Blockchain-Based Privacy-Preserving Federated Learning for Mobile Crowdsourcing

Haiying Ma, Shuanglong Huang, Jiale Guo, Kwok-Yan Lam, *Senior Member, IEEE*, Tianling Yang

***Abstract*—Mobile crowdsourcing (MCS) is an emerging paradigm that enables the outsourcing of a complex task to a group of mobile devices. The ability to utilize the collective power of mobile devices and human intelligence makes MCS a significant tool in various scenarios. Nevertheless, it faces practical challenge in protecting user privacy due to the sensitive nature of information collected by mobile devices. Additionally, the inherent openness of MSC and the heterogeneity of mobile devices raise reliability concerns among participants. To address these challenges, by integrating Federated Learning with the pairwise additive masking technique and the Chinese Remainder Theorem, we propose a Blockchain-based Privacy-preserving Federated Learning (BPFL) framework for mobile crowdsourcing, which allows mobile participants to collaboratively solve a crowdsourced machine learning task while preserving privacy. Besides, it employs blockchain technology to record the training process in a transparent and tamper-proof ledger. This ledger guarantees the verifiability of aggregation results and the fair distribution of training rewards, thereby enhancing trust and fairness. We prove that our BPFL supports privacy protection and trust mechanism simultaneously and resists inference and collusion attacks. Experimental results show that our BPFL can achieve high performance in terms of computation cost, communication cost and model accuracy, which is friendly for mobile users with resource-constrained devices in MCS ecosystems.**

***Index Terms*—Mobile Crowdsourcing, Federated Learning, Blockchain, privacy preservation, Chinese Remainder Theorem.**

I. Introduction

With the rapid advances of mobile and communication technologies, mobile devices equipped with various sensors have become an essential part of our daily life. By taking advantage of mobile devices and human intelligence, mobile crowdsourcing (MCS) has been identified as an emerging paradigm that allows affordable outsourcing a complex task to an unspecified group of mobile devices [1]. An MCS system is open for mobile devices engaging in various sensing and computing tasks. This system not only enables instant data collection and data processing in a flexible and cheap way, but also provides a way to utilize human and machine intelligence [2]. Due to these benefits, MCS finds extensive application across various scenarios, such as environmental monitoring [3], traffic and navigation [4], social networking [5], *etc*.

Despite the significant benefits that MCS brings, it still faces crucial challenges in terms of privacy and trust [1-14]. First, the data sensed by mobile devices inevitably contains sensitive information of user privacy, such as personal health information and financial records. The lack of privacy protection mechanisms not only may discourage the engagement of mobile participants in MCS due to concerns about potential privacy breaches [3], but also exposes the MCS system to the risk of violating the relevant data privacy regulations. Second, establishing a trust mechanism among service providers, task issuers, and mobile participants also poses a great challenge. When a task issuer turns to service provider to complete a certain task, it is reasonable to provide him with a trust way to verify the correctness of processed results [2]. Moreover, attracting mobile participants to actively engage in MCS tasks often involves integrating incentive mechanisms [8, 10, 12]. However, the effectiveness of such mechanisms can be hindered by a variety of factors. There should be a transparent and verifiable mechanism to guarantee that the honest mobile participants who complete the task duly receive their well-earned rewards. Motivated by these crucial issues, an open and challenging research question is how to construct a mobile crowdsourcing solution that supports privacy protection and trust mechanism simultaneously.

To protect the privacy of participants in MCS, some popular privacy-preserving methods have been introduced in existing literature, *e.g.*, differential privacy (DP) [10], homomorphic encryption (HE) [15], *k*-anonymity [16], uploading cloaked information [17] or adding random noise [18]. Jin *et al.* [10] proposed the privacy-preserving incentives for MCS systems, that added a random perturbation to the bidding information by using difference privacy. Su *et al.* [15] presented a privacy-preserving verifiable incentive mechanism for MCS by using homomorphic encryption. Wu *et al.* [16] used *k*-anonymity to cluster participants into one group to protect location privacy of MCS. The scheme [18] divided a whole sensing area into several subareas based on privacy budget and random noise. However, the methods based on DP [10], uploading cloaked information [17], and adding random noise [18] may affect the

This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative and Strategic Capability Research Centres Funding Initiative, the Nantong Natural Science Foundation (No. JC2021128, No. JC22022036), the National Natural Science Foundation of China (No. 61762044, No. 62072259, No. 61402244, No. 11371207). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority. (*Corresponding author: Haiying Ma,* e-mail: mhy8855@ntu.edu.cn, *Kwok-Yan Lam,* e-mail: kwokyan.lam@ntu.edu.sg)

Haiying Ma, Shuanglong Huang and Tianling Yang are with School of Information Science and Technology, Nantong University, China, 226019. (e-mail: mhy8855@ntu.edu.cn, 2010320057@stmail.ntu.edu.cn, yangtianling@stmail.ntu.edu.cn). Haiying Ma and Shuanglong Huang contribute equally to this work.

Jiale Guo and Kwok-Yan Lam are with School of Computer Science and Engineering, Nanyang Technological University, Singapore, 639798. (e-mail: jiale001@e.ntu.edu.sg, kwokyan.lam@ntu.edu.sg)

accuracy of results. When faced with large-scale volumes of data, HE-based method [15] needs perform complex cryptographic calculations and unavoidably leads to ciphertext expansion, which greatly increases computation and communication overheads. Additionally, these schemes may not make full use of the computing power of mobile devices.

In this context, federated learning (FL) [19] has attracted much attention as a promising solution to achieve privacy preserving in MCS. In the traditional FL paradigm, each participant may use its local data to train a local model gradient, and the local updated gradient of each iteration will be uploaded to an aggregation server to update a global model. Nevertheless, the shared gradients can still leak privacy information about participants' training data in FL [20-22]. To enhance the privacy preservation of FL, many schemes have been proposed, which are mainly based on differential privacy technique [23-27], homomorphic encryption [28-36], secure multi-party computation (MPC) [37-40], and masking-based approach [41-43]. However, integrating these solutions into practical MCS systems may not be straightforward. In general, DP-based schemes improve privacy by disturbing the participants' local gradients with noise before uploading them to the aggregation server. However, it may affect the accuracy of the global model due to noise accumulation. Alternatively, HE-based methods allow participants to encrypt their local gradients and then the aggregation server will perform the encrypted model aggregation according to homomorphic operations. However, such methods may not quite practical for resource-constrained mobile device because they introduce high computational and communication overheads. MPC-based schemes usually let participants secretly share their local gradients with others or multiple servers. While each secret share is as long as the size of the entire data vector, *i.e*., the entire local gradients. Then, the communication overhead of MPC-based methods limits its application in practical MCS scenarios. To further improve the efficiency, masking-based approach [41] blinds participants' local models using a well-designed mask matrix that can be automatically canceled during aggregation process. However, it requires each participant to run the Diffie-Hallman key agreement protocol to agree on a key and share his/her secret key with all other participants in each round.

In addition, the traditional centralized FL usually relies on a central server to aggregate the local updates, which may make it vulnerable to several threats, such as single-point-of-failure, malicious aggregation, *etc*. Furthermore, these schemes are also unable to prevent the misbehaviors of the aggregation server in local model collection and aggregation [44-46]. To address the mentioned trust issue, the integration of blockchain technology into FL becomes a promising solution through its decentralized architecture. Specifically, this integration decentralizes the model aggregation to the blockchain nodes, and uses the immutable blockchain ledger to record the training process, thereby improving the robustness of FL [25, 47]. Besides, the consensus protocol of blockchain can be adapted to guarantee that only correct aggregation results are stored on the blockchain. In this way, the task publisher can have a transparent overview of the whole FL training process and verify the correctness of the final result. Moreover, leveraging on smart contracts, it can deploy an automatically executed incentive mechanism to ensure honest and equitable reward distribution among participants. Overall, blockchain applied in FL can establish a feasible trust mechanism among service providers, task publishers and mobile participants. Therefore, FL and blockchain technologies can be combined to address privacy and trust issues in MCS.

Inspired by these considerations, we propose a new blockchain-based privacy-preserving federated learning for mobile crowdsourcing, called BPFL, which not only can preserve participants' privacy but also enhance the trust among task publisher and all participants. Our contributions are summarized as follows:

(1) A privacy-enhanced FL is proposed which allows mobile participants to collaboratively solve a crowdsourced machine learning task in a privacy-preserving manner.

(2) The proposed scheme is enhanced with a transparent and tamper-proof ledger and fairly distribute the training rewards, hence allows the verifiability of aggregation results and the fair distribution of training rewards, thereby enhancing trust and fairness.

(3) We prove that our BPFL protocol supports privacy protection and trust simultaneously for MCS, and resists inference and collusion attacks. Experimental results show that our BPFL can achieve high performance in terms of computation cost, communication cost and model accuracy, which is friendly for mobile users with resource-constrained devices in MCS ecosystems.

The rest of the paper is organized as follows. First, we introduce some related works in Section II. In section III, we describe the preliminaries. Then, we construct the our BPFL scheme in detail in Section IV. Section V gives the security analysis. Experiment evaluation is presented in Section VI.

II. Related Works

A. Masking-Based Privacy-Preserving FL

With the growing concerns over personal data privacy, privacy-preserving federated learning (PPFL) has attracted tremendous attention from both academia and industry [22, 48]. Based on the variant ElGamal, Yang *et al.* [49] proposed a secure aggregation protocol without a trusted third party, and introduced a super-increasing sequence to compress multi-dimensional data into one-dimensional. However, complex encryption and decryption computations may not be suitable for mobile devices. As mentioned in [41-43], masking-based approach can be adopted to mask the model gradients to protect participants' privacy and reduce the computational overheads. To improve the communication cost of SecAgg [41], some follow-up works [50, 51] are designed to allow the users to communicate with only a subset of all users. For example, SecAgg+ [50] adopts the aggregation technology of non-group architecture and Harary directed graph to reduce the communication complexity from $O(n^2)$ to $O(n\log n)$. But it still requires clients to run a Diffie-Hellman key exchange with each other and share their secrets with all other clients. TurboAgg [51] divides n users into $n/\log n$ groups and follows

a multi-group circular structure for aggregation. However, these schemes require additional communication rounds to process between groups. To verify the correctness of the aggregation results, Xu *et al*. [43] proposed the first verifiable PPFL framework, using the homomorphic hash function integrated with pseudorandom technology to verify the correctness of results with acceptable overhead. Therefore, the above schemes may not be suitable for mobile crowdsourcing, due to their high communication costs.

To improve the efficiency of the masking-based FL scheme, another direction is to replace the Diffie-Hellman key exchange with a lightweight or non-interactive algorithm. For example, with the assistance of two non-colluding cryptographic secret providers, *i.e*., aided servers, the seed of each pair of users can be generated in a non-interactive way [52]. However, the trust distribution of [52] is limited to the number of aided servers and non-colluding assumptions, hence limiting its application scenarios. Fu *et al*. [42] combined Lagrange interpolation method and pairwise additive masking techniques to propose a verifiable FL scheme, which greatly reduces communication overhead by generating pairwise masking seeds through a trusted entity. Nonetheless, this scheme requires clients to compute the Lagrange interpolation function that introduces excessive extra computational costs. Besides, we should note these schemes usually rely on a single server to aggregate the local updates, which may make the system vulnerable to several attacks, such as DDoS attack. Additionally, these schemes are also unable to prevent the misbehaviors of the aggregation server in local model collection and aggregation. In this backdrop, some researches attempt to integrate blockchain with FL to enhance security.

