ITU Journal on Future and Evolving Technologies, Volume 6, Issue 2, June 2025

ARTICLE

Incentive mechanism design for semi-asynchronous blockchain-based federated edge learning

Xuanzhang Liu[1], Jiyao Liu[1], Xinliang Wei[2], Yu Wang[1]

[1] Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA, [2] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, 518055, China

Corresponding author: Xuanzhang Liu, xzliu@temple.edu

Federated learning at edge systems not only mitigates privacy concerns by keeping data localized but also leverages edge computing resources to enable real-time AI inference and decision-making. In a blockchain-based federated learning framework over edge clouds, edge servers as clients can contribute private data or computing resources to the overall training or mining task for secure model aggregation. To overcome the impractical assumption that edge servers will voluntarily join training or mining, it is crucial to design an incentive mechanism that motivates edge servers to achieve optimal training and mining outcomes. In this paper, we investigate the incentive mechanism design for a semi-asynchronous blockchain-based federated edge learning system. We model the resource pricing mechanism among edge servers and task publishers as a Stackelberg game and prove the existence and uniqueness of a Nash equilibrium in such a game. We then propose an iterative algorithm based on the Alternating Direction Method of Multipliers (ADMM) to achieve the optimal strategies for each participating edge server. Finally, our simulation results verify the convergence and efficiency of our proposed scheme.

Keywords: Blockchain, edge AI, edge computing, federated learning, incentive mechanism, semi-asynchronous, Stackelberg game

1. INTRODUCTION

Federated Learning (FL) was first proposed by Google [1] to enhance user data privacy by enabling decentralized training of machine learning models directly on local devices, reducing the need to share raw data. This paradigm is particularly crucial for supporting emerging AI applications that rely on vast amounts of sensitive and distributed data, such as personalized healthcare, smart cities, and autonomous systems. The integration of FL into edge computing environments, where computation is performed closer to data sources, such as edge servers and IoT devices, offers a promising solution to further enhance privacy, efficiency, and scalability. Unlike traditional cloud-based AI models, FL at edge systems (i.e., federated edge learning [2, 3, 4, 5]) not only mitigates privacy concerns by keeping data localized but also leverages edge computing resources to enable real-time AI inference/decision-making and scalable AI-driven applications [6, 7], such as real-time federated analytics, intelligent transportation systems, and industrial automation.

Despite these advantages, deploying FL in edge environments presents several key challenges [8]. Traditional FL architectures rely on a centralized parameter server (often located in a remote cloud platform) to aggregate model updates from clients, leading to two significant limitations: (1) high communication costs and congestion at the parameter server, especially in large-scale FL deployments where frequent model transmissions from clients can saturate limited and heterogeneous network resources, increasing latency and energy consumption [9]; (2) security and transparency concerns, as model aggregation at a centralized and potentially untrusted server creates vulnerabilities to adversarial attacks, model poisoning, and privacy breaches. Furthermore, self-interested clients may

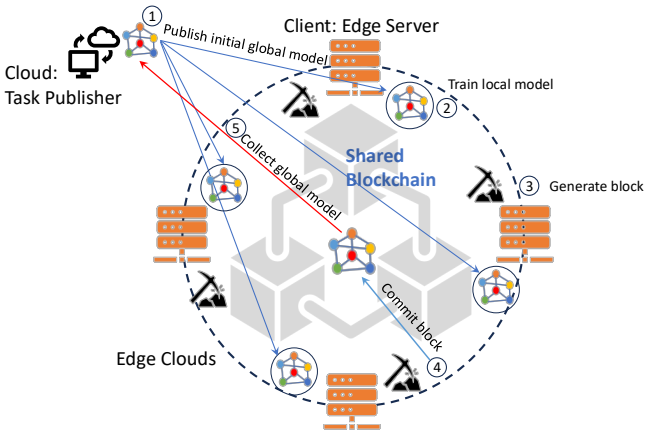


Figure 1 – The overall architecture of blockchain-based federated edge learning, where edge servers can collaboratively train the AI model and mine the block for committing the aggregated model.

exhibit strategic behavior, such as free-riding or malicious updates, further degrading model performance and trustworthiness. Given these challenges, a more efficient and robust federated learning approach tailored for edge computing is essential to address the inherent limitations of centralized FL, to avoid the single point of failure, communication bottleneck, and malicious attacks.

Blockchain has been integrated into the federated edge learning system to improve its security and transparency [10, 11, 12] in recent years. Specifically, the usage of a decentralized blockchain as the aggregator allows the elimination of a central server in FL training [10]. The local updates are recorded as transactions and appended to the block. After the consensus process, a shared ledger of ordered links of blocks is used to aggregate the global model and distribute global updates to clients for direct computation at devices. Thus, every node can easily trace the origin where a model parameter is modified or updated during the training process through transaction logs, which avoids the malfunction or malicious behavior of clients. Blockchain not only mitigates the risk of single-point failures but also ensures high trust and transparency for the FL system. Fig. 1 shows a general architecture of a blockchain-based federated edge learning [13, 14]. It generally includes a group of edge servers (clients) in a connected blockchain network. Each edge server can collect the training data for the training model and also perform mining to maintain the blockchain network. The model owner sits in the remote cloud and announces/collects the global model from the blockchain network.

Since the performance of this blockchain-based FL heavily depends on participation from clients [15] for contributing data, training local model, and mining, incentive mechanism design becomes critical to stimulate these participants to provide private data or computing resources for the overall FL training/mining task. In this paper, we mainly investigate the incentive mechanism design for a blockchain-based federated edge learning system by motivating the task publisher and edge servers to work together to achieve efficient training and mining processes.

Besides, according to the aggregation protocol adopted by blockchain-based FL systems, we can classify them into two categories, *synchronous update* and *asynchronous update*. For example, Wang et al. [13] use the synchronous update, where the aggregator has to wait for receiving updates from all clients to perform the aggregation. This may cause the straggler issue, where the straggler's local update may experience long delays or fail to upload to the aggregator. In contrast, Feng et al. [14] propose an asynchronous FL framework under the Proof-of-Work (PoW)-based blockchain system. However, due to frequent mining work (since every local update requires consensus by all nodes), the consensus process needs massive peer-to-peer communications among all participating nodes which causes long latency for the overall learning time. In this paper, we instead consider a *semi-asynchronous update* protocol [16], which can balance the learning convergence speed and the frequency of the consensus process.

Our major contributions of this paper can be summarized as follows.

- We model and analyze the interaction between edge servers (clients) and a task publisher as a Stackelberg game in a blockchain-based semi-asynchronous FL over edge clouds. We then prove the existence and uniqueness of Nash Equilibrium (NE) in the formulated game so that no one has the motivation to change the optimal strategies derived.
- To efficiently obtain such optimal strategies, we propose an iterative algorithm based on the Alternating Direction Method of Multipliers (ADMM) [17] that jointly considers the high-dimensional strategy space and distributed manner.
- We conduct simulations to evaluate the performance of the proposed method. Results show that our iterative algorithm converges to the optimal solution quickly and achieve better utilities under different scenarios compared with several baseline methods.

The rest of this work is organized as follows. In Section 2, we review the related research regarding blockchain-based FL. We then present the system model of blockchain-based FL architecture in Section 3 and the utility model of each entity in Section 4. In Section 5, we construct a Stackelberg game to analyze the equilibrium among edge servers and the cloud. In Section 6, we propose an iterative algorithm to solve the formulated game. Simulation results and performance analysis are presented in Section 7, followed by our conclusion in Section 8. A preliminary version of this paper appears as [18].

