

A Platform-Free Proof of Federated Learning Consensus Mechanism for Sustainable Blockchains

Yuntao Wang[†], Haixia Peng[‡], Zhou Su^{†*}, Tom H Luan[†], Abderrahim Benslimane[§], and Yuan Wu[¶]

[†]School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an, China

[‡]School of Information and Communications Engineering, Xi'an Jiaotong University, Xi'an, China

[§]Laboratory of Computer Sciences, Avignon University, France

[¶]State Key Laboratory of Internet of Things for Smart City, University of Macau, Macau, China

*Corresponding author: Zhou Su (zhousu@ieee.org)

Abstract—Proof of work (PoW), as the representative consensus protocol for blockchain, consumes enormous amounts of computation and energy to determine bookkeeping rights among miners but does not achieve any practical purposes. To address the drawback of PoW, we propose a novel energy-recycling consensus mechanism named platform-free proof of federated learning (PF-PoFL), which leverages the computing power originally wasted in solving hard but meaningless PoW puzzles to conduct practical federated learning (FL) tasks. Nevertheless, potential security threats and efficiency concerns may occur due to the untrusted environment and miners' self-interested features. In this paper, by devising a novel block structure, new transaction types, and credit-based incentives, PF-PoFL allows efficient artificial intelligence (AI) task outsourcing, federated mining, model evaluation, and reward distribution in a fully decentralized manner, while resisting spoofing and Sybil attacks. Besides, PF-PoFL equips with a user-level differential privacy mechanism for miners to prevent implicit privacy leakage in training FL models. Furthermore, by considering dynamic miner characteristics (e.g., training samples, non-IID degree, and network delay) under diverse FL tasks, a federation formation game-based mechanism is presented to distributively form the optimized disjoint miner partition structure with Nash-stable convergence. Extensive simulations validate the efficiency and effectiveness of PF-PoFL.

Index Terms—Blockchain, AI-inspired consensus, federated learning, dynamic pool formation.

I. INTRODUCTION

Blockchain, as a disruptive next paradigm innovation, offers a decentralized solution to immutably store information, transparently execute transactions, and automatically establish trust in open and trustless environments [1]–[4]. In blockchain systems, the consensus protocol is the core component which determines the system scalability, security, and consistency. The goal of consensus protocols is to ensure that all involved entities agree on an identical and consistent ledger without the need for a central authority. The representative consensus protocol is the proof of work (PoW) [5], which is widely adopted in various blockchains such as Bitcoin and Ethereum. In PoW, miners compete for the bookkeeping rights and

rewards by solving a cryptographic puzzle (which is hard to solve but easy to validate) through brute-forcing, i.e., *mining* [6]. Unfortunately, the mining process is computation-hungry and consumes an enormous amount of computation and energy in solving meaningless PoW puzzles, controversial to the trend of sustainable and environment-friendly technology. As reported, the annual electricity consumption of Bitcoin is comparable to that of Thailand, and the electricity that a single Bitcoin transaction consumes can power a U.S. household for about 3 weeks [7].

To mitigate the energy waste in PoW, research efforts have been made from two aspects: *energy saving* and *energy recycling*. Proof of stake [8] and proof of space [9] are two typical energy-saving alternatives, which conserve energy by reducing mining difficulty for wealthy stakeholders who invest more in cryptocurrency or storage space. An explicit side effect of these methods is the non-democracy and corresponding Matthew's effects as rich participants have a higher chance for bookkeeping to earn more rewards. Meanwhile, there still exists "useless" resource waste on computation or storage in reaching consensus, remaining far from being really "useful".

Another line of work studies the energy-recycling consensus approaches, where the wasted energy in solving hard but meaningless puzzles in PoW is recycled to perform practical useful works, known as proof of useful work (PoUW) [10]. In PoUW, PoW mining tasks can be replaced with practical computational problems such as searching prime chains [11], image segmentation [12], and all-pairs shortest path [10], which offers a suitable substitution of PoW for better sustainability of the blockchain system. Recently, various works have attempted to design PoUW mechanisms using machine learning as the basis for useful tasks [13], [14]. Essentially, similar to PoW mining, training artificial intelligence (AI) models usually costs considerable computing power and time while its verification process is much easier. Besides, the current trend of going deeper in AI models inevitably results in the ever-increasing demand for computing power. Thereby, it is natural to reinvest the computation power in PoW to train valuable AI models and employ trained models as proofs for completing the consensus.

However, training a qualified AI model requires considerable training data as "fuel", and a single miner or the task initiator may lack sufficient data samples on its device.

This work was supported in part by NSFC (nos. U20A20175, U1808207), the Fundamental Research Funds for the Central Universities, FDCT 0162/2019/A3 and SKL-IOTSC(UM)-2021-2023. This article has been accepted for publication in IEEE Journal on Selected Areas in Communications with DOI 10.1109/JSAC.2022.3213347

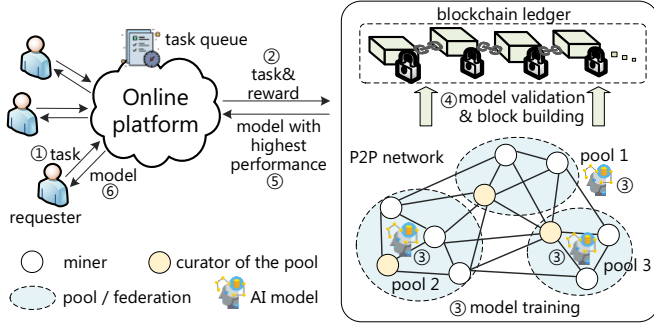


Fig. 1. Architecture of the PoFL scheme [15]. (①: Requesters upload FL tasks with rewarding information to the FL platform; ②: The FL platform maintains a task queue and selects an unfinished task to start PoUW; ③: All members in a mining pool train the requested FL model and each pool uses the trained model as a proof; ④: The blockchain performs model validation and ranking; ⑤: The model with the highest performance is delivered to the platform and the pool who produces it earns the mining reward.)

Moreover, privacy concerns make it risky or even illegal in centralizing data samples from data owners (i.e., miners) for model training [3]. As an attempt to fully unleash the power of AI using distributed data, Qu *et al.* [15] recently introduced a general proof of federated learning (PoFL) consensus framework by reinvesting miners’ computing power in solving meaningless PoW puzzles to train high-quality federated learning (FL) models. FL is a distributed AI paradigm which usually contains (i) a crowd of data owners (e.g., *miners*) who cooperatively train an AI model by sharing intermediate gradients and model parameters instead of their local datasets, and (ii) a central *curator* for learning coordination. Under FL, miners’ training datasets are kept on local devices and only the locally computed model updates are shared, thereby greatly alleviating data privacy concerns. As shown in Fig. 1, in PoFL [15], users first outsource their FL tasks to the online third-party platform (e.g., Kaggle and Codalab), then all the members within a mining pool collaboratively train the requested FL model before the deadline, and finally different pools compete to earn rewards by using the trained models as proofs.

Although the work [15] has made significant progress, there are still significant challenges that remain to be tackled. First, the working of PoFL in [15] heavily depends on the trusted third-party platform which hosts FL tasks issued by requesters, determines the order of tasks for mining, and delivers rewards to the winning pool in the race. The malfunction or breakdown of the platform can cause a system failure, which is essentially a single point of failure (SPoF). Besides, the platform may collude with certain mining pools and disclose the FL tasks to them in an early stage, so that they can start training before others and gain benefits. Therefore, *it remains a concern to design a robust and fully decentralized PoFL scheme which gets rid of the central platform.* Next, the work [15] directly leverages off-the-shelf pooled-mining structures in PoW-based blockchains to train FL models. Different from the homogeneous PoW tasks, miners usually have distinct training samples and non-IID degrees and correspondingly distinct contributions when conducting different

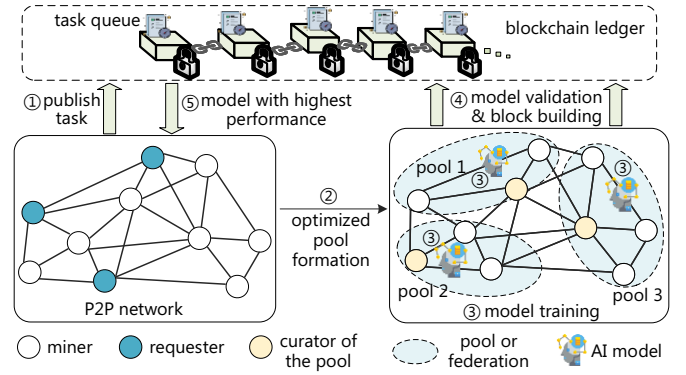


Fig. 2. Architecture of our PF-PoFL scheme. (①: Requesters upload FL tasks with rewards to the blockchain which maintains an unfinished task queue; ②: Miners dynamically form the optimized pool structure to perform PoUW for an unfinished task; ③: All members in a pool train the requested FL model in a differentially private manner and each pool uses the trained model as a proof; ④: Selected validators in blockchain perform model validation and ranking for the uncompleted task; ⑤: The model with the highest performance is delivered to its requester and the pool who produces it earns the mining reward.)

FL tasks under PoFL, causing a more complex pool formation problem. *There exists a challenge in dynamically forming optimized pooled structures with stable miner-pool association for diverse FL tasks.* Furthermore, in the fully distributed and autonomous blockchain system, well-designed incentives for self-interested miners are benign impetuses to build a healthy consensus system with enhanced efficiency. Due to the intrinsic selfishness and behavior diversity of miners, *how to offer precise incentives to heterogenous miners for optimized consensus process is a challenging issue.*

To address these issues, we propose a novel platform-free proof of federated learning (PF-PoFL) scheme to build a sustainable and robust blockchain ecosystem by removing the central platform and forming a dynamically optimized pooled structure for AI model training. As shown in Fig. 2, PF-PoFL involves three types of blockchain nodes: (i) *requesters* that produce FL tasks, (ii) *model trainers* (consisting of *curators* and *miners*) that train FL models within a pool, and (iii) *validators* (selected from miners) that are in charge of model rankings and proposing new blocks. Specifically, requesters directly announce their FL tasks with rewards to the blockchain, and a group of validators collectively maintain an ordered queue of unfinished tasks. For an unfinished task, a crowd of miners with distinct training data size, non-IID degree, and network delay dynamically form a stable pooled structure via the federation formation game. Next, coordinated by the corresponding curator, different pools compete to train the FL model and the pool with the best model earns the task reward via the consensus process. The validators are evenly compensated with the transaction fees. The main contributions of this work are summarized below.

- *Sustainable and robust blockchain framework.* We propose PF-PoFL, an energy-recycling consensus framework which leverages the computing power of blockchain for efficient AI task outsourcing and reward distribution in a fully decentralized manner. Under PF-PoFL, we devise a

novel block structure, new transaction types, and credit-based incentives to support FL model ranking and realize efficient consensus in the blockchain.

- *User-level privacy-preserving federated mining.* We enhance the record-level differential privacy (DP) by designing a user-level DP (UDP) algorithm to prevent the leakage of a user’s whole data records during federated mining. Then, we rigorously analyze the sensitivity bound of a global aggregation and prove that the federated mining process satisfies (ϵ, δ) -UDP by adapting the variance of Gaussian noise added by curators.
- *Game-theoretical optimized pool formation.* We formulate the disjoint pool formation problem for performing distinct FL tasks as a social welfare maximization problem, and propose a distributed federation formation game-based algorithm, which converges to Nash-stable equilibrium. By introducing the switch gain, both miners and pools execute an iterative two-sided matching process, where each miner decides its switch strategy to switch to the optimal pool while each pool decides its admission strategy to accept the optimal miner.
- *Extensive simulations for performance evaluation.* We evaluate the efficiency and effectiveness of PF-PoFL using extensive simulations. Numerical results show that the PF-PoFL can attain not only stable blockchain throughput and desirable model performance with UDP guarantees, but also lower computation cost in reaching consensus and higher efficiency in federated mining, compared with conventional approaches.

The remainder of the paper is organized as follows. In Section II, we review the related works. Section III introduces the system model. Section IV provides a detailed construction of the PF-PoFL framework. We analyze the game-theoretical stable pool formulation process in Section V. We evaluate the performance of our proposed framework in Section VI. Section VII concludes this paper.

II. RELATED WORKS

In the literature, there has been an increasing interest in designing energy-recycling consensus mechanisms by substituting the hash puzzles in PoW with valuable works which are hard to solve but easy to verify. Initially, early efforts focus on replacing the PoW nonce with practical computational problems. Ball *et al.* [10] proposed a novel concept proof of useful work (PoUW) to replace the PoW mining with mathematic problems such as all-pairs shortest path and orthogonal vectors. King [11] proposed a novel cryptocurrency named Primecoin to generate scientific value in the mining process, where miners are required to search long prime chains as proofs of work with difficulty adjustability and non-reusability. Shoker [16] devised a sustainable consensus protocol named proof of exercise, in which miners choose to solve matrix-oriented computation issues hosted on a third-party platform instead of calculating hashes in PoW. Daian *et al.* [17] developed PieceWork to mitigate the energy waste in PoW by solving outsourced tasks such as denial-of-service (DoS) defense and spam prevention.

Recently, many attempts have been made in the integration of AI and blockchain by exploiting the computing capacity of blockchain for AI model training. Li *et al.* [12] proposed a PoUW mechanism to reuse the wasted computing power in PoW for deep learning (DL)-based biomedical image segmentation to help clinical diagnosis. Chenli *et al.* [14] developed a proof of deep learning (PoDL) consensus protocol by forcing miners to perform DL training and employing the trained models as proofs to earn rewards for outsourced DL tasks. Lan *et al.* [13] devised an efficient proof of learning consensus mechanism to ensure data integrity and prevent malicious behaviors without sacrificing model performance. Nevertheless, a single user may lack sufficient training data, and the centrally aggregated training/test dataset for public model training/verification may sacrifice user privacy and frustrate the adoption of blockchain.

Distinguished from the above works, our work combines blockchain and AI under the FL paradigm in a fully decentralized manner with privacy preservation and dynamic pool formation functions.

III. SYSTEM MODEL

We introduce the system model in this section, which consists of the blockchain model and adversary model. Then, we discuss the desired properties of PF-PoFL.

A. Blockchain Model

The proposed decentralized blockchain network in this paper mainly consists of three types of nodes: requesters, trainers, and validators, as illustrated in Fig. 3.

- *Requesters.* Every node in the blockchain can act as a task requester to publish its FL tasks (e.g., semantic analysis and biomedical image recognition) to the blockchain platform with specific completion deadline and performance requirements, as well as the corresponding rewards as incentives. Let $\mathcal{R} = \{1, \dots, r, \dots, R\}$ be the set of task requesters in the blockchain network. The set of published tasks and unfinished tasks in the blockchain are denoted as $\mathcal{T} = \{1, \dots, t, \dots, T\}$ and $\mathcal{T}_u = \{1, \dots, \tau, \dots, T_u\}$, respectively.
- *Trainers.* Analogy to the pooled-mining in PoW-based blockchains such as Bitcoin, a set of model trainers in PF-PoFL can form the pooled (or federated) structure [15], which is featured with intra-pool cooperation and inter-pool competition. A set of pools or federations are formed in the blockchain network, and the set of which is defined as $\mathfrak{S} = \{\mathfrak{S}_1, \dots, \mathfrak{S}_j, \dots, \mathfrak{S}_J\}$. In each pool/federation $\mathfrak{S}_j \in \mathfrak{S}$ managed by the corresponding curator ϕ_j , a group of miners, denoted as $\mathcal{M}_j = \{1, \dots, m, \dots, M_j\}$, cooperatively train a qualified FL model for an unfinished task $\tau \in \mathcal{T}_u$ within the pool, and compete to earn the task reward with other pools in $\mathfrak{S} \setminus \{\mathfrak{S}_j\}$.
- *Validators.* In PF-PoFL, validators are a group of consensus nodes, denoted as $\mathcal{V} = \{1, \dots, v, \dots, V\}$, which are responsible for transaction verification and new block construction. An ordered queue of uncompleted tasks, denoted as $\widetilde{\mathcal{T}}_u = \text{ordered}(\mathcal{T}_u)$, is collectively maintained

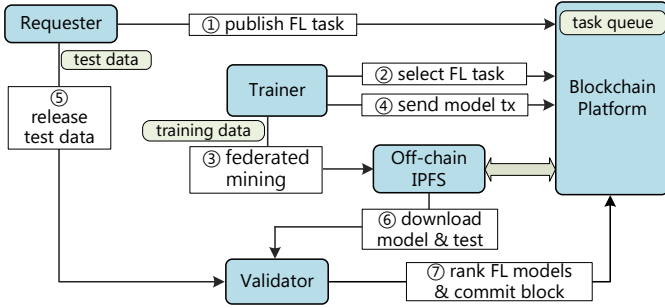


Fig. 3. Entities and workflow of PF-PoFL.

by validators and is updated before the block height grows. Validators also perform the evaluation and ranking operations for released FL models, which are trained by each pool, via the consensus process.