B. Blockchain-Based Privacy-Preserving FL

As blockchain has the properties of decentralization, transparency, and immutability, it can establish trust between nodes in an untrusted environment [53]. To address above issues, some works [54, 55] use a smart contract pre-deployed in blockchain to aggregate local models, but impose heavy computation and communication burdens on the nodes in blockchain. Therefore, Weng *et al.* [47] propose a blockchain-based privacy-preserving FL, using the Algorand consensus mechanism to calculate and verify the aggregation results, and utilizes a threshold HE to protect the local model privacy. However, the threshold HE greatly increases the computational and communication overhead of the participants. Zhao *et al.* [25] also use the Algorand consensus mechanism to solve the single-point-of-failure and malicious aggregation issues, but use DP technology to protect the privacy of participants. While improving the computational efficiency of the participants, it may affect the accuracy of the model.

Overall, the abovementioned researches have achieved great developments on privacy-preserving FL, but they may not be applicable in mobile crowdsourcing scenarios due to their communicational costs [43, 50, 51], computation costs [42], or both [47, 54, 55]. Besides, the scheme proposed in [52] cannot be deployed if there is only one server. If high model accuracy is required, the scheme proposed in [25] may not be suitable. Regarding these issues, the goal of this work is to design a PPFL that is computation and communication efficient for resource-limited devices and can achieve lossless accuracy.

III. PRELIMINARIES

This section briefly describes the preliminaries of the technologies used in this paper, including the Chinese Remainder Theorem, blockchain and Algorand consensus protocol, and incentive mechanism. Federated Learning technology used in this paper is a well-known distributed machine learning technology allowing a set of participants to collaboratively train a machine learning model using their local data. We refer interested readers to [19, 56] for the details.

A. The Chinese Remainder Theorem

Theorem 1. For non-negative integers $\{p_k \mid k = 1, 2, \cdots, m\}$ such that $gcd\,(p_i, p_j) = 1$ for any $i \neq j$ and integers $\{b_k \mid k = 1, 2, \cdots, m\}$ consider the system of congruential equations [57, 58]:

$$\begin{cases} y \equiv b_1 \pmod{p_1} \\ y \equiv b_2 \pmod{p_2} \\ \quad \cdots \\ y \equiv b_m \pmod{p_m} \end{cases}. \tag{1}$$

This system always has a solution in the sense of modulo S:

$$y \equiv \sum_{k=1}^{m} b_k S_k T_k \pmod{S}, \tag{2}$$

where $S = \prod_{k=1}^{m} p_k$, $S_k = S/p_k$, $T_k \equiv S_k^{-1} \pmod{p_k}$. In this paper, since $\{p_k \mid k = 1, 2, \cdots, m\}$ is kept secret for the aggregator, it can be used to protect privacy of the global aggregation model. Moreover, the CRT contributes to reduce the size of ciphertext and the communication costs.

B. Blockchain and Algorand Consensus Protocol

Blockchain is a distributed ledger, which may store time series data of transactions into a chain structure, and has excellent characteristics such as immutability, transparency, and auditability [53]. The consensus mechanism is the cornerstone part of the blockchain, and is used to regulate the behavior of the consensus nodes, ensuring the integrity and consistency of the stored data [53]. Due to these properties, blockchain has emerged as a promising technology that is used in various applications. For example, Haddad *et al.* [59, 60] employ blockchain as a distributed public key management system in their authentication and key agreement scheme for 5G network, it allows the network entities to authenticate each other without using digital certificates, thereby reducing the computation and communication overhead. Besides, blockchain technology can also be used in cloud storage [61], crowdsourcing [6], and so on.

In this work, we employ the Algorand consensus protocol to select an aggregator in each round to aggregate the local models and then ensure the correctness and integrity of the aggregated model. Algorand is a consensus protocol based on proof-of-stake (PoS) and Byzantine Fault Tolerance (BFT). It employs committee-based PoS to generate blocks and relies on the improved Byzantine agreement to reach block consensus [62]. Specifically, reaching consensus requires the following steps:

> <

(1) the block proposal phase: After receives the block of the previous round, each worker runs Verifiable Random Function (VRF) locally to generate a *hash* and a *proof*, where the *hash* is used to compute the number *j* of the selected sub-users, the *proof* is used by other workers to confirm whether the worker is selected *j* sub-users. Then each worker broadcasts its own *hash* and *proof*. Therefore, after each worker receives *proofs* from other workers, it can determine whether it is the leader (*i.e.*, judge whether it is the highest priority according to the size of *j*), if the worker is selected as a leader, it has the right to generate a new block and broadcast it.

(2) the block finalization phase: After all workers receive the new block, they run VRF to determine whether they can vote as validators (a worker whose number *j* of selected sub-users exceeds a predetermined threshold is a validator). Then, validators verify the correctness of the transaction data stored in this new block, and broadcast their votes. When more than 2/3 of the validators agree on the leader's block, the new block is accepted. Otherwise, the next priority worker becomes the leader and repeats the new block generation and voting process.

C. Incentive Mechanism

To attract more users to contribute to MCS tasks, we introduce an incentive mechanism. Many works use data quality as an important criterion for measuring participants' contributions [33-35]. For example, Zhang *et al.* [63] designed an incentive mechanism, where the coins awarded to each participant are dependent on the size and the centroid distance between the positive class and the negative class of dataset, *i.e.*, using the distance to measure the quality of the dataset. Kang *et al.* [64] proposed an incentive mechanism by combining reputation with contract theory to train the learning model, and use the earth mover's distance (EMD) to measure the local data quality of unreliable participants. The EMD is the distance between two distributions, which is a measure of image similarity [65].

IV. Blockchain-Based Privacy-Enhanced Federated Learning for Mobile Crowdsourcing

A. System Model

Fig. 1 shows the system architecture of our BPFL, which consists of five primary components: System Manager (SM), Task Publisher (TP), participants, blockchain and Distributed File Storage system (DFS).

System Manager (SM). The SM is a trusted entity (such as a government authority) that constructs a consortium blockchain, on which the task publishers and participants can register. The SM needs to generate and distribute the secure parameters for each group of participants using a distributed method such as the one in [66]. Moreover, the SM can decrypt the final model parameters and send them to the corresponding TPs via their secure channels.

Task Publisher (TP). The TP can be a mobile device manufacturer that raises a request to train high-quality learning models to solve data-driven problems in MCS [67-69]. Leveraging the FL framework, the TP crowdsources the model training task to the participants, to improve model performance. By the model-based pricing for machine learning in recent works [70, 71], TPs can utilize the consensus protocol and the incentive mechanism of blockchain to solve the collusion attack and the reward fairness problems.

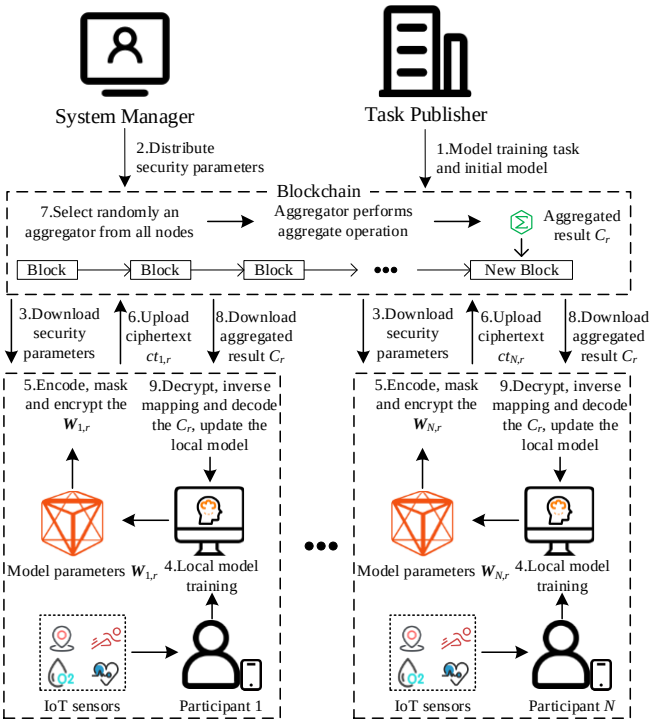


Fig. 1. The system architecture of our BPFL protocol.

Participants. Each participant has many data collected from mobile devices satisfying a kind of crowdsourcing requirement. However, the participant is unable to perform the whole training task alone, it can take part in an FL task that trains a model locally. To preserve privacy, participants will encrypt their model parameters and broadcast the ciphers to the blockchain to generate the aggregated ciphers. Then, participants download the aggregated cipher to update their local model parameters for the next round of model training. Moreover, we assume that participants are honest but curious, and they are willing to upload the correct model gradients. But they will attempt to learn privacy from all received messages. Besides, they may also collude with other participants or the selected leader to obtain other participants' model parameters.

The design goal of this paper is to protect the data privacy of participants and does not involve the anonymity and unlinkability of participants' transactions. For the issue of malicious participants influencing the aggregation results, we will address it in the future works.

Blockchain. The blockchain is used to record the training process of the FL task permanently, aggregate and verify the aggregation results. In this work, by leveraging the consensus protocol and smart contract of blockchain, we design a transparent and tamper-proof ledger. The ledger guarantees the verifiability of aggregation results and the fair distribution of training rewards.

Distributed File Storage System (DFS). In our platform, this DFS is designed in a multi-layered, modular design encompassing storage nodes/cloud servers located in multiple geographic locations and dynamically scalable entities

responsible for the efficient encapsulation, retrieval, and replication of stored ciphertexts. We also have a centralized metadata server that coordinates these operations, making our platform robust and scalable.

TABLE I
SUMMARY OF NOTATIONS

Symbol	Definition
r	r-th round of global model training
N	number of participants
P_i	participant i
m, ($m \geq 3$)	encryption complexity parameter
l	used to control the computing precision
$p_0, p_1, p_2, \cdots, p_m$	m+1 positive primes such that $gcd(p_i, p_j)=1$, $(i \neq j)$
$\boldsymbol{M}$	$N \times N$ positive integer matrix
$\{M_i^{(j,0)}\}$, $\{M_j^{(i,0)}\}$ ($j = 1, 2, \cdots, N$)	two seed vectors received by P_i
S, S_k, T_k, ($k = 1, 2, \cdots, m$)	public parameters for CRT encryption
$f(x, \boldsymbol{W})$	neural network model
η	learning rate
D_i	the dateset of participant i
$L_f(\cdot)$	loss function
$\boldsymbol{W}_{r-1}$	TP_j's model parameter
$\boldsymbol{W}_{i,r}$	the model parameters trained by participant i in r-th round training
$\bar{\boldsymbol{W}}_{i,r}$	participant i's encoded model parameters in r-th round
$\hat{\boldsymbol{W}}_{i,r}$	participant i's blinded model parameters in r-th round
$ct_{i,r}$	participant i's encrypted model parameters in r-th round
C_r	aggregated model parameters ciphertext in r-th round
$\{B_k^{(r)} \mid k = 1, 2, \cdots, m\}$	the aggregated discrete values via decrypt the ciphertext C_r
$\{\hat{B}_k^{(r)} \mid k = 1, 2, \cdots, m\}$	from the finite field to integers for $\{B_k^{(r)} \mid k = 1, 2, \cdots, m\}$
$\bar{\boldsymbol{W}}_r$	the encoded model parameters in r-th round
$\boldsymbol{W}_r$	the real aggregation parameters in r-th round

To address the size constraints of the blocks in a blockchain, we utilize the DFS to off-chain store the ciphertext of local and global gradients. Upon the user's submission of ciphertext into our DFS, hashing algorithms parse the input and yield a unique, cryptographically-secure identifier to guarantee the integrity of the stored data. This identifier paired with its associated node spatial descriptors can be recorded on a blockchain-based ledger and used to locate the exact file across the DFS systems. This hybrid-storage mode can make the blockchain light-weight and significantly improve the data storage capacity of the blockchain.

The BPFL scheme consists of five phases: global system initialization, local model training, model gradient aggregation, local model update, and obtain final model gradients.

Notations. Notations used in the paper are summarized in Table I.