2. RELATED WORK

In this section, we first review the existing research on blockchain-based federated learning, followed by a discussion of current solutions for incentive mechanism design in blockchain-based FL systems.

2.1 Blockchain-based FL

While blockchain-based FL holds promise in enhancing the security and transparency of FL training, it still faces several performance challenges [8, 11, 12, 19], e.g. prolonged latency due to mining, challenging resource management, learning performance/efficiency trade-offs, and security/privacy concerns from curious miners/trainers. Therefore, recently, there have been several efforts to design various blockchain-based FL structures to meet different requirements.

Feng *et. al* [14] have suggested a Blockchain-based Federated Learning (BAFL) system to engender efficient and secure FL. BAFL employs an asynchronous strategy to hasten the convergence of the global model. The blockchain is used to prevent single-point failures and ensure the security of FL. The authors also utilized an entropy-based participant evaluation, optimal control of the block generation rate, and Pareto optimization strategy to trade-off between device energy consumption and learning delay, thus enhancing learning efficiency. In a similar vein, Pokhrel and Choi [20] propose an enhanced FL with a blockchain framework, facilitating efficient communication among autonomous vehicles. The use of blockchain mitigates centralized malfunctioning issues and augments the intelligent vehicular network's capability by attracting untrustworthy vehicles with a reward mechanism to improve FL performance. By analyzing the latency of FL training and model updating at a vehicle with its associated miner, they design both offline and online algorithms to enhance the overall model learning performance. Lu *et. al* [21] also propose a blockchain-based FL scheme to enhance communication security and user privacy in digital twin edge networks. Since their scenario is latency-sensitive, the blockchain operates on a Delegated Proof of Stake (DPoS) consensus mechanism, where the verifiers are selected based on their computing contribution to the global model. A multi-agent deep reinforcement learning algorithm is further proposed to solve a latency optimization problem with respect to learning accuracy conditions and bandwidth allocation constraints.

Huang *et. al* [22] investigate the imbalanced data distribution and resource allocations of IoT devices under the multilayer blockchain-enabled Hierarchical FL (HFL) network. Through analyzing the bottleneck of model accuracy, the authors have derived the upper bound of model error, which can be represented by the total data distance (composed of device association and local data distribution). Therefore, they design a distance-aware HFL to jointly optimize the device association and resource allocation to improve the model accuracy and minimize the learning latency. Fan *et. al* [23] design a hybrid blockchain-based resource trading system for FL in edge computing. Acknowledging the high overhead in cross-blockchain data synchronization within traditional blockchain networks, their system enables requesters and edges to engage with public and consortium blockchains separately, achieving higher credibility and better system performance. Within the consortium blockchain part, data quality-driven reverse auctions are introduced to facilitate automatic, autonomous, and auditable auctions among edge nodes. The payment channel is integrated into the public blockchain using the smart contract to enable credible, fast, low-cost, and high-frequency payment transactions among requesters and edge nodes.

Besides, FL can also facilitate the blockchain. Qu *et. al* [24] propose a new consensus mechanism called Proof-of-Federated-Learning (PoFL) to reduce the amounts of energy wasted by Proof-of-Work (PoW). To protect the privacy of training data, a reverse game-based data trading mechanism is proposed, where a trainer maximizes its utility only when it trains the model without any data leakage. A privacy-preserving model verification mechanism, which employs homomorphic encryption and secure two-party computation in label prediction and comparison, respectively, is also designed to verify the accuracy of a trained model while preserving the privacy of the task requester's test data, as well as avoiding the submitted model from being plagiarized by others.

In this paper, we consider a semi-asynchronous blockchain-based FL in a cloud-edge system, which is different from the works above.

2.2 Incentive mechanism design

The performance of FL relies on the contributions (both training data and computational resources) from its participants; therefore, it is necessary to design a proper incentive mechanism to motivate their participation [15]. This is also true for blockchain-based FL systems. Without sufficient incentives, clients may not be willing to join the training or perform the mining in blockchain, thus hurting the scalability of blockchain-based FL. On the other hand, since blockchain provides the features of robust and tamper-proof to FL, it has been considered as a promising method to facilitate transparent economic mechanism designs, so as to boost FL training.

Yuan *et. al* [25] propose a blockchain-based platform called CoopEdge which addresses the incentive and trust

issues systematically in a decentralized manner. CoopEdge rewards edge servers that participate in cooperative computing with credits so that they can use the credits to retrieve help later on. Edge servers could not only obtain credits when they win a peer-offloaded task and complete it on time as a task executor but also receive a bookkeeping reward when they successfully commit a task transaction onto the blockchain as a task recorder. Bao *et. al* [26] adopt the blockchain to provide auditable FL with trust and incentive attributes. Their FL system will drop the incorrect computation result and punish the generator by adopting a reward for misbehavior detector and compensation for affected c lients. To r educe the time cost of blockchain query, they adopt a double dual counting bloom filter.

Wang *et. al* [13] also investigate the issue of incentive mechanism design for the hierarchical FL. Considering the resource heterogeneity of clients and edge servers with multidimensional individual properties under incomplete information, a game-based incentive mechanism is proposed to model the interaction between clients, edge servers, and the cloud. In addition, to address the challenge of unreliable participants, the authors introduce blockchain technology to ensure the privacy of local updates and provide a credible and transparent environment.

Although much work has been devoted to designing incentive mechanisms for blockchain-based FL, they mainly focus on motivating FL without considering the incentives for the blockchain consensus process. In addition, none of the previous work on incentive mechanisms considers their blockchain-based FL in a semi-asynchronous manner. Instead, in this paper, we consider a price-based incentive mechanism that motivates the edge servers to join in both the learning process and the consensus process under the semi-asynchronous blockchain-based federated edge learning.

3. SYSTEM MODEL

The cloud-client architecture for our blockchain-based federated edge learning (as shown in Fig. 1) consists of two components: the cloud (i.e., the task publisher) and the clients (i.e., edge servers).

Clients: There is a set of clients $C = \{c_1, c_2, \ldots, c_N\}$ in the edge system. Each client c_i is an individual edge server that has computing resources and training data that can be used for training FL tasks. Concurrently, each client can also join in the mining game since it is responsible for managing the blockchain system. Hereafter, we use the terms *clients* and *edge servers* interchangeably.

Table 1 – Description of major notations used.