In the blockchain network, a node can simultaneously perform all three roles or two of them. The set of all nodes in blockchain is denoted as $\mathcal{N} = \{1, \dots, n, \dots, N\}$. The blockchain ledger \mathcal{B} is composed of a series of growing hash-chained blocks, i.e., $\mathcal{B} = \{\mathcal{B}_i \leftarrow \mathcal{B}_{i-1} \leftarrow \dots \leftarrow \mathcal{B}_0\}$, where \mathcal{B}_0 is called the genesis block, and \mathcal{B}_i is the latest block with height i . Specifically, \mathcal{B}_i contains two parts (i.e., block header and block body) and can be denoted by

$$\mathcal{B}_i = \left\{ \underbrace{\text{Hash}(\mathcal{B}_{i-1}) || T_{\text{stamp}} || \text{root}_i || \text{meta}_i}_{\text{header}} || \underbrace{\mathcal{T}\mathcal{X}_i}_{\text{body}} \right\}. \quad (1)$$

In Eq. (1), $\mathcal{T}\mathcal{X}_i$ is a set of timestamped transactions in the block, which is organized into a Merkle tree structure with root value root_i . $\text{Hash}(\mathcal{B}_{i-1})$ is a cryptographic hash of the previous block to maintain a chained structure. T_{stamp} is the timestamp for block creation. meta_i is the metadata for validating and ranking the trained FL models, which ensures the validity of block \mathcal{B}_i (details are shown in Sect. IV-F). The metadata meta_i can be analogical to the random nonce in Bitcoin, which serves a verifiable proof of the work.

B. Adversary Model and Assumptions

In the system, model trainers (i.e., the curator ϕ_j and miners in \mathcal{M}_j) in each pool $\mathfrak{S}_j \in \mathfrak{S}$ are assumed to be *semi-honest*. In other words, they will honestly obey the FL procedure in AI model training, while the miners may plagiarize others' trained FL models and the curator may be curious about miners' privacy information and refuse to pay at last. Besides, the blockchain platform is assumed to be *secure* and its stored transactions are *immutable* and *non-repudiable*. Particularly, the following four kinds of attacks are considered:

Spoofing Attack. The model trainers may carry out spoofing attacks to gain high illegal profits. For example, they may publish a perfect model by training on the released testing data to win the competition with paltry cost. They could also plagiarize the FL models trained by other pools [15].

Sybil Attack. During FL model training, malicious trainers may launch Sybil attacks by submitting multiple meaningless local updates (to deteriorate model performance) or uploading the same local update multiple times (to earn more benefits).

After FL model training, malicious pools may submit multiple meaningless models (to waste system resources) or send the same model multiple times (to earn more benefits), using Sybil identities. Besides, adversarial validators may create an arbitrary number of Sybil pseudonyms to influence the process of reaching an agreement on blockchain ledgers [18].

Untrusted Third-Party Platform. In traditional PoFL [15], the central FL platform operated by third parties is responsible for hosting and ranking the FL tasks to be performed, as well as the payment delivery, whose malfunction or breakdown can result in a SPoF. The malicious platform may also collude with part of miners to conduct market manipulation by revealing FL tasks to them in advance. In addition, the platform may be compromised by attacks such as DDoS, causing leakage of miners' sensitive information that is stored in it.

Implicit Privacy Leakage. According to [19]–[21], user privacy (e.g., the participation status in a task and the data used for training) may still be leaked unconsciously under FL in sharing raw local updates through differential attacks and advanced inference attacks.

C. Desired Properties

The design goal of the PF-PoFL is to achieve the following desirable properties simultaneously.

1) Fully decentralized operation. The whole process of PF-PoFL including FL task outsourcing, AI model training, model evaluation and ranking, and reward distribution should be operated in a fully decentralized paradigm.

2) Dynamically optimized pool structure. PF-PoFL aims for an optimized pool structure to competitively train FL tasks by considering dynamic user characteristics such as miners' cooperation and competition.

3) Spoofing and Sybil prevention. PF-PoFL should resist the Sybil identities and defend against spoofing attacks.

4) User-level differential privacy (UDP). Existing works in FL mainly focus on the record-level differential privacy (RDP) [22]–[24], namely, whether a certain data sample is used for training. Different from them, we focus on the user-level privacy protection, which offers stronger provisions to protect the whole data samples of a user, i.e., the learned model does not leak whether a user participates in FL or not.

IV. PF-POFL: PLATFORM-FREE PROOF OF FEDERATED LEARNING FRAMEWORK

A. Design Overview

The workflow of PF-PoFL consensus framework consists of four successive phases: (i) FL task publication, (ii) FL model training, (iii) model ranking and rewarding, and (iv) block building and commit.

- *FL task publication.* Requesters publish their FL tasks \mathcal{T} with rewards to the blockchain platform which maintains an ordered queue of uncompleted tasks $\widetilde{\mathcal{T}}_u$.
- *Federated mining with UDP.* A group of trainers form an optimized pool/federation structure \mathfrak{S} and choose an uncompleted task $\tau \in \widetilde{\mathcal{T}}_u$ to perform model training within the pool $\mathfrak{S}_j \in \mathfrak{S}$ under FL in a privacy-preserving

manner. Different pools compete to train and submit the best FL model before the task deadline.

- **Model ranking and rewarding.** The validators mutually execute the model ranking contract to evaluate and rank the trained FL models. Besides, validators perform the block rewarding contract to automatically enforce financial rewarding and update credits for participants.
- **Block building and commit.** Each validator executes the credit-based Algorand Byzantine agreement protocol to efficiently reach consensus on the new block to be added into the blockchain with deterministic finality.

In PF-PoFL, three types of blockchain transactions are implemented as follows.

- **Payment Transaction** refers to transferring or redeeming the cryptocurrency from one node to another node for task payment, participation fee, etc.
- **Task Publication Transaction** in which a task requester proposes its FL tasks for learning competition.
- **FL Model Transaction** in which a training pool uploads its trained FL model as a solution for a particular FL task.

B. System Initialization

For system initialization, a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ is chosen, where \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_3 are groups of prime order q . The generators of \mathbb{G}_1 and \mathbb{G}_2 are g_1 and g_2 , respectively. Two hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_2$ and $H_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ are selected as random oracles. Taking a security parameter φ as an input, the bilinear group generator $\mathbb{G}(\varphi)$ outputs the system parameters as $params = (q, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e)$. Both $\mathbb{G}(\cdot)$ and $params$ are recorded in the genesis block.

Based on these public information, each participant $n \in \mathcal{N}$ sets up its secret key in the blockchain via $sk_n \xleftarrow{\mathbb{R}} \mathbb{Z}_q$ and generates its public key via $pk_n \leftarrow g_2^{sk_n}$, where $\xleftarrow{\mathbb{R}}$ represents randomly sampling. The wallet address of each node n can be computed as $wa_n \leftarrow H_2(pk_n)$. Each blockchain node n stores (sk_n, pk_n, wa_n) into its local tamper-proof device.

C. FL Task Publication

In PF-PoFL, the machine learning problems such as sentiment analysis and image classification are produced by the requesters along with the corresponding rewards. Specifically, each requester $r \in \mathcal{R}$ can publish an AI task $t \in \mathcal{T}$ at any time by sending a task publication transaction tx_t to the blockchain. The detailed transaction format is as below.

- **Inputs:**
 - Task reward p_t and task hosting fee ξ_1 of task t ;
 - Initial AI model parameters $\Theta_t(0)$;
 - Performance metrics ψ to evaluate model performance such as accuracy;
 - The testing release block height i_t , i.e., the deadline (measured by block height [25]) to release test dataset $\mathcal{D}_{\text{test}}$;
 - Cryptographic hash of the test dataset $dID \leftarrow \text{Hash}(\mathcal{D}_{\text{test}})$.
- **Outputs:** Task publication transaction tx_t :

$$tx_t = \left\{ tx_t^{\text{ID}}, p_t, \xi_1, \Theta_t(0), \psi, i_t, dID, T_{\text{stamp}}, pk_r, \text{Sig}_{sk_r}(tx_t^{\text{ID}}) \right\}, \quad (2)$$

where tx_t^{ID} is the unique transaction identifier, i.e., hash of the transaction tx_t . T_{stamp} is the timestamp for transaction generation. $\text{Sig}_{sk_r}(tx_t^{\text{ID}}) \leftarrow H_1(tx_t)^{sk_r}$ is the digital signature of requester r on the transaction. A small task hosting fee ξ_1 is utilized to encourage the validators to package tx_t into the block and prevent malicious requesters from sending multiple meaningless tasks to cause a DoS.

The task publication transaction tx_t also contains a payment transaction in which the requester transfers the task reward p_t and task hosting fee ξ_1 to an escrow address $esAdd$ under public supervision. The formal transaction format is as follows.

- **Inputs:** Task identifier tx_t^{ID} and user identifier pk_r .
- **Outputs:** Payment transaction tx_p :

$$tx_p = \left\{ tx_p^{\text{ID}}, tx_t^{\text{ID}}, p_t, \xi_1, T_{\text{stamp}}, pk_r, \text{Sig}_{sk_r}(tx_p^{\text{ID}}) \right\}, \quad (3)$$

where tx_p^{ID} is the identifier (i.e., hash) of the transaction tx_p . Here, if the requester fails to release the test dataset for task t after the testing release block height i_t , it will lose these cryptocurrencies.

After receiving the task and payment transactions, validators mutually validate each of them via the function $verify(tx_t)$ by checking: (i) the task requester r holds enough funds to pay the task hosting fee ξ_1 and reward p_t ; and (ii) the difference between the testing release block height i_t and the current block height i is positive and larger than the minimum model training period preset by the system. If verification passes, we have $verify(tx_t) = true$, otherwise $verify(tx_t) = false$. The invalid transactions are discarded and only valid ones are forwarded to the network.

For all valid tasks to be completed, each validator $v \in \mathcal{V}$ independently ranks these tasks in ascending order of timestamp and maintains a ready queue $\widetilde{\mathcal{T}}_u$ of uncompleted tasks. Generally, the task reward reflects the importance and urgency of a machine learning task. As nodes in blockchain are selfish and rational, it is reasonable to assume that the priority of each running task t for miners is determined by joint effect of the announced task reward p_t and the remaining task training duration $i_t - i$. In other words, miners prefer FL tasks with higher rewards and longer remaining training duration. For the case that multiple tasks have the same priority, these tasks are ordered by their publication timestamp, i.e., the earlier-arrived one gets a better ranking. After reaching an agreement among validators, the ready queue $\widetilde{\mathcal{T}}_u$ is updated and recorded in the blockchain. In practice, we further implement a soft handover mechanism in which the PF-PoFL will be rolled back to PoW once the ready queue is empty.

D. Federated Mining with UDP

Currently, there is a trend of pooled-mining in blockchain, which has a similar clustering structure with FL [15]. In this work, we study PF-PoFL under the pooled-mining structure, as shown in Fig. 4. The detailed federated mining process consists of the following steps.

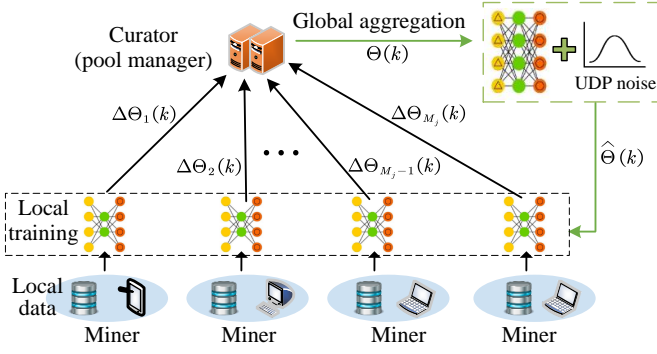


Fig. 4. Illustration of federated mining with UDP in a pool under FL.

Step 1: Task selection & pool formation. Each trainer can pick any task $\tau \in \widetilde{\mathcal{T}}_u$ published on the blockchain which is still in its training phase (i.e., $i \leq i_\tau$). Let $\mathcal{M}_\tau \subseteq \mathcal{M}$ be the set of trainers or miners that involve in training FL task τ . All trainers in the set \mathcal{M}_τ dynamically form an optimized stable pooled structure \mathfrak{S}^* (details are shown in Sect. V) in a distributed manner. For each pool $\mathfrak{S}_j \in \mathfrak{S}^*$, a group of trainers within the pool collaboratively train a qualified AI model for that task under the FL paradigm using their private data. Besides, in each pool $\mathfrak{S}_j \in \mathfrak{S}^*$, a curator ϕ_j is randomly selected from the pool members whose experienced network latency is less than a predefined threshold D_{th} . Distinguished with the PoW protocol, in our PF-PoFL consensus protocol, different mining pools can concurrently train on different FL tasks that are in the training phase.

Step 2: FL model training with UDP within a pool. In pooled-mining under FL, as depicted in Fig. 4, each pool member (i.e., miner) computes the intermediate gradients (i.e., local model update) of the current global model based on its local dataset and uploads them to the curator (i.e., pool manager) for periodic global aggregation (i.e., global model update). Let $\Psi_{j,\tau}^{global}$ be the number of communication rounds in training task τ . Initially, each miner in pool \mathfrak{S}_j initializes its local model parameters as the initial model parameters $\Theta_t(0)$ downloaded from the transaction tx_τ . At k -th communication round ($k = 1, 2, \dots, \Psi_{j,\tau}^{global}$), it consists of a parallel local training stage on miners followed by a global aggregation and perturbation stage on the curator.

1) *Local training at miner's side.* After receiving the previous global model $\widehat{\Theta}(k-1)$, each miner $m \in \mathfrak{S}_j$ individually calculates its local model $\Theta_m(k)$ using its local dataset via mini-batch stochastic gradient descent (SGD) with learning rate η , i.e.,

$$\Theta_m(k) \leftarrow \widehat{\Theta}(k-1) - \eta \nabla \mathcal{L}(\widehat{\Theta}(k-1)), \quad (4)$$

where $\nabla \mathcal{L}$ is the gradient of loss function $\mathcal{L}(\cdot)$ on the mini-batch. Then miner m calculates the local updates by

$$\Delta \Theta_m(k) \leftarrow \Theta_m(k) - \widehat{\Theta}(k-1), \quad (5)$$

and uploads $\Delta \Theta_m(k)$ to the curator.

2) *Global aggregation and perturbation at curator's side.* The curator ϕ_j aggregates the local updates into a global model $\Theta(k)$. Besides, to ensure user-level privacy preservation under

FL, the curator applies a randomized function \mathcal{F} to add the UDP noise to the sum of scaled local updates.

The above training process is repeated until a desirable accuracy of the global model is attained or the communication round reaches its maximum value. In the following, we first give the definition of UDP and then present the detailed UDP implementation mechanism.

Definition 1. (User-level Differential Privacy (UDP)): A randomized function $\mathcal{F} : \mathcal{D} \rightarrow \mathcal{R}$ satisfies (ϵ, δ) -UDP if for any two user-adjacent datasets $y, y' \in \mathcal{D}$ and for any subset of outputs $\mathcal{Y} \subseteq \mathcal{R}$, the following inequality holds:

$$\Pr[\mathcal{F}(y) \in \mathcal{Y}] \leq e^\epsilon \Pr[\mathcal{F}(y') \in \mathcal{Y}] + \delta, \quad (6)$$

where \mathcal{D} and \mathcal{R} are the domain (e.g., possible training datasets) and range (e.g., possible trained global models) of \mathcal{F} , respectively. Two datasets y, y' are user-neighboring if y can be formed from y' by removing or adding all data samples related to a single miner.