B. Global System Initialization Phase

First, the SM sets up the blockchain with the Algorand protocol. All participants and TPs can register to have a valid account (*i.e.*, a pair of public key/secret key (*pk*, *sk*), a unique identifier *id*, a wallet address *wa*) and a deposit on the blockchain, where each entity can use its wallet address to launch transactions.

All participants, TPs and workers need to lock a part of their coins on the blockchain as the deposit, containing the initial transactions recording ownership statements of their coins. The TPs need to state their assets, which enables them to find participants to fulfill their FL tasks. The asset statement only contains some description of FL tasks, and the asset can be used for publishing an FL task.

Each TP can launch a transaction to publish an FL task, *i.e.*, the initial model parameters W_0, model index *mid*, model topic, learning rate η, training time per round t (assuming this time period is large enough, all participants can complete model training and transaction uploading steps within this time period), model accuracy requirements for completing FL tasks θ, the number of participants N and other model description.

Suppose that the TP_j launches a transaction to state its asset as follows: $TX_{TPj} = \{mid_j, t_j, H(\boldsymbol{W}_{j,0}), \sigma(sk_j, H(\boldsymbol{W}_{j,0})), N_j, \eta_j, \theta_j,$ "*Keywords*"$\}$, where $H(\boldsymbol{W}_{j,0})$ is the hash value of the initial model parameters $\boldsymbol{W}_{j,0}$ indicating its storage location in DFS, sk_j is TP_j's secret key, $\sigma(sk_j, H(\boldsymbol{W}_{j,0}))$ is TP_j's signature which proofs that TP_j indeed possesses the model, and "*Keywords*" denotes model description of this FL task.

According to the FL model "*Keywords*", participants can voluntarily join the TP_j's FL training for benefits. Before FL training starts, the participant P_i (i =1, 2, ⋯, N) should generate a transaction to state its data asset D_i as follows: $TX_{Di} = \{sid, H(D_i), \sigma(sk_i, H(D_i)), H(TX_{TPj})$, "*Keywords*"$\}$, where *sid* is the index of dataset D_i, sk_i is P_i's secret key, $H(D_i)$ is the hash value of D_i without leaking the sensitive content of D_i, $\sigma(sk_i, H(D_i))$ is the P_i's digital signature proofing that P_i indeed possesses the dataset D_i, and $H(TX_{TPj})$ is the hash value of TX_{TPj} proving that P_i joined the FL task created by TP_j. Then, each TP can establish an FL group and audit the participants' assets to ensure authenticity of asset ownership using the method in [47].

The TP_j will retrieve the number of participants after TX_{Di} is sent to the blockchain. When at least N participants have joined the FL task, the TP_j queries all transactions and selects N participants from them, and then generates a list *List* of participants' wallet addresses and public keys. Next, the TP_j launches a transaction to employ statement as follows: $TX_{employ} = \{List, H(TX_{TPj}), H(List), \sigma(sk_j, H(List))$, "*Keywords*"$\}$. Then, the TP_j sends TX_{employ} to the blockchain.

For ease of notation, we remove subscript j of $\{mid_j, t_j, N_j, \eta_j, \theta_j, sk_j, \boldsymbol{W}_{j,0}, TP_j\}$, focus on a representative TP form now. Suppose there are N ($N \geq 4$) participants $P_1, P_2, \cdots, P_N$ who can build a group to train a machine learning model for the TP's FL task. The SM needs to generate and distribute the following parameters and Pseudo-random Generator (*PRG*) as follows:

(1) The SM first gets the number of participants N, model accuracy requirements θ, the learning rate η and initial model parameters $\boldsymbol{W}_0$ from the TP's transaction TX_{TP}, chooses a complexity parameter m which is tunable, a positive integer l which controls the computing precision, and a training start time t_0. Then the SM generates $m + 1$ positive primes $p_0, p_1, p_2, \cdots, p_m$ that are pairwise co-prime $gcd(p_i, p_j) = 1$, $(i \neq j)$.

(2) The SM randomly generates an $N \times N$ positive integer

matrix $\boldsymbol{M}$ as a seed matrix, and generates two vectors $\{\boldsymbol{M}_i^{(j,0)} \mid j = 1, 2, \cdots, N\}$ and $\{\boldsymbol{M}_j^{(i,0)} \mid j = 1, 2, \cdots, N\}$ which denote the i-th row and the i-th column of the seed matrix $\boldsymbol{M}$ for each participant. Besides, the SM also generates the $PRG(\cdot)$ satisfying $PRG(\cdot+*) = (PRG(\cdot)+PRG(*)) \pmod{p_0}$.

(3) Finally, the SM reads the list *List* according to the TP_j's transaction TX_{employ} to obtain the wallet addresses and the public keys $\{P_i.wa, P_i.pk \mid i = 1, 2, \cdots, N\}$ of all participants. Next, the SM sends these security parameters $\{m, l, p_0, p_1, p_2, \cdots, p_m\}$ and two vectors and the $PRG(\cdot)$ to N participants via their secure channels. Specifically, the steps of the SM sending before-mentioned security parameters to the i-th participant P_i are shown as follows: (i) The SM encrypts $\{m, l, p_0, p_1, p_2, \cdots, p_m, \{\boldsymbol{M}_i^{(j,0)} \mid j = 1, 2, \cdots, N\}, \{\boldsymbol{M}_j^{(i,0)} \mid j = 1, 2, \cdots, N\}, PRG(\cdot)\}$ with the public key $P_i.pk$ of the i-th participant P_i to generate the ciphertext SP_i; (ii) The SM stores the ciphertext SP_i onto the DFS, and gets the storage hash address $H(SP_i)$ in DFS. Then it generates a transaction $TX_{Pi} = \{P_i.wa, t_0, H(TX_{TP}), H(SP_i), \sigma(\text{SM}.sk, H(SP_i))$, "*Keywords*"$\}$; (iii) The SM sends TX_{Pi} to the blockchain.

C. Local Model Training Phase

Each participant P_i (i =1, 2, $\cdots$, N) downloads the initial model parameters $\boldsymbol{W}_0$ from the DFS according to the hash address $H(\boldsymbol{W}_0)$, and obtains the model performance requirement θ and learning rate η from the TP's transaction TX_{TP}. Before performing the encryption operation, P_i queries the transaction TX_{Pi} from the blockchain according to his wallet address $P_i.wa$, then downloads ciphertext from DFS according to the address $H(SP_i)$, and decrypts it using his own secret key sk_i.

Next, P_i uses its local dataset to train the machine learning model. After completing each round of the model training, P_i encrypts the model parameters and uploads the ciphertext onto the DFS. Therefore, each participant P_i needs to perform three steps for each global round: model training, encoding and encryption and uploading parameter ciphertext.

Suppose that the participant P_i holds the dataset $D_i = \{<x_k, y_k> \mid k = 1, 2, \cdots, n\}$. Let $f(x, \boldsymbol{W})$ denote the neural network model, where x is the input and $\boldsymbol{W}$ is the model parameter. We use the cross-entropy function as the loss function:

$$L_f(D_i, \boldsymbol{W}_r) = -\frac{1}{n}\sum_{k=1}^{n}\left[y_k log\left(f(x_k, \boldsymbol{W}_{r-1})\right) + (1-y_k) log\left(f(x_k, \boldsymbol{W}_{r-1})\right)\right], \quad (3)$$

where D_i is the P_i's dataset, $<x_k, y_k> \in D_i$, x_k is the input, y_k is the label, n is the size of D_i. For a multi-classification problem, we can combine the *softmax* function and the cross-entropy function to obtain the output probability of each class [72].

(1) Model Training

The P_i uses TP's model parameter $\boldsymbol{W}_{r-1}$ to compute the r-th round local model parameters $\boldsymbol{W}_{i,r}$. The goal of training the neural network model is to find the model parameters $\boldsymbol{W}_{i,r}$ for minimizing the value of the loss function $L_f(D_i, \boldsymbol{W}_{r-1})$. To achieve this goal, many optimization methods based on gradient descent are used, which can update the model parameters by back-propagating the error at each iteration [72]. Then P_i computes the gradient of its loss function in r-th as follows:

Algorithm 1 Model parameters encryption.

Input: Private model parameters $\boldsymbol{W}_{i,r}$;
Security parameters: m, l, $\{\boldsymbol{M}_i^{(j,r)}\}$ and $\{\boldsymbol{M}_j^{(i,r)}\}$ ($j \in N$).
Output: Model parameters ciphertext $ct_{i,r}$.
1: Encode the $W_r^{(j,i)}$:

$$\bar{\boldsymbol{W}}_{i,r} = \varphi(\boldsymbol{W}_{i,r});$$

2: Blind the $\bar{\boldsymbol{W}}_{i,r}$:

$$\hat{\boldsymbol{W}}_{i,r} = \bar{\boldsymbol{W}}_{i,r} - \sum_{j=1}^{N}\boldsymbol{M}_i^{(j,r)} + \sum_{j=1}^{N}\boldsymbol{M}_j^{(i,r)};$$

3: Randomly split $\hat{\boldsymbol{W}}_{i,r}$ into m parts $\{b_k^{(i,r)} \mid k = 1, 2, \cdots, m\}$, that satisfies:

$$\hat{\boldsymbol{W}}_{i,r} = \sum_{k=1}^{m} b_k^{(i,r)};$$

4: Use the CRT to encrypt $\{b_k^{(i,r)} \mid k = 1, 2, \cdots, m\}$ and package:

$$ct_{r,i} \equiv \sum_{k=1}^{m} b_k^{(i,r)} S_k T_k \pmod{S};$$

5: Return ciphertext $ct_{i,r}$.

$$\nabla L_f\left(D_i^*, \boldsymbol{W}_{r-1}\right) = \frac{\partial L_f\left(D_i^*, \boldsymbol{W}_{r-1}\right)}{\partial \boldsymbol{W}_{r-1}}, \quad (4)$$

where ∇L_f is derivative of the loss function $L_f(\cdot)$, D_i^* is a random subset of D_i. P_i computes the model parameters as follows:

$$\boldsymbol{W}_{i,r} = \boldsymbol{W}_{r-1} - \eta \nabla L_f\left(D_i^*, \boldsymbol{W}_{r-1}\right). \quad (5)$$

(2) Encoding and Encryption

After participant P_i completes the model training, the local model parameters $\boldsymbol{W}_{i,r}$ of the r-th round are obtained. To protect its privacy, P_i will encrypt its model parameters according to **Algorithm 1**.

First, P_i needs to convert each element in the model parameter $\boldsymbol{W}_{i,r}$ from a real number to an integer. Thus, we define the following function for the conversion:

$$\varphi(x) = \begin{cases} \lfloor x \times 10^l \rfloor, & if\ x \times 10^l - \lfloor x \times 10^l \rfloor < 0.5, \\ \lfloor x \times 10^l \rfloor + 1, & otherwise. \end{cases} \quad (6)$$

where l is a positive integer (such as $l = 5$) such that adjusting l's value can control the computing precision of the model parameters. $\lfloor x \times 10^l \rfloor$ is the largest integer less than or equal to $x \times 10^l$. Given this function $\varphi(x)$, we let $\bar{\boldsymbol{W}}_{i,r} = \varphi(\boldsymbol{W}_{i,r})$.

Next, in r-th round of the training, P_i uses the $PRG(\cdot)$ to generate two random sequences $\{M_i^{(j,r)}\}$ and $\{M_j^{(i,r)}\}$ from two seed vectors. Specifically, $M_i^{(j,r)} = PRG(M_i^{(j,r-1)})$ and $M_j^{(i,r)} = PRG(M_j^{(i,r-1)})$. Then P_i uses the two random sequences to blind the encoded parameter $\bar{\boldsymbol{W}}_{i,r}$:

$$\hat{\boldsymbol{W}}_{i,r} = \bar{\boldsymbol{W}}_{i,r} - \sum_{j=1}^{N}\boldsymbol{M}_i^{(j,r)} + \sum_{j=1}^{N}\boldsymbol{M}_j^{(i,r)}, \quad (7)$$

where $\hat{\boldsymbol{W}}_{i,r}$ is the blinded model parameter.