Variable	Description
c_i, C	jth edge server, set of servers
N	number of edge servers (clients)
$w_i^{(t)}$, $w^{(t)}$	ith client's local model, global model at time t
α_t, $\sigma()$	learning rate at t, staleness function
$C^{(t)}$	set of clients whose local models were received at t
x_i, $\mathcal{X}_i$,	training data size, max training data size at c_i
θ_i, ϵ_i, q_i	data quality, local accuracy, weighted model quality of client c_i
η, cr	data quality coefficient, crash probability
R^T	expected reward for one local update
r_{cloud}	training payment from cloud to all servers
f_i, d_i	CPU frequency, number of CPU cycles for training unit data at c_i
τ	number of iterations during local training in each round
E_i^{cmp}, E_i^{comm}	energy consumption for training/transferring the model at c_i
Ω	size of model parameters
p_i, v_i	transmission power and rate of c_i
C_i^E,	unit cost of data collected by c_i
U_i^T, U_i^M	training/mining utility of server c_i
m_i, $\mathcal{M}_i$	mining resources actually used by c_i, max mining resource at c_i
h_i, W_i	computing power ratio and winning probability of mining game at c_i
κ, λ	consensus delay per transaction, process parameter of mining a block
n_i	number of transactions in the block
R^M, γ^M	fixed reward, unit reward rate for commission reward
K_i	unit mining cost at c_i
B_i	revenue budget of server c_i
b	weight parameter to balance mining and training utilities
P	unit training price paid by the cloud
$g()$	training gain function of the cloud
δ	satisfaction degree of task publisher
U_i, U_{pub}	total utility of client c_i and the cloud
ι, q	index of inner loop, outer loop
Ξ	threshold for ending ADMM
$\varphi_{k,i}^{(q)}$	scaled dual variables
ρ	damping factor

Cloud: The role of the cloud is to receive training tasks from specific task publishers and distribute these tasks to the clients. The goal for each training task is to generate a satisfactory model, leveraging the FL training from selected clients. The cloud fulfills the role of providing payment to clients as an incentive for participation in model training. Hereafter, we use the terms *cloud* and *task publisher* interchangeably.

We summarize all notations used in Table 1.

Now we introduce the workflow of task training and mining of a single FL task of a global model w. Our system has two phases: *negotiation phase* and *execution phase*.

In the negotiation phase, the interaction between cloud and clients is formed as a Stackelberg game to determine the optimal policy for each participant (including deciding both prices and allocated resources). Here, we assume that all participants in the system are rational entities, i.e., always aiming to maximize their utilities. The key part of this phase is how to design the incentive mechanism such that everyone can reach an equilibrium.

In the execution phase, each participant would perform the steps shown in Fig. 1 and provide the corresponding rewards or resources to train the model under the management by blockchain according to the decision made during the negotiation phase. The execution of FL training is performed as follows.

(1) **Initialization:** In each round, the global model is distributed to each client (Step ① in Fig. 1).

(2) **Local training at clients:** Upon receiving the global model w with its timestamp from the cloud, the client performs a local update based on its local training data (Step ②).

(3) **Mining for generating the block:** Once the client finishes one local update, it writes the model to its own new block and broadcasts the local model at the blockchain layer (Step ③). When other miners receive the local model, they verify its accuracy and record it into their new blocks. After a certain time period, each client runs a consensus mechanism until it finds the desired nonce or receives the generated block (Step ④). We assume the latest global model it received was with timestamp t', at the moment of t it is about to aggregate a set of local updates from a group of clients (denoted by $C^{(t)}$), and each local model from a client $c_i \in C^{(t)}$ were with timestamp t_i. Then, the weighted global model at t is aggregated as:

$$w^{(t)} \longleftarrow w^{(t')} + \alpha_{t'} \frac{\sum_{c_i \in C^{(t)}} \sigma\left(w_i^{(t_i)}\right)}{|C^{(t)}|}, \tag{1}$$

where $|C^{(t)}|$ is the number of received local models during the time period, $\alpha_{t'}$ is the learning rate, $\sigma\left(w_i^{(t_i)}\right)$ is the stale gradient computed by client c_i [27], e.g., $\sigma\left(w_i^{(t_i)}\right) = w_i^{(t_i)}/\sigma(t_i - t')$.

(4) **Aggregation on cloud:** After several rounds or a certain time period, the cloud reads the global model from the blockchain to generate the final global model (Step ⑤).

4. STACKELBERG GAME

We represent the FL process as a two-stage Stackelberg game, which includes a cloud subgame and a client subgame. The Stackelberg games have proven extremely useful in sequentially analyzing interactions among rational agents [28, 29, 30].

4.1 Stage II: Client subgame

Each client should adapt training or mining tasks to fit their specific budgets and capabilities. This leads to two categories of utility models.

Training process: When clients with sufficient dataset receive the FL task, they could choose to train the model with a data size of x_i. Besides, we define the data quality $\theta_i = -\eta/\log(\epsilon_i)$ where ϵ_i is the local accuracy of client c_i and η indicates the data quality coefficient. Let

$$q_i = \frac{\theta_i x_i}{\sum_{c_j \in C} \theta_j x_j}$$

represent the weighted model quality of client c_i [31]. Owing to the possibility of some local updates not being added to the global model in semi-asynchronous aggregation, we introduce crash probability cr [16] to reflect such likelihood that a local update is not committed into the global model in one round. The expected reward for one local update is calculated as:

$$R^T = r_{cloud}((1 - cr) + \sigma cr(1 - cr)), \tag{2}$$

where r_{cloud} is the training reward provided by the task publisher. σ is the stale rate due to the lagged update.

Besides, each client should incur certain expenses to collect data and have to consume energy to reach the predetermined accuracy [32]. Let f_i and d_i denote the CPU cycle frequency and the number of CPU cycles for training unit data of client c_i, respectively. We assume the number of iterations in each round as τ. Consequently, the energy consumption for training can be depicted as

$$E_i^{cmp} = \tau d_i x_i f_i^2. \tag{3}$$

Similarly, we can also define the energy consumption for sending the model to the cloud as

$$E_i^{comm} = \frac{\Omega p_i}{v_i}, \tag{4}$$

where Ω is the size of model, p_i and v_i represent the client c_i's transmission power and rate, respectively. Let C_i^E denote the cost of unit data collected by the client. Thus, the utility of training clients is formulated as:

$$U_i^T = q_i R^T - E_i^{cmp} - E_i^{comm} - C_i^E x_i. \tag{5}$$

The first part of the equation consists of the cloud's paid reward, followed by computational, communication, and data collection expenditures incurred during the training phase.

Mining process: Given the incentive of receiving mining rewards for successfully appending a new block, each client might devote as much computation resource to win the mining game. As such, the winning probability of c_i by solving the PoW problem depends on the relative computing power m_i compared to the collective computing power of all participating miners (clients), which can be formulated as

$$h_i = \frac{m_i}{\sum_{c_k \in C} m_k}. \tag{6}$$

After a block is successfully mined, it will be broadcast across the network immediately to complete the consensus process. Thus, the consensus delay becomes an influential determinant for winning the mining reward. Based on [30, 17], we assume that the consensus delay (including both propagation delay and verification time) is approximately linear to the number of transactions n_i in the block, which can be expressed as κn_i. In this context, κ is the parameter determined by the network scale, channel capacity at links, and the average verification speed of nodes. Assuming that the block mining duration follows the Poisson distribution, the probability of client c_i winning the mining game is calculated as:

$$W_i = h_i e^{-\lambda \kappa n_i}, \tag{7}$$

where λ is the process parameter referring to the complexity of mining a block.

Once the client wins the mining competition, it will gain the corresponding mining reward which consists of a fixed reward R^M and a commission reward $\gamma^M n_i$, where γ^M is the unit reward rate. On the other hand, the total cost of client c_i is composed of using its computation resource. Therefore, the utility of the mining process is defined as:

$$U_i^M = (R^M + \gamma^M n_i) W_i - K_i m_i, \tag{8}$$

where K_i is the unit mining cost.