Remark 1: In conventional record-level DP (RDP) mechanisms [22]–[24], the datasets y, y' are defined to be record-neighboring, i.e., y can be formed from y' by removing or adding a single data record. Instead of protecting a single data sample of the miner, we design a UDP mechanism \mathcal{F} to protect the miner's whole data samples in the training process. As such, any adversary observing the published final model cannot deduce the participation of any specific miner and the usage of any specific data samples in model training with strong probability.

The Gaussian mechanism is adopted to design such function \mathcal{F} by sanitizing the sum of all updates with low utility decrease. To enforce the bounded sensitivity of local updates, the curator ϕ_j scales each local update by

$$\Delta \tilde{\Theta}_m(k) \leftarrow \frac{\Delta \Theta_m(k)}{\max\left\{1, \frac{\|\Delta \Theta_m(k)\|_2}{A}\right\}}. \quad (7)$$

Thereby, $\|\Delta \tilde{\Theta}_m(k)\|_2 \leq A$ can be ensured, and the sensitivity in the summing operation of all scaled updates is upper bounded by A . According to [26], we choose $A = \text{median}\{\|\Delta \Theta_m(k)\|_2 : m \in \mathcal{M}_j \cup \{\phi_j\}\}$.

The Gaussian noise scaled to sensitivity A with zero mean, i.e., $\mathcal{G}(0, \sigma^2 A^2)$, is added to the sum of the scaled updates to prevent individual privacy leakage, i.e.,

$$\widehat{\Theta}(k) \leftarrow \widehat{\Theta}(k-1) + \frac{1}{M_j+1} \left[\sum_{m=1}^{|\mathcal{M}_j \cup \{\phi_j\}|} \Delta \tilde{\Theta}_m(k) + \mathcal{G}(0, \sigma^2 A^2) \right], \quad (8)$$

where the parameter σ controls the scale of Gaussian noise. After that, the curator ϕ_j delivers the perturbed global model $\widehat{\Theta}(k)$ to all miners in the pool.

Step 3: FL model submission. Once pool \mathfrak{S}_j accomplishes the FL learning process, the curator ϕ_j as the pool manager submits its trained final model $\Theta_{j,\tau}$ by sending a FL model transaction, whose formal format is as follows.

- Inputs:
 - Hash of the trained model $\Theta_{j,\tau}$, i.e., $\text{mID} \leftarrow \text{Hash}(\Theta_{j,\tau})$;

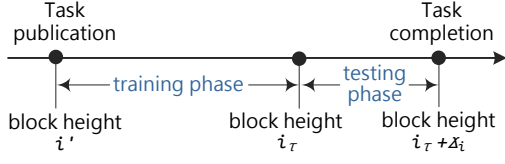


Fig. 5. Illustration of training and testing phases of a FL task in PF-PoFL.

- Reference score rs in model training phase;
- Participation fee ξ_2 ;
- Aggregated public key apk_j of trainers in the pool;
- Multi-signature $\text{MulSig}(tx_m^{\text{ID}})$ of trainers in the pool.
- Outputs: FL model transaction tx_m :

$$tx_m = \left\{ tx_m^{\text{ID}}, tx_\tau^{\text{ID}}, \text{mID}, rs, \xi_2, T_{\text{stamp}}, apk_j, \text{MulSig}(tx_m^{\text{ID}}) \right\}, \quad (9)$$

where tx_m^{ID} is the transaction identifier. Notably, each mining pool only submits the hash value of the trained AI model (i.e., mID) at this stage, to prevent from being plagiarized by other rivals. Once the test dataset is released, all involved pools will upload their trained models to an off-chain IPFS platform. Besides, the training score rs can be a reference value of model performance and is not considered for model ranking. In addition, a small participation fee ξ_2 is involved in tx_m to defend against Sybil attacks conducted by malicious trainers/pools during/after FL model training. The participation fee ξ_2 is evenly assigned to all members within each pool, which can be redeemed to the trainers whose model performance is above a certain threshold in consensus model ranking. Similar to transaction tx_τ , a payment transaction is involved, i.e.,

$$tx_p = \left\{ tx_p^{\text{ID}}, tx_m^{\text{ID}}, \xi_2, T_{\text{stamp}}, pk_{\phi_j}, \text{Sig}_{sk_{\phi_j}}(tx_p^{\text{ID}}) \right\}. \quad (10)$$

For trainers in pool \mathfrak{S}_j , their aggregated public key can be derived as

$$apk_j = \prod_{n \in \mathcal{M}_j \cup \{\phi_j\}} pk_n^{\text{H}_2(pk_n, \{pk_1, \dots, pk_{M_j}, pk_{\phi_j}\})}. \quad (11)$$

We define $\varsigma_n \leftarrow \text{H}_2(pk_n, \{pk_1, \dots, pk_{M_j}, pk_{\phi_j}\})$ to ease the formulation. For each pool member $n \in \mathcal{M}_j \cup \{\phi_j\}$, its signature can be computed as $\text{Sig}_{sk_n}(tx_m^{\text{ID}}) \leftarrow \text{H}_1(tx_m^{\text{ID}})^{\varsigma_n sk_n}$. Within a pool \mathfrak{S}_j , the multi-signature can be computed by aggregating all members' signatures [27], i.e.,

$$\text{MulSig}(tx_m^{\text{ID}}) \leftarrow \prod_{n \in \mathcal{M}_j \cup \{\phi_j\}} \text{Sig}_{sk_n}(tx_m^{\text{ID}}). \quad (12)$$

The correctness of multi-signature can be verified by checking whether $e(\text{MulSig}(msg), g_2^{-1}) \cdot e(\text{H}_1(msg), apk_j) = 1_{\mathbb{G}_3}$.

Once the training phase for a task τ has elapsed (i.e., current block height $i \geq i_\tau$), the task requester proceeds to submit the corresponding test dataset $\mathcal{D}_{\text{test}}$ identified by the previously published hash value dID to the off-chain data storage. The InterPlanetary File System (IPFS) [28] is adopted to serve as the distributed off-chain data store, where data is stored in the form of distributed files and is uniquely addressed through the hash pointer of the data. After that, all involved pools proceed to upload their FL models to the IPFS identified by the hash

Algorithm 1 Model Ranking Contract

- 1: **Input:** 1) Task publication transaction tx_τ ; 2) Stable pooled structure \mathfrak{S}^* ; 3) Model transactions tx_m and payment transactions tx_p related to the tx_τ .
- 2: **Output:** Model ranking $mrList_\tau$.
- 3: Extract $\{tx_\tau^{\text{ID}}, p_\tau, \xi_1, \psi, i_\tau, \text{dID}\} \leftarrow tx_\tau$.
- 4: If $\text{verify}(tx_\tau) = \text{true}$, proceed to line 5, otherwise terminate.
- 5: Add the model transactions tx_m and payment transactions tx_p related to the tx_τ into the local memory pool.
- 6: **for** $tx_m \in \mathcal{T}\mathcal{X}_m^\tau$ **do**
- 7: Extract $\{\text{mID}, rs, \xi_2, apk_j\} \leftarrow tx_m$
- 8: If $\text{deposit}_j \geq \xi_2$ && $i_m \leq i_\tau$, proceed to line 9, otherwise terminate.
- 9: **if** the model $\Theta_{j,\tau}$ identified by mID is successfully downloaded from IPFS **then**
- 10: **if** the current height $i_\tau \leq i \leq i_\tau + \Delta_i$ **then**
- 11: Download task τ 's test dataset $\mathcal{D}_{\text{test}}$ identified by dID.
- 12: Evaluate performance score $ts_{j,\tau}$ on the test dataset $\mathcal{D}_{\text{test}}$.
- 13: Insert $\langle \text{mID}, ts_{j,\tau} \rangle$ into the local model ranking $mrList_\tau$ depending on the performance metric.
- 14: **else if** the current height $i < i_\tau$ **then**
- 15: Wait until $i \geq i_\tau$, i.e., reaching the test phase.
- 16: **else if** the current height $i > i_\tau + \Delta_i$ **then**
- 17: Terminate.
- 18: **end if**
- 19: **else**
- 20: Set $ts_{j,\tau} = \text{"unevaluated"}$.
- 21: Insert $\langle \text{mID}, \text{"unevaluated"} \rangle$ into $mrList_\tau$.
- 22: **end if**
- 23: **end for**

value mID. The training and testing phases for a FL task are depicted in Fig. 5. To prevent malicious trainers from training a model using the released test data, the model transactions are regarded as invalid if they are generated during the task testing phase (i.e., the current block height is larger than the testing release block height).

E. Model Ranking and Rewarding

For each FL task to be completed, the more trained models associated with it, the higher chance for validators to earn more participation fees. Consequently, for all running tasks that are in the testing phase, it is reasonable to assume that validators are incentivized to reach consensus on the most urgent and profitable task (i.e., the task with the largest number of model transactions and shortest remaining test duration). The validators in the set \mathcal{V} independently execute the model ranking contract and block rewarding contract to compute the model ranking and complete financial settlements. The model ranking contract and block rewarding contract are shown in Algorithms 1 and 2, respectively.

Step 1: Model ranking. As shown in Algorithm 1, each validator $v \in \mathcal{V}$ first validates the task publication transaction

Algorithm 2 Block Rewarding Contract

- 1: **Input:** 1) Task publication transaction tx_τ ; 2) Stable pooled structure \mathfrak{S}^* and curator $\phi_j, \forall \mathfrak{S}_j \in \mathfrak{S}^*$; 3) Consensus ranking $mrList_\tau$.
 - 2: **Output:** Financial rewarding transactions and node credits.
 - 3: Transfer $p_\tau \rightarrow wa_{j^*}$ from $esAdd$ to the pool \mathfrak{S}_{j^*} that ranks best in $mrList_\tau$.
 - 4: Transfer $\xi_1 \cdot \frac{1}{|\mathcal{V}'|} + \xi_2 \cdot \frac{|\mathfrak{S}^* \setminus \mathcal{W}_\tau|}{|\mathcal{V}'|} \rightarrow wa_v$ from $esAdd$ to all non-faulty validators in the set \mathcal{V}' .
 - 5: Transfer $\xi_2 \rightarrow wa_j$ from $esAdd$ to all pools in \mathcal{W}_τ whose model is performed above the preset threshold.
 - 6: Update $cre_v \leftarrow cre_v + \chi_1, \forall v \in \mathcal{V}'$.
 - 7: Update $cre_m \leftarrow cre_m + \chi_2, \forall m \in \mathfrak{S}_{j^*}$.
 - 8: Update $cre_{m'} \leftarrow cre_{m'} + \chi_3, \forall m' \in \mathfrak{S}_j \subseteq \mathcal{W}_\tau, j \neq j^*$.
-

tx_τ by using $verify(tx_\tau)$ defined in Sect. IV-C (lines 3-4). Then, validator v adds all model transactions tx_m and payment transactions tx_p associated with the unfinished FL task $\tau \in \widetilde{\mathcal{T}}_u$ into its memory pool (line 5). For each model transaction tx_m issued by pool \mathfrak{S}_j , validator v verifies (i) whether the deposit $deposit_j$ is no less than the participation fee ξ_2 , and (ii) whether the issuing time (measured by block height) of transaction tx_m (i.e., i_m) is no larger than the testing release block height i_τ (line 8). For all valid model transactions, validator v downloads the corresponding FL models identified by the hash pointer mID from the off-chain IPFS (line 9).

Once the task τ elapses its training phase and enters the testing phase (i.e., $i_\tau \leq i \leq i_\tau + \Delta_i$), validator v downloads the test dataset $\mathcal{D}_{\text{test}}$ identified by the previously published hash pointer dID from the IPFS and evaluates all the downloaded models (line 11). Here, Δ_i is the time duration (measured by block height) of the testing phase. For each model $\Theta_{j,\tau}$, it computes the performance score $ts_{j,\tau}$ on the test dataset $\mathcal{D}_{\text{test}}$ (line 12). Based on the performance scores, each validator creates a candidate model ranking $mrList_\tau$, i.e., an ordered list of model-score pairs on the testing data, which is sorted either in descending or ascending order depending on the specific metrics ψ (line 13). Due to network latency, models that cannot be timely evaluated during the time limit Δ_i obtain a special “unevaluated” score in $mrList_\tau$ (lines 20–21).

Step 2: Block rewarding. After reaching consensus on the model ranking for the selected task $\tau \in \widetilde{\mathcal{T}}_u$, validators will independently carry out the block rewarding procedure to enforce the reward and punishment for blockchain nodes, as shown in Algorithm 2. Specifically, the following financial transactions for rewarding and punishment will be added to the candidate block built by the validator.

1) A payment transaction transferring the task reward p_τ escrowed by the escrow address $esAdd$ to the wallet of the pool \mathfrak{S}_{j^*} that owns the top-performing model in the consensus ranking $mrList_\tau$ (line 3).

2) A payment transaction transferring the task hosting fee ξ_1 from $esAdd$ to all honest validators in the set \mathcal{V}' (line 4). The fee ξ_1 is evenly distributed among all honest validators in \mathcal{V}' .

3) A payment transaction redeeming the participation fee

ξ_2 to all pools in \mathcal{W}_τ whose models are performed above the preset threshold such as first quartile (line 5).

4) A payment transaction transferring the participation fee ξ_2 of all pools in $\mathfrak{S}^* \setminus \mathcal{W}_\tau$ whose models are either invalid or performed below the preset threshold to all honest validators (line 4). The total fees $\xi_2 \cdot |\mathfrak{S}^* \setminus \mathcal{W}_\tau|$ are evenly divided.

Apart from the economic rewards, the credits (as a virtual token) can offer incentives for blockchain nodes. The credit values of blockchain nodes can reflect their honest active participation statuses [29]. After reaching consensus, the current credit value of the honest validators, members in the winning pool \mathfrak{S}_{j^*} , and trainers in \mathcal{W}_τ will be increased by χ_1 , χ_2 , and χ_3 , respectively (lines 6–8). Here, $\chi_k > 0$ ($k = 1, 2, 3$).

F. Block Building and Commit

A validator committee \mathcal{V} is responsible for proposing and adding a new block to the blockchain by carrying out the proposed credit-based Algorand Byzantine agreement (BA) protocol with the following steps.

Step 1: Validator committee formation. In conventional Algorand BA protocol [18], nodes are weighted based on their account balance which can be only used for cryptocurrency applications. Instead, in our work, each node is weighted based on its owned credit value. Thereby, our work is also applicable to non-cryptocurrency applications and can encourage nodes’ participation willingness and honesty. Specifically, members of the validator committee are randomly chosen based on their weights via a cryptographic sortition approach implemented by a verifiable random function (VRF).

A simple implementation of VRF using digital signature and hash function is adopted. At the beginning stage $s = 1$, each node $n \in \text{PK}(i - k) \subseteq \mathcal{N}$ independently validates whether it is a member of committee $\mathcal{V}_{i,s}$ by checking

$$\text{Hash}(\text{Sig}_{sk_n}(i, s, seed_i)) \leq \frac{cre_n^i}{C} \cdot \frac{|\mathcal{V}_{i,s}|}{\#\text{PK}(i-k)}. \quad (13)$$

In Eq. (13), $seed_i$ is a public random seed for height i which is included in the previous block with height $i - 1$. cre_n^i is the current credit value of node n . $C = \sum_{n \in \mathcal{N}} cre_n^i$ is the total credit of blockchain nodes. $\#\text{PK}(i - k)$ represents the number of public keys involved in consensus process between heights $i - k$ and i . The signature $\sigma_n^{i,s} = \text{Sig}_{sk_n}(i, s, seed_i)$ means the credential of node n to prove its role as a validator in $\mathcal{V}_{i,s}$ by revealing its public key pk_n . $\text{Hash}(\sigma_n^{i,s})$ is a random 256-bit long string uniquely determined by sk_n and i , which is indistinguishable from random to any node that does not know the private key sk_n . The symbol “ \leq ” indicates the transformation of hash value into a decimal in $(0, 1]$.