Then, P_i randomly splits the blinded parameter $\hat{\boldsymbol{W}}_{i,r}$ into m parts $\{b_k^{(i,r)} \mid k = 1, 2, \cdots, m\}$ such that satisfies: $\hat{\boldsymbol{W}}_{i,r} = \sum_{k=1}^{m} b_k^{(i,r)}$. To further protect the privacy of model parameters and reduce the spatial overhead [73]. P_i uses the CRT to encrypt and package the $\{b_k^{(i,r)} \mid k = 1, 2, \cdots, m\}$. In specific, given the following system of congruences:

> <

$$\begin{cases} ct_{i,r} \equiv b_1^{(i,r)} \pmod{p_1} \\ ct_{i,r} \equiv b_2^{(i,r)} \pmod{p_2} \\ \cdots \\ ct_{i,r} \equiv b_m^{(i,r)} \pmod{p_m} \end{cases}, \tag{8}$$

according to the CRT, the unique solution of equations (8) in the sense of modulo S can be obtained:

Algorithm 2 Time contract.

Input: The transaction of the participant P_i: $TX_{i,r}$.
Output: Transaction: TX_{check}.
1: Checking the timestamp of the participant's transaction:
function *CheckTime*($TX_{i,r}$){
Query the recorded transaction $TX_{i,r}.H(TX_{TP})$ and $TX_{i,r}.H(TX_{Pi})$ from the blockchain to get the training time per round t and the training start time t_0, then calculate the deadline of the current round:
$t_r = t_0 + TX_{i,r}.r \times t$
if $TX_{i,r}.timestamp > t_r$ then
return *false*
else return *true*
end if
}
$pass = CheckTime(TX_{i,r})$
2: Generate a transaction recording the checking result:
$TX_{check} = \{r, pass, H(TX_{i,r}), H(TX_{TP}),$ "*Keywords*"$\}$

$$ct_{i,r} \equiv \sum_{k=1}^{m} b_k^{(i,r)} S_k T_k \pmod{S}, \tag{9}$$

where $S = \prod_{k=1}^{m} p_k$, $S_k = S/p_k$, $T_k \equiv S_k^{-1} \pmod{p_k}$ is the inverse element of S_k of module p_k.

We can know that: $b_k^{(i,r)} S_k T_k \equiv b_k^{(i,r)} \times 1 \equiv b_k^{(i,r)} \pmod{p_k}$, for $\forall k \in \{1, 2, \cdots, m\}$, $k \neq j$, $b_j^{(i,r)} S_j T_j \equiv 0 \pmod{p_k}$, then $ct_{i,r}$ satisfies:

$$\begin{aligned} ct_{i,r} &\equiv b_k^{(i,r)} S_k T_k + \sum_{j \neq k} b_j^{(i,r)} S_j T_j \pmod{S} \\ &\equiv b_k^{(i,r)} + \sum_{j \neq k} 0 \,(mod\ p_k) \equiv b_k^{(i,r)} \pmod{p_k} \end{aligned}. \tag{10}$$

According to equation (10), $b_k^{(i,r)}$ is mapped from the integer $\mathbb{Z}$ to the finite field GF(p_k):

$$g\left(b_k^{(i,r)}\right) = \begin{cases} b_k^{(i,r)}, & if\ b_k^{(i,r)} \geq 0, \\ b_k^{(i,r)} + p_k, & otherwise. \end{cases} \tag{11}$$

where $b_k^{(i,r)}$ is k-th part of the blinded parameter $\hat{\boldsymbol{W}}_{i,r}$. To avoid overflow error in the calculation process, the prime $\{p_k \mid k = 1, 2, \cdots, m\}$ must be large enough and satisfies $p_k \gg N \times 10^l$ to ensure $b_k^{(i,r)} \in [-(p_k - 1)/(2N), (p_k - 1)/(2N)]$.

Thus, after executing the specified CRT operation, the encryption and decryption operation can be performed in the finite field $GF(p_k)$ ($k = 1, 2, \cdots, m$). To ease description, we write the packaged data $ct_{i,r}$ as CRT[$b_1^{(i,r)}$, $b_2^{(i,r)}$, $\cdots$, $b_m^{(i,r)}$].

(3) Upload Parameter Ciphertext

The participants upload their encrypted parameters to DFS for off-chain storage, and generate transactions to record the hash address of the stored file into blockchain ledger so that others can retrieve the file from DFS via this address. We use $TX_{i,r} = \{r, H(ct_{i,r}), \sigma(sk_i, H(ct_{i,r})), H(TX_{TPj}), H(TX_{Pi}),$ "*Keywords*"$\}$ to denote the transaction that P_i ($1 \leq i \leq N$) sends to the blockchain, where $H(ct_{i,r})$ is the hash value of P_i's encrypted local model parameter. Next, the time contract checks whether N participants finish uploading their encrypted parameter before the deadline t_r. If some participants fail to upload their parameters, the monetary penalty mechanism will be performed to forfeit a part of its deposit and reward it to other participants who perform honestly. Then the time-out participant re-executes the ciphertext uploading. Only if N participants succeed in uploading the ciphertexts of their model parameters, the parameter aggregation phase will be performed.

Algorithm 3 Model aggregation.

Input: Participant P_i: $TX_{i,r}$.
Output: Block: $block_r$.
1: All workers verify the transactions uploaded by the participants;
2: A leader is selected from all workers by the VRF;
3: The leader computes the aggregation value of the ciphertexts for tasks whose local model training phase has ended:
$C_r = \sum_{i=1}^{N} ct_{i,r}$;
4: The leader generates a transaction TX_{Cr}, and packages all transactions to $block_r$:
$TX_{Cr} = \{mid, r, H(C_r), \sigma(sk, H(C_r)), H(TX_{TP}),$ "*Keywords*"$\}$;
$block_r = \{TX_{Cr}, TX_{1,r}, TX_{2,r}, \cdots, TX_{N,r}\}$;
5: The validators verify and vote on $block_r$:
$TX_{vote} = \{H(block_r), \sigma(sk, H(block_r)), H(TX_{TP}),$ "*Keywords*"$\}$;
if more than 2/3 of the validators sign and agree $block_r$:
The $block_r$ will be appended onto the blockchain;
return $block_r$;
Otherwise:
Confiscate the leader's deposit and award it to other committee members; Then, the next priority worker becomes the leader, repeat the process 3, 4, 5.

Time contract: The time contract is designed to constrain participants to upload the ciphertext of the model parameters on time. In general, the time contract judges whether the parameter uploading has timed out based on the timestamp of the transaction $TX_{i,r}$. The execution of the time contract is described in **Algorithm 2**.

D. Model Gradient Aggregation Phase

Once all participants' model upload transactions are received, the workers on the blockchain need to verify the validity of transactions uploaded by participants. Then, leader and validators selected by the consensus mechanism complete the aggregation and verification operations. The model aggregation algorithm is described in **Algorithm 3.**

***Step 1*:** *Workers verify the validity of transaction*. When a participant P_i sends the transaction $TX_{i,r}$ to the blockchain, the workers will check its digital signature to confirm that the uploaded transaction is from a legal participant, and then put the verified transactions into the transaction pool specified by the TP's task.

Step 2: *The elected leader performs the aggregation operation and generates a new block.* After all participants upload their locally trained models, the leader selected by the Algorand consensus algorithm obtains these encrypted model parameters from DFS, and computes the aggregation value of these ciphertexts:

> <

$$C_r = \sum_{i=1}^{N} ct_{i,r}\,, \tag{12}$$

since the CRT satisfies additive homomorphism, the equation holds:

$$\begin{aligned} C_r &= CRT\left[\sum_{i=1}^{N} b_1^{(i,r)}, \sum_{i=1}^{N} b_2^{(i,r)}, \ldots, \sum_{i=1}^{N} b_m^{(i,r)}\right], \\ &= CRT\left[B_1^{(r)}, B_2^{(r)}, \ldots, B_m^{(r)}\right] \end{aligned} \tag{13}$$

where C_r is the aggregated ciphertext for the r-th round, $B_k^{(r)} = \sum_{i=1}^{N} b_k^{(i,r)}$.

Algorithm 4 Decryption and update.

Input: Aggregated ciphertext C_r;
Security parameters: m, l, p_k ($k = 1, 2, \cdots, m$).
Output: Plaintext model parameters $\boldsymbol{W}_r$.
1: Decrypt ciphertext C_r:

$$B_k^{(r)} = C_r \pmod{p_k}\,;$$

2: Use the $g^{-1}(B_k^{(r)})$ to converts $\{B_k^{(r)} \mid k = 1, 2, \cdots, m\}$ from finite field to integers:

$$g^{-1}(B_k^{(r)}) = \begin{cases} B_k^{(r)} - p_k, & p_k / 2 \le B_k^{(r)} < p_k, \\ B_k^{(r)}, & 0 \le B_k^{(r)} < p_k / 2. \end{cases};$$

3: Calculate the sum of $\hat{B}_k^{(r)} = g^{-1}(B_k^{(r)})$, $k = 1, 2, \cdots, m$:

$$\bar{\boldsymbol{W}}_r = \sum_{k=1}^{m} \hat{B}_k^{(r)}\,;$$

4: Decode $\bar{\boldsymbol{W}}_r$:

$$\boldsymbol{W}_r = \frac{\bar{\boldsymbol{W}}_r}{10^l \times N}\,;$$

5: Return $\boldsymbol{W}_r$.

After the aggregation is completed, the leader generates a transaction TX_{Cr} = {*mid*, *r*, $H(C_r)$, $\sigma(sk, H(C_r))$, $H(TX_{TP})$, "*Keywords*"}, where *sk* is the secret key of the leader, $H(C_r)$ is the hash address of the ciphertext of the aggregated model parameter stored in DFS. Next, the leader packages all transactions generated in *r*-th round of the specific model training task into a new block, *i.e.*, $block_r$ = {TX_{Cr}, $TX_{1,r}$, $TX_{2,r}$, $\cdots$, $TX_{N,r}$ }, and broadcasts the block.

Step 3: *The validators validate and vote on the new block.* The validators verify whether the operations performed by the leader are correct or not. Only if more than 2/3 of the validators sign and agree on it, a transaction will be generated TX_{vote} = {$H(block_r)$, $\sigma(sk, H(block_r))$, $H(TX_{TP})$, "*Keywords*"}, where $H(block_r)$ is hash value of the $block_r$, the new block is appended to the blockchain, and the leader gains a reward from the blockchain. Otherwise, the next priority worker becomes the leader, repeating the ***Step 2*** and ***Step 3*** until the leader and validators reach a consensus among them.

E. Local Model Update Phase

Each participant P_i ($1 \le i \le N$) can request the hash value of the aggregated ciphertext from TX_{Cr} in $block_r$, then downloads the aggregated ciphertext C_r from DFS followed by decrypting and decoding it to update its local model. **Algorithm 4** shows the decryption and update algorithm for P_i in the *r*-th round.