Client subgame: Hence, the objective in this stage is to maximize the utility of each client by determining the strategy of x_i and m_i simultaneously.

$$\begin{aligned} \max_{x_i, m_i} \ & U_i \\ \text{s.t.} \ & C_i^E x_i + K_i m_i \le B_i. \\ & 0 \le x_i \le \mathcal{X}_i, \\ & 0 \le m_i \le \mathcal{M}_i. \end{aligned} \tag{9}$$

Here, the utility of client c_i is interpreted as:

$$U_i = b \cdot U_i^M + U_i^T, \tag{10}$$

where b is a weight parameter to balance the relative impact of mining and training utilities. The budget constraint for each client c_i is defined as B_i. In addition, each client has its own data size capacity $\mathcal{X}_i$ and computing capacity $\mathcal{M}_i$.

4.2 Stage I: Cloud subgame

Unlike the role of clients in our scheme, the cloud exclusively publishes a single training task in each period by providing a total corresponding reward:

$$r_{cloud} = \sum_{c_i \in C} \theta_i x_i P > 0. \tag{11}$$

Here P denotes the unit price the cloud is willing to pay for training data. Additionally, the task publisher must cover the costs associated with committing transactions to the blockchain.

The gain of the cloud is determined by the performance of the global model (which is related to the global accuracy). This relationship can be formulated as a concave function: [33]

$$g(\sum_{c_i \in C} \theta_i x_i) = 0.5 \cdot \ln(1 + \sum_{c_i \in C} \theta_i x_i). \tag{12}$$

Given these groundwork constructions, we can define the general form of the cloud's utility per round as:

$$U_{pub} = \delta g(\sum_{c_i \in C} \theta_i x_i) - \gamma^M n_i - \sum_{c_i \in C} \theta_i x_i P. \tag{13}$$

Here $\delta > 0$ identifies the satisfaction degree parameter of the cloud.

Cloud subgame: In this stage, it aims to solve the following problem to maximize the utility of the cloud U_{pub}:

$$\begin{aligned} \max_{P} \ & U_{pub} \\ \text{s.t.} \ & P \ge 0. \end{aligned} \tag{14}$$

5. EQUILIBRIUM ANALYSIS

In this section, we prove the existence and uniqueness of Nash Equilibrium (NE) in the formulated game so that no one has the motivation to change the optimal strategies derived. The formal definition of NE is as follows.

Definition 1 *Let* $\mathbf{X}^*$, $\mathbf{M}^*$, P^* *denote the optimal strategies of each entity. The NE is the point that no one has the incentive to deviate from the current state. We can represent such claims as follows:*

$$U_i(\mathbf{X}^*, \mathbf{M}^*) \geq U_i(\mathbf{X}, \mathbf{M}), \tag{15}$$

$$U_{pub}(P^*) \geq U_{pub}.(P) \tag{16}$$

In the following section, we will prove the NE in each stage of the Stackelberg game, first for Stage II (Theorem 1) and then for Stage I (Theorem 2).

Theorem 1 *In Stage II, the existence and uniqueness of NE for clients are guaranteed given the cloud's reward strategy.*

Proof. Relying on Eq. (*(9)*), we calculate the first-order and second-order derivatives of utility U_i regarding x_i and m_i as follows:

$$\frac{\partial U_i}{\partial m_i} = b(R^M + \gamma^M n_i)e^{-\lambda\kappa n_i}\frac{\partial h_i}{\partial m_i} - K_i, \tag{17}$$

$$\frac{\partial^2 U_i}{\partial (m_i)^2} = b(R^M + \gamma^M n_i)e^{-\lambda\kappa n_i}\frac{\partial^2 h_i}{\partial (m_i)^2}, \tag{18}$$

$$\frac{\partial U_i}{\partial x_i} = R^T\frac{\partial q_i}{\partial x_i} - \tau d_i f_i^2 - C_i^E, \tag{19}$$

$$\frac{\partial^2 U_i}{\partial (x_i)^2} = R^T\frac{\partial^2 q_i}{\partial (x_i)^2}, \tag{20}$$

$$\frac{\partial^2 U_i}{\partial m_i \partial x_i} = \frac{\partial^2 U_i}{\partial x_i \partial m_i} = 0. \tag{21}$$

The explicit-form expressions of $\frac{\partial h_i}{\partial m_i}$, $\frac{\partial^2 h_i}{\partial (m_i)^2}$, $\frac{\partial q_i}{\partial x_i}$, and $\frac{\partial^2 q_i}{\partial (x_i)^2}$ are computed as below:

$$\frac{\partial h_i}{\partial m_i} = \frac{\sum_{c_k \in C, c_k \neq c_i} m_k}{(\sum_{c_k \in C} m_k)^2}, \tag{22}$$

$$\frac{\partial^2 h_i}{\partial (m_i)^2} = -\frac{2\sum_{c_k \in C, c_k \neq c_i} m_k}{(\sum_{c_k \in C} m_k)^3}. \tag{23}$$

$$\frac{\partial q_i}{\partial x_i} = \frac{\theta_i \sum_{c_k \in \mathcal{G}_i, c_k \neq c_i} \theta_k x_k}{(\sum_{c_k \in \mathcal{G}_i} \theta_k x_k)^2}, \tag{24}$$

$$\frac{\partial^2 q_i}{\partial (x_i)^2} = -\frac{2\theta_i^2 \sum_{c_k \in \mathcal{G}_i, c_k \neq c_i} \theta_k x_k}{(\sum_{c_k \in \mathcal{G}_i} \theta_k x_k)^3}. \tag{25}$$

Subsequently, we demonstrate that the Hessian matrix given below is positive definiteness for each client:

$$H_i = \begin{bmatrix} U_{mm}^i & U_{mx}^i \\ U_{xm}^i & U_{xx}^i \end{bmatrix}, \tag{26}$$

where $U_{mm}^i = \frac{\partial^2 U_i}{\partial m_i^2}$, $U_{xx}^i = \frac{\partial^2 U_i}{\partial x_i^2}$, $U_{mx}^i = \frac{\partial^2 U_i}{\partial m_i \partial x_i}$, and $U_{xm}^i = \frac{\partial^2 U_i}{\partial x_i \partial m_i}$.

Considering $U_{mm}^i < 0$, while the determinant $\det(H_i) = U_{mm}^i U_{xx}^i - U_{mx}^i U_{xm}^i > 0$, Hessian Matrix H_i exhibits negative definite. Therefore, the utility function U_i is strictly concave within the feasible region of (x_i, m_i). We can conclude that the NE is not only existent but also unique at this stage. □

Theorem 2 *In Stage I, the existence and uniqueness of NE for the cloud are guaranteed.*

Proof. The proof is similar to those of Theorem 1. The key idea is to prove that Eq. (*(14)*) is a concave function, thereby establishing the presence of a unique point within the cloud's subgame. By calculating the first-order derivative of the utility

$$\frac{\partial U_{pub}}{\partial P} = -\sum_{c_i \in C} \theta_i x_i < 0, \tag{27}$$

it reveals that U_{pub} is a decreasing function of P. With an assumption that all participants exercise rationality within this game, the utility of cloud and clients should be non-negative. Consequently, the range of P can be established as follows:

$$U_{pub} \geq 0 \Rightarrow P \leq \frac{\delta g(\sum_{c_i \in C} \theta_i x_i) - \gamma^M n_i}{\sum_{c_i \in C} \theta_i x_i}, \tag{28}$$

$$U_i \geq 0 \Rightarrow P \geq \frac{(E_i^{cmp} + E_i^{comm} + C_i^E x_i)}{\theta_i x_i((1-cr) + \sigma cr(1-cr))}, \forall c_i \in C. \tag{29}$$

Following these computations, we can obtain the maximum utility of the task publisher when P reaches the minimum value through the implementation of a convex optimization algorithm. □

By combining the results from both theorems, we can infer that the unique NE exists in the whole Stackelberg game based on the optimal reward strategy.