Step 2: Candidate block building and propagation. All members in $\mathcal{V}_{i,1}$ are responsible for building candidate blocks and disseminating them to the whole network for verification. The validator $v \in \mathcal{V}_{i,1}$ validates each transaction (i.e., tx_τ , tx_p , and tx_m) in its memory pool that has not been included in the blockchain and is associated with an unfinished FL task $\tau \in \widetilde{\mathcal{T}}_u$. After that, it builds a candidate ranking $mrList_v$ according to the model ranking contract in Algorithm 1. Then it orders all valid ones by timestamps, compresses them into a Merkle tree, and constructs a candidate block \mathcal{B}_v^{ca} .

Step 3: Mutual verification via multi-stage voting. The members in committee $\mathcal{V}_{i,s}$ are replaced in each consensus stage s at height i to prevent being targeted by adversaries. At stage $s > 1$ of height i , $n \in \text{PK}(i-k)$ is a validator in $\mathcal{V}_{i,s}$, if formula (13) holds. To mitigate block propagation delay, validators in $\mathcal{V}_{i,s}$ independently vote for hashes of candidate blocks, instead of the entire block proposal. The final agreement can be reached via the following two phases.

Phase 1. All validators in $\mathcal{V}_{i,2}$ runs the (N, f) -graded consensus (GC) protocol [30] to determine a value-grade pair (μ_v, ρ_v) . Here, $f < N$ is the number of malicious or Byzantine nodes in PF-PoFL. Specifically, each validator $v \in \mathcal{V}_{i,2}$ delivers its signed vote message $\mu_v = \text{voteMsg}_v$ to other validators for mutual verification via the gossiping protocol, i.e.,

$$\text{voteMsg}_v = \langle \text{Hash}(\mathcal{B}_v^{ca}), i, s, \text{seed}_i, T_{\text{stamp}}, pk_v, \sigma_n^{i,s}, \text{Sig}_{sk_v}(\text{voteMsg}_v) \rangle, \quad (14)$$

where \mathcal{B}_v^{ca} is its voted candidate block.

For each validator $v \in \mathcal{V}_{i,2}$, if and only if $\#_v^2(\text{voteMsg}) \geq 2f + 1$, it sends the vote message voteMsg to other validators. Besides, it sets the value-grade pair as $(\mu_v = \text{voteMsg}, \rho_v = 2)$. If $\#_v^2(\text{voteMsg}) \geq f + 1$, it sets $(\mu_v = \text{voteMsg}, \rho_v = 1)$. Otherwise, it sets $(\mu_v = \perp, \rho_v = 0)$. Here, $\#_v^s(\text{msg})$ means the number of validators from which v has received the message msg in stage s .

Phase 2. Each validator in $\mathcal{V}_{i,s}(s > 2)$ executes the improved binary BA (BBA*) protocol [18] (with initial input 0 if $\rho_v = 2$, and 1 otherwise) to determine the final consensus block and model ranking. If the result of BBA* is $\text{out}_v = 0$, then the network reaches consensus on a candidate block voted by μ_v . Otherwise, an empty block is recognized as the consensus block.

Step 4: Adding the consensus block. After reaching consensus, the newly built block \mathcal{B}_i is successfully added into the blockchain which is linearly linked to the previous block with height $i - 1$ via a hash pointer. Fig. 6 illustrates the structure of a block in our PF-PoFL. The metadata meta_i in the newly built block \mathcal{B}_i contains the identity of the completed FL task τ , the consensus ranking mrList_τ , the random seed seed_i for VRF, and a script Script that aggregates the signed votes of validators during the multi-stage voting for model ranking and consensus building. The script Script for each block is agreed upon by the consensus process, which allows new users to catch up with the validation process of the block by processing these votes. Besides, the Script (as a certificate) allows any user to prove the safety of a block and efficiently verify the model ranking and credit updating process.

G. Security and Privacy Analysis

In this subsection, we give the privacy and security analysis of our PF-PoFL scheme. To analyze the query sensitivity of the UDP mechanism in Eqs. (7)–(8), an estimator \hat{f} is defined as $\hat{f}(\mathfrak{S}_j) = \frac{1}{|\mathfrak{S}_j|} \sum_{m \in \mathfrak{S}_j} \Delta\Theta_m$ in every communication round. For privacy protection, the sensitivity of the query function \hat{f} , i.e., $\mathcal{S}(\hat{f}) = \max_{\mathfrak{S}_j, m} \|\hat{f}(\mathfrak{S}_j \cup \{m\}) - \hat{f}(\mathfrak{S}_j)\|_2$ should be controlled. Theorem 1 analyzes the sensitivity bound of $\mathcal{S}(\hat{f})$.

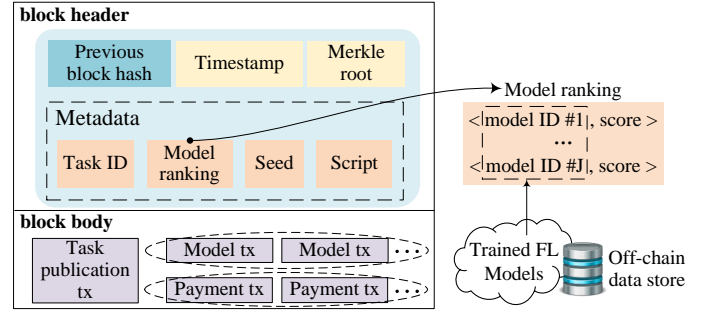


Fig. 6. Structure of a block in PF-PoFL.

Theorem 1: If $\|\Delta\tilde{\Theta}_m\|_2 \leq A$ holds $\forall m \in \mathfrak{S}_j$, the sensitivity of \hat{f} is bounded as $\mathcal{S}(\hat{f}) \leq \frac{2A}{|\mathfrak{S}_j|}$.

Proof: For any $m \in \mathfrak{S}_j$, as $\|\Delta\tilde{\Theta}_m\|_2 \leq A$, we have

$$\begin{aligned} & \|\hat{f}(\mathfrak{S}_j \cup \{m\}) - \hat{f}(\mathfrak{S}_j)\| \\ &= \left\| \frac{\sum_{n \in \mathfrak{S}_j \cup \{m\}} \Delta\Theta_n}{|\mathfrak{S}_j \cup \{m\}|} - \frac{\sum_{n \in \mathfrak{S}_j} \Delta\Theta_n}{|\mathfrak{S}_j|} \right\| \\ &= \left\| \frac{\Delta\Theta_m}{|\mathfrak{S}_j| + 1} - \frac{\sum_{n \in \mathfrak{S}_j} \Delta\Theta_n}{|\mathfrak{S}_j| (|\mathfrak{S}_j| + 1)} \right\| \\ &\leq \left\| \frac{\Delta\Theta_m}{|\mathfrak{S}_j| + 1} \right\| + \left\| \frac{\sum_{n \in \mathfrak{S}_j} \Delta\Theta_n}{|\mathfrak{S}_j|} \right\| \frac{1}{|\mathfrak{S}_j| + 1} \leq \frac{2A}{|\mathfrak{S}_j|}. \end{aligned} \quad (15)$$

Theorem 1 is proved. \blacksquare

Theorem 2: Given the number of communication rounds $\Psi_{j,\tau}^{\text{global}}$ and the sample ratio $\lambda_s, \forall \epsilon < 2(\lambda_s)^2 \log(1/\lambda_s) \Psi_{j,\tau}^{\text{global}}$ and $\forall \delta > 0$, the proposed mechanism satisfies (ϵ, δ) -UDP, if

$$\sigma \geq \frac{2\lambda_s \sqrt{\Psi_{j,\tau}^{\text{global}} \log(1/\delta)}}{\epsilon}. \quad (16)$$

Proof: According to the Lemma 3 of [22], given $\sigma > 1$ and $\lambda_s < \frac{1}{16\sigma}$, for any positive integer $\gamma \leq \sigma^2 \ln(1/\lambda_s \sigma)$, the moment of UDP mechanism \mathcal{F} satisfies

$$\alpha_{\mathcal{F}}(\gamma) \leq \frac{(\lambda_s)^2 \gamma (\gamma + 1)}{(1 - \lambda_s) \sigma^2} + \mathcal{O}\left(\left(\frac{\lambda_s \gamma}{\sigma}\right)^3\right), \quad (17)$$

which is bounded by $\alpha_{\mathcal{F}}(\gamma) \leq \frac{(\lambda_s)^2 \gamma (\gamma + 1)}{(1 - \lambda_s) \sigma^2}$. Besides, based on the Theorem 2 of [22], to be (ϵ, δ) -UDP, it suffices that

$$\frac{\Psi_{j,\tau}^{\text{global}} (\lambda_s)^2 \gamma^2}{\sigma^2} \leq \frac{\gamma \epsilon}{2}, \text{ and } \delta \geq \exp\left(-\frac{\gamma \epsilon}{2}\right). \quad (18)$$

According to formula (18), we can derive that

$$\sigma \geq \frac{\lambda_s \sqrt{2\Psi_{j,\tau}^{\text{global}} \gamma \epsilon}}{\epsilon} \geq \frac{2\lambda_s \sqrt{\Psi_{j,\tau}^{\text{global}} \log(1/\delta)}}{\epsilon}. \quad (19)$$

As $\sigma > 1$ and $\lambda_s \in (0, 1]$, we have $\ln(1/\lambda_s \sigma) < \ln(1/\lambda_s)$. According to $\delta \geq \exp(-\frac{\gamma \epsilon}{2})$, we have $\gamma \epsilon \geq 2 \log(1/\delta)$. By transforming $\gamma \leq \sigma^2 \ln(1/\lambda_s \sigma)$, we have

$$\begin{aligned} \epsilon &\leq \frac{4(\lambda_s)^2 \Psi_{j,\tau}^{\text{global}} \log(1/\delta) \log(1/\lambda_s \sigma)}{\gamma \epsilon} \\ &< 2(\lambda_s)^2 \Psi_{j,\tau}^{\text{global}} \log(1/\lambda_s \sigma) < 2(\lambda_s)^2 \Psi_{j,\tau}^{\text{global}} \log(1/\lambda_s). \end{aligned} \quad (20)$$

Theorem 2 is proved. \blacksquare

In the blockchain, the reasonable security assumption is adopted, where the fraction of the credit value held by malicious blockchain nodes is assumed to be less than $1/3$. In the following, we analyze the finality, consistency, and efficiency of our PF-PoFL system, as well as security analysis in defending against adversaries defined in Sect. III-B.

- *Finality and consistency.* According to [18], [30], under the above majority honest assumption, our PF-PoFL system is resilient up to $1/3$ credit values held by malicious or Byzantine nodes, and all honest nodes can reach an agreement on the same block with overwhelming probability. It ensures the finality and consistency of the blockchain, as well as the prevention of blockchain forks and double-spending threats.
- *Consensus efficiency.* Under the *strong synchrony* assumption (i.e., messages sent by most honest nodes can be received by most other honest nodes within a known time bound), the BBA* protocol is proved to reach agreement within one stage, thereby ensuring the high efficiency of consensus protocol in common situations.
- *Defense of spoofing attack.* In the model ranking smart contract, a model transaction will not be included in the candidate block by honest validators if it is published within the testing phase of the task. Hence, trainers do not have incentives to cheat by training a perfect model on the test data. Besides, as only the hashes of trained FL models are released in federated mining process, adversaries cannot plagiarize the FL models trained by other pools. Besides, the payment delivery to members of the winning pool is executed automatically via the block rewarding smart contract, the refusal-to-pay threat can be avoided. Thereby, PF-PoFL can resist spoofing attacks.
- *Defense of Sybil attack.* By introducing the participation fee, trainers/pools have to pay a participation fee before their trained models are ranked, which increases the cost for nodes to carry out Sybil attacks. Besides, as the committee members are randomly formed via VRF based on their credit values, Sybil attacks to the committee selection process can be prevented, as well as ensuring fairness for blockchain nodes. Thereby, PF-PoFL can mitigate Sybil attacks.
- *Defense of the central platform.* PF-PoFL offers a fully decentralized solution for the life-cycle process including task outsourcing, model training, model ranking, and reward sharing. In PF-PoFL, as the third-party platform is removed, potential risks it brings including SPoF, collusion, and sensitive information leakage can be avoided.
- *Privacy leakage prevention.* According to Theorem 2, PF-PoFL ensures UDP for miners to prevent user-level privacy leakage during federated mining. Besides, as the committee selection is pseudo-random and non-interactive and the committee members are replaced in every stage, adversaries cannot target a committee member until that validator starts participation. Thereby, PF-PoFL also ensures membership unpredictability for validators to prevent from being targeted.

V. DYNAMIC OPTIMIZED POOL STRUCTURE FORMATION

In this section, we first analyze the federation utility of each pool, then we present a stable pool formation algorithm based on the federation formation game, followed by the property analysis of the proposed algorithm.

A. Federation Utility Function

The federation utility of pool \mathfrak{S}_j is determined by the accuracy loss of the trained global model and the federation cost. First, we characterize the expected accuracy loss of the global model under federated mining in Sect. IV-D.

1) *Non-IID effect.* The non-IID data can affect the accuracy of the global model under FL. We adopt the widely used average earth mover's distance (EMD) to measure the heterogeneity of data distribution among diverse miners [31]. In a typical classification problem with Y classes defined over a label space $\mathcal{Y} = \{1, \dots, Y\}$ and a compact space \mathcal{X} . Let $\Pr_m(y = k)$ denote the proportion of miner m 's local training samples with label $k \in \mathcal{Y}$, and $\Pr(y = k)$ be the proportion of training samples with label $k \in \mathcal{Y}$ in all miners' training data. Thereby, $\Pr(y), \forall y \in \mathcal{Y}$ denotes the data distribution for each data sample $\{\mathbf{x}, y\}$ distributed over $\mathcal{X} \times \mathcal{Y}$. The EMD of miner $m \in \mathcal{M}_\tau$ is computed as

$$\Xi_m = \sum_{k=1}^Y \left| \Pr_m(y = k) - \Pr(y = k) \right|. \quad (21)$$

The average EMD in pool \mathfrak{S}_j is computed as the weighted sum of all miners' EMDs, which is expressed as

$$\bar{\Xi}_j = \sum_{m=1}^{|\mathfrak{S}_j|} \frac{s_m}{\sum_{m=1}^{|\mathfrak{S}_j|} s_m} \Xi_m, \quad (22)$$

where s_m is miner m 's local data size, and $|\mathfrak{S}_j|$ is the number of members in pool \mathfrak{S}_j .

2) *Latency analysis.* In a global communication round k , the total latency consists of (i) local training latency D_{tl}^k in local model training and (ii) network latency D_{nl}^k for uploading the trained local model and receiving the latest global model using wired/wireless networks. According to [32], we have $D_{nl}^k \gg D_{tl}^k$, thereby D_{tl}^k can be negligible. Here, the maximum waiting time (i.e., D_j^{\max}) of pool \mathfrak{S}_j in a global communication round is defined as

$$D_j^{\max} = \beta \bar{D}_{nl} = \beta \sum_{m=1}^{|\mathfrak{S}_j|} \bar{D}_{nl,m}, \quad (23)$$

where $\beta > 0$ is an adjustment factor to control the ratio of participants in a global communication round. \bar{D}_{nl} is the average network latency of all miners in the pool \mathfrak{S}_j . $\bar{D}_{nl,m}$ is the expected network latency of miner $m \in \mathfrak{S}_j$.

3) *Number of training samples.* In a global communication round, the total number of data samples used for model training in pool \mathfrak{S}_j is denoted as $S_j = \sum_{m=1}^{|\mathfrak{S}_j|} \alpha_m s_m$. Here, $\alpha_m = \{0, 1\}$ is a binary variable, i.e.,

$$\alpha_m = \begin{cases} 1, & \text{if } \bar{D}_{nl,m} \leq D_j^{\max}, \\ 0, & \text{otherwise.} \end{cases} \quad (24)$$

It means that only miners with $\bar{D}_{nl,m} \leq D_j^{\max}$ are admitted to join the model training.

