataset by dropping all entries with missing values. For the labels, we only keep “fully paid” and “charged off” to simplify the task as a binary classification task. We randomly select 80% of the data as the training set and use the rest of the data as the test set. The training set is split into K subsets of equal size and distributed across K clients. Within each client, 20% of the local data are randomly selected as the validation set.

2) *Implementation*: There are $K = 50$ clients in the system, and each client is initialized with 64 tokens. We use a three-layer multilayer perceptron (MLP) as the network backbone. Apart from the last layer, each layer of the MLP has 128 hidden nodes. We use a standard Adam [46] optimizer with fixed learning rate 10^{-3} and batch size 128. No data augmentation is applied. We use the binary accuracy as both the local validation score and evaluation metric. In our experiments, malicious clients are randomly selected before the training according to

³For example, the state can record the global model and tokens of each client.

⁴<https://www.kaggle.com/datasets/wordsforthewise/lending-club>

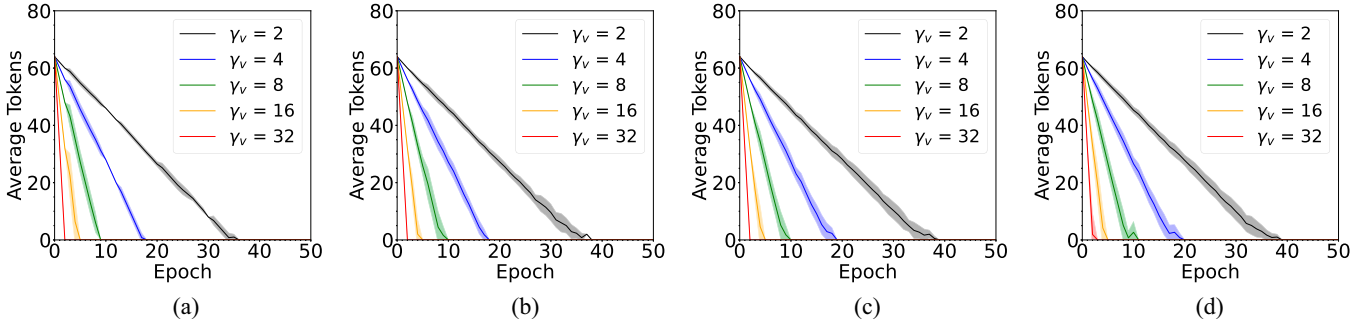


Fig. 4. Token distribution results for malicious voters when all proposers are honest. The malicious voters’ tokens decrease quickly as the number of epochs increases and a large γ_p leads to a high decreasing rate. This empirically validates our proof of Theorem 1. The solid line denotes the mean over five runs with different random seeds and the shaded region denotes one standard deviation around the mean. (a) $\eta = 0.1$. (b) $\eta = 0.2$. (c) $\eta = 0.3$. (d) $\eta = 0.4$.

the ratio η . In each training round, if a malicious client is selected as the proposer or voter, it will act maliciously as described in Section IV.5. We consider a simple data poisoning attack [14], where malicious clients are trained to confuse the model. Specifically, the local models are trained to lower the model performance by using the wrong labels, but maintaining a low weight divergence from the aggregated weights from the last round. All baselines are implemented in PyTorch 1.12.1 [47] on one NVIDIA Tesla T4 GPU. We leverage Ethereum smart contracts to deploy our reward-and-slash design in a private blockchain and simulate the training process using the Python library `web3.py`.⁶ We set $\epsilon = 0.05$ based on empirical experience.⁷

3) *Baselines*: So far, there is no such blockchain-based FL baseline suitable for the comparative evaluation of counteracting malicious behaviors in our problem setting. It is worth mentioning that the proposed framework can be integrated with the existing FL method. In our experiments, we use FedAVG as the backbone FL method to illustrate our defending mechanism. We consider four baselines. The first one is an *Oracle* approach, a centralized baseline without malicious attacks. The *Oracle* should provide the upper-bound performance of the experiment. The second one is FedAVG without malicious attacks (denoted as FedAVG w/o mal), which is equivalent to FedAVG under $\eta = 0$ and should provide the upper-bound performance for a decentralized environment. The third one is FedAVG under malicious attacks (denoted as FedAVG w/ mal), where η of clients are malicious. The fourth one is the proposed method, FedAVG with blockchain under malicious attacks (denoted as FedAVG w/ block). For FL baselines, 10% of clients are randomly selected to perform local training at each epoch. For FedAVG w/ block, we simply use the remaining 90% of the clients as voters.

⁵If a malicious client acts honestly, then it will be considered as an honest one and makes no harm to the system.

⁶<https://web3py.readthedocs.io/en/v5/>

⁷We notice that too small ϵ can cause large oscillation, which slows the convergence, and too large ϵ can facilitate the convergence at the expense of decreased detection performance, i.e., the system fails to remove the majority of malicious clients.