6. ALGORITHM DESIGN

In our blockchain-based FL framework, the leader (the task publisher at the cloud) first moves and the followers (clients) then respond. Considering the efficiency and accuracy of the desired algorithm, we design an iterative algorithm based on ADMM to obtain the optimal strategies in our game. The algorithm (Algorithm 1) contains two-tier iterations. In the inner loop, based on the

Algorithm 1 Iterative ADMM-based algorithm for the formulated Stackelberg Game

Input: Generated initial feasible strategy ($\mathbf{X}$, $\mathbf{M}$, P), threshold Ξ, iteration $q = 0$.
Output: Optimal strategy ($\mathbf{X}^*$, $\mathbf{M}^*$, P^*).

1: Calculate the current values of U_i, set as $U_i^{(q)}$;
2: **repeat**
3: $q \leftarrow q + 1$;
4: (Inner Loop) Utility optimization for clients through ADMM: Based on the current reward strategy (P^{q-1}) given by cloud, client iteratively update the training data $\mathbf{X}$ and computing resource $\mathbf{M}$ according to Eq. *(30)* and *(31)*;
5: ($\mathbf{X}^*$, $\mathbf{M}^*$) $\leftarrow$ the current outputs;
6: (Outer Loop) Obtain the pricing strategy P^q of task publisher using Eq. *(32)*;
7: **until** $\|U_{pub}^{(q)} - U_{pub}^{(q-1)}\| \leq \Xi$
8: $P^* \leftarrow P^q$;
9: **return** ($\mathbf{X}^*$, $\mathbf{M}^*$, P^*).

unit price of data (P) given by the task publisher, clients employ the Lagrange multiplier method to obtain the optimal amount of data size $\mathbf{X}$ and computing resource $\mathbf{M}$ that maximize their utilities. In the outer loop, the task publisher updates its price strategy according to the data size and computing resources from the clients.

6.1 Inner loop

In the inner loop (Line 4 in Algorithm 1), the ADMM algorithm solves the optimal strategies of clients in the qth outer loop, where q is the index of outer loop iterations. We use Lagrange's multipliers to find the iteration strategy of m_i and x_i:

$$\begin{aligned}\{x_i^{(q)}(\iota), m_i^{(q)}(\iota)\} &= \arg\max(U_i^{(q)}) + \varphi_{1,i}^{(q)}(\iota-1)x_i^{(q)}(t)m_i^{(q)}(\iota-1) \\ &+ \frac{\rho}{2}\|C_i^E x_i^{(q)}(\iota-1) + K_i m_i^{(q)}(\iota-1) - B_i\|_2^2 \\ &+ \varphi_{2,i}^{(q)}(\iota-1)(|\mathcal{X}_i| - x_i^{(q)}(\iota-1)) + \frac{\rho}{2}\||\mathcal{X}_i| - x_i^{(q)}(\iota-1)\|_2^2 \\ &+ \varphi_{3,i}^{(q)}(\iota-1)(|\mathcal{M}_i| - m_i^{(q)}(\iota-1)) + \frac{\rho}{2}\||\mathcal{M}_i| - m_i^{(q)}(\iota-1)\|_2^2, \end{aligned} \tag{30}$$

where ι represents the iteration index in the inner loop, ρ is a damping factor. Updating the dual variables of the inner loop is described as follows:

$$\begin{aligned}\varphi_{1,i}^{(q)}(\iota) &= \varphi_{1,i}^{(q)}(\iota-1) + \rho(C_i^E x_i^{(q)}(\iota) + K_i m_i^{(q)}(\iota) - B_i), \\ \varphi_{2,i}^{(q)}(\iota) &= \varphi_{2,i}^{(q)}(\iota-1) + \rho(|\mathcal{X}_i| - x_i^{(q)}(\iota)), \\ \varphi_{3,i}^{(q)}(\iota) &= \varphi_{3,i}^{(q)}(\iota-1) + \rho(|\mathcal{M}_i| - m_i^{(q)}(\iota)).\end{aligned} \tag{31}$$

6.2 Outer loop

After client c_i reaches its current strategies, the task publisher updates its price P to maximize its profit by invoking ADMM as

$$P^{(q)} = P^{(q-1)} + \nabla \frac{\partial U_{pub}}{\partial P}. \tag{32}$$

The iterative steps for the task publisher continuously adjust its pricing strategy until the following condition holds:

$$\|U_{pub}^{(q)} - U_{pub}^{(q-1)}\| \leq \Xi, \tag{33}$$

where Ξ is the predefined exiting threshold of ADMM.

6.3 Complexity analysis

We now provide a time complexity analysis of our algorithm by proving the following theorem.

Theorem 3 *The time complexity of Algorithm 1 is* $O(\frac{N^2}{\Xi^3} + \frac{N^2}{\Xi})$.

Proof. A single iteration of the ADMM algorithm exhibits a complexity of $O(1/\Xi^2)$, where Ξ denotes the predefined convergence tolerance [34]. Given the distributed nature of the system with N participating clients, each inner loop iteration consequently requires $O(N/\Xi^2)$ operations to coordinate all clients' updates. In the outer loop architecture, the task publisher computes reward strategies for its associated clients, resulting in $O(N)$ operations per outer iteration.

The convergence analysis demonstrates that the ADMM algorithm requires $O(N/\Xi)$ global iterations to reach the prescribed tolerance. Consequently, the total computational complexity combines these components is calculated as follows:

$$O\left(\frac{N}{\Xi}\right) \times \left[O\left(\frac{N}{\Xi^2}\right) + O(N)\right] = O\left(\frac{N^2}{\Xi^3} + \frac{N^2}{\Xi}\right). \tag{34}$$

□

7. SIMULATION EVALUATION

We conduct simulations including one task publisher situated in the cloud with ten clients implemented to verify the efficiency of our proposed scheme. We first evaluate the convergence of the proposed ADMM method, and then compare learning performance with several baseline algorithms.

Table 2 – Default parameter settings in simulations.