4) *Total global communication rounds.* The number of global communication rounds of pool \mathfrak{S}_j can be computed as $\Psi_{j,\tau}^{\text{global}} = \lfloor T_\tau^{\text{train}} / D_j^{\max} \rfloor$. Here, T_τ^{train} is the total training time duration (measured by block height) of a FL task, as shown in Fig. 5.

5) *Expected accuracy loss.* The accuracy loss of the global model after $\Psi_{j,\tau}^{\text{global}}$ rounds can be measured by the prediction loss between the optimal parameter Θ^* and the parameter $\Theta(\Psi_{j,\tau}^{\text{global}})$, i.e., $\mathcal{L}(\Theta(\Psi_{j,\tau}^{\text{global}})) - \mathcal{L}(\Theta^*)$. According to [33], [34], when adopting mini-batch SGD for local training, the expected accuracy loss Π_j under non-IID case is bounded by $\mathcal{O}\left(\mathcal{E}(\bar{\Xi}_j) \left(\frac{1}{\sqrt{S_j \Psi_{j,\tau}^{\text{global}}}} + \frac{1}{\Psi_{j,\tau}^{\text{global}}} \right)\right)$. Here, $\mathcal{E}(\bar{\Xi}_j)$ is the relative accuracy loss function. Obviously, the expected accuracy loss decreases with the increase of the number of global iterations $\Psi_{j,\tau}^{\text{global}}$ and the number of training samples S_j .

Apart from the federated mining strategy, miners can also employ the solo mining strategy, in which the miner m chooses to work alone and forms a singleton, i.e., $\mathfrak{S}_j = \{m\}$. Under this case, the expected accuracy loss bound can be approximated as $\mathcal{O}\left(\mathcal{E}(\Xi_m) \left(\frac{1}{\sqrt{s_m \Psi_{\max,\tau}^{\text{local}}}} + \frac{1}{\Psi_{\max,\tau}^{\text{local}}} \right)\right)$, where $\Psi_{\max,\tau}^{\text{local}}$ is the maximum number of local epoches in training task τ to avoid overfitting. To summarize, the explicit form of the expected accuracy loss is expressed as

$$\Pi_j = \begin{cases} \mathcal{E}(\bar{\Xi}_j) \left(\frac{1}{\sqrt{S_j \Psi_{j,\tau}^{\text{global}}}} + \frac{1}{\Psi_{j,\tau}^{\text{global}}} \right), & \text{if } |\mathfrak{S}_j| > 1, \\ \mathcal{E}(\Xi_m) \left(\frac{1}{\sqrt{s_m \Psi_{\max,\tau}^{\text{local}}}} + \frac{1}{\Psi_{\max,\tau}^{\text{local}}} \right), & \text{if } |\mathfrak{S}_j| = 1. \end{cases} \quad (25)$$

6) *Federation satisfaction function.* The federation satisfaction function $\mathcal{S}(\Pi_j)$ measures the satisfaction that the pool \mathfrak{S}_j receives in the expected accuracy loss Π_j . Generally, the lower the accuracy loss, the higher the federation satisfaction. Besides, it becomes harder to further reduce the accuracy loss when the loss is smaller, indicating that the faster the drop rate of accuracy loss, the higher the obtained federation satisfaction. Thereby, $\mathcal{S}(\Pi_j)$ should satisfy $\mathcal{S}(\Pi_j) \geq 0$, $\frac{d\mathcal{S}(\Pi_j)}{d\Pi_j} < 0$, and $\frac{d^2\mathcal{S}(\Pi_j)}{d\Pi_j^2} \geq 0$. A well-suited satisfaction function satisfying these requirements can be the exponential decay function [35] given by

$$\mathcal{S}(\Pi_j) = \gamma_s \exp(-\gamma_d \cdot \Pi_j), \quad (26)$$

where $\gamma_s > 0$ is the satisfaction parameter, and $\gamma_d > 0$ is the decay parameter which controls the decay speed. A larger γ_d indicates a faster decay of satisfaction.

Next, we analyze the federation cost in pool formulation.

7) *Federation cost function.* The miners in pool \mathfrak{S}_j need to frequently synchronize the latest global model from the curator. According to [36], [37], the federation cost $\mathcal{C}(\mathfrak{S}_j)$ can be measured by the communication cost which varies linearly with the federation size $|\mathfrak{S}_j|$. We have

$$\mathcal{C}(\mathfrak{S}_j) = \begin{cases} \lambda_c |\mathfrak{S}_j|, & \text{if } |\mathfrak{S}_j| > 1, \\ 0, & \text{if } |\mathfrak{S}_j| = 1, \end{cases} \quad (27)$$

where $\lambda_c > 0$ is a scaling factor.

Finally, the federation utility of pool \mathfrak{S}_j can be obtained as the difference between the federation satisfaction and the federation cost, i.e.,

$$\begin{aligned} \mathcal{U}(\mathfrak{S}_j) &= \mathcal{S}(\Pi_j) - \mathcal{C}(\mathfrak{S}_j) \\ &= \begin{cases} \gamma_s \exp\left(-\gamma_d \mathcal{E}(\bar{\Xi}_j) \left(\frac{1}{\sqrt{\sum_{m=1}^{|\mathfrak{S}_j|} \alpha_m s_m} \lfloor \frac{T_\tau^{\text{train}}}{D_j^{\max}} \rfloor} + \lfloor \frac{T_\tau^{\text{train}}}{D_j^{\max}} \rfloor^{-1} \right)\right) \\ \quad - \lambda_c |\mathfrak{S}_j|, & \text{if } |\mathfrak{S}_j| > 1, \\ \gamma_s \exp\left(-\gamma_d \mathcal{E}(\Xi_j) \left(\frac{1}{\sqrt{s_j \Psi_{\max,\tau}^{\text{local}}}} + \frac{1}{\Psi_{\max,\tau}^{\text{local}}} \right)\right), & \text{if } |\mathfrak{S}_j| = 1. \end{cases} \end{aligned} \quad (28)$$

Let $W(\mathfrak{S})$ denote the social welfare of a partition \mathfrak{S} , i.e., the sum of federation utilities of disjoint pools in a partition \mathfrak{S} . The target of PF-PoFL is to maximize the social welfare by forming the optimal partition structure \mathfrak{S}^* , where the optimization problem for task τ is formulated as:

$$\begin{aligned} \text{Problem : } \max W(\mathfrak{S}) &:= \sum_{j=1}^{|\mathfrak{S}|} \mathcal{U}(\mathfrak{S}_j). \\ \text{s.t. } \mathfrak{S}_j \cap \mathfrak{S}_{j'} &= \emptyset, \forall j \neq j', \cup_{j=1}^{|\mathfrak{S}|} \mathfrak{S}_j = \mathcal{M}_\tau. \end{aligned} \quad (29)$$

B. Stable Pool Formation Algorithm

To solve the problem in (29), the pooled-mining formation process among miners is modeled as a federation formation game with transferable utility (FFG-TU), where distributed miners (as game players) tend to form various disjoint federations to maximize their individual profits [38].

Definition 2 (FFG-TU game): For each uncompleted FL task τ , a FFG-TU game for optimized pool structure is defined by a triple $(\mathcal{M}_\tau, \mathcal{U}, \mathfrak{S})$, where the specific formulation is shown as below.

- *Players:* The players of the game is the set of miners (i.e., \mathcal{M}_τ) that participate FL task τ .
- *Transferable federation utility:* $\mathcal{U}(\mathfrak{S}_j)$ is the federation utility of each federation (or pool) $\mathfrak{S}_j \subseteq \mathcal{M}_\tau$, which can be apportioned arbitrarily among the members of \mathfrak{S}_j .
- *Pooled structure:* A pooled structure (or a federation partition) is defined as the set $\mathfrak{S} = \{\mathfrak{S}_1, \dots, \mathfrak{S}_J\}$ that partitions the miner set \mathcal{M}_τ . Here, $\mathfrak{S}_j, j = 1, \dots, J$ are disjoint federations such that $\mathfrak{S}_j \cap \mathfrak{S}_{j'} = \emptyset, \forall j \neq j'$, and $\cup_{j=1}^J \mathfrak{S}_j = \mathcal{M}_\tau$.
- *Strategy:* Each miner decides whether to work alone by adopting the *solo mining* strategy or choose a federation to join to cooperatively train a FL model using the *federated mining* strategy.

Definition 3 (Switch operation): A switch operation $\Phi_{l,k}(m)$ is defined as the transfer of miner m from the current pool $\mathfrak{S}_l \in \mathfrak{S}$ to another pool $\mathfrak{S}_k \in \mathfrak{S} \cup \{\emptyset\}$, mathematically,

$$\Phi_{l,k}(m) : \mathfrak{S}_l \triangleright \{\mathfrak{S}_l^-, \{m\}\} \text{ and } \{\mathfrak{S}_k, \{m\}\} \triangleright \mathfrak{S}_k^+. \quad (30)$$

Remark 2: The switch operation consists of two successive parts: the split operation (i.e., $\mathfrak{S}_l \triangleright \{\mathfrak{S}_l^-, \{m\}\}$) and the merge operation (i.e., $\{\mathfrak{S}_k, \{m\}\} \triangleright \mathfrak{S}_k^+$), where $\mathfrak{S}_l^- = \mathfrak{S}_l \setminus \{m\}$ and $\mathfrak{S}_k^+ = \mathfrak{S}_k \cup \{m\}$. In the case that $\mathfrak{S}_l = \{m\}$, $\Phi_{l,k}(m)$ means the merge of \mathfrak{S}_l with \mathfrak{S}_k , causing the number of federations decrease by one. In the case that $\mathfrak{S}_k = \{\emptyset\}$, $\Phi_{l,k}(m)$ means

the formation of a new singleton $\mathfrak{S}_k = \{m\}$, causing the number of federations increase by one. In the case that $k = l$, it means miner n stays in the current federation and the number of federations remains unchanged.

Definition 4 (Switch gain): The switch gain $\Omega_m(\Phi_{l,k})$ related to the switch operation $\Phi_{l,k}(m)$ is defined as:

$$\begin{aligned} \Omega_m(\Phi_{l,k}) &:= v_m(\mathfrak{S}_k \cup \{m\}) - v_m(\mathfrak{S}_l) \\ &= [\mathcal{U}(\mathfrak{S}_k \cup \{m\}) - \mathcal{U}(\mathfrak{S}_k)] - [\mathcal{U}(\mathfrak{S}_l) - \mathcal{U}(\mathfrak{S}_l \setminus \{m\})]. \end{aligned} \quad (31)$$

Here, $v_m(\mathfrak{S}_k \cup \{m\})$ and $v_m(\mathfrak{S}_l)$ mean the extra utility value that miner m brings to the pools $\mathfrak{S}_k \cup \{m\}$ and \mathfrak{S}_l , respectively.

Remark 3: Specially, we define $\Omega_m(\Phi_{l,k}) = 0$ when $k = l$. Besides, when $\mathfrak{S}_k = \{\emptyset\}$, we have $\Omega_m(\Phi_{l,\emptyset}) = \mathcal{U}(\{m\}) - v_m(\mathfrak{S}_l)$. The transferable switch gain in the pool switch operation accords with the transferable utility of our proposed federation formation game. Besides, the federation gain of any switch operation only relies on the utilities of involved pools, rather than the manner that federation utilities are shared among their members.

Definition 5 (Preference order): For any miner $m \in \mathcal{M}_\tau$, the preference order \succeq is defined as a complete and transitive binary relation between two switch operations $\Phi_{l,k}(m)$ and $\Phi_{l',k'}(m)$ such that:

$$\Phi_{l,k}(m) \succeq \Phi_{l',k'}(m) \Leftrightarrow \Omega_m(\Phi_{l,k}) \geq \Omega_m(\Phi_{l',k'}). \quad (32)$$

Similarly, for the strict preference order \succ , we also have $\Phi_{l,k}(m) \succ \Phi_{l',k'}(m) \Leftrightarrow \Omega_m(\Phi_{l,k}) > \Omega_m(\Phi_{l',k'})$.

As miners can transfer from one federation to another, the federation partition (i.e., \mathfrak{S}) varies with time. The solution of the proposed FFG-TU game is a stable pooled structure \mathfrak{S}^* . In general, the stable partition outcome can be attained using exhaustive search methods, where the possible partition iterations increase exponentially with the number of involved miners [37]. In the following, we design a distributed stable pool formation algorithm with low computational complexity in Algorithm 3 to obtain the optimal federation partition strategy of each player (i.e., miner m) in the game. The proposed algorithm consists of the following three phases.

Phase 1: Pool initialization (line 3). At the beginning ($h = 0$), an initial partition $\mathfrak{S}^{(0)}$ is generated according to the specific application. For example, the initial partition can be the result of the stable partition outcome for the previously finished FL task.

Phase 2: Switch strategy-making on miner's side (lines 5–8). Given the current federation partition $\mathfrak{S}^{(h)} = \{\mathfrak{S}_1^{(h)}, \dots, \mathfrak{S}_{J^{(h)}}^{(h)}\}$, each miner can opt three strategies: (i) stay in the current federation; (ii) split from the current federation and merge with any other non-empty federation; and (iii) split from the current federation and act alone. The first two strategies are federated mining strategies and the last one is the solo mining strategy. To formulate each miner's switch strategy, the switchable candidate pool is first defined as below.

Definition 6 (Switchable candidate pool): For each miner $m \in \mathfrak{S}_l$, its switchable candidate pool set is defined as $\mathcal{C}_m^{\text{pool}}$, where $\Omega_m(\Phi_{l,k}) > 0, \forall \mathfrak{S}_k \in \mathcal{C}_m^{\text{pool}} \subseteq \mathfrak{S} \cup \{\emptyset\}$. In other words, only the pool with positive switch gain can be added into the switchable candidate pool set.

Algorithm 3 Distributed Pool Formation Algorithm

- 1: **Input:** $\mathcal{T}_u, \mathcal{M}_\tau, \Xi_m, s_m, \beta, \bar{D}_{nl,m}, T_\tau^{\text{train}}, \Psi_{m,\tau}^{\text{local}}, \gamma_s, \gamma_d, \lambda_c$.
 - 2: **Output:** Stable federation structure \mathfrak{S}^* .
 - 3: **Initialization:** Set $h = 0$ and the initial pool structure $\mathfrak{S}^{(0)}$.
 - 4: **Repeat**
 - 5: **for** $m \in \mathcal{M}_\tau$ **do**
 - 6: Calculate all candidate pools (including the empty set $\{\emptyset\}$) with positive switch gain, and record them into the set $\mathcal{C}_m^{\text{pool}}$.
 - 7: Send transfer request to most preferred pool $\mathfrak{S}_{j^*}^{(h)} \in \mathcal{C}_m^{\text{pool}}$ such that

$$\Omega_m(\Phi_{l,j^*}) \geq \Omega_m(\Phi_{l,j}), m \in \mathfrak{S}_l^{(h)}, \forall \mathfrak{S}_j^{(h)} \in \mathcal{C}_m^{\text{pool}}, \quad (33)$$
 where $\mathfrak{S}_l^{(h)}$ is the current federation of miner m at h -th iteration.
 - 8: **end for**
 - 9: **for** $\mathfrak{S}_j^{(h)} \in \mathfrak{S}^{(h)}$ **do**
 - 10: Record the candidate miners that send a transfer request into the set $\mathcal{C}_j^{\text{miner}}$.
 - 11: Accept the most preferred miner $m^* \in \mathfrak{S}_j^{(h)}$ through the admission rule:

$$\Omega_{m^*}(\Phi_{l',j}) \geq \Omega_m(\Phi_{l,j}), m \in \mathfrak{S}_l^{(h)}, \forall m \in \mathcal{C}_j^{\text{miner}}. \quad (34)$$
 - 12: Reject other candidate miners in the set $\mathcal{C}_j^{\text{miner}} \setminus \{m^*\}$.
 - 13: Do split operation, i.e., $\mathfrak{S}_{l'}^{(h)} \triangleright \{\mathfrak{S}_{l'}^{(h+1)}, \{m^*\}\}$, where $\mathfrak{S}_{l'}^{(h+1)} = \mathfrak{S}_{l'}^{(h)} \setminus \{m^*\}$.
 - 14: Do merge operation, i.e., $\{\mathfrak{S}_j^{(h)}, \{m^*\}\} \triangleright \mathfrak{S}_j^{(h+1)}$, where $\mathfrak{S}_j^{(h+1)} = \mathfrak{S}_j^{(h)} \cup \{m^*\}$.
 - 15: **end for**
 - 16: $h = h + 1$.
 - 17: Update the federation structure as $\mathfrak{S}^{(h)} = \{\mathfrak{S}_1^{(h)}, \dots, \mathfrak{S}_{J^{(h)}}^{(h)}\}$.
 - 18: **Until** no miner do switch operations, i.e., $\Phi_{l,l}(m) \succeq \Phi_{l,j}(m), m \in \mathfrak{S}_l^{(h)}, j \neq l, \forall m \in \mathcal{M}_\tau, \forall \mathfrak{S}_j^{(h)} \in \mathfrak{S}^{(h)} \cup \{\emptyset\}$.
-

Remark 4: If $\Omega_m(\Phi_{l,k}) \leq 0, \forall \mathfrak{S}_k \in \mathfrak{S} \cup \{\emptyset\}$, there exists no available federation in \mathfrak{S} nor the empty set to transfer for the miner m (i.e., $\mathcal{C}_m^{\text{pool}} = \emptyset$), indicating that the miner intends to stay in the current federation \mathfrak{S}_l (i.e., $\Phi_{l,l}(m)$). Otherwise, the miner makes its strategy based on the following switch rule.