B. Results

1) *Empirical Analysis on Malicious Voters*: To empirically validate the theoretical result in Section IV-E, we first simulate a hypothetical scenario where there are only honest proposers. As there are more honest proposers than malicious proposers at each round on average, the effect of malicious weights can be seen as slowing the convergence and decreasing the global performance, which will be validated in Section V-A3. Here, we further simplify the scenario to focus on the behavior of malicious voters. As shown in Fig. 4, given the set of the hyperparameter for slashing voters $\gamma_r = \{2, 4, 8, 16, 32\}$, the malicious voters will be eliminated from the system shortly (i.e., their average tokens decline to 0 within ≈ 40 epochs).

2) *Comparison With Baselines*: Following Theorem 1 and Section V-B1, we now are certain that there will be no *de facto* malicious voters. Thus, in the following experiments, we focus on the scenarios where malicious clients only upload harmful weights but make honest votes. We evaluate the proposed framework against the baselines described in Section V-A3. We provide the learning curves in Fig. 5 and the accuracy for all four approaches after convergence (the mean accuracy of the last 50 epochs) in Table III. The performance of FedAVG w/ block is competitive with FedAVG w/o mal (i.e., $\eta = 0$) and consistently outperforms FedAVG w/ mal. As η increases, the performance of FedAVG w/ mal decreases significantly, with a larger standard deviation and increased instability. In contrast, FedAVG w/ block maintains robust performance, with only slightly lower results compared to FedAVG w/o mal.

3) *Analysis of Token Distributions*: Fig. 6 depicts the average tokens remaining in honest and malicious proposers during the FL training process when $\gamma_p = 8$. We observe that honest proposers gradually accumulate more tokens while malicious proposers own fewer tokens as training progresses. Eventually, most malicious proposers lose the eligibility to participate in staking and are removed from the FL system, as their remaining tokens are insufficient. Looking more closely at the case where few malicious proposers are in the system [Fig. 6(a)], we note that proposals are initially accepted until a revert and slash step is performed around epoch 32 (indicated by the sudden drop in tokens for malicious proposers). As more malicious proposers are in the system [Fig. 6(d)], revert and slash steps

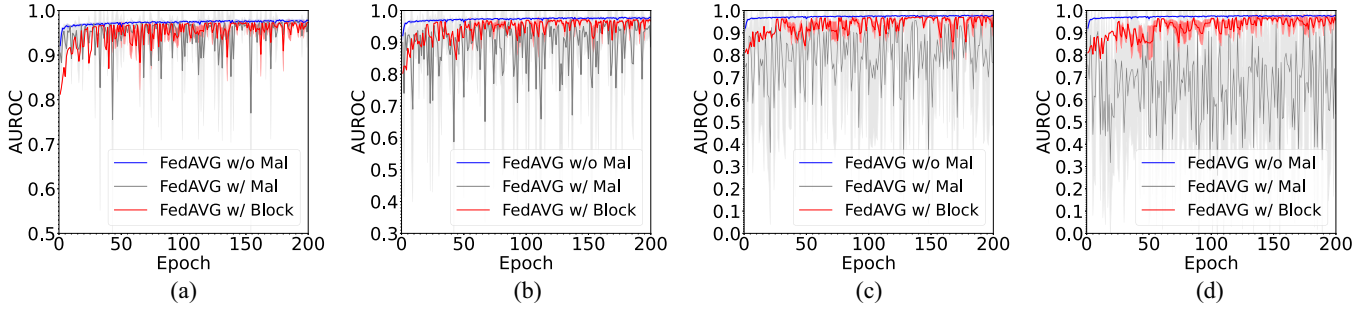


Fig. 5. Federated training under different values of the ratio of malicious clients (η). Each subfigure shows the training AUROCs when the training time (i.e., the number of epochs) increases. The solid lines are the mean AUROCs and the shaded regions are one standard deviation around the means. We compare the performances of FedAVG with blockchain (i.e., w/ Block), FedAVG with malicious clients (i.e., w/ mal), and FedAVG without malicious clients (i.e., w/o mal). We observe that FedAVG w/ Block significantly outperforms FedAVG w/ mal, while being comparable with FedAVG w/o mal, the performance upper bound under this setup. (a) $\eta = 0.1$. (b) $\eta = 0.2$. (c) $\eta = 0.3$. (d) $\eta = 0.4$.

TABLE III
PERFORMANCE COMPARISON UNDER DIFFERENT VALUES OF THE RATIO OF MALICIOUS CLIENTS (η)

Model	$\eta = 0.1$	$\eta = 0.2$	$\eta = 0.3$	$\eta = 0.4$
FedAVG w/ mal	0.963 ± 0.017	0.946 ± 0.034	0.801 ± 0.222	0.709 ± 0.266
FedAVG w/ block (Ours)	0.965 ± 0.008	0.969 ± 0.003	0.952 ± 0.020	0.955 ± 0.021
FedAVG w/o mal	0.975 ± 0.004	0.975 ± 0.004	0.975 ± 0.004	0.975 ± 0.004
Oracle	0.971 ± 0.007	0.971 ± 0.007	0.971 ± 0.007	0.971 ± 0.007

Note: The reported numbers of the performance are mean and standard deviation under five random seeds.