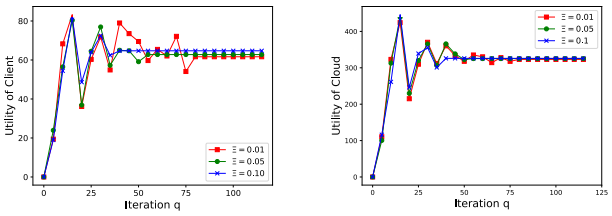
Parameters	Values/Distribution
Staleness rate σ	10
Quality of data θ_i	$\mathcal{U}(0.2, 0.9)$
CPU cycle capacity d_i	$\mathcal{U}(10, 20)$
Device CPU frequency f_i	$\mathcal{U}(1, 10)GHz$
Client transmission rate ν_i	$\mathcal{U}(100, 200)kHz$
Client transmission power p_i	$24dBm$
Updated model size Ω	2 000 bits
Mining complexity parameter λ	0.1
Unit commission reward rate γ^M	100
Computing capacity of client m_i	$\mathcal{U}(0, 5)$
Unit mining cost of server K_i	$\mathcal{N}(30, 5)$
Consensus delay effect κ	0.01
Satisfaction degree δ	0.1

7.1 Simulation setting

Our federated learning framework follows the benchmark implementation in [16], evaluated on the CIFAR-10 digit classification dataset containing 50 000 training and 10 000 test samples. The dataset is partitioned into N mutually exclusive subsets distributed across clients to preserve data locality. To emulate resource heterogeneity, clients are configured with constrained computational capabilities, processing limited batches per second during model training. The learning architecture employs the Convolutional Neural Network (CNN) with two convolutional layers followed by two fully connected layers with ReLu as the activation function. The federated learning rate is 0.01 and the mini-batch size of stochastic gradient descent used for optimizing is 32. Each client conducts $\tau = 20$ local iterations before parameter aggregation, spanning 50 global communication rounds. The temporal configuration fixes each round at 10-minute intervals through empirical calibration. We also assume that client devices experience failure updates with identical independent probability during each global aggregation. Besides, we simulate the process of generating the block as a Poisson process with the block size limited to $2MB$ [30, 35]. Unless otherwise specified, the default simulation parameters shown in Table 2 are used.

7.2 Convergence of ADMM

First, we verify the convergence of our proposed ADMM iterative algorithm. To better illustrate the convergence, we set $cr = 0.3$, client budget $B_i = 2\ 000$, and mining reward $R^M = 5\ 000$. Fig. 2 depicts the trend in the average utilities of clients and the cloud for our iterative algorithm under different thresholds Ξ. We can see that the convergence speed will be slightly faster as Ξ gets larger. After about 80 iterations, all plots tend to be stable. This result further attests that our proposed iterative algorithm can efficiently achieve the NE in the Stackelberg game.



(a) clients (b) cloud

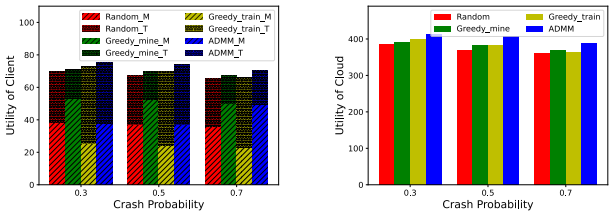
Figure 2 – Convergence of average utilities of (a) all clients and (b) the cloud under different values of existing threshold Ξ.

7.3 Utility performance of proposed method

We then compare the system performance of the proposed iterative algorithm (labeled as ADMM in figures) under different parameters compared with the following baselines.

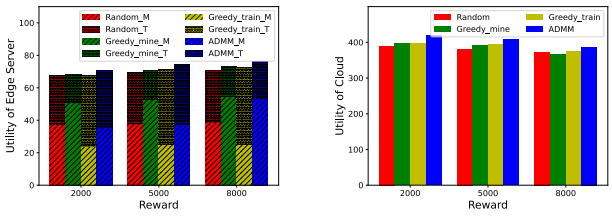
- **Random strategy** (Random): This strategy allows that each client c_i randomly selects whether to participate in training or mining. Based on these selections, the strategy identifies a feasible profile for participation in either process without explicit optimization considerations.
- **Greedy mining strategy** (Greedy_mine): In this strategy, clients are ranked in descending order according to their computing resources. The top half of the clients are selected to perform mining at full capacity, while the remaining clients contribute their data for training, subject to the budget constraints imposed by the edge server.
- **Greedy training strategy** (Greedy_train): In contrast to the mining strategy, clients are sorted in descending order based on their dataset cardinality. The top half of the clients are mandated for model training, with the remaining clients provisioning computational resources subject to the edge server's budget.

Fig. 3a and Fig. 3b show the impact of different crash probabilities on the utilities of clients and the cloud when fixing $R^M = 5\ 000$ and $B_i = 2\ 000$. To better highlight the contributions of different tasks in the utility, we compute the average utility and use the shadow composed of slash and star to distinguish the mining utility (append with '_M' in the label, e.g. 'Greedy_mine_M') and training utility (append with '_T' in the label, e.g. 'Greedy_mine_T'), respectively. The results show that our proposed scheme can always achieve higher utility compared to other baseline schemes. As the crash probability increases, the average utility of all participants decreases slightly. This is because a higher crash probability discourages clients from selecting training tasks, which in turn reduces the



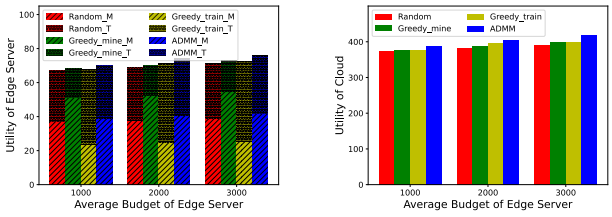
(a) client utility vs crash prob (b) cloud utility vs crash prob

Figure 3 – Utilities of the participants ((a) clients and (b) cloud) with different crash probabilities cr.



(a) client utility vs reward (b) cloud utility vs reward

Figure 4 – Utilities of the participants ((a) clients and (b) cloud) with different reward R^M.



(a) client utility vs budget (b) cloud utility vs budget

Figure 5 – Utilities of the participants ((a) clients and (b) cloud) with different Budget B_i.

utility of the task publisher, whose primary gains come from training tasks. Consequently, the task publisher lowers the rewards offered to clients, leading to a decline in their average utility.

Fig. 4 shows the impact of different mining rewards (R^M) on participant utilities, with a fixed crash probability of $cr = 0.3$ and an average edge server budget of 2 000. Similar to the trend observed in the previous figure, our proposed scheme consistently achieves the highest utility across different reward levels, demonstrating its superiority. By analyzing the utility components, we observe that as R^M increases, client utility rises while the task publisher's utility declines. This occurs because a higher mining reward provides greater incentives for mining, thereby reducing clients' motivation to contribute training data.

We also analyze the impact of the edge server budget on participant utilities, as shown in Fig. 5. When the crash probability and mining reward are fixed, increasing the

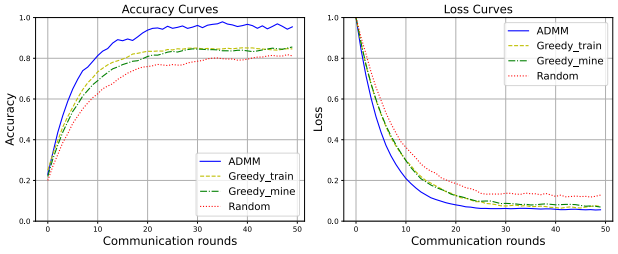


Figure 6 – Learning performance (left: accuracy, right: loss) of different mechanisms.