Definition 7 (Switch rule): Given the switchable candidate pool set $\mathcal{C}_m^{\text{pool}} \cup \{\emptyset\}$, each miner $m \in \mathfrak{S}_l$ sends transfer request to the candidate pool \mathfrak{S}_{j^*} with the largest switch gain, i.e.,

$$\mathfrak{S}_{j^*} = \arg \max \Omega_m(\Phi_{l,j}), m \in \mathfrak{S}_l, \forall \mathfrak{S}_j \in \mathcal{C}_m^{\text{pool}} \cup \{\emptyset\}, \quad (35)$$

where \mathfrak{S}_l is the current federation of miner m .

Remark 5: In the case $\arg \max \{\Omega_m(\Phi_{l,j}) : \forall \mathfrak{S}_j \in \mathcal{C}_m^{\text{pool}} \cup \{\emptyset\}\} = \{\emptyset\}$, it means that the miner m prefers the solo mining strategy and will split from

the current federation (i.e., $\Phi_{l,\emptyset}(m)$). In the other case $\arg \max \{\Omega_m(\Phi_{l,j}) : \forall \mathfrak{S}_j \in \mathcal{C}_m^{\text{pool}} \cup \{\emptyset\}\} = \mathfrak{S}_{j^*}$, it means that the miner m prefers splitting from the current federation \mathfrak{S}_l and merging with another federation \mathfrak{S}_{j^*} (i.e., $\Phi_{l,j^*}(m)$).

To summarize, $\forall \mathfrak{S}_j \in \mathcal{C}_m^{\text{pool}} \cup \{\emptyset\}$, the optimal switch strategy of each miner $m \in \mathfrak{S}_l$ can be expressed as

$$\Phi_{l,k}^*(m) = \begin{cases} \Phi_{l,l}(m), & \text{if } \mathcal{C}_m^{\text{pool}} = \emptyset, \\ \Phi_{l,\emptyset}(m), & \text{if } \arg \max \Omega_m(\Phi_{l,j}) = \{\emptyset\}, \\ \Phi_{l,j^*}(m), & \text{if } \arg \max \Omega_m(\Phi_{l,j}) = \mathfrak{S}_{j^*}. \end{cases} \quad (36)$$

Phase 3: Admission strategy-making on pool's side (lines 9–17). When multiple miners request to join the same pool \mathfrak{S}_j , the switch order can affect the federation utilities and the federation formation outcome. To capture the pool's preference for the transfer order, the admission rule is defined as below.

Definition 8 (Admission rule): For a set of candidate miners (i.e., $\mathcal{C}_j^{\text{miner}}$) that send transfer request to the same pool \mathfrak{S}_j , only the candidate m^* with the largest switch gain is permitted by the pool \mathfrak{S}_j , i.e.,

$$m^* = \arg \max \Omega_m(\Phi_{l,j}), m \in \mathfrak{S}_l, \forall m \in \mathcal{C}_j^{\text{miner}}, \quad (37)$$

and other candidate miners in the set $\mathcal{C}_j^{\text{miner}} \setminus \{m^*\}$ are rejected.

According to the admission rule, each pool $\mathfrak{S}_j \in \mathfrak{S}$ accepts the most preferred miner $m^* \in \mathfrak{S}_{l'}$ (in line 11) and rejects other candidates (in line 12). Correspondingly, the switch operation is performed between two pools $\mathfrak{S}_j, \mathfrak{S}_{l'} \in \mathfrak{S}$ consisting of the split operation (i.e., $\mathfrak{S}_{l'}^{(h)} \triangleright \{\mathfrak{S}_{l'}^{(h+1)}, \{m^*\}\}$) and the merge operation (i.e., $\{\mathfrak{S}_j^{(h)}, \{m^*\}\} \triangleright \mathfrak{S}_j^{(h+1)}$), where $\mathfrak{S}_{l'}^{(h+1)} = \mathfrak{S}_{l'}^{(h)} \setminus \{m^*\}$ and $\mathfrak{S}_j^{(h+1)} = \mathfrak{S}_j^{(h)} \cup \{m^*\}$. Then, the federation structure is updated as $\mathfrak{S}^{(h)} \rightarrow \mathfrak{S}^{(h+1)}$ (in line 17).

The above phases 2-3 are repeated until no miner $m \in \mathfrak{S}_l^* \subseteq \mathcal{M}_\tau$ in the final pooled structure \mathfrak{S}^* intends to execute switch operations to increase the overall gain by transferring to another pool $\mathfrak{S}_j^* \in \mathfrak{S}^* \cup \{\emptyset\}$ with $j \neq l$, i.e., $\Phi_{l,l}(m) \succeq \Phi_{l,j}(m)$.

C. Property Analysis

Given any initial partition $\mathfrak{S}^{(0)}$, the federation formation procedure can be formulated as a sequence of switch operations, i.e.,

$$\{\mathfrak{S}^{(0)} \rightarrow \dots \rightarrow \mathfrak{S}^{(h)} \rightarrow \dots \rightarrow \mathfrak{S}^{(H)} = \mathfrak{S}^*\}, \quad (38)$$

where H is total number of transformations.

Lemma 1: Any successive transformation of the federation partition, i.e., $\mathfrak{S}^{(h)} \rightarrow \mathfrak{S}^{(h+1)}$, $h = \{0, 1, \dots, H-1\}$, can improve the social welfare for miners in \mathcal{M}_τ .

Proof: According to Definitions 6–7, for each switch operation $\Phi_{l,k}(m)$, it results in a strictly positive gain $\Omega_m(\Phi_{l,k}) > 0$ for the two involved pools \mathfrak{S}_j and \mathfrak{S}_k , while the utilities of other pools in $\mathfrak{S} \setminus \{\mathfrak{S}_j, \mathfrak{S}_k\}$ remain unchanged. Let $\sum^{(h)}$ be the set of permitted switch operations during the transformation $\mathfrak{S}^{(h)} \rightarrow \mathfrak{S}^{(h+1)}$. Therefore, we can obtain

$$W(\mathfrak{S}^{(h+1)}) = W(\mathfrak{S}^{(h)}) + \sum_{\Phi_{l,k}(m) \in \sum^{(h)}} \Omega_m(\Phi_{l,k}). \quad (39)$$

The second term in the right side of (39) means the improved social welfare (i.e., the sum of added switch gains) during $\mathfrak{S}^{(h)} \rightarrow \mathfrak{S}^{(h+1)}$. As $\sum_{\Phi_{l,k}(m) \in \sum^{(h)}} \Omega_m(\Phi_{l,k}) > 0$, we have $W(\mathfrak{S}^{(h+1)}) > W(\mathfrak{S}^{(h)})$. Lemma 1 is proved. ■

In the following, we first prove the convergence of federation partition transformations in Theorem 3. Then, we prove the Nash-stability, near-optimality, and low computational complexity in Theorems 4, 5, 6, respectively.

Theorem 3: Given an arbitrary initial pooled structure $\mathfrak{S}^{(0)}$, the proposed Algorithm 3 can always converge to a final disjoint federation partition \mathfrak{S}^* after a finite number of transformations.

Proof: According to the miner's switch strategy defined in Eq. (36), we can observe that a single switch operation either yields (i) a previously visited partition with a non-cooperative miner (i.e., a singleton) or (ii) an unvisited new partition.

Case (i). When it comes to the partition structure $\mathfrak{S}^{(h)}$ where miner m forms a singleton, the non-cooperative miner m should either join a new federation or decide to remain non-cooperative in the next iteration $\mathfrak{S}^{(h+1)}$.

- If miner m chooses to join a new federation, an unvisited partition without this non-cooperative miner m will be formed via the switch operation.
- If miner m chooses to remain non-cooperative, any visited partition will not appear in the transformation of the current partition structure.

Case (ii). When it comes to the partition structure $\mathfrak{S}^{(h)}$ where a single switch operation leads to an unvisited partition in the next iteration $\mathfrak{S}^{(h+1)}$, the maximum number of partitions among miners in \mathcal{M}_τ can be given by the well-known Bell number function, i.e.,

$$b_{|\mathcal{M}_\tau|} = \sum_{m=1}^{|\mathcal{M}_\tau|} \binom{|\mathcal{M}_\tau|-1}{m} b_m, \forall m \in \mathcal{M}_\tau \ \& \ b_0 = 1. \quad (40)$$

According to Eq. (40), the number of transformations of federation partitions in formula (38) is finite. Thereby, in all cases, the transformation sequence in formula (38) will always terminate and converge to a final partition $\mathfrak{S}^* = \mathfrak{S}^{(H)}$, which is composed of a number of disjoint federations after finite H iterations. Theorem 3 is proved. ■

Definition 9 (Nash-stability): A federation partition $\mathfrak{S} = \{\mathfrak{S}_1, \dots, \mathfrak{S}_J\}$ is Nash-stable if $\forall m \in \mathcal{M}_\tau, \forall \mathfrak{S}_j \in \mathfrak{S}, \forall \mathfrak{S}_l \in \mathfrak{S} \cup \{\emptyset\}, \Omega_m(\Phi_{j,l}) \leq 0$.

Remark 6: The Nash-stability implies that, under the Nash-stable partition, any single switch operation will not result in a strictly positive gain. In other words, no miner has incentives to move from its current federation to another one or to deviate and act alone in the Nash-stable partition.

Theorem 4: The final partition \mathfrak{S}^* from Algorithm 3 is Nash-stable.

Proof: The Nash-stability of partition \mathfrak{S}^* is proved by contradiction. Suppose that the final partition \mathfrak{S}^* resulting from Algorithm 3 is not Nash-stable. As a consequence, there exist a miner $m \in \mathfrak{S}_j$ and a federation $\mathfrak{S}_{j'} \in \mathfrak{S}^* \cup \{\emptyset\}$, $j \neq j'$ such that $\Phi_{j,j'}(m) \succeq \Phi_{j,j}(m)$. Under this circumstance, miner

m will execute the switch operation $\Phi_{j,j'}(m)$, which contradicts with the fact that \mathfrak{S}^* is the final partition result of Algorithm 3. Theorem 4 is proved. ■

Theorem 5: The final partition outcome from Algorithm 3 attains a near-optimal performance.

Proof: According to Lemma 1, the consecutive transformation of the federation partition, i.e., $\mathfrak{S}^{(h)} \rightarrow \mathfrak{S}^{(h+1)}$, can improve the overall social welfare. Moreover, according to Theorem 3, the overall system welfare derived by Algorithm 3 is convergent after several iterations. Besides, instead of allowing multiple miners to simultaneously transfer to a single pool, the switch operation to a single pool is executed in a greedy manner in our work, where only the miner with the largest positive switch gain is permitted. Thereby, the partition result derived through Algorithm 3 is near-optimal, which is also evaluated using simulations in comparison with the exhaustive optimal solution in Sect. VI-C. ■

Theorem 6: The computational complexity of the proposed Algorithm 3 is $\mathcal{O}(H \cdot J \cdot |\mathcal{M}_\tau|)$, where $J = |\mathfrak{S}^*|$ is the number of federations in the Nash-stable partition \mathfrak{S}^* .

Proof: The complexity of the distributed pool formation algorithm in Algorithm 3 mainly consists of two parts: (i) the first for-loop for each miner to decide its switch strategy (lines 5-8); and (ii) the second for-loop for each federation to decide its admission strategy (lines 9-15). For the first part, it has a complexity of $\mathcal{O}(H|\mathcal{M}_\tau|J)$. The second part has a complexity of $\mathcal{O}(HJ|C_j^{\text{miner}}|)$. As $\max\{|C_j^{\text{miner}}|\} \leq |\mathcal{M}_\tau|$, the overall computational complexity yields $\mathcal{O}(HJ|\mathcal{M}_\tau|)$. Theorem 6 is proved. ■

VI. PERFORMANCE EVALUATION

In this section, the simulation setup is first introduced in Sect. VI-A, followed by the numerical results and discussions in Sects. VI-B~VI-D.

A. Simulation Setup

We implement the prototype of PF-PoFL on a private blockchain network based on the open-source Go-Algorand project [39] under both local and distributed settings. For the distributed setting (i.e., **setting 1**), the prototype is deployed across 20 Azure B8ms virtual machines (VMs), with 8 CPU cores and 32 GB of RAM. The VMs spread across five locations: West US, Canada East, UK South, Korea South, and North Europe. Each VM hosts 5 blockchain miners, and the total number of blockchain miners (i.e., N) is 100. For the local setting (i.e., **setting 2**), the prototype is deployed and tested on a single computer with Intel Core i7-8700 CPU (3.6GHz) and 32GB RAM, where the number of blockchain miners is set as 100.

We use Go 1.17 to create and manage the private multi-node networks, as well as handle all networking and distributed systems aspects. We use the Turing-complete PyTeal [40] to write the smart contract in Python 3.10 and use the `compileProgram` method to produce the bytecode-based TEAL source code, which can be deployed on the blockchain. Besides, miners in a pool use PyTorch to perform federated mining with UDP to produce SGD updates during training,

TABLE I
USED DATASETS AND MODELS IN PF-POFL

Dataset	#Training samples	#Test samples	#Classes	Model
MNIST	60,000	10,000	10	CNN
CIFAR-10	50,000	10,000	10	ResNet18

and our prototype can support any AI model which can be optimized via SGD.

Dataset. We consider two types of FL tasks trained on two typical datasets: one type is the handwritten digits recognition tasks on the MNIST dataset [41]; and another is the image recognition tasks on the CIFAR-10 dataset [42]. MNIST consists of 60,000 training images and 10,000 test images with size of 28×28 in 10 classes. CIFAR-10 contains 60,000 32×32 images evenly divided in 10 classes, with 50,000 training images and 10,000 test samples. For the dataset partition among miners, the non-IID setting is adopted. Specifically, each miner is assigned with z classes of 600 training samples for MNIST (resp. 500 training samples for CIFAR-10) and the training samples of different miners can be repeated, where the integer z is randomly selected from $\{1, 2, \dots, 10\}$.

Model. For federated mining within each pool in PF-PoFL, the 3-layer CNN model is adopted for MNIST, while the ResNet18 model is employed for CIFAR-10. The total number of communication rounds is set as $\Psi_{j,\tau}^{\text{global}} = 200$, and the sample ratio is set as $\lambda_s = 0.35$. For the learning algorithm, the mini-batch SGD with local batch size equal to 50 is adopted for all miners, where the initial learning rate η is set as 0.01 and declines to 0.001 after 120-th communication round. Besides, the local epoch of miners is set as 1 under the federated mining strategy, while the maximum local epoch $\Psi_{\text{max},\tau}^{\text{local}}$ of miners is set as 1000 under the solo mining strategy.