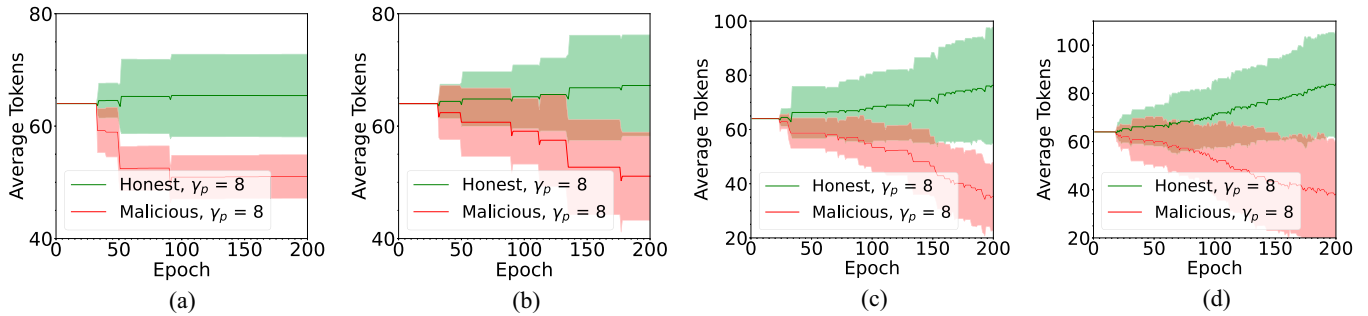


Fig. 6. Token distribution results for clients when setting the parameter for slashing proposers as $\gamma_p = 8$. The expected average token of malicious proposers fluctuates down during the training process. (a) $\eta = 0.1$. (b) $\eta = 0.2$. (c) $\eta = 0.3$. (d) $\eta = 0.4$.

occur more frequently. This is further highlighted by Fig. 7(a) and 7(b), which demonstrates for each epoch if it corresponds to an award or a revert and slash step. Fig. 7(c) and 7(d) further provides the cumulative sum over the number of award and slash epochs, illustrating that the majority of the epochs consist of award epochs and that the fraction of revert and slash episodes increases with the rising fraction of malicious proposers. Finally, we depict the average tokens remaining in honest and malicious proposers under various configurations of γ_p in Figs. 8 and 9. We observe the same phenomenon as Fig. 6, which aligns with the expectations of our system design and reinforces the effectiveness of our approach.

4) *Survival Analysis of Clients*: As shown in Fig. 10, the anticipated survival time of malicious proposers experiences a decrease as γ_p increases. This effect can be attributed to the incentive mechanism in place, whereby a higher value of γ_p results in a greater penalty for proposers who act maliciously.

Fig. 11 shows the survival time of honest proposers under different values of γ_p and exhibits noteworthy behavior. In cases where the malicious ratio η is high, the expected survival time of honest proposers may decrease with a large γ_p . This is due to the fact that, in each epoch, all randomly selected proposers will be slashed if the performance of the aggregated global model does not show improvement. Therefore, it is worth noting that balancing the token slashing parameter γ_p is crucial because setting an excessively high value can harm honest proposers, whereas a small value can lead to slow convergence (see Fig. 12).

5) *Sensitivity to Malicious Client Ratio*: The results presented in Fig. 5 demonstrate the robustness of our proposed method, FedAVG w/ block, against different malicious client ratios, as its performance remains unaffected even under large η values. However, it is important to note that the malicious client ratio can impact the token distribution and survival time of clients. Specifically, when there are more malicious clients

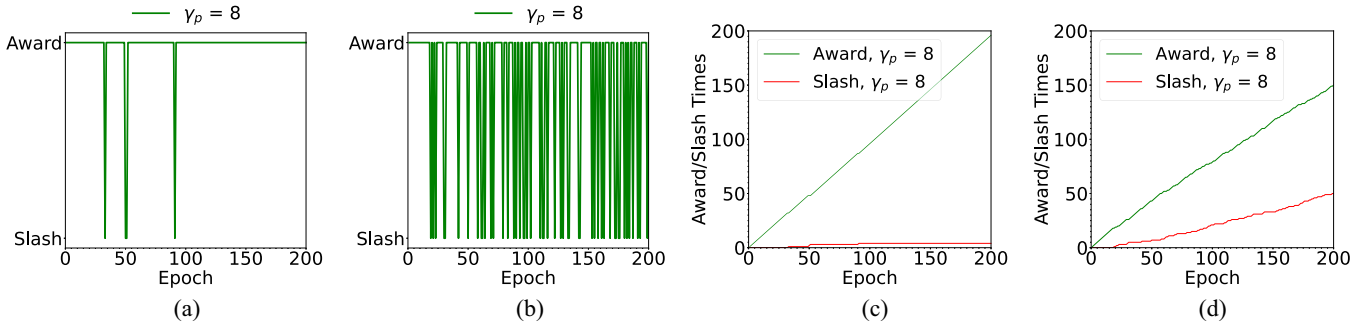


Fig. 7. (a) and (b) Number of award and slash epochs for clients when setting the parameter for slashing proposers as $\gamma_p = 8$. (c) and (d) The number of slash epochs increases when the number of malicious proposers increases. (a) $\eta = 0.1$. (b) $\eta = 0.4$. (c) $\eta = 0.1$. (d) $\eta = 0.4$.

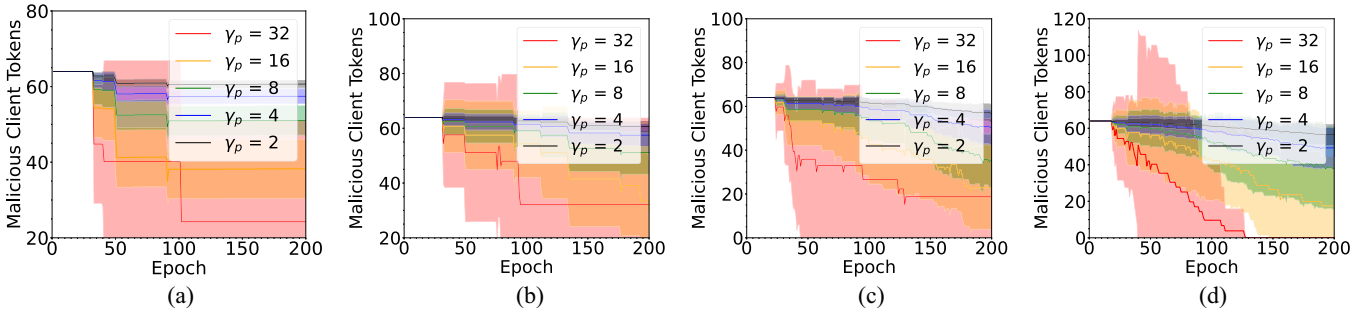


Fig. 8. Token distribution results for malicious clients when choosing $\gamma_p = 2, 4, 8, 16, \text{ and } 32$. The expected average token of malicious proposers exhibits a higher rate of decrease when a large value of γ_p is selected. (a) $\eta = 0.1$. (b) $\eta = 0.2$. (c) $\eta = 0.3$. (d) $\eta = 0.4$.

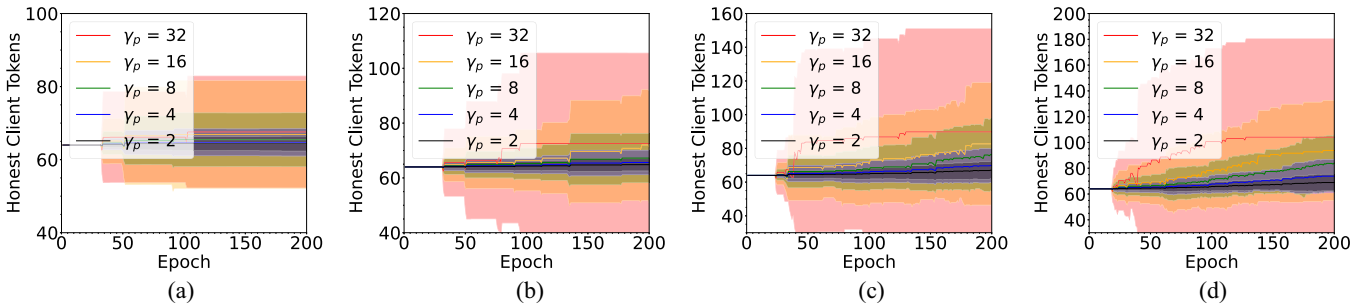


Fig. 9. Token distribution results for honest clients when choosing $\gamma_p = 2, 4, 8, 16, \text{ and } 32$. The expected average token of honest proposers displays a higher rate of growth when a large value of γ_p is selected. (a) $\eta = 0.1$. (b) $\eta = 0.2$. (c) $\eta = 0.3$. (d) $\eta = 0.4$.

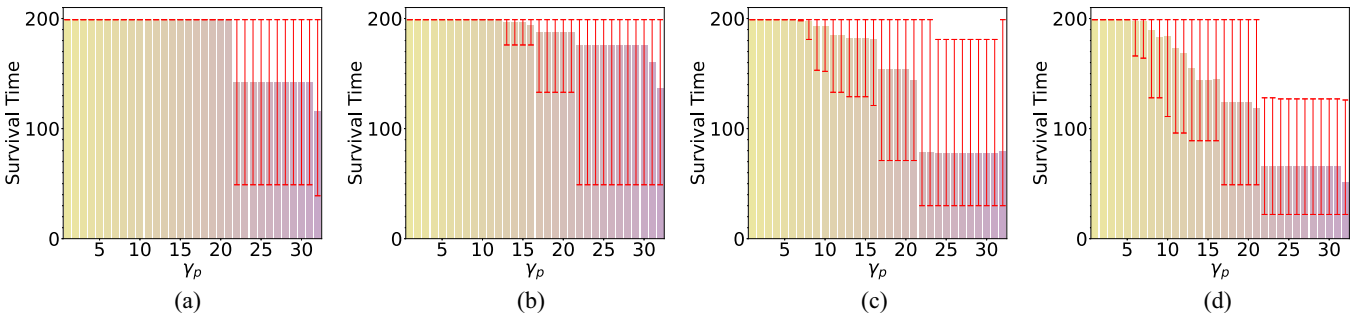


Fig. 10. Malicious proposers survival time with various γ_p . The expected survival time of malicious proposers declines as γ_p increases. (a) $\eta = 0.1$. (b) $\eta = 0.2$. (c) $\eta = 0.3$. (d) $\eta = 0.4$.

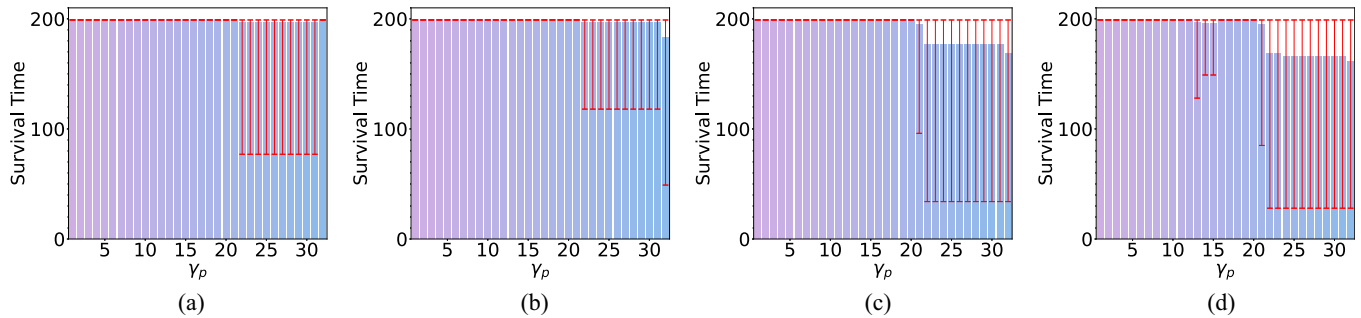


Fig. 11. Honest proposers survival time with various γ_p . A large γ_p can also decrease the expected survival time of honest proposers when the malicious rate η is large. This is because, in each epoch, the randomly selected proposers will be all slashed when the performance of the aggregated global model does not increase. (a) $\eta = 0.1$. (b) $\eta = 0.2$. (c) $\eta = 0.3$. (d) $\eta = 0.4$.

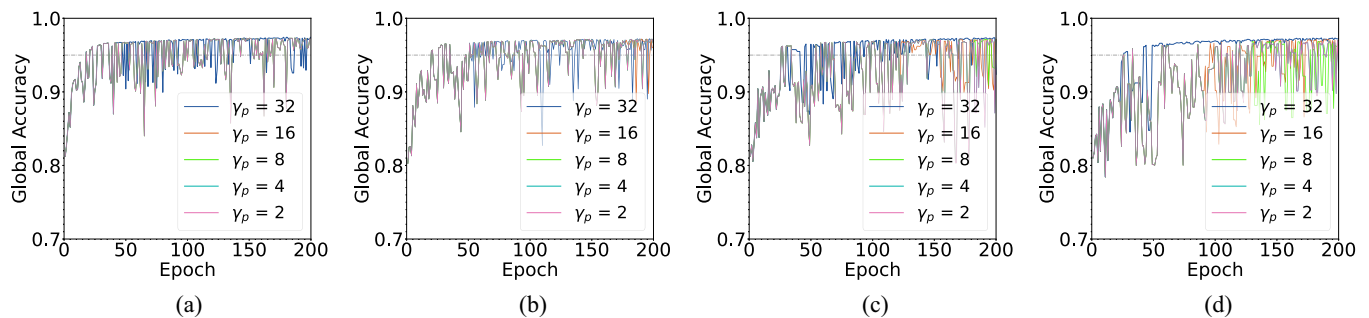


Fig. 12. Global accuracy results when choosing $\gamma_p = 2, 4, 8, 16$, and 32 . The global accuracy is not sensitive to the value of γ_p but the convergence tasks more epochs for larger η . (a) $\eta = 0.1$. (b) $\eta = 0.2$. (c) $\eta = 0.3$. (d) $\eta = 0.4$.

present in the system, honest clients tend to accumulate more assets on average [c.f. Fig. 9(a) and 9(d)]. Nevertheless, they also face a higher risk of being slashed during an epoch, which can ultimately shorten their survival time [c.f. Fig. 11(a)–11(d)].

6) *Limitations*: In this work, as the experimental results aim to evaluate the robustness of the proposed framework, several practical challenges are simplified, e.g., staleness [48], storage, and privacy [44]. Further, the proposed method requires more computational power than traditional methods due to mining (blockchain computing) and voting. Finally, large models have gained in popularity in practical applications, e.g., ViT [49] and GPT-3 [50]. This raises the question of how to efficiently handle on-chain aggregation for large models. Future work thus will aim to address these limitations to facilitate the research and development of FL with blockchain.

C. Cost Analysis

In our experiments, we adopt the Kaggle Lending Club dataset⁸ to simulate a realistic financial application scenario. The total client number is $K = 50$. For each client, the generated model update file is with a size of 587 kB. Therefore, the communication cost incurred by a proposer or a voter is 587 kB and the storage cost on the blockchain is 587×50 kB = 28.66 MB.

VI. ADDITIONAL EXPERIMENTS

In Section V, we examine the performance of the proposed system on a simple learning task on IID data. In this section, we further evaluate the robustness of the proposed system with a more complex task on non-IID data.

A. Experimental Setup

1) *Data and Task*: We consider a standard multilabel classification (MLC) task [51] to simulate a realistic clinical application scenario. We use the ChestX-ray14⁹ dataset [20] and leverage the first 6×10^4 chest X-ray images (CXRs) as our non-IID dataset. We use 80% of the data as the training set and use the rest of the data as the test set. The training set is split into K subsets of equal size and distributed across K clients in a non-IID fashion. Within each client, 20% of local data are randomly selected as the validation set. Because ChestX-ray14 has a long-tailed label distribution, we choose the ten most common diseases to ensure that each client can contain labels for all diseases of interest.

2) *Implementation*: There are $K = 50$ clients in the system and each client is initialized with 64 tokens. Following [51], we use DenseNet121 [52] as the network backbone. We use a standard Adam [46] optimizer with fixed learning rate 10^{-3} and batch size 256. We process each CXR with instance normalization [53] and no data augmentation is applied. We

⁸<https://www.kaggle.com/datasets/wordsforthewise/lending-club>

⁹<https://nihcc.app.box.com/v/ChestXray-NIHCC>

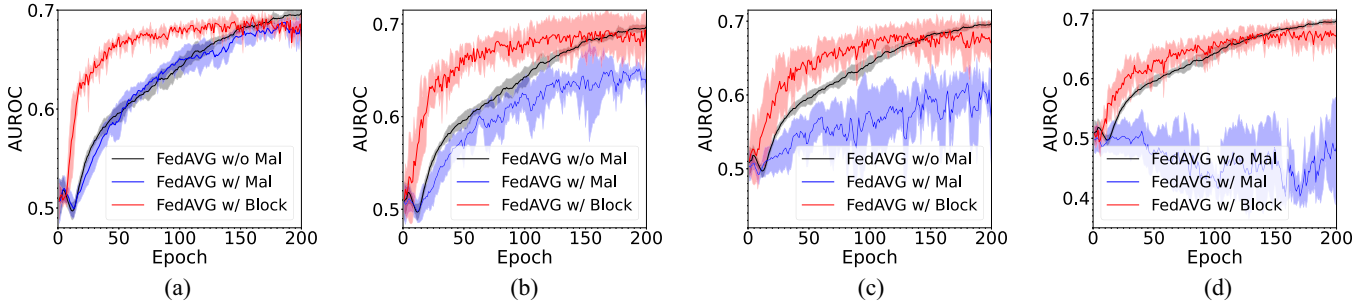


Fig. 13. Federated training under different values of ratio of malicious clients (η). The solid lines are the mean AUROCs and the shaded regions are one standard deviation around the means. (a) $\eta = 0.1$. (b) $\eta = 0.2$. (c) $\eta = 0.3$. (d) $\eta = 0.4$.

TABLE IV
PERFORMANCE COMPARISON UNDER DIFFERENT VALUES OF RATIO OF MALICIOUS CLIENTS (η)

Model	$\eta = 0.1$	$\eta = 0.2$	$\eta = 0.3$	$\eta = 0.4$
FA w/ M	0.677 ± 0.014	0.640 ± 0.012	0.594 ± 0.030	0.483 ± 0.080
FA w/ B	0.683 ± 0.008	0.688 ± 0.024	0.676 ± 0.023	0.673 ± 0.021
FA w/o M	0.695 ± 0.004	0.695 ± 0.004	0.695 ± 0.004	0.695 ± 0.004
Oracle	0.733 ± 0.014	0.733 ± 0.014	0.733 ± 0.014	0.733 ± 0.014

Note: The reported numbers are mean and standard deviation under five random seeds.

use the mean area under the receiver operating characteristic curve over the 10 diseases as both the local validation score and evaluation metric. We set $\epsilon = 0.05$ and $\gamma = 32$ based on empirical experience. The rest of the implementation details follow Section V-A2.

3) *Baselines*: We consider the same four baselines as in Section V-A3.

B. Results

We run each baseline with five random seeds and report both mean and standard deviation under different values of η . The training results are visualized in Fig. 13.

1) *Performance Comparison*: FedAVG w/ block shows competitive performance with FedAVG w/o mal (i.e., $\eta = 0$) and outperforms FedAVG w/ mal consistently. In addition to the learning curves in Fig. 13, we also report the mean AUROCs for all 4 methods after they fully converge in Table IV. When η increases, the performance of FedAVG w/ mal drops significantly and becomes more unstable (i.e., larger standard deviation). FedAVG w/ block remains robust performance and is only slightly lower than FedAVG w/o mal.

2) *Convergence Analysis*: It can be shown that FedAVG w/ block converges faster than both FedAVG w/o mal and FedAVG w/ mal under various values of η . We hypothesize that the proposed global aggregation mechanism can facilitate federated optimization. Intuitively, this can be explained with gradient descent. FedAVG averages gradients optimized for different directions at different clients, which might not be an optimal global gradient. Under malicious attacks, gradients from malicious clients are intentionally optimized away from the optimal direction, which slows down the training process of FedAVG.

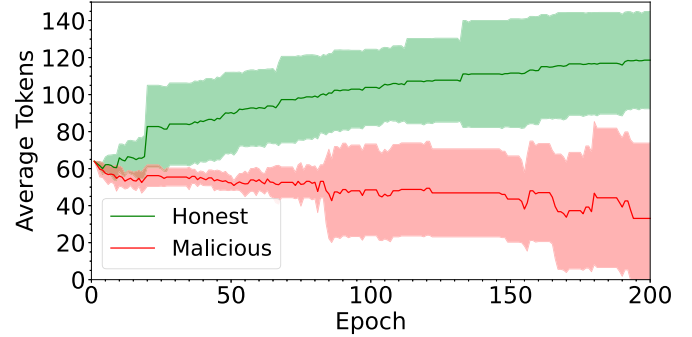


Fig. 14. Tokens of honest and malicious clients when $\eta = 0.3$ and $\gamma = 32$.

However, the proposed consensus mechanism mitigates this issue as it only aggregates when consensus is achieved.

3) *Token Analysis*: As we use tokens to filter out malicious clients, we plot the average tokens left in honest and malicious clients (e.g., Fig. 14). After enough training epochs, honest clients will have more tokens and malicious clients will have fewer tokens. At the end of training, almost all malicious clients do not have enough tokens to stake, i.e., they are removed from the FL system. It is worth mentioning that γ has only trivial effect on the learning performance but large γ can overkill honest clients and small γ can cause slow convergence.

4) *Impact of Non-IID Data*: Due to the non-IID nature of the medical task, the task setup in this section is more complex than the binary classification task in the previous section. In contrast to Section V, there are two important findings. First, the proposed method is robust under the non-IID setup. Second, surprisingly, while the task is more difficult, the performance gain between the proposed method and the baselines becomes larger.

VII. CONCLUSION

In this work, we explore an under-explored research direction, namely using FL and blockchain to defend against poisoning attacks. The defense mechanism is twofold. We use on-chain smart contracts to replace the traditional central server and propose a stake-based majority voting mechanism to detect client-side malicious behaviors. We not only provide a solution

to the problem of interest but also show the robustness of the proposed method and provide the first empirical understanding of the problem. Last but not least, the results of this work suggest that the integration of FL and blockchain is an emerging solution to trustworthy ML. We believe that blockchain can not only play an important role in decentralization and the incentivization of participants for real-world FL applications in fields such as finance and medicine but also can be leveraged to defend against poisoning attacks.

REFERENCES

- [1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
- [2] A. Vaswani et al., "Attention is all you need," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 6000–6010.
- [3] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [4] European Commission, "General data protection regulation," 2016. Accessed: Apr. 25, 2023. [Online]. Available: https://ec.europa.eu/info/law/law-topic/data-protection/data-protection-eu_en
- [5] U.S. Department of Health and Human Services, "Health insurance portability and accountability act," 2017. Accessed: Apr. 25, 2023. [Online]. Available: <https://www.cdc.gov/phlp/publications/topic/hipaa.html>
- [6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, PMLR, 2017, pp. 1273–1282.
- [7] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2014, pp. 19–27.
- [8] 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.
- [9] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.
- [10] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, "When machine learning meets blockchain: A decentralized, privacy-preserving and secure design," in *Proc. IEEE Int. Conf. Big Data*, Piscataway, NJ, USA: IEEE, 2018, pp. 1178–1187.
- [11] L. Soltanisehat, R. Alizadeh, H. Hao, and K.-K. R. Choo, "Technical, temporal, and spatial research challenges and opportunities in blockchain-based healthcare: A systematic literature review," *IEEE Trans. Eng. Manage.*, vol. 70, no. 1, pp. 353–368, Jan. 2023.
- [12] M. M. Queiroz, R. Telles, and S. H. Bonilla, "Blockchain and supply chain management integration: A systematic review of the literature," *Supply Chain Manage.*, vol. 25, no. 2, pp. 241–254, 2020.
- [13] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Netw.*, vol. 35, no. 1, pp. 234–241, Jan./Feb. 2021.
- [14] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, vol. 108, PMLR, 2020, pp. 2938–2948.
- [15] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2020, pp. 480–501.
- [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] A. P. Kalapaaking, I. Khalil, and X. Yi, "Blockchain-based federated learning with SMPC model verification against poisoning attack for healthcare systems," *IEEE Trans. Emerg. Topics Comput.*, vol. 12, no. 1, pp. 269–280, Jan.–Mar. 2024.
- [18] S. Bano et al., "SOK: Consensus in the age of blockchains," in *Proc. ACM Conf. Adv. Financial Technol.*, 2019, pp. 183–198.
- [19] Wikipedia, "The resistance (game)," Accessed: Feb. 12, 2023. [Online]. Available: [https://en.wikipedia.org/wiki/The_Resistance_\(game\)](https://en.wikipedia.org/wiki/The_Resistance_(game))
- [20] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "ChestX-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2097–2106.
- [21] X. Yan, Y. Miao, X. Li, K.-K. Raymond, X. Meng, and R. H. Deng, "Privacy-preserving asynchronous federated learning framework in distributed IoT," *IEEE Internet Things J.*, vol. 10, no. 15, pp. 13281–13291, Aug. 2023.
- [22] L. Cui et al., "Security and privacy-enhanced federated learning for anomaly detection in IoT infrastructures," *IEEE Trans. Ind. Inform.*, vol. 18, no. 5, pp. 3492–3500, May 2022.
- [23] K. Toyoda and A. N. Zhang, "Mechanism design for an incentive-aware blockchain-enabled federated learning platform," in *Proc. IEEE Int. Conf. Big Data*, Piscataway, NJ, USA: IEEE, 2019, pp. 395–403.
- [24] J. Zhang, Y. Wu, and R. Pan, "Incentive mechanism for horizontal federated learning based on reputation and reverse auction," in *Proc. Web Conf.*, 2021, pp. 947–956.
- [25] C. Ma et al., "When federated learning meets blockchain: A new distributed learning paradigm," *IEEE Comput. Intell. Mag.*, vol. 17, no. 3, pp. 26–33, Aug. 2022.
- [26] Y. Qu, C. Xu, L. Gao, Y. Xiang, and S. Yu, "FL-SEC: Privacy-preserving decentralized federated learning using signsgd for the internet of artificially intelligent things," *IEEE Internet Things Mag.*, vol. 5, no. 1, pp. 85–90, Mar. 2022.
- [27] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: <https://ssrn.com/abstract=3440802>
- [28] J. Garay and A. Kiayias, "Sok: A consensus taxonomy in the blockchain era," in *Proc. Topics Cryptol.—CT-RSA: Cryptographers' Track RSA Conf.*, San Francisco, CA, USA, Feb. 24–28, 2020, Cham, Switzerland: Springer, 2020, pp. 284–318.
- [29] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "SOK: Research perspectives and challenges for bitcoin and cryptocurrencies," in *Proc. IEEE Symp. Secur. Privacy*, Piscataway, NJ, USA: IEEE, 2015, pp. 104–121.
- [30] V. Gatteschi, F. Lamberti, C. Demartini, C. Pranteda, and V. Santamaría, "Blockchain and smart contracts for insurance: Is the technology mature enough?" *Future Internet*, vol. 10, no. 2, p. 20, 2018.
- [31] J. Bao, D. He, M. Luo, and K.-K. R. Choo, "A survey of blockchain applications in the energy sector," *IEEE Syst. J.*, vol. 15, no. 3, pp. 3370–3381, Sep. 2021.
- [32] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with IoT. Challenges opportunities," *Future Gener. Comput. Syst.*, vol. 88, pp. 173–190, 2018.
- [33] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends® Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021.
- [34] Y. Wan, Y. Qu, W. Ni, Y. Xiang, L. Gao, and E. Hossain, "Data and model poisoning backdoor attacks on wireless federated learning, and the defense mechanisms: A comprehensive survey," 2023, *arXiv:2312.08667*.
- [35] H. Zhang, S. Jiang, and S. Xuan, "Decentralized federated learning based on blockchain: Concepts, framework, and challenges," *Comput. Commun.*, vol. 216, pp. 140–150, 2024.
- [36] 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.
- [37] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1279–1283, Jun. 2020.
- [38] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2438–2455, Sep./Oct. 2021.
- [39] D. C. Nguyen et al., "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12806–12825, Aug. 2021.
- [40] M. Ali, H. Karimpour, and M. Tariq, "Integration of blockchain and federated learning for internet of things: Recent advances and future challenges," *Comput. Secur.*, vol. 108, 2021, Art. no. 102355.
- [41] D. Li et al., "Blockchain for federated learning toward secure distributed machine learning systems: A systemic survey," *Soft Comput.*, vol. 26, no. 9, pp. 4423–4440, 2022.

- [42] N. Dong, M. Kampffmeyer, I. Voiculescu, and E. Xing, "Federated partially supervised learning with limited decentralized medical images," *IEEE Trans. Med. Imag.*, vol. 42, no. 7, pp. 1944–1954, Jul. 2023.
- [43] N. Dong, M. Kampffmeyer, and I. Voiculescu, "Learning underrepresented classes from decentralized partially labeled medical images," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervention*, Cham, Switzerland: Springer, 2022, pp. 67–76.
- [44] M. Abadi et al., "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.
- [45] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.
- [46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015.
- [47] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 8026–8037.
- [48] W. Dai, Y. Zhou, N. Dong, H. Zhang, and E. Xing, "Toward understanding the impact of staleness in distributed machine learning," in *Proc. Int. Conf. Learn. Represent.*, 2019.
- [49] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent.*, 2021.
- [50] T. Brown et al., "Language models are few-shot learners," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1877–1901.
- [51] P. Rajpurkar et al., "CheXNet: Radiologist-level pneumonia detection on chest x-rays with deep learning," 2017, *arXiv:1711.05225*.
- [52] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.
- [53] W. Dai, N. Dong, Z. Wang, X. Liang, H. Zhang, and E. P. Xing, "Scan: Structure correcting adversarial network for organ segmentation in chest x-rays," in *Proc. Deep Learn. Med. Image Anal. Multimodal Learn. Clin. Decis. Support*, 2018, pp. 263–273.