First, P_i uses the module operation to decrypt the ciphertext C_r, and obtains the aggregated discrete values for $k = 1, 2, \cdots, m$, *i.e.*, P_i computes:

$$B_k^{(r)} \equiv C_r \pmod{p_k}\,. \tag{14}$$

Next, P_i needs to convert $\{B_k^{(r)} \mid k = 1, 2, \cdots, m\}$ from the finite field to the integer field ($GF(p_k) \to [-(p_k - 1)/2, (p_k - 1)/2]$, $k = 1, 2, \cdots, m$) by using the function $g^{-1}(B_k^{(r)})$, where $g^{-1}(B_k^{(r)})$ is the inverse function of $g(b_k^{(i,r)})$:

$$g^{-1}(B_k^{(r)}) = \begin{cases} B_k^{(r)} - p_k, & p_k / 2 \le B_k^{(r)} < p_k, \\ B_k^{(r)}, & 0 \le B_k^{(r)} < p_k / 2. \end{cases} \tag{15}$$

Let $\hat{B}_k^{(r)} = g^{-1}(B_k^{(r)})$, $k = 1, 2, \cdots, m$. Then, P_i computes the real aggregated model parameter $W_r^{(j)}$ in *r*-th round as follows:

$$\bar{\boldsymbol{W}}_r = \sum_{k=1}^{m} \hat{B}_k^{(r)}\,,\ \boldsymbol{W}_r = \frac{\bar{\boldsymbol{W}}_r}{10^l \times N}\,. \tag{16}$$

Afterwards, each participant P_i ($1 \le i \le N$) replaces its local model parameter $\boldsymbol{W}_{i,r}$ with $\boldsymbol{W}_r$ for the next round of model training.

Algorithm 5 Incentive contract.

Input: Participant P_i: *wa*, D_i;
Participant list: *List*;
Incentive coefficient: *u, v*.
Output: Transaction: $TX_{award[Pi]}$.
1: The FL task is completed, P_i performs the following operations:

work.finished ← *true;*
work.size ← *size*(D_i);
work.dis ← *distance*(D_i);
contri[*mid*][*Pi*] ← *work*;

2: if $P_i.wa \in$ *List* and *contri*[*mid*][P_i].*finished* is *true*:

coins[*mid*][P_i] = *contri*[*mid*][P_i].*size*×*u* + *contri*[*mid*][P_i].*dis*×*v*;

3: Record the reward:

$TX_{award[Pi]}$ = {$TP_j.wa$, *mid*, *coins*[*mid*][P_i], $P_i.wa$, $H(TX_{TP})$, "*Keywords*"}

4: Send $TX_{award[Pi]}$ to the blockchain.

F. Obtain Final Model Parameters

After *r* rounds of model training (*e.g.*, $r \ge 50$), *N* participants use their own datasets to test the model performance. If their model accuracy meets the specified termination conditions, the participant will upload a transaction to the blockchain. Only when all *N* participants have uploaded transactions to the blockchain, the SM downloads the aggregated ciphertext according to the transaction TX_{Cr} of the final round. Then, SM decrypts the ciphertext to obtain the final model parameters, and sends it to the TP via its secure channels.

The TP uses secret key *sk* to decrypt the ciphertext and checks whether the model parameters meet its requirement. If the requirement of model accuracy is not satisfied, these participants will proceed to the next round of model training. When this model training task is completed, the TP must pay the corresponding rewards to the participants according to the incentive mechanism, which is the result of the incentive contract of **Algorithm 5** $TX_{award[Pi]}$. If the TP fails to pay the participants, he/she will be held accountable. The SM confiscates his deposit and pays the corresponding participants' rewards, and then blacklists the TP in the system.

Incentive mechanism. Since the reward is related to the quality of the user's data, we use the methods of [63, 64] to measure the contribution of each participant, as the **Algorithm 5** shows. This incentive mechanism requires participants to

> <

honestly compute the size and distance of the local dataset. We leave the design of incentive mechanisms in the presence of malicious participants as future research work.

In **Algorithm 5**, participant P_i will compute the size and distance of the local dataset after the FL task is completed. Then P_i packages the completion tag *finished*, local dataset *size*, and the *distance* into a *work* structure. Next, it invokes the contract: the *work* will be uploaded to *contri*; then the contract computes rewards *coins*[*mid*][P_i] according to *size* and *dis*; in the end, the contract packages the computation results to a transaction $TX_{award[Pi]}$ and sends it to the blockchain. The blockchain will automatically distribute coin rewards based on recorded $TX_{award[Pi]}$ transactions. This incentive mechanism helps TPs to motivate more participants with high-quality datasets to participate in federated learning tasks, hence improving the model performance.

V. Security Analysis

This section gives the security analysis of BPFL in terms of correctness, data privacy, and verifiability.

A. Correctness

Theorem 2. In our BPFL, if each entity honestly performs the protocol and the consensus mechanism is secure, participants may get the correct aggregated model parameters to update their local models.

Proof: If each entity honestly performs the BPFL protocol, each participant can encrypt its model parameters to generate its ciphertext $ct_{i,r}$ according to **Algorithm 1**. If the consensus mechanism is secure, validators may verify the correctness of the aggregation results, and only the correct aggregated ciphertext is appended to the blockchain by the aggregator.

The aggregator performs the aggregation operation according to the equation (12):

$$\begin{aligned} C_r &= \sum_{i=1}^{N} ct_{i,r} \\ &= \sum_{i=1}^{N}\sum_{k=1}^{m} b_k^{(i,r)} S_k T_k \\ &= \sum_{i=1}^{N} \hat{\boldsymbol{W}}_{i,r} \sum_{k=1}^{m} S_k T_k, \end{aligned} \tag{17}$$

where

$$\begin{aligned} \sum_{i=1}^{N} \hat{\boldsymbol{W}}_{i,r} &= \sum_{i=1}^{N} \left(\bar{\boldsymbol{W}}_{i,r} - \sum_{j=1}^{N} \boldsymbol{M}_i^{(j,r)} + \sum_{j=1}^{N} \boldsymbol{M}_j^{(i,r)} \right) \\ &= \sum_{i=1}^{N} \bar{\boldsymbol{W}}_{i,r} + \sum_{i=1}^{N}\sum_{j=1}^{N} (-\boldsymbol{M}_i^{(j,r)} + \boldsymbol{M}_j^{(i,r)}) \\ &= \sum_{i=1}^{N} \bar{\boldsymbol{W}}_{i,r} = \bar{\boldsymbol{W}}_r. \end{aligned} \tag{18}$$

Hence,

$$\begin{aligned} C_r &= \sum_{i=1}^{N} \hat{\boldsymbol{W}}_{i,r} \sum_{k=1}^{m} S_k T_k \\ &= \bar{\boldsymbol{W}}_r \sum_{k=1}^{m} S_k T_k. \end{aligned} \tag{19}$$

Then the aggregator should record this aggregation result and all the encrypted local model parameters received from participants into a block. A committee of voters will verify this block to ensure the correctness of the aggregation result. Only if C_r is indeed the aggregation of all received $ct_{i,r}$, the block can be accepted and added into the blockchain ledger.

After that each participant obtains the aggregated model parameters C_r from the blockchain and computes $B_k^{(r)}$ (k = 1, 2, ⋯, m) according to equation (14):

$$B_k^{(r)} \equiv \bar{\boldsymbol{W}}_r \sum_{k=1}^{m} S_k T_k \pmod{p_k}, \tag{20}$$

as $S_k T_k \equiv 1 \pmod{p_k}$, and $S_k = S/p_k$,

$$B_k^{(r)} \equiv \bar{\boldsymbol{W}}_r \pmod{p_k}. \tag{21}$$

Next, according to equations (15) and (16), each participant can get the real aggregated model parameters.

B. Privacy Protection

Theorem 3. In our BPFL, if the aggregator without colluding with participants, the aggregator cannot obtain participants' real model parameters and the aggregated model parameters.

Proof: In our BPFL protocol, the local model parameters are protected by pairwise masking and CRT techniques. The aggregator can get a participant P_i's real local model at round r only if it can obtain the security parameters $\{p_1, p_2, \cdots, p_m\}$, $\{\boldsymbol{M}_i^{(j,r)} \mid j = 1, 2, \cdots, N\}$, and$\{\boldsymbol{M}_j^{(i,r)} \mid j = 1, 2, \cdots, N\}$. But the security parameters are sending to participant P_i via a secret channel, they are kept secret from the aggregator. Therefore, the aggregator cannot decrypt P_i's ciphertext nor recover its pairwise masks. Similarly, the aggregator cannot obtain the real aggregated model parameters.

Theorem 4. In our BPFL protocol, even if k ($k \le N - 2$) participants collude, they still cannot get the real model parameters of other participants.

Proof: Considering that in the worst-case scenario, if $N - 2$ participants collude, they get some security parameters, but cannot obtain the two random sequences used by other participants. Hence, they only get according to equation (7):

$$\begin{aligned} \bar{\boldsymbol{W}}_{N-1,r} &= \hat{\boldsymbol{W}}_{N-1,r} + \sum_{j=1}^{N} \boldsymbol{M}_{N-1}^{(j,r)} - \sum_{j=1}^{N} \boldsymbol{M}_j^{(N-1,r)} \\ &= \tilde{\boldsymbol{W}}_{N-1,r} + \boldsymbol{M}_{N-1}^{(N-1,r)} + \boldsymbol{M}_{N-1}^{(N,r)} - \boldsymbol{M}_{N-1}^{(N-1,r)} - \boldsymbol{M}_N^{(N-1,r)} \\ &= \tilde{\boldsymbol{W}}_{N-1,r} + \boldsymbol{M}_{N-1}^{(N,r)} - \boldsymbol{M}_N^{(N-1,r)}, \end{aligned} \tag{22}$$

$$\begin{aligned} \bar{\boldsymbol{W}}_{N,r} &= \hat{\boldsymbol{W}}_{N,r} + \sum_{j=1}^{N} \boldsymbol{M}_N^{(j,r)} - \sum_{j=1}^{N} \boldsymbol{M}_j^{(N,r)} \\ &= \tilde{\boldsymbol{W}}_{N,r} + \boldsymbol{M}_N^{(N-1,r)} + \boldsymbol{M}_N^{(N,r)} - \boldsymbol{M}_{N-1}^{(N,r)} - \boldsymbol{M}_N^{(N,r)} \\ &= \tilde{\boldsymbol{W}}_{N,r} + \boldsymbol{M}_N^{(N-1,r)} - \boldsymbol{M}_{N-1}^{(N,r)}. \end{aligned} \tag{23}$$

According to equations (22) and (23), they cannot obtain the two pivotal parameters $M_N^{(N-1,r)}$, $M_{N-1}^{(N,r)}$ of other participants. In addition, when k ($k < N - 2$) participants collude, according to equations (22) and (23), they cannot obtain the $N - k$ pivotal parameters of other participants.

C. Verifiability

Theorem 5. In our BPFL protocol, validators can detect the forged aggregated results and reward misdistribution.

Proof: In BPFL, participants are required to upload their encrypted model parameters to the DFS, and the addresses of their encrypted model parameters can be recorded into the blockchain. Then the aggregator selected by the consensus protocol will aggregate these ciphertexts by using all the encrypted local model parameters, and upload them onto the DFS, and generate a block recording the addresses of the aggregation results. When the task is finished, the transaction of reward allocation is also recorded into a block. The blocks can be admitted after more than 2/3 of the validators have approved it. If the aggregated model parameters of the

> <

aggregator are found to be forged, or the reward misdistribution is detected, the consensus protocol will confiscate the tokens pledged by the aggregator and re-select a new aggregator. This security of the consensus protocol guarantees that the forged results can be detected. Besides, all participants can get blocks and verify the correctness of the aggregation results and reward allocation. In a word, properties of the blockchain guarantees the verifiability of BPFL.

VI. Performance Evaluation

This section evaluates our BPFL protocol's performance in model accuracy, computational time, and communication overhead, and blockchain evaluation, respectively.

Our experiments are performed on a Linux workstation with 2 Intel Xeon Silver 4110 CPUs (2.10GHz) and 128 GB of RAM, running CentOS 7 operating system. We implement the BPFL scheme in Python and utilize the Pytorch library [74] for developing and training models. In addition, the performance is evaluated on the Fashion-MNIST dataset [75] containing 60000 training samples and 10000 testing samples, from which we randomly select the same number of samples to construct each participant's local dataset. We chose a convolutional neural network (CNN) as the training model, the architecture is 28×28 (input) - 3×3×16 (conv,1 stride) - 2×2 (max-pooling) - 3×3×32 (conv,1 stride) - 2×2 (max-pooling) - (flatten) – 10 (output). Note that both the global and the local models trained by participants have the same model architecture, the total number of parameters is 20490 in this model.