Table I summarizes the datasets and models used in PF-PoFL. For the UDP model, the privacy parameters (σ, ϵ) are selected from $\{(3, 5.52), (14, 1.06), (24, 0.61)\}$. Besides, we set $\delta = 1 \times 10^{-6}$. For the federation formation game model, we set $\gamma_d = 30$, $\gamma_s = 23$, $\lambda_c = 0.01$. In addition, we set miner's initial credit value as $cre_n^0 = 1$ and the credit updates as $\chi_1 = 2$, $\chi_2 = 4$, and $\chi_3 = 1$.

Comparing method. We evaluate the performance of the PF-PoFL in comparison with the PoFL scheme [15]. In PoFL [15], its working relies on a central FL platform, and the fixed mining pool structure is employed for federated mining. PoFL ensures privacy preservation and prevents model plagiarism for trained models using homomorphic encryption (HE)-based label prediction and secure two-party computation (2PC)-based label comparison. As PoFL does not specify the detailed agreement protocol to acquire consensus, to be objective and fair, we implement the conventional Algorand BA protocol [18] for miners in PoFL to reach consensus on a new block. Here, the number of miners is set as $N = 100$ and the number of malicious or Byzantine miners is set as $f = 20$.

In the following, we first evaluate the blockchain throughput, block latency, and system overhead of our PF-PoFL in Figs. 7–8 and Tables II–III. Next, we analyze the impacts of non-IID, number of miners, and privacy parameters on the federated mining process with UDP in Figs. 9–13. Finally, we

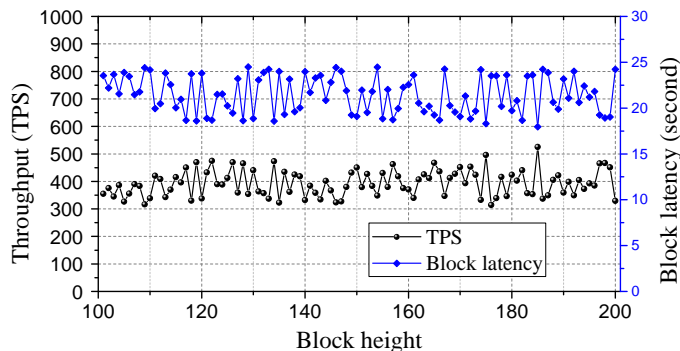


Fig. 7. Throughput and block latency in different block heights in PF-PoFL (setting 1).

analyze the efficiency and stability of the federation formation game approach for optimized pool structure in Figs. 14–15.

B. System Throughput and Latency

Fig. 7 shows the throughput (measured by transactions per second (TPS)) and the block latency (i.e., the consensus time for a block from being pending to be confirmed) of the blockchain system in the distributed setting. In the blockchain, the shorter the consensus time, the better the consensus efficiency. Here, the consensus time is mainly determined by the delay in model downloading, model ranking, model verification, block propagation, and voting-based agreement process. In the simulation, the ratio of malicious or Byzantine miners is set to be 20%. From Fig. 7, it can be seen that, when the block height grows from 100 to 200, both the TPS and the block latency of our proposed PF-PoFL consensus mechanism remain relatively stable. Specifically, the throughput mainly varies between [300, 500] TPS, while the block latency changes within [18, 25] seconds, which indicates a relatively stable block packaging frequency of our PF-PoFL consensus mechanism. It can be explained as follows.

In the traditional PoW consensus protocol, all miners compete to solve the same PoW puzzle and the fastest node to solve it earns the block rewards. Distinguished from the PoW protocol, in our PF-PoFL consensus protocol, different mining pools can concurrently train on different FL tasks that are in the training phase; meanwhile, in the current block height, the validators prefer to reach consensus on the most urgent and profitable task from all tasks that are in the testing phase. For example, for a FL task τ with a high difficulty level, in its training phase, validators will carry out consensus on other tasks that are in the testing phase; until the FL task τ reaches its testing phase, validators will perform consensus on it if the task τ is most urgent and profitable in current block height. As such, for FL tasks with different difficulty levels, the relatively stable throughput of the blockchain system can be obtained.

Fig. 8 shows the throughput and the block latency of PF-PoFL in the distributed setting when the block size increases from 0.5 MB to 10 MB. It can be observed that our PF-PoFL system can efficiently handle hundreds of transactions per second and the block latency is below 47 seconds. As seen in Fig. 8, with the increase of block size, the block latency keeps increasing while the blockchain throughput first

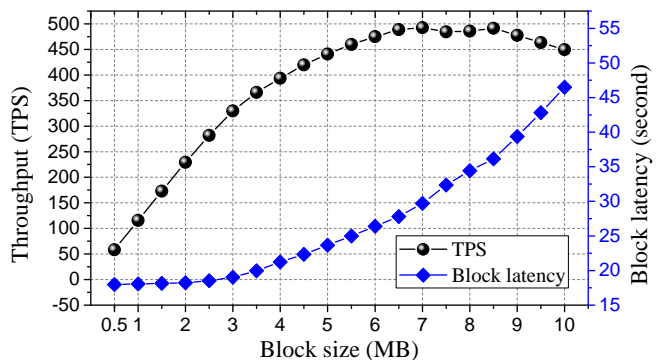


Fig. 8. Throughput and block latency vs. block size in PF-PoFL (setting 1).

TABLE II
TIME COST OF EXECUTING MODEL RANKING CONTRACT IN PF-PoFL (SETTING 1)

Operation	Time for the fastest validator	Time for the slowest validator	Ave. time for validators
Running	3.215 s	6.246 s	4.952 s
Reaching agreement	16.583 s		
Overall latency			≈ 21.1 s

increases then decreases when the block size is over 8.5 MB. It can be explained as follows. On one hand, the higher block size indicates that more transactions are included within a block, resulting in a higher latency in transaction processing and block propagation. On the other hand, the relatively higher block size means that more transactions to be processed in a new block, which brings a higher TPS. However, the TPS cannot grow with the block size all the time. The reason is when the block size is too large (i.e., over 8.5 MB), the time to verify and broadcast the block can be extended, so that the distributed blockchain cannot be synchronized in time.

Table II shows the time cost of executing model ranking contract in PF-PoFL under the distributed setting. The execution time of model ranking contract mainly consists of two parts: the contract running time (to produce local model ranking result) and the time to reach agreement on the final model ranking result (to be committed on blockchain). Note that in Algorithm 1, each validator can process multiple FL model transactions (including model downloading and evaluation) in parallel to reduce the contract running time. As seen in Table II, the average contract running time of validators is 4.952 seconds and the time to reach agreement is 16.583 seconds, leading to an overall latency of about 21.1 seconds.

Table III compares the time cost of model operations for pools and validators in PoFL and PF-PoFL under both local and distributed settings. As seen in Table III, PoFL involves huge computation cost in model evaluation process (i.e., calculating model performance), especially for pools due to the expensive HE operations. Compared with the PoFL scheme, our PF-PoFL adds the Gaussian noise according to Eqs. (7) and (8) to ensure UDP for miners, and the time cost for pools in this process can be negligible. For model ranking and verification operations performed by validators, PoFL only needs to determine the best model. It indicates that if the model with the best performance is verified as true, there is no need to verify other models. In our PF-PoFL,

TABLE III
COMPUTATION COST OF MODEL OPERATIONS FOR POOLS AND VALIDATORS UNDER POFL AND OUR PF-POFL

Setting	Scheme	Pool		Validator		
		Model evaluation	Downloading	Model verification	Model ranking	Total consensus delay
Setting 1	PoFL [15]	1.7×10^6 s	Negligible	1.675 s	59 us	12.2 s
	Ours	Negligible	Negligible	3.241 s	184 us	9.4 s
Setting 2	PoFL [15]	2.3×10^6 s	1.21 s	1.296 s	51 us	39.8 s
	Ours	Negligible	1.88 s	2.927 s	172 us	21.3 s

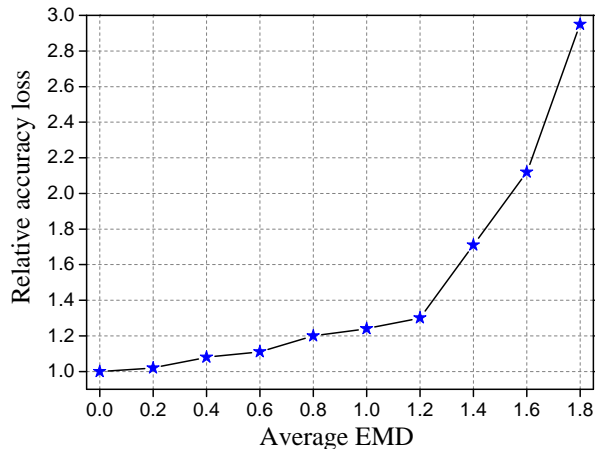


Fig. 9. Relative accuracy loss $\mathcal{E}(\bar{\Xi}_j)$ vs. average EMD $\bar{\Xi}_j$ (CIFAR-10).

each validator needs to test and sort all relevant models to determine the financial rewards and credit incentives in the block rewarding contract, causing a longer running time (e.g., 172 microseconds for model ranking and 2.927 seconds for model verification in setting 2) than that in the PoFL (e.g., 51 microseconds for model ranking and 1.296 seconds for model verification in setting 2). Besides, Table III shows that the total consensus time for reaching agreement among validators on a new block is about 9.4 seconds in our PF-PoFL in setting 1 (resp. 21.3 seconds in setting 2), which is faster than that in the PoFL, i.e., 12.2 seconds in setting 1 (resp. 39.8 seconds in setting 2). The reasons are as follows. On one hand, the block proposal in PoFL involves huge storage space for encrypted model parameters in HE and 2PC, resulting in a higher block propagation latency. On the other hand, our credit-based Algorand BA protocol can employ more honest miners in the validator committee and incentivize miners to act legitimately and actively, thereby reducing the voting stages and the delay in reaching an unambiguous agreement.

C. Federated Mining with UDP

In this subsection, we evaluate the performance of the federated mining process with UDP on MNIST and CIFAR-10, in comparison with the baseline scheme, by changing the number of miners and values of privacy parameters. In the baseline scheme, the curator in a pool directly delivers the raw global update (instead of the perturbed version) to all miners. Fig. 9 illustrates the relationship between average EMD $\bar{\Xi}_j$ and relative accuracy loss $\mathcal{E}(\bar{\Xi}_j)$ in Eq. (25) on CIFAR-10 dataset under non-IID settings. As seen in Fig. 9, the relative accuracy loss first grows slowly and tends to increase fast when $\bar{\Xi}_j > 1.2$. Besides, when $\bar{\Xi}_j = 0$, it corresponds to the

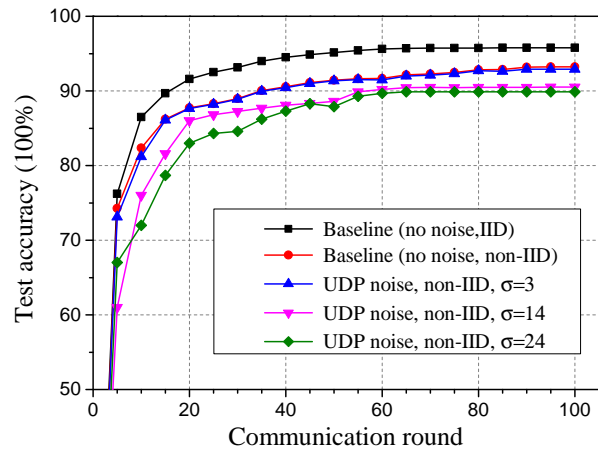


Fig. 10. Test accuracy of federated mining vs. noise parameter σ and data distribution, under different communication rounds (MNIST, $N = 100$).

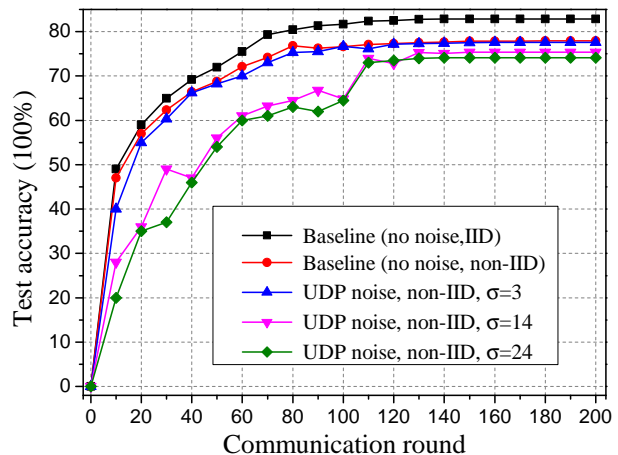


Fig. 11. Test accuracy of federated mining vs. noise parameter σ and data distribution, under different communication rounds (CIFAR-10, $N = 100$).

IID case and we have $\mathcal{E}(\bar{\Xi}_j) = 1$. According to Eq. (22), a higher EMD indicates a higher non-IID degree, and thereby a larger drop on test accuracy.

Figs. 10 and 11 show the evolution of test accuracy on MNIST and CIFAR-10 under different privacy parameters in our UDP noise-adding mechanism, respectively, compared with baseline schemes. In this simulation, the number of participating miners is fixed as 100. As seen in Figs. 10 and 11, the test accuracy of the trained model via federated mining with $\sigma = 3$ UDP noise is very close to the baseline under non-IID, while the accuracies of the trained model with $\sigma = 14$ and $\sigma = 24$ are decreased to a certain degree. It is because a higher σ implies a higher privacy preservation level, causing larger perturbations to be added to the global model update

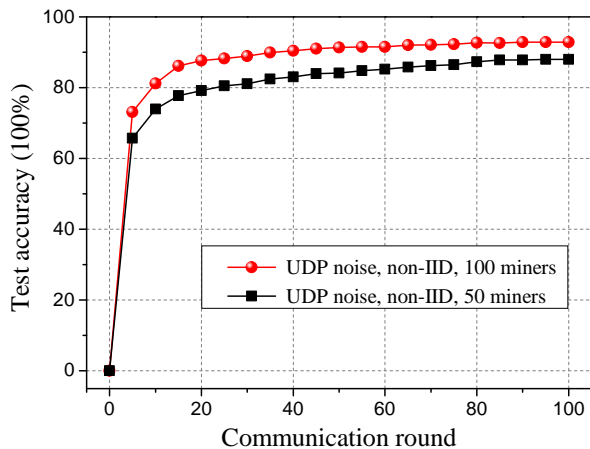


Fig. 12. Test accuracy of federated mining vs. number of participating miners $|\mathcal{S}_j|$, under different communication rounds (MNIST, $\sigma = 3$).

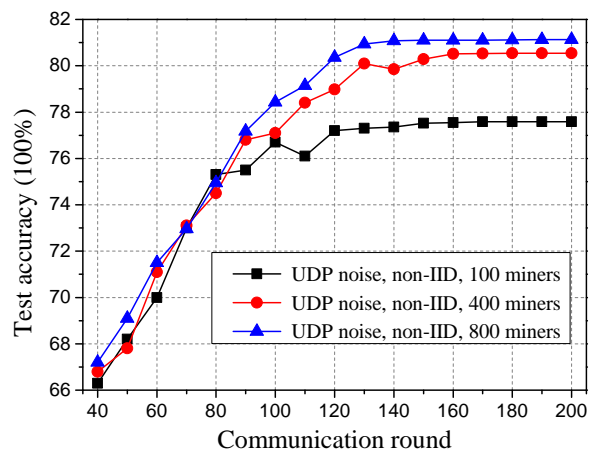


Fig. 13. Test accuracy of federated mining vs. number of participating miners $|\mathcal{S}_j|$, under different communication rounds (CIFAR-10, $\sigma = 3$).

in Eq. (8) which eventually deteriorates the model accuracy. Moreover, compared with the IID case, the existence of non-IID results in an accuracy drop to the baseline.