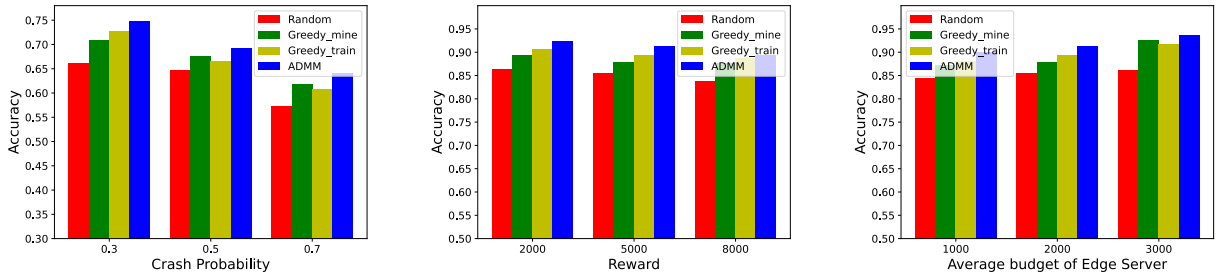
edge server's budget clearly leads to higher utilities for all participants. The reason inside is that a larger budget enables the edge server to allocate more resources for both training data and computing power. However, the utility gain from increasing the budget from 2 000 to 3 000 is smaller than that from 1 000 to 2 000. This is due to the fact that when client capacity is fixed and limited, simply increasing the budget does not necessarily translate to additional resources for mining or training tasks.

7.4 Learning performance of proposed federated edge learning method

Finally, we also evaluate the learning performance of the proposed mechanism. As shown in Fig. 6, we compare the learning performance of our proposed scheme with the aforementioned baseline strategies under the same settings in Section 7.2 in terms of the loss function and learning accuracy. The results indicate that our scheme achieves higher global model accuracy and relatively faster convergence. As shown in Fig. 7, our approach also yields the highest average model accuracy among all compared methods with different settings, attributed to the larger volume of training data collected. We notice that the trend of average model accuracy aligns closely with that of the cloud's utility under different parameters. This can be explained by the fact that the relationship between the data accuracy and the amount of training data follows a concave function [36]. Since we model the utility of each participant per round, the average model accuracy is computed after predefined rounds which effectively reflects the long-term impact of the incentive mechanism. The evaluation demonstrates that our scheme improves average model accuracy by approximately 7% compared to the random strategy and 3% compared to the greedy strategies. These results highlight the effectiveness of our approach in incentivizing clients to contribute training data, ultimately enhancing the task publisher's model accuracy.

8. CONCLUSION

This paper presents an incentive mechanism for semi-asynchronous blockchain-based federated edge learning.



(a) model accuracy vs crash prob (b) model accuracy vs reward (c) model accuracy vs budget

Figure 7 – Effects of crash probability, reward and budget on the model accuracy.

We model the interactions between the task publisher and clients as a two-stage Stackelberg game and demonstrate a unique Nash equilibrium within the proposed game. Furthermore, considering the high dimensionality of strategy space and the distributed manner of the system, we design an iterative algorithm based on ADMM to obtain the optimal solution efficiently. Simulation results validate its effectiveness, showing superior performance over baseline schemes. This research aims to advance the development of federated learning frameworks that fully leverage edge computing infrastructure, making it well-suited for next-generation AI applications while ensuring robustness, scalability, and security in decentralized environments.

In future work, we plan to consider investigating a similar incentive mechanism of a hierarchical FL over a cloud-edge-client architecture, where mobile clients can participate in both learning and mining.

ACKNOWLEDGEMENT

This work is partially supported by the US National Science Foundation under Grant No. CNS-2006604 and CNS-2128378.

REFERENCES

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data". In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282.

[2] Xinliang Wei, Jiyao Liu, and Yu Wang. "Joint Participant Selection and Learning Scheduling for Multi-Model Federated Edge Learning". In: *19th IEEE International Conference on Mobile Ad-Hoc and Smart Systems (MASS 2022)*. 2022.

[3] Xinliang Wei, Jiyao Liu, Xinghua Shi, and Yu Wang. "Participant Selection for Hierarchical Federated Learning in Edge Clouds". In: *IEEE International Conference on Networking, Architecture, and Storage (NAS 2022)*. 2022.

[4] Xinliang Wei, Jiyao Liu, and Yu Wang. "Joint Participant Selection and Learning Optimization for Federated Learning of Multiple Models in Edge Cloud". In: *Journal of Computer Science and Technology* 38.4 (2023), pp. 754–772.

[5] Xinliang Wei, Kejiang Ye, Xinghua Shi, Cheng-Zhong Xu, and Yu Wang. "Joint Participant and Learning Topology Selection for Federated Learning in Edge Clouds". In: *IEEE Transactions on Parallel and Distributed Systems* 35.8 (2024), pp. 1456–1468.

[6] Xiaofei Wang, Yiwen Han, Chenyang Wang, Qiyang Zhao, Xu Chen, and Min Chen. "In-Edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning". In: *IEEE Network* 33.5 (2019), pp. 156–165.

[7] Youqi Li, Shuangji Liu, Yanchen Meng, Shenyi Qi, Zhe Qu, Fan Li, and Yu Wang. "Towards Collaborative Intelligence for Metacomputing-driven IIoT based on Vertical Federated Learning with Fast Convergence". In: *IEEE Internet of Things Journal* (2025).

[8] Dun Li, Dezhi Han, Tien-Hsiung Weng, Zibin Zheng, Hongzhi Li, Han Liu, Arcangelo Castiglione, and Kuan-Ching Li. "Blockchain for federated learning toward secure distributed machine learning systems: a systemic survey". In: *Soft Computing* 26.9 (2022), pp. 4423–4440.

[9] Zhiyuan Wang, Hongli Xu, Jianchun Liu, Yang Xu, He Huang, and Yangming Zhao. "Accelerating Federated Learning With Cluster Construction and Hierarchical Aggregation". In: *IEEE Transactions on Mobile Computing* 22.7 (2023), pp. 3805–3822.

[10] Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. "Blockchained on-device federated learning". In: *IEEE Communications Letters* 24.6 (2019), pp. 1279–1283.

[11] Chuan Ma, Jun Li, Long Shi, Ming Ding, Taotao Wang, Zhu Han, and H Vincent Poor. "When federated learning meets blockchain: A new distributed learning paradigm". In: *IEEE Computational Intelligence Magazine* 17.3 (2022), pp. 26–33.

[12] Dinh C Nguyen, Ming Ding, Quoc-Viet Pham, Pubudu N Pathirana, Long Bao Le, Aruna Seneviratne, Jun Li, Dusit Niyato, and H Vincent Poor. "Federated learning meets blockchain in edge computing: Opportunities and challenges". In: *IEEE Internet of Things Journal* 8.16 (2021), pp. 12806–12825.

[13] Xiaofei Wang, Yunfeng Zhao, Chao Qiu, Zhicheng Liu, Jiangtian Nie, and Victor CM Leung. "InFEDge: A blockchain-based incentive mechanism in hierarchical federated learning for end-edge-cloud communications". In: *IEEE Journal on Selected Areas in Communications* 40.12 (2022), pp. 3325–3342.

[14] Lei Feng, Yiqi Zhao, Shaoyong Guo, Xuesong Qiu, Wenjing Li, and Peng Yu. "BAFL: A Blockchain-Based Asynchronous Federated Learning Framework". In: *IEEE Transactions on Computers* 71.05 (2022), pp. 1092–1103.

[15] Youqi Li, Fan Li, Song Yang, Chuan Zhang, Liehuang Zhu, and Yu Wang. "A Cooperative Analysis to Incentivize Communication-Efficient Federated Learning". In: *IEEE Transactions on Mobile Computing* 23.10 (2024), pp. 10175–10190.

[16] Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen Jarvis. "SAFA: A semi-asynchronous protocol for fast federated learning with low overhead". In: *IEEE Transactions on Computers* 70.5 (2020), pp. 655–668.

[17] Zehui Xiong, Jiawen Kang, Dusit Niyato, Ping Wang, and H Vincent Poor. "Cloud/edge computing service management in blockchain networks: Multi-leader multi-follower game-based ADMM for pricing". In: *IEEE Transactions on Services Computing* 13.2 (2019), pp. 356–367.

[18] Xuanzhang Liu, Jiyao Liu, Xinliang Wei, and Yu Wang. "Incentive Mechanism Design in Semi-Asynchronous Blockchain-based Federated Learning". In: *Proceedings of 2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall)*. 2024.

[19] Zhilin Wang and Qin Hu. "Blockchain-based federated learning: A comprehensive survey". In: *arXiv preprint arXiv:2110.02182* (2021).

[20] Shiva Raj Pokhrel and Jinho Choi. "Federated learning with blockchain for autonomous vehicles: Analysis and design challenges". In: *IEEE Transactions on Communications* 68.8 (2020), pp. 4734–4746.

[21] Yunlong Lu, Xiaohong Huang, Ke Zhang, Sabita Maharjan, and Yan Zhang. "Communication-efficient federated learning and permissioned blockchain for digital twin edge networks". In: *IEEE Internet of Things Journal* 8.4 (2020), pp. 2276–2288.

[22] Xiaoge Huang, Yuhang Wu, Chengchao Liang, Qianbin Chen, and Jie Zhang. "Distance-aware hierarchical federated learning in blockchain-enabled edge computing network". In: *IEEE Internet of Things Journal* 10.21 (2023), pp. 19163–19176.

[23] Sizheng Fan, Hongbo Zhang, Yuchen Zeng, and Wei Cai. "Hybrid blockchain-based resource trading system for federated learning in edge computing". In: *IEEE Internet of Things Journal* 8.4 (2020), pp. 2252–2264.

[24] Xidi Qu, Shengling Wang, Qin Hu, and Xiuzhen Cheng. "Proof of federated learning: A novel energy-recycling consensus algorithm". In: *IEEE Transactions on Parallel and Distributed Systems* 32.8 (2021), pp. 2074–2085.

[25] Liang Yuan, Qiang He, Siyu Tan, Bo Li, Jiangshan Yu, Feifei Chen, Hai Jin, and Yun Yang. "Coopedge: A decentralized blockchain-based platform for cooperative edge computing". In: *Proceedings of the Web Conference 2021*. 2021, pp. 2245–2257.

[26] Xianglin Bao, Cheng Su, Yan Xiong, Wenchao Huang, and Yifei Hu. "FLChain: A blockchain for auditable federated learning with trust and incentive". In: *2019 5th International Conference on Big Data Computing and Communications (BIGCOM)*. IEEE. 2019, pp. 151–159.

[27] Zihao Zhou, Yanan Li, Xuebin Ren, and Shusen Yang. "Towards efficient and stable K-asynchronous federated learning with unbounded stale gradients on non-IID data". In: *IEEE Transactions on Parallel and Distributed Systems* 33.12 (2022), pp. 3291–3305.

[28] Youqi Li, Fan Li, Song Yang, Pan Zhou, Liehuang Zhu, and Yu Wang. "Three-stage Stackelberg Long-term Incentive Mechanism and Monetization for Mobile Crowdsensing: An Online Learning Approach". In: *IEEE Transactions on Network Science and Engineering (TNSE)* 8.2 (2021), pp. 1385–1398.

[29] Wenji He, Haipeng Yao, Tianle Mai, Fu Wang, and Mohsen Guizani. "Three-Stage Stackelberg Game Enabled Clustered Federated Learning in Heterogeneous UAV Swarms". In: *IEEE Transactions on Vehicular Technology* (2023).

[30] Shaoyong Guo, Yao Dai, Song Guo, Xuesong Qiu, and Feng Qi. "Blockchain meets edge computing: Stackelberg game and double auction based task offloading for mobile blockchain". In: *IEEE Transactions on Vehicular Technology* 69.5 (2020), pp. 5549–5561.

[31] Yanru Zhang, Lanchao Liu, Yunan Gu, Dusit Niyato, Miao Pan, and Zhu Han. "Offloading in software defined network at edge with information asymmetry: A contract theoretical approach". In: *Journal of Signal Processing Systems* 83 (2016), pp. 241–253.

[32] Nguyen H Tran, Wei Bao, Albert Zomaya, Minh NH Nguyen, and Choong Seon Hong. "Federated learning over wireless networks: Optimization model design and analysis". In: *IEEE INFOCOM 2019-IEEE conference on computer communications*. IEEE. 2019, pp. 1387–1395.

[33] Jiawen Kang, Zehui Xiong, Dusit Niyato, Shengli Xie, and Junshan Zhang. "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory". In: *IEEE Internet of Things Journal* 6.6 (2019), pp. 10700–10714.

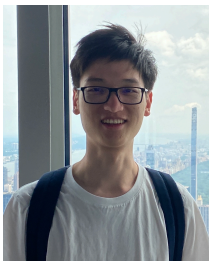
[34] Yuna Jiang, Yi Zhong, and Xiaohu Ge. "IIoT data sharing based on blockchain: A multileader multifollower Stackelberg game approach". In: *IEEE Internet of Things Journal* 9.6 (2021), pp. 4396–4410.

[35] Mengting Liu, F Richard Yu, Yinglei Teng, Victor CM Leung, and Mei Song. "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing". In: *IEEE Transactions on Wireless Communications* 18.1 (2018), pp. 695–708.

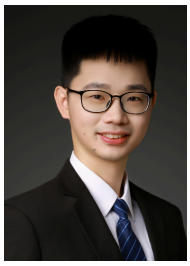
[36] Yufeng Zhan, Peng Li, Zhihao Qu, Deze Zeng, and Song Guo. "A learning-based incentive mechanism for federated learning". In: *IEEE Internet of Things Journal* 7.7 (2020), pp. 6360–6368.

AUTHORS

Xuanzhang Liu received a master's degree in computer science from the University of Delaware in 2020. He is currently pursuing his PhD degree at the Department of Computer and Information Science, Temple University. His research interests include edge computing, blockchain, and federated learning.



Jiyao Liu is a Ph.D. student at the Department of Computer and Information Sciences at Temple University. He received his B.E. degree in information security from North China University of Technology in 2020. His research interests include AI, security, and edge computing.



Xinliang Wei is an assistant professor in Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. He holds a Ph.D. in computer and information services from Temple University, an M.S. and a B.E. in software engineering from SUN Yat-sen University, China. His research interests include edge computing, federated learning, reinforcement learning, and Internet of Things. He is a recipient of *CST Outstanding Research Assistant Award* (2022) and *Scott Hibbs Future of Computing Award* (2023) from Temple University.

Yu Wang is a Professor and Chair of the Department of Computer and Information Sciences at Temple University. He holds a Ph.D. from Illinois Institute of Technology, an MEng and a BEng from Tsinghua University, all in computer science. His research interest includes wireless networks, smart sensing, and distributed computing. He is a recipient of *Ralph E. Powe Junior Faculty Enhancement Awards* from Oak Ridge Associated Universities (2006), *Outstanding Faculty Research Award* from the University of North Carolina at Charlotte (2008), *Fellow of IEEE* (2018), *ACM Distinguished Member* (2020), and *IEEE Benjamin Franklin Key Award* (2024). He has served as an associate editor for *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Cloud Computing*, among others.