A. Model Accuracy

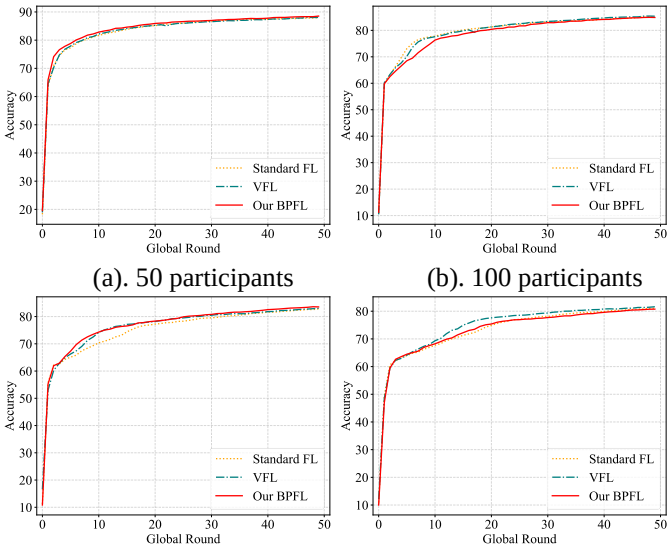
In this experiment, we initiate 50, 100, 150 and 200 participants respectively, each with a local dataset. In BPFL, we set the complexity parameter m = 4 and the computing precision l = 5. Each element of the prime list $\{p_k \mid k = 0, 1, 2, 3, 4\}$ satisfies $p_k \in (200\times N\times 10^l, 400\times N\times 10^l)$. The CRT parameters S, S_k, T_k (k = 1, 2, ⋯, m) can be computed according to the prime list $\{p_k \mid k = 1, 2, 3, 4\}$. Besides, the seed matrix $\boldsymbol{M}$ is set to 50×50, each element in this matrix is produced randomly in (1, 1000), and then passed to a *PRG*: *rand* = (10 × *seed*) mod p_0. Lastly, the number of learning rate η is 0.001. The local batch size we set is 32.

Fig. 2 shows the model accuracy in 50 global rounds using the BPFL scheme with different numbers of participants, a standard FL scheme [19] and VFL scheme [42]. In the VFL scheme, we define the prime $p_k \in (4\times 10^{13}, 8\times 10^{13})$, and the random number sequence $a_i \in (1, 50)$ and $b_i \in (51, 100)$ for all $i \in \{1, 2, 3, 4\}$ to construct the constant sequences $\{a_i \mid i = 1, 2, 3, 4\}$ and $\{b_i \mid i = 1, 2, 3, 4\}$, respectively. Due to keeping consistency, other parameters in the VFL scheme remain the same as those set in the BPFL scheme. From **Fig. 2**, we can observe the BPFL scheme, standard FL scheme with different numbers of participants, their model accuracy is almost the same at the convergence. This experimental result shows that model accuracy of our BPFL protocol is the same as other schemes. In conclusion, our scheme has practicality and security advantages.

We also investigate the effect of the computing precision l on the model accuracy in the BPFL scheme. From **Fig. 3**, we can see that the model performance will improve with the increased computing precision l. However, when the value of l is too large, it will increase computational time cost (see next section). Therefore, it is very necessary to choose a smaller computation precision l when meeting the model accuracy requirements.

B. Computational Time

In this section, we evaluate the computational time of the BPFL scheme compared with the VFL scheme [42] and the Fedv scheme [76] in four phases: security parameter generation, encryption, aggregation, and decryption. Suppose the number of participants is N =10, then according to $p_k \gg N\times 10^l$ and equation (9), the computing precision l and the complexity parameter m affect the ciphertext length. Therefore, we set the complexity parameter $m \in \{3, 4, 5, 6,$



(a). 50 participants (b). 100 participants

(c). 150 participants (d). 200 participants

Fig. 2. The evaluation of model accuracy for different schemes.

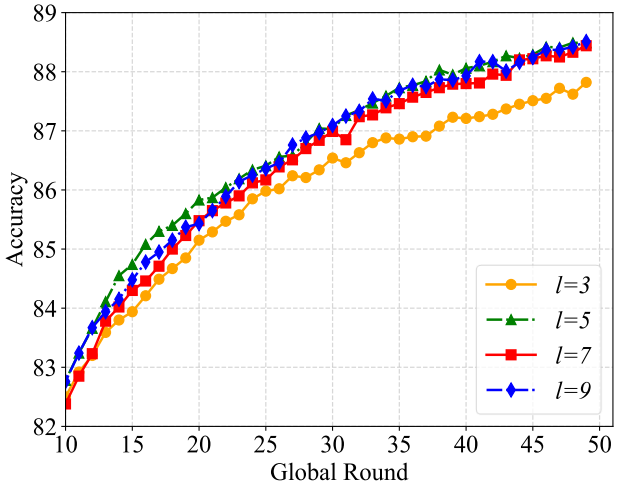


Fig. 3. The evaluation of model accuracy via different computing precision value l in our BPFL.

> <

7}, the computing precision $l \in \{3, 5, 8\}$. In the Fedv scheme, we set the key length $K \in \{1024, 1536, 2048\}$ in the Paillier homomorphic encryption algorithm.

(1) Security parameter generation time

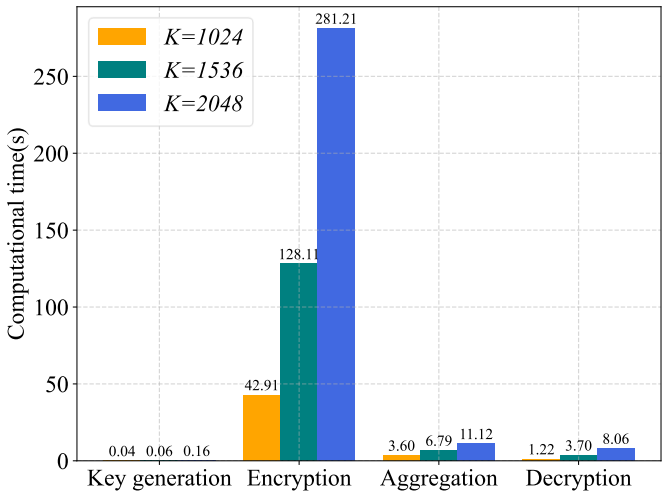
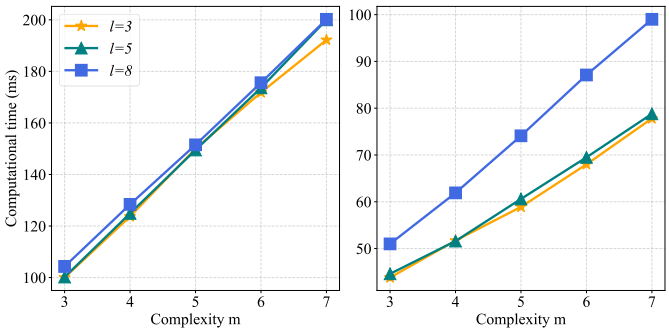


Fig. 4. The computational time of Fedv scheme under different key lengths.



(a). Encryption (b). Decryption

Fig. 5. The encryption and decryption time of a participant under different security parameters in our BPFL.

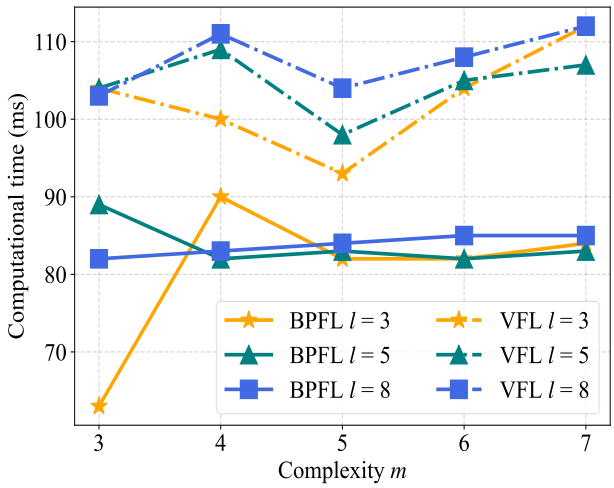


Fig. 6. The aggregation time of the aggregator under different security parameters.

We use different system parameters to generate the security parameters for 10 participants, and record their generation time. **Fig. 4** shows the computational time cost of the Fedv scheme increases linearly with the increase of the key value K. When K = 2048, the computational time for parameter generation is 161ms. Similarly, as shown in Table II, the security parameter generation time in the VFL scheme increases with the increase of the values of m and l. However, the generation time increases slowly in BPFL. When m = 4, l = 8, it costs 13ms to generate parameters in BPFL, while this number is 417ms in the VFL scheme. To prevent overflow errors, the VFL scheme must generate large primes for Lagrange interpolation. In conclusion, in the parameter generation phase, the BPFL scheme outperforms other related schemes, especially in achieving a more secure FL (increasing

TABLE II
THE SECURITY PARAMETER GENERATION TIME

(m, l)	VFL [42] (ms)	**Our BPFL** (ms)
(3, 3)	17	**11**
(4, 3)	14	**10**
(5, 3)	16	**10**
(6, 3)	16	**9**
(7, 3)	14	**9**
(3, 5)	25	**11**
(4, 5)	32	**11**
(5, 5)	32	**11**
(6, 5)	37	**11**
(7, 5)	37	**11**
(3, 8)	294	**12**
(4, 8)	417	**13**
(5, 8)	518	**13**
(6, 8)	617	**14**
(7, 8)	712	**15**

TABLE III
THE ENCRYPTION TIME OF A PARTICIPANT

(m, l)	VFL [42] (ms)	**Our BPFL** (ms)
(3, 3)	7672	**100**
(4, 3)	12732	**123**
(5, 3)	21048	**150**
(6, 3)	28004	**171**
(7, 3)	37659	**192**
(3, 5)	7515	**100**
(4, 5)	13808	**124**
(5, 5)	19587	**149**
(6, 5)	28053	**173**
(7, 5)	37872	**200**
(3, 8)	7455	**104**
(4, 8)	13078	**128**
(5, 8)	19759	**151**
(6, 8)	27911	**175**
(7, 8)	37959	**200**

TABLE IV
THE DECRYPTION TIME OF A PARTICIPANT

(m, l)	VFL [42] (ms)	**Our BPFL** (ms)
(3, 3)	7316	**43**
(4, 3)	12537	**51**
(5, 3)	19452	**58**
(6, 3)	27724	**68**
(7, 3)	37217	**77**
(3, 5)	7191	**44**
(4, 5)	13556	**51**
(5, 5)	19320	**60**
(6, 5)	27613	**69**
(7, 5)	37083	**78**
(3, 8)	7201	**51**
(4, 8)	12577	**61**
(5, 8)	19264	**74**
(6, 8)	27535	**87**
(7, 8)	37263	**99**

> <

m, *l*).

(2) Encryption time cost

We compute the average encryption time cost of 10 participants under different system parameters. As shown in **Fig. 4**, the encryption time increases with the increase of the key size in the Fedv scheme, in which the minimum encryption time is 42.910s. Thus, it is unsuitable for many resource-constrained mobile devices. As shown in **Fig. 5**(a), the encryption time increases linearly with the increase of the complexity parameter value *m*, under the given value of computing precision *l*. From Table III, when $m = 7$, $l = 8$, our BPFL protocol takes only 200ms to perform the encryption algorithm. However, the VFL scheme takes 37959ms to complete the encryption algorithm, because the building of Lagrange interpolation function is time-consuming. Under our proposed application area and parameters, the encryption time required by our protocol is at least 70times faster than VFL. Therefore, the BPFL encryption efficiency is approx. 98% higher than that of VFL scheme.

(3) Aggregation time cost

The aggregation time is the total run time that the aggregation server sums the encrypted model gradient values of all participants, *i.e.*, the computation time of running the equation (12). We test the aggregation time by aggregating 10 participants' local updates generated from different schemes. As shown in **Fig. 4**, the minimum aggregation time is 3.6s in the Fedv scheme. However, from **Fig. 6**, the aggregation time is about 85ms in the BPFL scheme, which is approx. 10% faster than VFL scheme.

(4) Decryption time cost

As shown in **Fig. 5**(b), the decryption time increases linearly with the increase of the complexity parameter *m*. From **Fig. 4** and Table IV, we find that BPFL is more efficient than VFL and Fedv schemes in the decryption. When $m = 7$, $l = 8$, our BPFL protocol takes only 99ms to perform the decryption algorithm, while the VFL scheme takes 37263ms to complete the decryption algorithm. Especially, the decryption time of the VFL scheme is more than 140 times that of our BPFL under different security parameters. Therefore, the BPFL decryption efficiency is approx. 99% higher than that of VFL scheme.

In summary, compared with encryption methods in VFL and Fedv schemes, the CRT used in BPFL scheme has the lowest computational overhead and is more suitable for mobile crowdsourcing.

C. Communication Overhead

As we know, the communication overhead for a participator is related to the size of the ciphertext. Thus, we investigate the communication overhead of our BPFL ($m = 4$, $l = 5$) compared with VFL ($m = 4$, $l_\omega = 10^5$) and Fedv ($K = 1536$) in terms of the file size of the encrypted parameters. In this experiment, the parameters of each scheme are stored and transmitted in JSON format, the experimental results are shown in Table V. We can observe that the size of a file that stores parameters in plaintext (*i.e.*, standard FL [19]) are 442 KB. In contrast, after encryption, the file size of the model parameter ciphertext reaches 675 KB, 1040KB, and 18545KB in BPFL, VFL scheme and Fedv scheme, respectively. Besides, as the size of the model parameters increases, the size of the ciphertext also increases correspondingly. However, the data expanding rate of the PBFL is only 1.53. Therefore, our BPFL protocol has the lowest data expanding rate than other two schemes [42, 76], greatly reduces the communication overhead.

TABLE V
THE SIZE OF FILE STORING MODEL PARAMETERS

Scheme	File size (KB)	Data expanding rate
Standard FL [19]	442	1
VFL [42]	1040	2.35
Fedv [76]	18545	41.96
Our BPFL	**675**	**1.53**

TABLE VI
THE TOTAL COMPUTATIONAL TIME IN ONE FL ROUND

Time	VFL [42]	Fedv [76]	**Our BPFL**
Parameter generation (ms)	32	62	**11**
Encryption (ms)	13808.7	128110	**124.9**
Aggregation (ms)	109	6792	**82**
Decryption (ms)	13556.6	3704.5	**51.6**
Tx generation (ms)	/	/	**26**
Consensus time (ms)	/	/	**4440**
Total time (s)	27.5063	138.6685	**4.7355**

TABLE VII
THE COMPUTATIONAL TIME FOR DIFFERENT NUMBER OF PARTICIPANTS IN ONE FL ROUND

Number of participants	**50**	**100**	**150**	**200**
Parameter generation (ms)	11	12	14	15
Encryption (ms)	137	136.7	137.4	135.7
Aggregation (ms)	250	485	735	949
Decryption (ms)	59	57.2	57.1	56.8
Tx generation (ms)	26	26	26	26
Consensus time (ms)	4440	4440	4440	4440
Total time (s)	4.923	5.1569	5.4095	5.6225

TABLE VIII
THE COIN REWARDS OF 10 PARTICIPANTS

ID	EMD	Coin Rewards	ID	EMD	Coin Rewards
1	1057.30	1040	6	1300.03	1210
2	1125.53	1087	7	1038.87	1027
3	1822.52	1575	8	1137.63	1096
4	1602.45	1421	9	1089.35	1062
5	1688.10	1481	10	1270.97	1189

D. The Evaluation of Blockchain

In this experiment, we build the Algorand blockchain network using Algorand sandbox, and utilize Algorand SDK to generate transactions and smart contracts and distribute the incentive coins (https://developer.algorand.org/docs/). We use 10 participants as an example in this experiment. In an Algorand transaction, the *note* field can store data up to 1000 Bytes. The TP will use this field to store the data (description information of the FL task) in BPFL. For participants, the *note* field is used to store the DFS address of the file-storing model parameter ciphertext.

(1) The total running time of each round

Each participant will generate one transaction in an FL round which contains the DFS address of the model parameter ciphertext. We repeatedly generate ten transactions on one Algorand node and record the total generation time as 260ms. Thus, the average generation time of one transaction is 26ms. Additionally, the Algorand blockchain takes an average of 4.44s to achieve a consensus on a new block. We use the tool *Postman* to get a transaction size of 808Bytes in JSON format

and a block size of 898Bytes without a transaction.

Table VI lists the total computational time of our BPFL (m = 4, l = 5) in each round, compared with that of the Fedv scheme (K = 1536) and VFL scheme (m = 4, $l_\omega = 10^5$), our BPFL execution efficiency is at least 82% higher than that of the other two schemes. Hence, it could be more practical to adopt BPFL in real-life applications.

In addition, we also evaluate the computational time of our BPFL with a different number of participants. The experimental result is shown in Table VII. Because calculating the sum of two random integer sequences takes little time, the participant's encryption and decryption time remains almost unchanged. Furthermore, as the number of uploaded ciphertext increases during aggregation, the aggregation time increases linearly with the number of participants. Therefore, the computational time of each round increases linearly as the number of participants increases.

(2) The evaluation of the incentive mechanism

We simulate the incentive mechanism and record the incentives received by participants in the Blockchain. We use the average EMD to measure its quality for multi-class image datasets, the size of EMD is directly proportional to the dataset's quality [63]. Specifically, we obtained the participant's EMD using the OpenCV library [77] as follows: (1) randomly select one image in each category as the baseline; (2) calculate the EMD between the baseline and other images with the same label using the histogram [65]; (3) get the total EMD of each category; (4) average all EMD values to get the final result.

The parameters u,v in the incentive mechanism are set to 0.3 and 0.7, respectively. In this experiment, each participant will train the local model on its local dataset of size 1000 and get a reward based on their contribution, which can be calculated by **Algorithm 5**. As shown in Table VIII, the participants train model on datasets with less similarity (the large EMD), the more rewards they will receive. The incentive mechanism may attract more participants with high-quality datasets to participate in FL, to improve model performance.

VII. Conclusion

We propose a novel BPFL scheme to tackle the privacy and trust issues for mobile crowdsourcing. This BPFL establishes a novel trust mechanism between task publishers and participants via the consensus-driven blockchain framework. Additionally, the blockchain is also used to implement cryptocurrencies and payment services, serving as incentive mechanism for mobile participants in the MCS systems. We prove that our BPFL protocol supports privacy protection and trust for MCS while remaining resilient against inference and collusion attacks. Experimental results show that our BPFL can achieve high performance in terms of computation cost, communication cost and model accuracy, which is friendly for mobile users with resource-constrained devices in MCS ecosystem. However, our BPFL protocol does not consider poisoning attacks from malicious participants, where fake local model parameters can affect aggregation results. Therefore, our future work is to address this issue.

Acknowledgments

We are grateful to the anonymous reviewers for their invaluable comments. This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative and Strategic Capability Research Centres Funding Initiative, the Nantong Natural Science Foundation (No. JC2021128, No. JC22022036), the National Natural Science Foundation of China (No. 61762044, No. 62072259). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

References

[1] K. Yang, K. Zhang, J. Ren, and X. Shen, "Security and privacy in mobile crowdsourcing networks: challenges and opportunities," *IEEE communications magazine*, vol. 53, no. 8, pp. 75-81, Aug. 2015.

[2] W. Feng, Z. Yan, H. Zhang, K. Zeng, Y. Xiao, and Y. T. Hou, "A survey on security, privacy, and trust in mobile crowdsourcing," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2971-2992, Aug. 2018.

[3] Y. Agarwal and M. Hall, "ProtectMyPrivacy: detecting and mitigating privacy leaks on iOS devices using crowdsourcing," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, Taipei, Taiwan, 2013, pp. 97–110.

[4] C. Wang, H. Liu, K.-L. Wright, B. Krishnamachari, and M. Annavaram, "A privacy mechanism for mobile-based urban traffic monitoring," *Pervasive and Mobile Computing*, vol. 20, pp. 1-12, 2015.

[5] L. Tan *et al.*, "Mining myself in the community: privacy preserved crowd sensing and computing," in *Wireless Algorithms, Systems, and Applications: 11th International Conference, WASA 2016, Bozeman, MT, USA, August 8-10, 2016. Proceedings 11*, 2016: Springer, pp. 272-282.

[6] Y. Guo, H. Xie, Y. Miao, C. Wang, and X. Jia, "Fedcrowd: A federated and privacy-preserving crowdsourcing platform on blockchain," *IEEE Transactions on Services Computing*, vol. 15, no. 4, pp. 2060-2073, 2020.

[7] Y. Gong, Y. Guo, and Y. Fang, "A privacy-preserving task recommendation framework for mobile crowdsourcing," in *2014 IEEE Global Communications Conference*, 2014: Austin, TX, USA, pp. 588-593.

[8] T. Dimitriou and I. Krontiris, "Privacy-respecting auctions as incentive mechanisms in mobile crowd sensing," in *Information Security Theory and Practice: 9th IFIP WG 11.2 International Conference, WISTP 2015, Heraklion, Crete, Greece, August 24-25, 2015. Proceedings 9*, 2015: Springer, pp. 20-35.

[9] J. Ren, Y. Zhang, K. Zhang, and X. Shen, "Exploiting mobile crowdsourcing for pervasive cloud services: challenges and solutions," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 98-105, 2015.

[10] H. Jin, L. Su, B. Ding, K. Nahrstedt, and N. Borisov, "Enabling privacy-preserving incentives for mobile crowd sensing systems," in *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, Nara, Japan, 2016, pp. 344-353.

[11] C. Tanas, S. Delgado-Segura, and J. Herrera-Joancomartí, "An integrated reward and reputation mechanism for MCS preserving users' privacy," in *Data Privacy Management, and Security Assurance: 10th International Workshop, DPM 2015, and 4th International Workshop, QASA 2015, Vienna, Austria, September 21–22, 2015. Revised Selected Papers 10*, 2016: Springer, pp. 83-99.

[12] Y. Wang, Z. Cai, G. Yin, Y. Gao, X. Tong, and G. Wu, "An incentive mechanism with privacy protection in mobile crowdsourcing systems," *Computer Networks*, vol. 102, pp. 157-171, 2016.

[13] Z. Xie, L. Hu, F. Wang, J. Li, and K. Zhao, "PEMM: A privacy-aware data aggregation solution for mobile sensing networks," in *Computational Intelligence and Intelligent Systems: 7th International Symposium, ISICA 2015, Guangzhou, China, November 21-22, 2015, Revised Selected Papers 7*, 2016: Springer, pp. 474-482.

[14] G. Zhuo, Q. Jia, L. Guo, M. Li, and P. Li, "Privacy-preserving verifiable data aggregation and analysis for cloud-assisted mobile crowdsourcing," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, San Francisco, CA, USA, 2016, pp. 1-9.

[15] J. Sun and H. Ma, "Privacy-preserving verifiable incentive mechanism for online crowdsourcing markets," in *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, Shanghai, China, 2014, pp. 1-8.

[16] Y. Wu, Y. Wu, H. Peng, H. Chen, and C. Li, "MagiCrowd: A crowd based incentive for location-aware crowd sensing," in *2016 IEEE Wireless Communications and Networking Conference*, Doha, Qatar, 2016, pp. 1-6.

[17] L. Pournajaf, L. Xiong, V. Sunderam, and S. Goryczka, "Spatial task assignment for crowd sensing with cloaked locations," in *2014 IEEE 15th International Conference on Mobile Data Management*, Brisbane, QLD, Australia, 2014, vol. 1, pp. 73-82.

[18] H. To, G. Ghinita, and C. Shahabi, "A framework for protecting worker location privacy in spatial crowdsourcing," *Proceedings of the VLDB Endowment*, vol. 7, no. 10, pp. 919-930, 2014.

[19] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, 2017: PMLR, pp. 1273-1282.

[20] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems*, vol. 32, 2019.

[21] B. Zhao, K. R. Mopuri, and H. Bilen, "idlg: Improved deep leakage from gradients," *arXiv preprint arXiv:2001.02610*, 2020.

[22] H. Yang, M. Ge, D. Xue, K. Xiang, H. Li, and R. Lu, "Gradient Leakage Attacks in Federated Learning: Research Frontiers, Taxonomy and Future Directions," *IEEE Network*, pp. 1-8, 2023.

[23] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," *arXiv preprint arXiv:1710.06963*, 2017.

[24] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, Vienna, Austria, 2016, pp. 308-318.

[25] Y. Zhao *et al.*, "Privacy-preserving blockchain-based federated learning for IoT devices," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1817-1829, 2020.

[26] B. Zhao, K. Fan, K. Yang, Z. Wang, H. Li, and Y. Yang, "Anonymous and privacy-preserving federated learning with industrial big data," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 9, pp. 6314-6323, 2021.

[27] L. Zhu, X. Liu, Y. Li, X. Yang, S.-T. Xia, and R. Lu, "A fine-grained differentially private federated learning against leakage from gradients," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11500-11512, 2021.

[28] D. Gao, Y. Liu, A. Huang, C. Ju, H. Yu, and Q. Yang, "Privacy-preserving heterogeneous federated transfer learning," in *2019 IEEE international conference on big data (Big Data)*, Los Angeles, CA, USA, 2019, pp. 2552-2559.

[29] J. Zhang, B. Chen, S. Yu, and H. Deng, "PEFL: A privacy-enhanced federated learning scheme for big data analytics," in *2019 IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA, 2019, pp. 1-6.

[30] F. Wang, H. Zhu, R. Lu, Y. Zheng, and H. Li, "A privacy-preserving and non-interactive federated learning scheme for regression training with gradient descent," *Information Sciences*, vol. 552, pp. 183-200, 2021.

[31] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "DeepFed: Federated deep learning for intrusion detection in industrial cyber–physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5615-5624, 2020.

[32] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, "Privacy-enhanced federated learning against poisoning adversaries," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4574-4588, 2021.

[33] Q. Kong *et al.*, "Privacy-preserving aggregation for federated learning-based navigation in vehicular fog," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 12, pp. 8453-8463, 2021.

[34] J. Zhao, H. Zhu, F. Wang, R. Lu, Z. Liu, and H. Li, "PVD-FL: A Privacy-Preserving and Verifiable Decentralized Federated Learning Framework," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2059-2073, 2022.

[35] Y. Li, H. Li, G. Xu, X. Huang, and R. Lu, "Efficient Privacy-Preserving Federated Learning With Unreliable Users," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11590-11603, 2021.

[36] Y. Li, H. Li, G. Xu, T. Xiang, X. Huang, and R. Lu, "Toward secure and privacy-preserving distributed deep learning in fog-cloud computing," *IEEE Internet of Things Journal*, vol. 7, no. 12, pp. 11460-11472, 2020.

[37] L. He, S. P. Karimireddy, and M. Jaggi, "Secure byzantine-robust machine learning," *arXiv preprint arXiv:2006.04747*, 2020.

[38] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A secure federated transfer learning framework," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 70-82, 2020.

[39] J. Shu *et al.*, "Clustered Federated Multi-Task Learning on Non-IID Data with Enhanced Privacy," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3453 - 3467, 2022.

[40] Y. Li, H. Li, G. Xu, T. Xiang, and R. Lu, "Practical Privacy-Preserving Federated Learning in Vehicular Fog Computing," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4692-4705, 2022.

[41] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, Dallas, Texas, USA, 2017, pp. 1175-1191.

[42] A. Fu, X. Zhang, N. Xiong, Y. Gao, H. Wang, and J. Zhang, "VFL: A verifiable federated learning with privacy-preserving for big data in industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3316-3326, 2020.

[43] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911-926, 2019.

[44] A. Fu, Z. Chen, Y. Mu, W. Susilo, Y. Sun, and J. Wu, "Cloud-based outsourcing for enabling privacy-preserving large-scale non-negative matrix factorization," *IEEE Transactions on Services Computing*, vol. 15, no. 1, pp. 266-278, 2019.

[45] P. H. Le, S. Ranellucci, and S. D. Gordon, "Two-party private set intersection with an untrusted third party," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, London, United Kingdom, 2019, pp. 2403-2420.

[46] 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 *IEEE INFOCOM 2019-IEEE conference on computer communications*, Paris, France, 2019, pp. 2512-2520.

[47] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2438-2455, 2019.

[48] Z. Liu, J. Guo, W. Yang, J. Fan, K.-Y. Lam, and J. Zhao, "Privacy-preserving aggregation in federated learning: A survey," *IEEE Transactions on Big Data*, pp. 1-20, 2022.

[49] X. Yang, Z. Liu, X. Tang, R. Lu, and B. Liu, "An Efficient and Multi-private Key Secure Aggregation for Federated Learning," *arXiv preprint arXiv:2306.08970*, 2023.

[50] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly) logarithmic overhead," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, Virtual Event, USA, 2020, pp. 1253-1269.

[51] J. So, B. Güler, and A. S. Avestimehr, "Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 479-489, 2021.

[52] K. Mandal, G. Gong, and C. Liu, "Nike-based fast privacy-preserving highdimensional data aggregation for mobile devices," *IEEE T Depend Secure*, pp. 142-149, 2018.

[53] M. N. M. Bhutta *et al.*, "A survey on blockchain technology: Evolution, architecture and security," *Ieee Access*, vol. 9, pp. 61048-61073, 2021.

[54] P. Ramanan and K. Nakayama, "Baffle: Blockchain based aggregator free federated learning," in *2020 IEEE international conference on blockchain (Blockchain)*, Rhodes, Greece, 2020, pp. 72-81.

[55] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1279-1283, 2019.

[56] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1-210, 2021.

[57] A. Kondracki, "The chinese remainder theorem," *Formalized Mathematics*, vol. 6, no. 4, pp. 573-577, 1997.

[58] D. Pei, A. Salomaa, and C. Ding, *Chinese remainder theorem: applications in computing, coding, cryptography*. World Scientific, 1996.

[59] Z. Haddad, M. M. Fouda, M. Mahmoud, and M. Abdallah, "Blockchain-based authentication for 5G networks," in *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, Doha, Qatar, 2020, pp. 189-194.

[60] Z. Haddad, M. Baza, M. M. Mahmoud, W. Alasmary, and F. Alsolami, "Secure and efficient AKA scheme and uniform handover protocol for

5G network using blockchain," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 2616-2627, 2021.
[61] S. Wang, X. Wang, and Y. Zhang, "A secure cloud storage framework with access control based on blockchain," *IEEE access*, vol. 7, pp. 112713-112725, 2019.
[62] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th symposium on operating systems principles*, Shanghai, China, 2017, pp. 51-68.
[63] W. Zhang *et al.*, "Blockchain-based federated learning for device failure detection in industrial IoT," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5926-5937, 2020.
[64] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10700-10714, 2019.
[65] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, p. 99, 2000.
[66] T. Nishide and K. Sakurai, "Distributed paillier cryptosystem without trusted dealer," in *Information Security Applications: 11th International Workshop, WISA 2010, Jeju Island, Korea, August 24-26, 2010, Revised Selected Papers 11*, 2011: Springer, pp. 44-60.
[67] T. Chen and S. Zhong, "Privacy-preserving backpropagation neural network learning," *IEEE Transactions on Neural Networks*, vol. 20, no. 10, pp. 1554-1564, 2009.
[68] Q. Zhang, L. T. Yang, and Z. Chen, "Privacy preserving deep computation model on cloud for big data feature learning," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1351-1362, 2015.
[69] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*, Dallas, Texas, USA, 2017, pp. 587-601.
[70] A. B. Kurtulmus and K. Daniel, "Trustless machine learning contracts; evaluating and exchanging machine learning models on the ethereum blockchain," *arXiv preprint arXiv:1802.10185*, 2018.
[71] L. Chen, P. Koutris, and A. Kumar, "Towards model-based pricing for machine learning in a data marketplace," in *Proceedings of the 2019 International Conference on Management of Data*, Amsterdam, Netherlands, 2019, pp. 1535-1552.
[72] Y. Zhou, X. Wang, M. Zhang, J. Zhu, R. Zheng, and Q. Wu, "MPCE: a maximum probability based cross entropy loss function for neural network classification," *IEEE Access*, vol. 7, pp. 146331-146341, 2019.
[73] W.-j. Lu and J. Sakuma, "More practical privacy-preserving machine learning as a service via efficient secure matrix multiplication," in *Proceedings of the 6th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, Toronto, Canada, 2018, pp. 25-36.
[74] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.
[75] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
[76] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, J. Joshi, and H. Ludwig, "Fedv: Privacy-preserving federated learning over vertically partitioned data," in *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, Virtual Event, Republic of Korea, 2021, pp. 181-192.
[77] G. Bradski, "The openCV library," *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120-123, 2000.

Haiying Ma received the B. S. degree from Henan Normal University, China, in 2001, the M. S. degree from Liaoning University, China, in 2004, and the Ph. D. degree from Tongji University, China, in 2015. Currently, she is an associate professor in Nantong University. Her research interests include information security, public key cryptography, blockchain, big data privacy protection, Internet of Things security, *etc*.

Shuanglong Huang received the B.Eng. degree in software engineering from Huaihai Institute of Technology, China, in 2019. He is currently pursuing the master's degree in computer technology in Nantong University, China. His research interests include Federated Learning, Blockchain and IoT security.

Jiale Guo received the B.Sc. from Shandong University, China, in 2017, and the Ph.D. degree in computer science from Nanyang Technological University, Singapore, in 2022. She is currently a Research Fellow with the Strategic Centre for Research in Privacy-Preserving Technologies and Systems (SCRiPTS), Nanyang Technological University. Her research interests include privacy-preserving machine learning and cyber security.

Kwok-Yan Lam is a Professor at the Nanyang Technological University (NTU), Singapore. He has been a Professor of the Tsinghua University, PR China (2002-2010) and a faculty member of the National University of Singapore and the University of London since 1990. He was a visiting scientist at the Isaac Newton Institute of the Cambridge University and a visiting professor at the European Institute for Systems Security. In 2012, he co-founded Soda Pte Ltd which won the Most Innovative Start Up Award at the RSA 2015 Conference. In 1998, he received the Singapore Foundation Award from the Japanese Chamber of Commerce and Industry in recognition of his R&D achievement in Information Security in Singapore. Prof Lam received his B.Sc. (First Class Honours) from the University of London in 1987 and his Ph.D. from the University of Cambridge in 1990. His research interests include Distributed Systems, IoT Security Infrastructure, Distributed Authentication, Biometric Cryptography, Homeland Security and Cybersecurity.

Tianling Yang received her Bachelor's degree in Computer Science and Technology from Nanjing University of Science and Technology ZiJin College in 2021. She is now studying for a master's degree in computer technology at Nantong University in China. She is currently studying information security with a focus on Federated Learning.