Figs. 12 and 13 show the evolution of test accuracy on MNIST and CIFAR-10 under different pool sizes (i.e., number of miners in a pool), respectively. In this simulation, we set $\sigma = 3$. From Figs. 12 and 13, we can observe that with a larger pool size, the performance of the trained model is improving in terms of both accuracy and stability. The reason is that given the fixed sample ratio, when the number of participating miners in a pool increases, the impact of the added UDP noise to the global update in Eq. (8) can be reduced, resulting in the higher accuracy and stability in training FL models.

D. Federation Formation Game

In this subsection, we evaluate the performance of the proposed federation formation game in comparison with the exhaustive optimal scheme, the non-cooperative scheme, and the PoFL scheme [15]. In the exhaustive optimal scheme, the optimal pool structure among miners is derived via the exhaustive searching method in a centralized manner. In the non-cooperative scheme, each miner behaves uncooperatively and applies the solo-mining strategy in completing FL tasks.

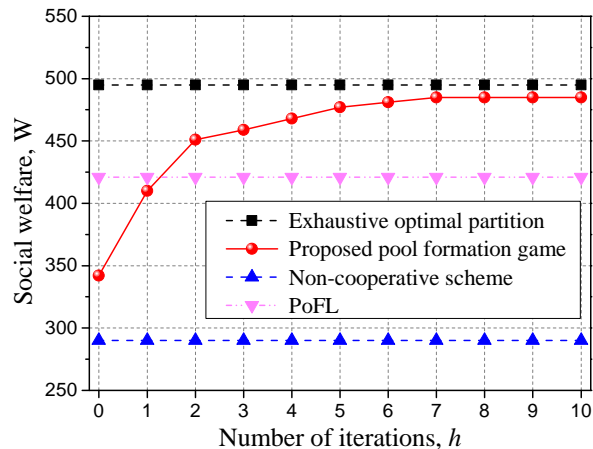


Fig. 14. Evolution of social welfare $W(\mathcal{S})$ of miner partition over time in four schemes ($N = 100$).

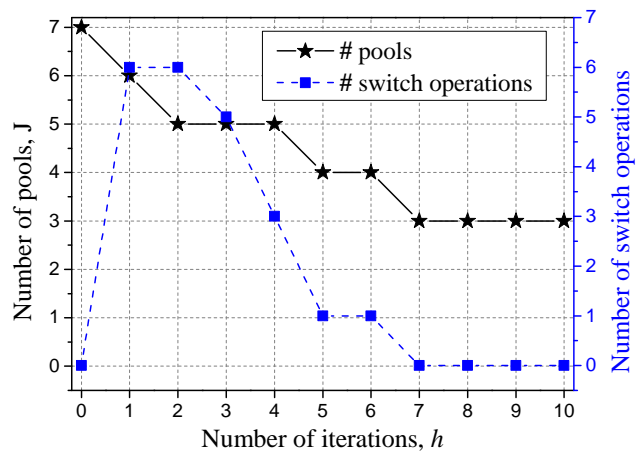


Fig. 15. Evolution of partition structure over time in terms of number of pools and number of switch operations ($N = 100$).

In PoFL scheme [15], all miners utilize the off-the-shelf fixed pooled-mining structure in PoW-based blockchains for federated mining. Here, the total number of miners is set as $N = 100$, where miners perform federated mining on CIFAR-10 using the ResNet18 model under the setting 1.

Fig. 14 compares the social welfare W defined in formula (29) in four schemes. As shown in Fig. 14, when the number of iterations grows, the social welfare of miner partition in the proposed FFG-TU game gradually converges to a stable value after around 7 iterations. Besides, the proposed FFG-TU game can yield a near-optimal performance, as its performance gap to the exhaustive optimal solution does not exceed 2.1% when $h \geq 7$. In addition, as seen in Fig. 14, the stable social welfare in our FFG-TU game is greater than that in the non-cooperative scheme and the PoFL scheme, and it brings up to about 67% improvement to the non-cooperative scheme and about 17.6% improvement to the PoFL scheme when $h = 10$. The reason is that in the non-cooperative scheme, all miners work alone and each of them forms a singleton in conducting FL tasks, resulting in the lowest social welfare. Besides, as different miners generally have distinct training samples and non-IID degrees under heterogeneous FL tasks, the fixed mining pool structure in the PoFL scheme cannot adapt to the varying

federated mining environment, resulting in relatively lower social welfare.

Fig. 15 further inspects the evolution of partition structure among miners over time in terms of the number of pools and the number of switch operations. As seen in Figs. 14 and 15, the switch operations at each iteration can improve the social welfare for miners, which accords with Lemma 1. Moreover, as observed in Fig. 15, the proposed pool formation algorithm converges to a final disjoint partition structure after 7 iterations and achieves Nash-stability, which accords with Theorems 3 and 4. Besides, Fig. 15 shows that, the number of switch operations $|\sum^{(h)}|$ at each iteration h satisfies $|\sum^{(h)}| \leq |\mathfrak{S}^{(h)}| + 1$, where $|\mathfrak{S}^{(h)}|$ is the number of pools in current partition $\mathfrak{S}^{(h)}$. The reason is that, according to the switch rule and admission rule in Definitions 7 and 8, each miner can either switch to another pool or form a singleton if it decides to leave the current pool, while each pool only admits the optimal miner greedily at each iteration.

According to the aforementioned results, our PF-PoFL can effectively recycle energy for efficient federated mining and attain stable throughput of the blockchain system, low block latency in reaching consensus, desirable model performance with UDP guarantees, and optimized federated structure with high social welfare.

VII. CONCLUSION

In this paper, we have proposed a novel energy-recycling consensus mechanism named PF-PoFL, where the wasted energy in solving cryptographic puzzles in PoW is reinvested to FL. To implement PF-PoFL in a fully decentralized manner, we utilize the blockchain to host the unfinished tasks, perform model ranking, and enforce financial settlement by devising a novel block structure, a model ranking contract, and a block rewarding contract. PF-PoFL also incorporates credit-based incentives to motivate miners' honest participation and improve consensus efficiency. Furthermore, a user-level privacy-preserving model training algorithm is designed to offer rigorous privacy protection for miners in each pool. In addition, based on the federation formation game, we present an optimized pool formulation algorithm, where miners with diverse characteristics (i.e., training samples, non-IID degree, and network delay) in heterogenous FL tasks can self-organize into a disjoint Nash-stable partition. Simulation results have validated the efficiency and effectiveness of the proposed PF-PoFL mechanism. For the future work, we will further investigate the optimized PF-PoFL consensus mechanism with optimal FL task rewarding, task difficulty adjustment function, and non-reusability guarantees.

REFERENCES

- [1] Y. Wang, Z. Su, J. Ni, N. Zhang, and X. Shen, "Blockchain-empowered space-air-ground integrated networks: Opportunities, challenges, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 160–209, 2022.
- [2] Z. Xiong, J. Kang, D. Niyato, P. Wang, and H. V. Poor, "Cloud/edge computing service management in blockchain networks: Multi-leader multi-follower game-based admm for pricing," *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 356–367, 2020.
- [3] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2021.
- [4] Y. Wang, Z. Su, N. Zhang, and A. Benslimane, "Learning in the air: Secure federated learning for UAV-assisted crowdsensing," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1055–1069, 2021.
- [5] R. Zhang and B. Preneel, "Lay down the common metrics: Evaluating proof-of-work consensus protocols' security," in *IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 175–192.
- [6] R. Chen, I.-P. Tu, K.-E. Chuang, Q.-X. Lin, S.-W. Liao, and W. Liao, "Endex: Degree of mining power decentralization for proof-of-work based blockchain systems," *IEEE Network*, vol. 34, no. 6, pp. 266–271, 2020.
- [7] Bitcoin energy consumption index. Accessed: Oct. 4, 2021. [Online]. Available: <https://digiconomist.net/bitcoin-energy-consumption/>
- [8] M. Saad, Z. Qin, K. Ren, D. Nyang, and D. Mohaisen, "e-PoS: Making proof-of-stake decentralized and fair," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 8, pp. 1961–1973, 2021.
- [9] G. Ateniese, I. Bonacina, A. Faonio, and N. Galesi, "Proofs of space: When space is of the essence," in *International Conference on Security & Cryptography for Networks*, 2014, pp. 538–557.
- [10] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan, "Proofs of useful work," *IACR Cryptology ePrint Archive*, vol. 2017, pp. 1–28, 2017.
- [11] K. Sunny, "Primecoin: Cryptocurrency with prime number proof-of-work," *self-published*, 2013.
- [12] B. Li, C. Chenli, X. Xu, T. Jung, and Y. Shi, "Exploiting computation power of blockchain for biomedical image segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, pp. 2802–2811.
- [13] Y. Lan, Y. Liu, B. Li, and C. Miao, "Proof of learning (PoLe): Empowering machine learning with consensus building on blockchains (demo)," in *Proceedings of AAAI*, vol. 35, no. 18, May 2021, pp. 16 063–16 066.
- [14] C. Chenli, B. Li, Y. Shi, and T. Jung, "Energy-recycling blockchain with proof-of-deep-learning," in *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019, pp. 19–23.
- [15] X. Qu, S. Wang, Q. Hu, and X. Cheng, "Proof of federated learning: A novel energy-recycling consensus algorithm," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 8, pp. 2074–2085, 2021.
- [16] A. Shoker, "Sustainable blockchain through proof of exercise," in *IEEE 16th International Symposium on Network Computing and Applications (NCA)*, 2017, pp. 1–9.
- [17] P. Daian, I. Eyal, A. Juels, and E. G. Sirer, "(short paper) PieceWork: Generalized outsourcing control for proofs of work," in *Financial Cryptography Workshops*, 2017, pp. 182–190.
- [18] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of 26th Symposium on Operating Systems Principles (SOSP)*, 2017, pp. 51–68.
- [19] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proceedings of NeurIPS*, 2019, pp. 1–11.
- [20] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proceedings of IEEE INFOCOM*, 2019, pp. 2512–2520.
- [21] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 3–18.
- [22] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of ACM CCS*, 2016, pp. 308–318.
- [23] M. Zhang, E. Wei, and R. Berry, "Faithful edge federated learning: Scalability and privacy," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3790–3804, 2021.
- [24] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.
- [25] F. Bravo-Marquez, S. Reeves, and M. Ugarte, "Proof-of-learning: A blockchain consensus mechanism based on machine learning competitions," in *IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*, 2019, pp. 119–124.
- [26] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, pp. 1–7, 2017.
- [27] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *Advances in Cryptology — ASIACRYPT*, vol. 2248, 2001.

- [28] J. Benet, "IPFS-content addressed, versioned, P2P file system," *arXiv preprint arXiv:1407.3561*, pp. 1–11, 2014.
- [29] J. Kang, Z. Xiong, X. Li, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Optimizing task assignment for reliable blockchain-empowered federated edge learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1910–1923, 2021.
- [30] J. Chen and S. Micali, "Algorand," *arXiv preprint arXiv:1607.01341*, pp. 1–75, 2016.
- [31] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," *arXiv preprint arXiv:1806.00582*, pp. 1–13, 2018.
- [32] J. Konecny, H. B. McMahan, D. Ramage, and P. Richtarik, "Federated optimization: Distributed machine learning for on-device intelligence," *ArXiv preprint arXiv:1712.07557*, pp. 1–38, 2016.
- [33] N. Ding, Z. Fang, and J. Huang, "Optimal contract design for efficient federated learning with multi-dimensional private information," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 186–200, 2021.
- [34] M. Li, T. Zhang, Y. Chen, and A. J. Smola, "Efficient mini-batch training for stochastic optimization," in *Proceedings of ACM SIGKDD*, 2014, pp. 661–670.
- [35] A. Das and M. M. Islam, "SecuredTrust: A dynamic trust computation model for secured communication in multiagent systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 2, pp. 261–274, 2012.
- [36] T. Wang, L. Song, Z. Han, and B. Jiao, "Dynamic popular content distribution in vehicular networks using coalition formation games," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 9, pp. 538–547, 2013.
- [37] W. Saad, Z. Han, A. Hjørungnes, D. Niyato, and E. Hossain, "Coalition formation games for distributed cooperation among roadside units in vehicular networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 1, pp. 48–60, 2011.
- [38] J. S. Ng, W. Y. B. Lim, Z. Xiong, X. Cao, J. Jin, D. Niyato, C. S. Leung, and C. Miao, "Reputation-aware hedonic coalition formation for efficient serverless hierarchical federated learning," *IEEE Transactions on Parallel and Distributed Systems*, 2021, doi: 10.1109/TPDS.2021.3139039.
- [39] Go-Algorand project. [Online]. Available: <https://github.com/algorand/go-algorand>
- [40] PyTeal: Algorand smart contracts in Python. [Online]. Available: <https://pyteal.readthedocs.io/en/latest/>
- [41] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [42] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Handbook of Systemic Autoimmune Diseases*, vol. 1, no. 4, 2009.

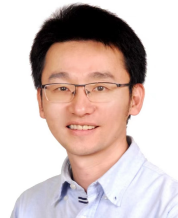


Yuntao Wang received the Ph.D degree in Cyber Science and Engineering from Xi'an Jiaotong University, Xi'an, China, in 2022. His research interests include security and privacy protection in wireless networks, vehicular networks, and UAV networks.



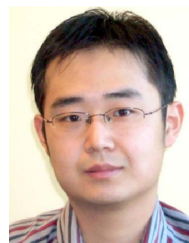
Haixia Peng received her Ph.D. degrees in Computer Science and Electrical & Computer Engineering from Northeastern University (Shenyang, China) in 2017 and the University of Waterloo (Waterloo, Canada) in 2021, respectively. She is currently an Associate Professor with the School of Information and Communications Engineering, Xi'an Jiaotong University, China. Her research interests include satellite-terrestrial vehicular networks, multi-access edge computing, resource management, artificial intelligence, and reinforcement learning.

She serves/served as a TPC member in IEEE VTC-fall 2016&2017, IEEE ICCEREC 2018, IEEE GlobeCom 2016-2022, and IEEE ICC 2017-2022 conferences and serves as an Associate Editor for the PEER-TO-PEER NETWORKING AND APPLICATIONS.



Zhou Su has published technical papers, including top journals and top conferences, such as IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE/ACM TRANSACTIONS ON NETWORKING, and INFOCOM. His research interests include multimedia communication, wireless communication, and network traffic. Dr. Su received the

Best Paper Award of International Conference IEEE ICC2020, IEEE Big-dataSE2019, and IEEE CyberSciTech2017. He is an Associate Editor of IEEE INTERNET OF THINGS JOURNAL, IEEE OPEN JOURNAL OF COMPUTER SOCIETY, and IET COMMUNICATIONS.



Tom H. Luan received the Ph.D. degree from the University of Waterloo, Canada, in 2012. He is currently a Professor with the School of Cyber Science and Engineering, Xi'an Jiaotong University, China. His research mainly focuses on content distribution and media streaming in vehicular ad hoc networks and peer-to-peer networking and the protocol design and performance evaluation of wireless cloud computing and edge computing. He served as a TPC Member for IEEE Globecom, ICC, and PIMRC.



Abderrahim Benslimane is Full Professor with the Laboratory of Computer Sciences at the Avignon University, France. He is Chair of the ComSoc Technical Committee of Communication and Information Security. He is EiC of Inderscience Int. J. of Multimedia Intelligence and Security (IJMIS), Area Editor of Security in IEEE IoT Journal, Area Editor of Wiley Security and Privacy Journal and Editorial Member of IEEE Wireless Communication Magazine, Elsevier Ad Hoc, IEEE Systems and Wireless Networks Journals.



Yuan Wu received the PhD degree in Electronic and Computer Engineering from the Hong Kong University of Science and Technology in 2010. He is currently an Associate Professor with the State Key Laboratory of Internet of Things for Smart City, University of Macau and also with the Department of Computer and Information Science, University of Macau. His research interests include resource management for wireless networks, green communications and computing, mobile edge computing and edge intelligence. He was a recipient of the Best

Paper Award from the IEEE ICC2016, and the Best Paper Award from IEEE Technical Committee on Green Communications and Computing in 2017. He is currently on the Editorial Boards of IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, IEEE INTERNET OF THINGS JOURNAL, and IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY.