Proof of Federated Learning: A Novel Energy-Recycling Consensus Algorithm

Xidi Qu, Shengling Wang, *Senior Member, IEEE*, Qin Hu, and Xiuzhen Cheng, *Fellow, IEEE*

Abstract—Proof of work (PoW), the most popular consensus mechanism for blockchain, requires ridiculously large amounts of energy but without any useful outcome beyond determining accounting rights among miners. To tackle the drawback of PoW, we propose a novel energy-recycling consensus algorithm, namely proof of federated learning (PoFL), where the energy originally wasted to solve difficult but meaningless puzzles in PoW is reinvested to federated learning. Federated learning and pooled-mining, a trend of PoW, have a natural fit in terms of organization structure. However, the separation between the data usufruct and ownership in blockchain lead to data privacy leakage in model training and verification, deviating from the original intention of federal learning. To address the challenge, a reverse game-based data trading mechanism and a privacy-preserving model verification mechanism are proposed. The former can guard against training data leakage while the latter verifies the accuracy of a trained model with privacy preservation of the task requester's test data as well as the pool's submitted model. To the best of our knowledge, our article is the first work to employ federal learning as the proof of work for blockchain. Extensive simulations based on synthetic and real-world data demonstrate the effectiveness and efficiency of our proposed mechanisms.

Index Terms—Blockchain, federated learning, consensus algorithm, incentive mechanism

1 Introduction

Blockchain, disrupting the current centralized models, is heralded as the next paradigm innovation in digital networks, which opens a door to uncharted cyberspace with ever-increasing security, verifiability and transparency concerns. The performance of blockchain heavily relies on the adopted consensus mechanisms in terms of efficiency, consistency, robustness and scalability. The aim of consensus mechanisms is orchestrating the global state machine so as to agree on the order of deterministic events and screen out invalid events. Undisputedly, the most popular consensus mechanism is PoW, which is adopted by two mainstream blockchain systems, namely Bitcoin and Ethereum.

PoW determines accounting rights and rewards through the competition among nodes (*miners*) to solve a hard cryptographic puzzle by brute-forcing, which is called *mining*, an extremely computation-hungry process. It is reported that the total electricity consumption of Bitcoin is comparable to that of Austria annually; the electricity that a single Bitcoin transaction expends is equal to 22.32 U.S. households powered for one day [1]. The energy-wasting way of PoW deviates from the sustainable and environment-friendly trend for current technology development, thus diluting its value and hindering its further application.

To tackle the drawback of PoW, researchers proposed solutions from two different perspectives: *energy-conservation* and *energy-recycling*. Proof of stake (PoS) [2] and voting-based consensus algorithms [3] are typically energy-conservation approaches. They economize on energy by cutting down the mining difficulty of rich stakeholders or their delegates. Non-democracy is an obvious side effect of these approaches since they have a bias toward wealthy peers. Energy-recycling consensus algorithms address the energy-wasting issue of PoW from a different angle. They recycle the energy which is originally employed to solve cryptographic puzzles for useful tasks. For instance, the mining energy can be repurposed for finding long prime chains [4], matrix computation [5], image segmentation [6] and deep learning [7]. The idea of energy-recycling consensus algorithms, i.e., turning the meaningless proof of work into practical tasks for completing the consensus of blockchain, undoubtedly deepens the integration of blockchain and other fields, expanding the application scope of blockchain.

In this paper, we propose a novel energy-recycling consensus algorithm: *proof of federated learning* (PoFL), where the energy originally wasted to solve difficult but meaningless puzzles in PoW is reinvested to federated learning. Federated learning [8] is a distributed machine learning approach, with the idea of bringing code to data rather than the reverse direction. In federated learning, a high-quality model maintained by a central server can be learned through aggregating locally-computed updates, which are sent by a loose federation of participating clients. Since the local training dataset of each client will not be sent to the central server, privacy and security risks are significantly reduced.

- *Xidi Qu and Shengling Wang are with the School of Artificial Intelligence, Beijing Normal University, Beijing 100875, China. E-mail: sindychanson@mail.bnu.edu.cn, wangshengling@bnu.edu.cn.*
- *Qin Hu is with the Department of Computer and Information Science, Indiana University-Purdue University Indianapolis, Indianapolis, IN 46202 USA. E-mail: qinhu@iu.edu.*
- *Xiuzhen Cheng is with the School of Computer Science and Technology, Shandong University, Jinan 250100, China. E-mail: xzcheng@sdu.edu.cn.*

Manuscript received 26 Apr. 2020; revised 23 Jan. 2021; accepted 1 Feb. 2021. Date of publication 4 Feb. 2021; date of current version 26 Feb. 2021. (Corresponding author: Shengling Wang.) Recommended for acceptance by S. Chen. Digital Object Identifier no. 10.1109/TPDS.2021.3056773

Besides PoFL can inherit the advantage of energy-recycling consensus algorithms, we propose it due to the following reasons:

- First, PoW and FL are functionally complementary. By combining FL with blockchain, miners' computing power can provide FL with sufficient resources for deep learning. In return, honest miners' computing capacity ensures the security and tamper-proofness of blockchain [7].
- Second, PoW and FL have a natural fit in terms of organizational structure. Due to the huge difficulty of individual mining, *pooled-mining* becomes a trend of PoW, where miners join pools, gathering their computational power, to figure out the cryptographic solution, and then the manager of each pool allocates rewards proportionally to each miner's contribution. In other words, both pooled-mining and FL come with a *clustering* structure. Thus, when the cryptographic puzzle in PoW is replaced with the FL task, the cluster head, namely the pool manager, can coordinate the locally-computed results updated by pool members (miners) to generate a high-quality model.
- Third, PoW and FL are matched in working mode. On one hand, the training process of the FL model is obviously more difficult than its verification, which is in line with the requirements of the useful work in energy-recycling solutions that verifying the work is much easier than completing the work. On the other hand, the training time, the accuracy of the FL model, and the computing capacity restrict and influence each other in FL, where the more computing power and the lower the accuracy requirements, the shorter the training time. Thus, PoFL consensus can control the average time consumption of block generation as PoW does, by controlling the accuracy requirements according to the computing resources in the whole blockchain.

However, the harmony of PoW and FL does not imply it is non-trivial to realize PoFL. In detail, the merit of federated learning lies in that all clients collaboratively train a high-quality model while keeping their training data private from others, where the usufruct and ownership of local training data is an integration. However, the openness of blockchain endows anyone with the right of mining, i.e., the local model training in PoFL, which results in the separation between the data usufruct and ownership. This may lead to data privacy leakage in model training and verification, deviating from the original intention of federal learning.

Our paper aims to address the above challenge so that PoFL can meet practical demands. To the best of our knowledge, our paper is the first work to employ federal learning as the proof of work for blockchain, where the main contributions are summarized as follows:

- A general framework of PoFL is introduced, which clarifies the interaction among all entities involved and designs a new PoFL block structure for supporting block verification so as to realize the consensus for blockchain.
- A reverse game-based data trading mechanism is proposed to leverage market power for guarding against training data leakage. This mechanism can determine the optimal data trading probability and pricing strategy even when a pool conceals his[1] profit from privacy disclosure. Driven by the proposed mechanism, a pool with a high risk of privacy disclosure has a low data trading chance and needs to pay a high purchase price, which further incentivizes pools to train models without any data leakage.
- A privacy-preserving model verification mechanism is designed to verify the accuracy of a trained model while preserving the privacy of the task requester's test data as well as the pool's submitted model. The proposed mechanism consists of two parts: the homomorphic encryption (HE)-based label prediction and the secure two-party computation (2PC)-based label comparison.

The remaining part of the paper proceeds as follows. In Section 2, we introduce an overview of our proposed PoFL. To implement PoFL, there are some potential security risks, which are analyzed in Section 3. To cope with the problems of training data exchange between data providers and pools, we proposed an incentive-compatible data trading mechanism based on the reverse game in Section 4. In order to calculate the accuracy of model without disclosing either the test data or the model itself, we proposed a privacy-preserving model verification mechanism based on HE and 2PC in Section 5. We conduct an experimental evaluation to illustrate the effectiveness and efficiency of our proposed PoFL in Section 6, and summarize the most related work in Section 7. The whole paper is concluded in Section 8.

2 Framework of PoFL

PoFL employs federated learning to solve realistic problems with practical value to achieve consensus in blockchain. In our proposed PoFL framework, the problems such as image recognition and semantic analysis are published as tasks on a platform by *requesters*, along with the corresponding rewards as incentives for mining. In addition, requesters may also publish the list of data provider candidates who can provide qualified training data. We set up a queue to store the ready PoFL tasks whose corresponding rewards and the quantity of available data meet the given requirements. Adopting the principle "first in first out", the platform takes the ready tasks from the queue to realize PoFL in turn. If the ready queue is empty, the PoFL will be rolled back to PoW to ensure that the block is generated on time.

As mentioned above, pooled-mining has become a development trend in blockchain currently, which has a similar organization structure with federated learning. Therefore, we investigate PoFL under the pooled-mining paradigm in this paper, whose framework is shown in Fig. 1. According to Fig. 1, pool members (miners) train machine learning (ML) models individually based on their private data to obtain locally-computed updates, which will be aggregated by the pool manager so as to achieve a high-quality model.

1. We denote the pool as "he" and the data provider as "she" for differentiation in the following.

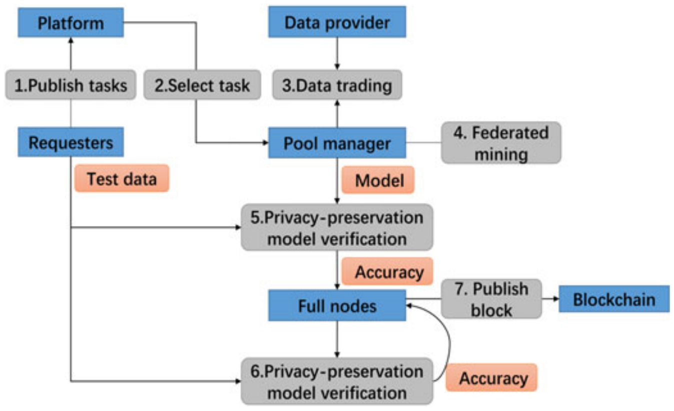


Fig. 1. Framework of PoFL.

This process is named as *federated mining*. After accuracy computation with the requester's test data, each pool manager packages transactions and generates a new block containing the information needed for model verification. Once receiving blocks, full nodes will identify the winner pool through verifying model accuracy. The winner pool should send his model to the requester, thus obtaining the accounting right and the corresponding reward.

2.1 Federated Mining

To implement federated learning, miners in the same pool will collectively train their models without any centralized storage, which is coordinated by the pool manager. With respect to the different storage characteristics of data, there are various types of federated learning [9], such as horizontal federated learning [10], vertical federated learning [11], and federated transfer learning (FTL)[12]. A general federated learning process is shown in Fig. 2 and described as follows:

1) The pool manager broadcasts an initial model as well as a public key to the pool.
2) Each miner individually calculates the gradient value based on the private training data and other information received from the pool manager, which is encrypted using the received public key.
3) The pool manager decrypts the locally-computed updates sent by miners, aggregating them for establishing a shared quality-improved model.
4) According to the deadline for submitting model accuracy published by the platform, the pool manager determines whether or not the next round of training is needed. If not, federated mining is terminated. Otherwise, the aggregated results will be sent back to each miner for implementing Steps 2 and 3 repeatedly.

In fact, which kind of FL is used in PoFL depends on the data obtained by miners. When pool managers can buy enough related sensitive data from data providers, horizontal or vertical federated learning can be adopted according to the different structure traits of the obtained dataset. The horizontal federated learning, initially proposed by Google [13], is used when the datasets of miners in the same pool share the same feature space but differ in samples. The application scenarios of the vertical federated learning are

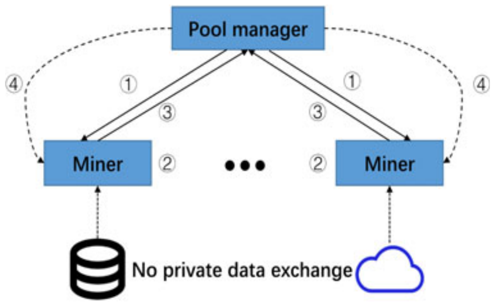


Fig. 2. Federated mining process.

just the opposite, in which datasets of the miners in the same pool have the same sampled individuals rather than feature space. These two types of FL are applicable to the datasets with either common features or common samples.

However, when some miners have poor quality sensitive training data or the set of common samples or features of datasets owned by the miners is small, there may be no enough training data for the whole pool to train a high-accuracy model. In this case, FTL is employed, where pool members are allowed to use rich labeled data of other related source domains to help training the dataset of current target domain by adopting different methods [14], [15] considering the different transfer learning criterions.

In different types of federated mining, the concrete computation of loss and gradient values in Step 2 are different. For example, FTL and horizontal FL require all the miners to compute the prediction results of loss and gradient values but such a job in vertical FL is only carried out in some miners with label values. In addition, vertical FL requires one more step before Step 2, which is called the encrypted sample alignment. This step aims to find the same samples with different features in different miners so that the federated training can be executed among these miners. For the above reason, FTL also requires the samples to align if the source domain overlaps with the target domain in the sample space. All in all, the different concrete algorithms of various types of FL have been studied [9], [10], [11], [12], [13], [14], [15] and the general federated mining framework of PoFL based on different types of FL is roughly similar, which will not be repeated here.

2.2 Block Structure of PoFL

The current block structure does not contain any model-related parameters, which makes the full nodes not able to verify blocks to achieve the final consensus in blockchain. To solve this problem, we design a new block structure, which is shown in Fig. 3. The proposed block header keeps some information in the existing block structure, such as the hash value of the previous block header for maintaining the chain structure, the Merkle tree root for securing transactions, the block height counting from the first block, and the tamper-resistant timestamp. In addition, in order to facilitate other nodes verifying the accuracy of the model for each pool in the network, we include *task*, V_m and M in the PoFL block header. To be specific, *task* is the current executed one by all miners, which is selected by the platform; V_m is the information of the model parameters used for HE

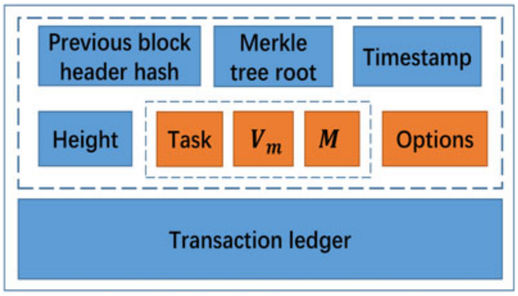


Fig. 3. Block structure of PoFL.

and 2PC when verifying model, detailed in Section 5; and M indicates the value of one or more evaluation metric(s) of model, which are selected for the competition of accounting rights according to different requirements of tasks. Typical evaluation metrics include accuracy, precision, recall, F1 score and so on. In the following, we illustrate our designed framework with accuracy as the model evaluation metric. Further, we add the item of *options* to the block structure. When the model needs other additional information in the header, it can be put into this reserved part.

3 Security Model

We assume that the task platform is neutral and completely credible, which means that the information published on the platform is believable. Also, the requesters are assumed to be credible to some extent. That is, requesters don't publish meaningless tasks, irrelevant data providers or provide unreliable test data in the platform.

Under the above assumptions, there are still security and privacy issues of training data. On one hand, the sensitive training data could be either owned by the pool members themselves or bought from some data providers. In the former case, the original model of federated learning can be applied perfectly to the pool and it is consistent with privacy preservation. In the latter case, data providers do not participate in the mining process of blockchain, while their sensitive data need to be transferred to pools for FL in our consensus mechanism. Hence, the ownership and the usage right of training data are separated. Under this circumstance, malicious data providers may fabricate data or provide tampered data to destroy model training without pools' awareness. Meanwhile, malicious pools also may resell the sensitive data to a third party, leading to privacy leakage.

Also, security and privacy issues of test data are too important to ignore. In our mechanism, once a pool accomplishes the training process, it is the pool manager who will calculate the accuracy of the final model based on the test data provided by the requester. The model accuracy is not calculated by the requester through receiving the trained model from the pool, because the pool manager is not willing to publish the trained model explicitly to the requester before the end of consensus competition to avoid being plagiarized by other competitors. Here a competitor may make an opportunistic choice via submitting not the exact training model but a slightly modified one based on the learning results published by others. However, the requester's test data may also be sensitive that are not suitable to be directly sent to the pool or any third party for accuracy verification due to the privacy leakage concern. In this case, the ownership and the usage right of test data are also separated, posing potential threats for security.

In order to deal with the above potential risks, we design a reverse game-based data trading mechanism for guarding against data fraud and data leakage in Section 4, which leverages the power of the market so that the lower the probability that the pool leaks data privacy, the higher the chance that the pool can buy sensitive data with a low price, thus incentivizing the pool to behave well. Meanwhile, there is no motivation for data providers to modify or manipulate data maliciously due to the payment delay mechanism. In this mechanism, the training data reward paid by the pool will be temporarily stored in the platform, and if the model trained by the training data deviates too much from the final result, data provider and pool need to negotiate under the coordination of the platform, who arbitrates the payment of data trading finally. Otherwise, the training data reward will be delivered to the data provider normally.

Furthermore, we propose a privacy-preserving model verification mechanism in Section 5. With the help of HE and 2PC, the test data and model will not be leaked directly to others, greatly reducing their privacy risks before the end of a task. Note that this mechanism can also be employed by full nodes to protect the privacy of the requester's test data for verifying the accuracy of all received models.

4 Reverse Game-Based Data Trading Mechanism

As mentioned above, when training data are bought from some data providers, there is a potential risk of privacy leakage due to the separation between the ownership and the usage right of the data. An intuitive countermeasure is that data providers encrypt their sensitive data and send them to the pool for miners' training. However, this purely cryptographic approach needs enormous computing power and makes the training time too long. Take the training on the encrypted MNIST dataset [16] as an example. It takes 570.11 seconds to run a single instance using CryptoNets model [17] and nearly 57 hours to train two epochs using CryptoCNN model [18]. Only when the miners computing power is strong enough and the training time of encrypted data can meet the tolerance of the block generation rate and overall performance of blockchain, can this approach be feasible. Otherwise, it is not practical to directly employ encrypted data to train ML models for consensus in blockchain.

To tackle the above challenge, we propose a reverse game-based data trading mechanism, which takes advantage of market power to make a rational pool maximize his utility only when he trains the model without any data leakage. The proposed mechanism leverages the reverse game to describe the cooperative and conflictive relationship between the pool and the data provider, which is an efficient tool to explore the solutions for a class of private-information games. The pool also has private information in reality, such as the net profit of pooled-mining and the profit of disclosing data privacy. This information is tightly related to the probability that the pool discloses training data, which will never be told to the data provider. Leveraging on the reverse game theory, the data provider can determine the optimal trading probability and the corresponding price without knowing the private information of the pool, thus preserving the privacy of the training data using the market as a tool.

In our mechanism, the data trading probability and its purchase price depend on not only the private information of the pool but also his reputation. In order to establish a credible reputation mechanism, we publish the pools' reputation information and the data trading records between the pool and the data provider on blockchain. Each piece of shared data can be marked with steganography techniques [19]. Thus secret information can be embedded into public information so that it cannot be obtained by a third party. More specifically, before sharing data, the data provider embeds the identity of the buyer into the original data without affecting its use. When this piece of data is detected to be disclosed, the hidden identity of the malicious pool(s) can be extracted for accountability.

Due to the features of transparency and traceability of blockchain, once a piece of data is disclosed and the malicious pool is detected offline by any blockchain participant, the update information of reputation value will be broadcasted as one piece of transaction which will be packaged into the newly generated block. After the transaction and the block where this transaction is packaged are validated, the pool will be accountable for data leakage with a reduced reputation. This further affects the data trading probability and the purchase price of the pool in the future.

The reverse game-based data trading mechanism selects the pool manager as the representative to conduct data trading with data providers. This design is based on two considerations: 1) It can avoid the waste of resources caused by two pool members in the same pool purchasing two copies of the same data; 2) The pool manager is able to have the knowledge of the data amount used for training in the entire pool, so that the amount of training data for each miner can be distributed as evenly as possible, which acts as the cornerstone of the average income allocation after successful mining. Once the data trading process is finished, the data provider directly sends the data to the corresponding miner, which helps avoid the communication overhead and possible privacy leakage risks caused by the pool manager's transmission.[2] We postulate that the miners in a pool trust each other as the pool has access control policy for miners before they join in mining, which determines whether they can become reliable participants of FL implemented in the pool. The access control can be made on the basis of the qualification of miners computation capability, the number of tokens they have, and so on.

To optimize the incomplete-information game between the data provider and the pool, the data provider is empowered to design a game rule, which can enforce the pool to derive the strategy based on his real private information. This requires the game rule to satisfy the incentive compatibility (IC) principle, implying that the game rule can enable the pool to obtain a higher utility when he develops the strategy based on the real private information rather than the fake one. In particular, the expected utility (U_{pool}) of the pool within a duration T is defined as

2. Each miner can acquire training data directly from one or more data providers which are assigned by the pool manager according to the data trading contracts, rather than obtain all data from the pool manager who need to collect training data from all data providers.

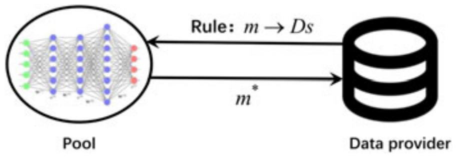


Fig. 4. Framework of reputation-based data trading mechanism.

$$U_{pool} = \int_0^T p \cdot (Q + \tilde{c}(r,V) - m - D_s)\mathrm{d}t. \tag{1}$$

In (1), Q is the legally expected net profit of a pool from this data trading; $\tilde{c}(r,V)$ is the expected profit of leaking those sensitive data, which is closely related to the pool's reputation $r \in [0,1]$ and the value (sensitivity) of the data, denoted by V, so we define it as $\tilde{c}(r,V) = \tilde{\alpha}(1-r) + \tilde{\beta}V$ with $\tilde{\alpha}, \tilde{\beta} \geq 0$ being coefficients; m and D_s are respectively the bid of the pool and the markup price proposed by the data provider, and thus the final price of the traded data is $m + D_s$; p is the probability that the pool can successfully purchase data from the data provider, which can be obtained by

$$p = \varepsilon_1 r + \varepsilon_2 \frac{D_s}{\bar{D}_s} + (1 - \varepsilon_1 - \varepsilon_2)\frac{m}{\bar{m}}. \tag{2}$$

In (2), $\bar{m}$ and $\bar{D}_s$ are respectively the highest values of m and D_s in the recent rounds of data trading; ε_1 and ε_2 are coefficients satisfying $\varepsilon_1, \varepsilon_2 \geq 0$ and $\varepsilon_1 + \varepsilon_2 \leq 1$. The above equation indicates that the higher the reputation of the pool or the higher the final price for the sensitive data, the more willing the data provider to sell the data. Our mechanism requires that at the beginning of the data trading process, the rule for calculating the successful trading probability shown in (2) will be informed to both parties, i.e., the data provider and the pool.

Similarly, the expected utility ($U_{provider}$) of the data provider within a duration T is defined as

$$U_{provider} = \int_0^T p \cdot (m + D_s - c(r,V))\mathrm{d}t, \tag{3}$$

In (3), $c(r,V)$ is the expected loss brought by sensitive data leakage, defined as $c(r,V) = \alpha(1-r) + \beta V$, which has the similar definition with $\tilde{c}(r,V)$ and $\alpha, \beta \geq 0$ are coefficients. With a higher value of the sensitive data, the markup price proposed by the data provider, namely D_s, should also be higher. So we define $V = \eta D_s$, with $\eta > 0$ being the coefficient.

In the reverse game-based data trading mechanism, the strategies of the pool and the data provider are respectively m and $D_s(m)$. In other words, the strategy of the data provider is not a value but a rule, i.e., a function, which empowers the data provider to force the pool to make the optimal bid based on his real private information, such as the legally expected net profit (Q) of a pool from this data trading and his expected profit ($\tilde{c}(r,V)$) from leaking sensitive data. Our proposed mechanism is illustrated in Fig. 4, which includes three phases:

- Phase 1: The data provider first designs an optimal game rule $D_s^*(m)$ which can maximize her utility.
- Phase 2: Once receiving $D_s^*(m)$, the pool needs to decide whether or not to accept the game rule. If yes, he will calculate the optimal bid m^* according to $D_s^*(m)$ for maximizing his utility. Otherwise, he just ignores the received message.
- Phase 3: If the data provider does not receive m^* before a given deadline, the trading negotiation is terminated. Otherwise, she will calculate D_s^* and thus the probability of data trading in this round, i.e., p, can also be derived in light of (2). Once the data provider and the pool reaches an agreement on the data trading, the final price of the training data is $m^* + D_s^*$.

In the following, we will introduce how to obtain the optimal strategies of both parties, i.e., $D_s^*(m)$ and m^*. Let $F_d = \ p \cdot (m + D_s - c(r,V)) = [\varepsilon_1 r + \varepsilon_2 \frac{D_s}{\bar{D}_s} + (1-\varepsilon_1-\varepsilon_2)\frac{m}{\bar{m}}] \cdot [m + D_s - \alpha(1-r) - \beta\eta D_s]$, the integrand of (3). To maximize $U_{provider}$, we adopt the variational method. In detail, through solving the Euler-Lagrange equation $\frac{\partial F_d}{\partial D_s} - \frac{\mathrm{d}}{\mathrm{d}r}\frac{\partial F_d}{\partial D_s'} = 0$ under $\frac{\partial^2 F_d}{\partial D_s^2} = \frac{2\varepsilon_2(1-\beta\eta)}{\bar{D}_s} < 0$, we have

$$D_s^*(m) = \frac{\varepsilon_2[m - \alpha(1-r)] + (1-\beta\eta)\bar{D}_s[\varepsilon_1 r + (1-\varepsilon_1-\varepsilon_2)\frac{m}{\bar{m}}]}{2\varepsilon_2(\beta\eta - 1)}. \tag{4}$$

Similarly, let the integrand of (1) be $F_p = p \cdot (Q + \tilde{c}(r,V) - m - D_s) = \ [\varepsilon_1 r + \varepsilon_2 \frac{D_s^*}{\bar{D}_s} + (1-\varepsilon_1-\varepsilon_2)\frac{m}{\bar{m}}] \cdot [Q + \tilde{\alpha}(1-r) + \tilde{\beta}\eta D_s^* - m - D_s^*]$. The optimal strategy of the pool can also be calculated through the variational method. That is

$$\begin{aligned} m^* = {} & \frac{A_0A_2A_3\varepsilon_1 r + (A_0A_2\varepsilon_2 + \bar{D}_sA_0^2A_1)[Q + \tilde{\alpha}(1-r)] - \varepsilon_1 r\bar{D}_sA_0^2}{2\varepsilon_2A_0A_2 + 2A_1A_0^2\bar{D}_s - 2\varepsilon_2(\tilde{\beta}\eta - 1)A_2^2 - 2A_0A_1A_2A_3} \\ & + \frac{[2\varepsilon_2A_2A_3 + A_0A_1A_3 - A_0\varepsilon_2][(1-\beta\eta)\bar{D}_s\varepsilon_1 r - \varepsilon_2\alpha(1-r)]}{2\varepsilon_2A_0A_2 + 2A_1A_0^2\bar{D}_s - 2\varepsilon_2(\tilde{\beta}\eta - 1)A_2^2 - 2A_0A_1A_2A_3}, \end{aligned} \tag{5}$$

in which

$$A_0 = 2\varepsilon_2(\beta\eta - 1), A_1 = (1-\varepsilon_1-\varepsilon_2)\frac{1}{\bar{m}},$$
$$A_2 = \varepsilon_2 + (1-\beta\eta)(1-\varepsilon_1-\varepsilon_2)\frac{\bar{D}_s}{\bar{m}}, A_3 = \bar{D}_s(\tilde{\beta}\eta - 1).$$

Theorem 4.1. *When $1 - \beta\eta < 0$, $D_s^*(m)$ in (4) is the equilibrium strategy of the data provider.*

Proof. According to the variational method, when $\frac{\partial^2 F_d}{\partial D_s^2} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial^2 F_d}{\partial D_s \partial D_s'} \leq 0$ and $\frac{\partial^2 F_d}{\partial (D_s')^2} \leq 0$, F_d can be maximized. Because F_d is not related to D_s', $\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial^2 F_d}{\partial D_s \partial D_s'} = 0$ and $\frac{\partial^2 F_d}{\partial (D_s')^2} = 0$. Thus, the condition of maximizing F_d is simplified to $\frac{\partial^2 F_d}{\partial D_s^2} \leq 0$, implying $\frac{2\varepsilon_2(1-\beta\eta)}{\bar{D}_s} \leq 0$ should be met. However, if $\frac{2\varepsilon_2(1-\beta\eta)}{\bar{D}_s} = 0$, $D_s^*(m)$ in (4) will be meaningless, so $\frac{\partial^2 F_d}{\partial D_s^2} = \frac{2\varepsilon_2(1-\beta\eta)}{\bar{D}_s} < 0$ should be satisfied. Due to $\varepsilon_2 > 0$ and $\bar{D}_s > 0$, when $1 - \beta\eta < 0$ is satisfied, F_d can be maximized. Thus, the theorem is proved. □

By the similar way, we can obtain the following theorem:

Theorem 4.2. *When $1 - \tilde{\beta}\eta < 0$, m^* in (5) is the equilibrium strategy of the pool.*

Theorem 4.3. *The game rule $D_s^*(m)$ designed by the data provider is incentive-compatible.*

Proof. We assume that $\hat{Q}$ and $\tilde{c}\widehat{(r,V)}$ are the fake private information, based on which a dishonest strategy $\hat{m}$ is reported to the data provider. Due to $\hat{Q} \neq Q$ and $\tilde{c}\widehat{(r,V)} \neq \tilde{c}(r,V)$, $\hat{m} \neq m^*(Q, \tilde{c}(r,V))$. Because U_{pool} is maximized only when his strategy is $m^*(Q, \tilde{c}(r,V))$ according to (5), $U_{pool}(\hat{m}(\hat{Q}, \tilde{c}\widehat{(r,V)})) \leq U_{pool}(m^*(Q, \tilde{c}(r,V))$. Thus, the pool can maximize his utility only if he reports the strategy based on the true privacy information. In other words, the game rule is incentive compatible. □

The game rule satisfying the incentive-compatibility principle drives the pool to calculate m^* based on his real private information. As the strategy of the data provider is a function of m^*, i.e., $D_s(m^*)$, it will also be derived based on real private information. This is equivalent to the situation where both the pool and the data provider make the optimal strategies for maximizing their utilities based on the global information known by two sides. In addition, the incentive-compatibility of the game rule enforces both m^* and $D_s(m^*)$ to reveal the risk of the data privacy leakage in this round of data trading. This is because they are calculated based on the real private information of the pool, i.e., his legally expected net profit (Q) from this data trading and expected extra profit ($\tilde{c}(r,V)$) from leaking sensitive data, while both rational players in the reverse game are utility-driven, making m^* and $D_s(m^*)$ closely related to whether the data traded in this round will be leaked or not. Therefore, the calculation of successful trading probability in (2) can reduce or even prevent privacy leakage behavior since it depends on not only the historical data privacy leakage behavior of the pool but also the privacy leakage risk in this round.

In addition, we add a payment delay mechanism to further prevent the data providers modifying or manipulating data maliciously. When the data provider transfers data to the pool, the payment paid by the pool will be temporarily stored in the platform. After FL training process based on these data, if the outcome deviates too much from the final result, the data provider and the pool need to negotiate under the coordination of the platform, who arbitrates the payment of the data trading finally. Otherwise, the profit will be delivered to the data provider normally[3]. It is worth noting that there is a hypothesis in the above mechanism - the profit of successful mining is higher than that of cheating on the training result to take back the expenses on the data trading. In this way, the pool will have no motivation to deliberately publish a model which deviates from the actual result (e.g., reporting a wrong result with low accuracy), so as to obtain data at low or even no cost.

5 Privacy-Preserving Model Verification Mechanism

As we mentioned above, after each epoch of training, the model accuracy should be calculated based on the test data of the requester. However, on one hand, the test data may be sensitive so that the requester is not willing to share

3. The specific deviation degree can be given by the blockchain system.

them; on the other hand, the trained model should not be published before the end of consensus competition to avoid being plagiarized by other competitors. To address this challenge, we design a privacy-preserving model verification mechanism to verify the accuracy of the trained model without information disclosure from either the requester or the pool. This mechanism can also be used by full nodes to verify the accuracy of models when receiving blocks from pools.

The accuracy of the trained model is evaluated by the number of the same predicted labels as the actual ones. Hence, the model verification in our mechanism is divided into two parts, i.e., label prediction and label comparison. To illustrate how our mechanism accomplishes label prediction and comparison, we take the deep feedforward network model[4] as an example, which is a typical type of deep learning network structure.

5.1 Homomorphic Encryption-Based Label Prediction

According to the design of the deep feedforward network, the value of a node i in each layer is calculated based on the inputs $\mathbf{x} = \{x_j\}$ from the last layer, which is $f(\mathbf{x}) = \sigma(z_i)$. Here, $\sigma(\cdot)$ represents the sigmoid function and $z_i = \mathbf{x}^T\mathbf{w}_i + b_i$, where $\mathbf{w}_i = \{w_{ij}\}$ is the weight vector and b_i is the bias of node i, with j denoting all nodes in the last layer. Particularly, the input of the first layer is the test data of the requester $A = [a_{ij}]_{I\times M}$, where a_{ij} is the jth attribute in the ith row of data with $1 \le i \le I$ and $1 \le j \le M-1$ with I indicating how many pieces[5] of test data that the requester owns and $M-1$ denoting the number of attributes in each piece of data; while the last item of each piece of test data, i.e., a_{iM} $(1 \le i \le I)$, is the label.

Since $A = [a_{ij}]_{I\times M}$ is sensitive for the requester, to avoid leaking A in the process of calculating the accuracy of models, an effective method is secure two-party computation (2PC). For example, Rouhani *et al.* [20] realized privacy-preserving label prediction utilizing 2PC. However, the communication and computation complexity of directly using 2PC is too high, making the efficiency of label prediction relatively low. To address this problem, we propose the homomorphic encryption (HE)-based label prediction. In detail, the requester only needs to send the encrypted A, denoted as $[\langle a_{ij}\rangle]_{I\times M}$, as well as the corresponding public key to the pool, who can use HE to calculate the outputs of the second layer, i.e., $f(\mathbf{x})$, based on the first (input) layer of encrypted data attributes. This is because the HE technology can output the desired calculation results in plaintext with operations on the ciphertext. Assuming that the number of nodes in the first layer is the same as the number of attributes, i.e., $M-1$, and the number of nodes in the second layer is represented by K. Take the first piece of data as an example, we can calculate the value of node k in the second layer as

$$\langle z_k\rangle = \sum_{j=1}^{M-1}(\langle a_{1j}\rangle \otimes \langle w_{kj}\rangle \oplus \langle b_k\rangle),\ 1 \le k \le K, \tag{6}$$

4. Other models can be processed in the same way.

5. For instance, one piece of data in a medical dataset denotes the data of one patient.

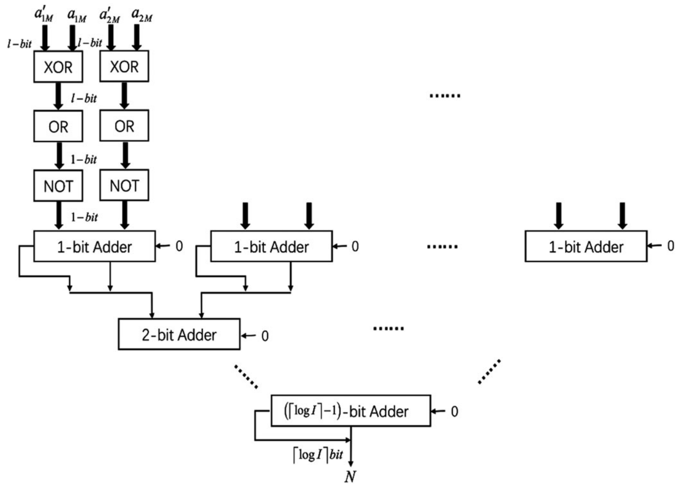


Fig. 5. Circuit for label comparison.

in which $\oplus$ and $\otimes$ are the corresponding computation of $+$ and $\times$ under HE, and $\sum$ is also operated with $\oplus$ calculation.

In order to prevent disclosing the real value of z_k $(1 \le k \le K)$ from the requester, the pool masks them with a random vector $[h_k]_K$, which can be encrypted to $[\langle h_k\rangle]_K$ with the requester's public key. So $\langle z_{ik} + h_k\rangle$ is calculated as follows:

$$\langle z_k + h_k\rangle = \langle z_k\rangle \oplus \langle h_k\rangle, \quad 1 \le k \le K. \tag{7}$$

Once the requester receives $\langle z_k + h_k\rangle$, she decrypts them with the private key and sends back $z_k + h_k$ $(1 \le k \le K)$ in plaintext to the pool, which helps to calculates $\sigma(z_k)$ serving as the input of the next layer in the deep feedforward network. After that, the rest layers in the deep feedforward network model can be calculated locally by the pool with no need to interact with the requester until deriving the predicted labels.

5.2 2PC-Based Label Comparison

After label prediction, label comparison starts, where the accuracy of the model can be derived through counting the number of the same predicted label a'_{iM} as the actual one a_{iM} $(1 \le i \le I)$. However, there still exists the privacy protection issue in this process. On one hand, the pool is not willing to disclose the predicted label a'_{iM} since the trained model might be inferred from this side information. On the other hand, the actual label a_{iM} is sensitive data for the requester as we mentioned above. To overcome this challenge, we propose a 2PC-based label comparison. The main reason of using 2PC lies in that it is easy to employ a garbled circuit (GC) to describe the label comparison process so that this process can be completed efficiently, since the efficiency of 2PC is closely related to the GC construction in both the communication and computation complexity [21], [22], [23].

To realize the proposed method, we first design the boolean circuit for comparison as shown in Fig. 5. The inputs of the circuit are a'_{iM} and a_{iM}, $1 \le i \le I$, both of which are assumed to be l-bit, and the output is the total number N of correct predictions. Based on our circuit design, if $a_{iM} = a'_{iM}$, the statistical total number N of correct prediction adds 1. Otherwise, this is an erroneous one and N adds 0.

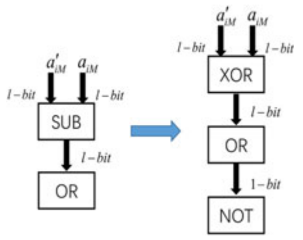


Fig. 6. Optimization of the non-free SUB gate.

Even though the accuracy should be calculated by N/I, we use the total number of correct predictions N to denote it for simplicity since I is stable in one round of calculation, verification and comparison.

The cost of GC is linearly correlated with the number of garbled gates [24]. Taking advantage of the *free-XOR* technology [22], XOR can be *free*, implying that XOR does not need associated garbled tables and the corresponding hashing or symmetric key operations. Therefore, a direct way to improve the performance is reducing the number of costly garbled gates, namely those non-XOR gates. In detail, to compare two values, a SUB gate is supposed to be exploited, which is a non-XOR gate. In our scheme, we replace the SUB gate with a combination of an XOR gate and a NOT gate shown in Fig. 6. Note that the NOT gate is also free since it can be implemented using an XOR gate with one of the inputs as constant 1. We summarize the number of non-free binary gates of our circuit in Table 1, where the n-bit Adder denotes all the adders from 1 bit to $\lceil logI \rceil - 1$ bit, namely $n \in \{1, \ldots, \lceil logI \rceil - 1\}$.

After the fundamental construction of the boolean circuit, we employ JustGarble [21] for garbling, due to its optimization in high efficiency, proven security and garbled row reduction. In short, to garble a circuit is to encrypt the inputs and outputs of the gates in the circuit and disorder the permutation of them. We omit the description of detailed operation for garbling because they are simple variable operations like a couple of shifts. It is worth mentioning that the time cost of the above garbling way is in the level of nanoseconds per gate, and the size of garbled tables is 10^1 order of magnitude, which is efficient enough to meet the requirement for model verification.

Note that GC is finally composed of garbled tables, which gives the keys used to encrypt the inputs and output of each gate. After constructing GC, the pool sends it to the requester along with the corresponding keys of predicted labels, denoted as $w(a'_{iM})$ $(1 \leq i \leq I)$. In order to verify the model accuracy without leaking the privacy of each other, the oblivious transfer (OT) [25], [26] is adopted, by which the requester can only find out the corresponding keys of her actual labels $w(a_{iM})$ $(1 \leq i \leq I)$ without any knowledge on the predicted labels, so neither the pool can know anything about the actual labels. After receiving the keys of inputs a'_{iM} and a_{iM}, the requester can calculate the corresponding garbled encrypted-output and evaluate GC to find out the actual output $\langle N \rangle$, which is the encrypted value of N. Once receiving $\langle N \rangle$, the pool decrypts it and packages it in the block as accuracy.

TABLE 1
Number of Non-Free Gates

Gate	OR	n-bit Adder	All Gates
Number	I	$\frac{I\lceil logI \rceil}{2}$	$I + \frac{I\lceil logI \rceil}{2}$

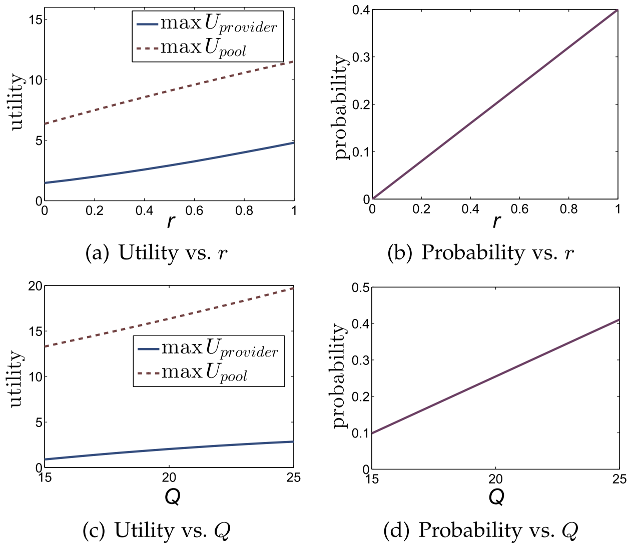


Fig. 7. Impacts of r and Q on data trading.

It is worth noting that our model verification mechanism can also be utilized by full nodes to verify a model's accuracy. Therefore, the encrypted weights and biases need to be packaged in the block with GC. All of these encrypted data are denoted by V_m, which are stored in the block header as we mentioned in Section 2.

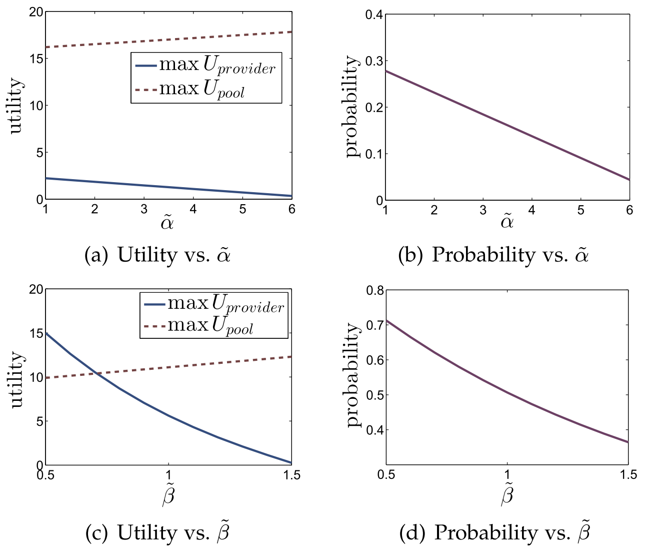
6 Experimental Evaluation

In this section, we evaluate PoFL through simulations based on both synthetic and real-world data. First of all, we study the performance of our data trading mechanism under various key parameters. And then we use a single computer with an Intel Core i7 Processor (3.20 GHz) and 8GB RAM to simulate the federated mining process where the number of miners vary from 20, 50 to 100. We change the amount of data that each miner owns and the learning rate to observe their impacts on federated mining process. Next, we analyze both computation and communication cost of HE and GC construction to verify the feasibility of our model verification mechanism. Finally, we analyze the time cost for full nodes to reach a consensus, which reflects the performance of our consensus mechanism to some extent.

6.1 Data Trading

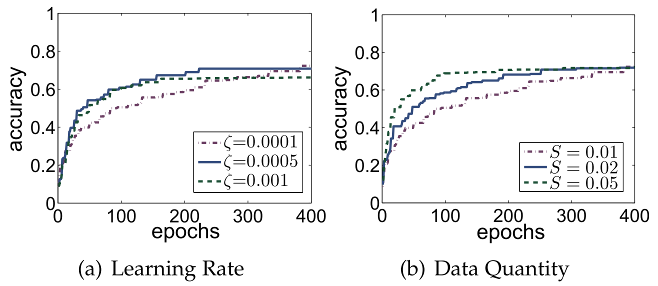
In this subsection, we study the impact of some key parameters in our proposed data trading mechanism. First, we study the impact of the pool's reputation r. Here we set $\varepsilon_1 = \varepsilon_2 = 0.4$, $\eta = 1.8$, $\alpha = \tilde{\alpha} = 1.5$, $\beta = \tilde{\beta} = 1$, and $Q = 8$ to satisfy Theorems 4.1 and 4.2.[6] As shown in Fig. 7a, when r increases, the maximum utilities of both

6. Other parameters satisfying requirements are also evaluated, which presents similar performance trend. So we omit these results to avoid redundancy.



(a) Utility vs. $\tilde{\alpha}$ (b) Probability vs. $\tilde{\alpha}$ (c) Utility vs. $\tilde{\beta}$ (d) Probability vs. $\tilde{\beta}$

Fig. 8. Impacts of $\tilde{\alpha}$ and $\tilde{\beta}$ on data trading.



(a) Learning Rate (b) Data Quantity

Fig. 9. Experiment results of federated mining.

the pool and the data provider quadratically increase. The similar trend happens on the probability that the pool can purchase training data as illustrated in Fig. 7b. The reason behind these facts is that the higher r will bring a higher data trading probability p in light of (2) and a lower expected loss brought by sensitive data leakage $c(r, V)$ according to (3), which together contribute to the increase of utilities for both sides.

Then we examine the impact of the pool's legally expected net profit Q, which is reported in Figs. 7c and 7d. Note that the parameter setting here is the same as the above ones except the varying Q and $r = 0.5$. It can be observed from figures that both maximum utilities and the data trading probability increase with Q. This is because the larger Q, the higher the data price that the pool is willing to provide, which not only increases p as defined in (2) but also improves the utilities for both sides due to (1) and (3).

Next, we evaluate the impacts of $\tilde{\alpha}$ and $\tilde{\beta}$ on data trading, which directly reflect the expected profit of leaking private data, i.e., $\tilde{c}(r, V)$. With the same ε_1, ε_2 mentioned above, we use $\eta = 1.8$, $\beta = \tilde{\beta} = 1$, $Q = 20$ in Figs. 8a and 8b and $\eta = 0.5$, $\alpha = \tilde{\alpha} = 5$, $Q = 20$ in Figs. 8c and 8d. As shown in Figs. 8a and 8c, with the increase of $\tilde{\alpha}$ and $\tilde{\beta}$, which is equivalent to the increase of $\tilde{c}(r, V)$, the maximum utility of the pool increases due to his selling of sensitive data, while the utility of the data provider decreases because of her increasing loss brought by data leakage. According to Figs. 8b and 8d, one can tell that as the extra profit that the pool can gain from leaking data increases, the probability he can obtain the data is decreasing significantly, which can act as a reverse driving force for the legal behavior of the pool.

6.2 Federated Mining

We simulate the federated mining process based on the CIFAR-10 dataset [27] with 50,000 training samples which is composed of 10 classes of 32×32 images with three RGB channels. Here we set that the initial model selected by the pool manager is AlexNet [28]. Even though there are other models with higher accuracy like more than 96.53 percent [29], AlexNet can function well as an example in our proposed mechanism. In fact, how to select the initial model is out of the scope of our work.

Specifically, we take the linear regression training as an example.[7] We use Batch Gradient Descent (BGD) to optimize the parameters of the model and assume that the total number of miners contributing to federated mining in the pool is $\mathcal{K}$ and the learning rate is ζ. As Fig. 9a illustrates, it is not the case where the higher the learning rate ζ the better the convergence speed to achieve certain accuracy [30]. This is because ζ represents the length of step in the direction of gradient descent. If ζ is too small, the training process can be time-consuming; while if it is too large, it may cause excessive loss for the gradient obtained from the training data is an approximate value of the real one. Pools can reach higher accuracy in one round of model training by finding better learning rate.

Assuming that the whole dataset is evenly distributed among miners in the pool and each miner has the same proportion S of the dataset, which is equivalent to $\frac{1}{\mathcal{K}}$. The number of miners $\mathcal{K} = 20, 50, 100$, respectively. As shown in Fig. 9b, the larger the portion of data each miner owns, the faster to reach the highest accuracy. This is because the gradient calculated by each miner will be closer to the real value for the fastest gradient descent with more data.

6.3 Accuracy Calculation

After federated mining, the model of each pool is translated and encrypted to calculate the accuracy based on the privacy-preserving model verification mechanism. We conduct the numerical analysis on both computation and communication cost of HE and GC construction affected by data quantity.

We first study the communication cost between the pool manager and the requester, changing with data quantity. Assuming that there are four layers with 32, 16, 16, 10 nodes respectively in the model, the communication cost of HE and OT of GC is shown in Fig. 10a. As we can see, the communication cost of HE is almost linear to the quantity of data. This is because the exchanged data are $\langle a_{ij} \rangle$, $\langle z_k + h_k \rangle$ and $z_k + h_k$, which are linearly increasing with data quantity. In addition, the communication cost of OT in GC is also increasing with data quantity since it is related to the number of garbled gates. According to Table 1, the number of

7. Other training functions can be implemented in a similar way.

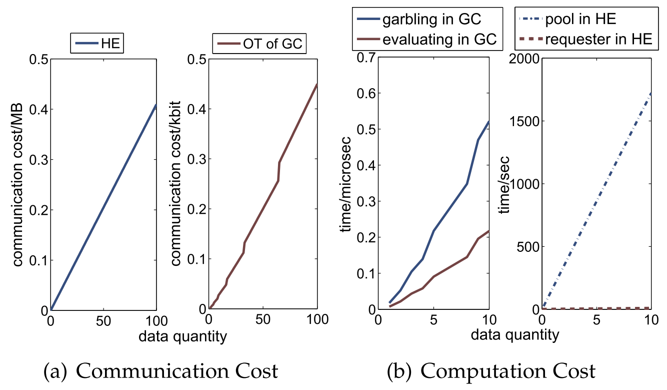


Fig. 10. Experimental results of the model verification mechanism.

garbled gates is changing with $I \times \lceil log(I) \rceil$, where I is exactly the data quantity.

Then we present the computation costs of both HE and GC in Fig. 10b. During HE, both the costs of the pool and the requester increase linearly with data quantity but the cost of the pool increases faster than that of the requester. This is because the requester can complete a lot of calculation offline, such as the encryption of the test data, then the online calculation only refers to the decryption of $\langle z_k + h_k \rangle$; while for the pool, he has to calculate a lot of intermediate results, such as z_k and h_k, which costs him more compared to that of the requester when data quantity increases. While the time cost of GC consists of two parts, garbling the circuit of the pool and evaluating GC of the requester. We use the best PRM-based garbling scheme of JustGarble system called GaX and our processor runs at 3.20 GHz [21]. The consumption of GC construction is proportional to the number of non-free gates in the circuit, which is also changing with $I \times \lceil log(I) \rceil$.

6.4 Accuracy Verification

When blocks are generated, it comes to the accuracy verification process of full nodes. The time for full nodes to verify accuracy and reach a consensus is very significant. The shorter the time, the better the efficiency and the higher the security. In our proposed mechanism, the main time spent on verification is accuracy sorting and testing. The time of sorting is based on the operating speed and the number of blocks. For the operating speed, we take ASIC in [31] as an example. Referring to the average sorting time of ASIC, the time consumption increases with the number of models as shown in Fig. 11a. While for the number of received blocks, according to the statistics of Bitcoin [32], there are 9,364 nodes at present. Thus, the time of sorting all the accuracy of received blocks in our system will not exceed 60 microseconds for full nodes.

After sorting the accuracy of models in a descending order, full nodes test these models from the first one. If the model with the highest accuracy is verified to be true, there is no need to test other models. Otherwise, the second one will be tested until finding the first model with verified accuracy being the same as that stored in the block header. The quantity of our test data is set as 10,000. The average time for full nodes to test the accuracy of models is between

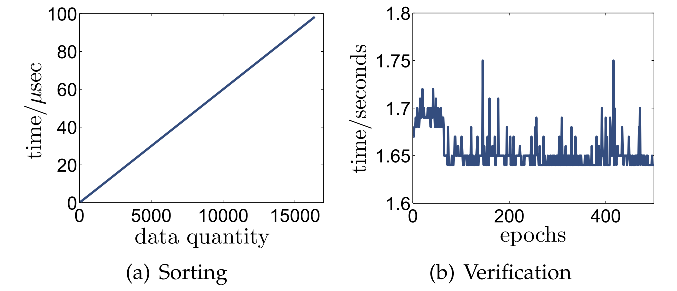


Fig. 11. Time of accuracy verification.

1.65 seconds and 1.7 seconds as shown in Fig. 11b. In addition to the time of sorting, all the time we need is no more than 2 seconds.

7 Related Work

Many attempts have been made to find valuable work as a substitute of puzzles in PoW, which should be difficult to solve but easy to verify. Initially, researchers take a small step forward replacing the nonce with some mathematic problems. Primecoin [4] requires miners to find long prime chains for the proof of work. There are many similar problems [33], such as orthogonal vectors, all-pairs shortest paths problem and rSUM, which requires to find r numbers to have their sum be zero. However, there is no significant practical value in solving these mathematical problems at the cost of ridiculously large amounts of energy.

Next, more diverse and improved attempts are proposed. In proof of exercise (PoX) [5], scientific computation matrix-based problems are sent by employers to a third-party platform where miners select tasks to solve. Permacoin [34] proposes proof of retrievability (PoR) to investigate the storage space and memory for a file or file fragment, where the mining is not associated with computation but storage resources. PieceWork [35] reuses the wasted work for additional goals such as spam prevention and DoS mitigation by outsourcing.

In recent, the combination of deep learning and blockchain has appeared. A substitution of PoW named proof of deep learning (PoDL) [7] first uses deep learning for blockchain maintenance instead of useless hash calculation, which only needs to add some new components to the block header and thus can be applied to the current cryptocurrency systems. However, the user needs to provide a complete training dataset for model training and test datasets for verification to all miners, which sacrifices data privacy and may frustrate the data provider to apply blockchain. Besides, image segmentation is employed in [6], which defines the segmentation model as proof. Furthermore, Coin.AI [36] proposes a proof-of-storage scheme to reward users for providing storage for deep learning models.

8 Conclusion

In this paper, we propose a novel energy-recycling consensus algorithm named PoFL, where the cryptographic puzzle in PoW is replaced with federated learning tasks. To realize PoFL, a general framework is introduced and a new PoFL block structure is designed for supporting block verification. To guarantee the privacy of training data, a reverse game-based data trading mechanism is proposed, which takes advantage of market power to make a rational pool maximize his utility only when he trains the model without any data leakage, thus further motivating pools to behave well. In addition, a privacy-preserving model verification mechanism is designed to verify the accuracy of a trained model while preserving the privacy of the task requester's test data as well as avoiding the pool's submitted model to be plagiarized by others, which employs homomorphic encryption and secure two-party computation in label prediction and comparison, respectively. Extensive simulations based on synthetic and real-world data demonstrate the effectiveness and efficiency of PoFL.

Acknowledgments

This work was supported by National Key R&D Program of China under Grant 2019YFB2102600, the National Natural Science Foundation of China under Grant 61772080, 62072044 and The Blockchain Core Technology Strategic Research Program of Ministry of Education of China under Grant 2020KJ010301 and Engineering Research Center of Intelligent Technology and Educational Application, Ministry of Education.

References

[1] Digiconomist, "Bitcoin energy consumption index," 2019. [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption
[2] S. King and S. Nadal, "PPCoin: Peer-to-peer crypto-currency with proof-of-stake," *Self-Published Paper*, vol. 19, p. 1, Aug. 2012.
[3] G.-T. Nguyen and K. Kim, "A survey about consensus algorithms used in blockchain," *J. Inf. Process. Syst.*, vol. 14, no. 1, pp. 101–128, 2018.
[4] S. King, "Primecoin: Cryptocurrency with prime number proof-of-work," vol. 1, no. 6, 2013.
[5] A. Shoker, "Sustainable blockchain through proof of exercise," in *Proc. IEEE 16th Int. Symp. Netw. Comput. Appl.*, 2017, pp. 1–9.
[6] B. Li, C. Chenli, X. Xu, T. Jung, and Y. Shi, "Exploiting computation power of blockchain for biomedical image segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 2802–2811.
[7] C. Chenli, B. Li, Y. Shi, and T. Jung, "Energy-recycling blockchain with proof-of-deep-learning," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency*, 2019, pp. 19–23.
[8] K. Bonawitz *et al.*, "Towards federated learning at scale: System design," 2019, *arXiv: 1902.01046*.
[9] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.
[10] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.
[11] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, and Q. Yang, "SecureBoost: A lossless federated learning framework," 2019, *arXiv: 1901.08755*.
[12] Y. Liu, T. Chen, and Q. Yang, "Secure federated transfer learning," 2018, *arXiv: 1812.03337*.
[13] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[14] D. Gao, Y. Liu, A. Huang, C. Ju, H. Yu, and Q. Yang, "Privacy-preserving heterogeneous federated transfer learning," in *Proc. IEEE Int. Conf. Big Data*, 2019, pp. 2552–2559.
[15] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A secure federated transfer learning framework," *IEEE Intell. Syst.*, vol. 35, no. 4, pp. 70–82, Jul./Aug. 2020.
[16] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[17] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 201–210.
[18] R. Xu, J. B. Joshi, and C. Li, "CryptoNN: Training neural networks over encrypted data," in *IEEE 39th Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 1199–1209.
[19] A. Yahya, *Steganography Techniques for Digital Images*. Springer, 2019.
[20] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "DeepSecure: Scalable provably-secure deep learning," in *Proc. 55th Annu. Des. Autom. Conf.*, 2018, Art. no. 2.
[21] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, "Efficient garbling from a fixed-key blockcipher," Cryptol. ePrint Arch., Report 2013/426, 2013. [Online]. Available: https://eprint.iacr.org/2013/426
[22] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," in *Proc. Int. Colloq. Automata Lang. Program.*, 2008, pp. 486–498.
[23] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams, "Secure two-party computation is practical," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2009, pp. 250–267.
[24] Y. Huang, C.-H. Shen, D. Evans, J. Katz, and A. Shelat, "Efficient secure computation with garbled circuits," in *Proc. Int. Conf. Inf. Syst. Secur.*, 2011, pp. 28–48.
[25] D. Harnik, Y. Ishai, E. Kushilevitz, and J. B. Nielsen, "OT-Combiners via secure computation," in *Proc. Conf. Theory Cryptogr.*, 2008, pp. 393–411.
[26] S. Jarecki and X. Liu, "Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection," in *Proc. Theory Cryptogr. Conf.*, 2009, pp. 577–594.
[27] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," *Citeseer*, 2009.
[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[29] B. Graham, "Fractional max-pooling," *CoRR*, 2014. [Online]. Available: http://arxiv.org/abs/1412.6071
[30] X. Qu, Q. Hu, and S. Wang, "Privacy-preserving model training architecture for intelligent edge computing," *Comput. Commun.*, vol. 162, pp. 94–101, 2020.
[31] S. Abdel-Hafeez and A. Gordon-Ross, "An efficient O (N) comparison-free sorting algorithm," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 6, pp. 1930–1942, Jun. 2017.
[32] BTC, 2019. [Online]. Available: https://bitnodes.earn.com/
[33] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan, "Proofs of useful work," *IACR Cryptol. ePrint Arch.*, vol. 2017, 2017, Art. no. 203.
[34] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, "Permacoin: Repurposing bitcoin work for data preservation," in *Proc. IEEE Symp. Secur. Privacy*, 2014, pp. 475–490.
[35] P. Daian, I. Eyal, A. Juels, and E. G. Sirer, "(Short paper) PieceWork: Generalized outsourcing control for proofs of work," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2017, pp. 182–190.
[36] A. Baldominos and Y. Saez, "Coin.Ai: A proof-of-useful-work scheme for blockchain-based distributed deep learning," *Entropy*, vol. 21, no. 8, p. 723, 2019.

Xidi Qu received the BS degree in computer science from Beijing Normal University, Beijing, China, in 2018. She is currently working toward the graduate degree with the School of Artificial Intelligence, Beijing Normal University, Beijing, China. Her research interests include blockchain, consensus mechanism, privacy preservation.

Shengling Wang (Senior Member, IEEE) received the PhD degree from Xi'an Jiaotong University, Xi'an, China, in 2008. She is currently a professor with the School of Artificial Intelligence, Beijing Normal University. After that, she did her postdoctoral research with the Department of Computer Science and Technology, Tsinghua University. Then, she worked as an assistant and associate professor from 2010 to 2013 with the Institute of Computing Technology, Chinese Academy of Sciences. Her research interests include mobile/wireless networks, game theory, crowdsourcing.

Qin Hu received the PhD degree in computer science from the George Washington University, Washington, DC, in 2019. She is currently an assistant professor with the Department of Computer and Information Science, Indiana University - Purdue University Indianapolis. Her research interests include wireless and mobile security, blockchain, Internet of Things, and crowdsourcing/crowdsensing.

Xiuzhen Cheng (Fellow, IEEE) received the MS and PhD degrees in computer science from the University of Minnesota Twin Cities, Minneapolis, Minnesota, in 2000 and 2002, respectively. She is currently a professor with the Department of Computer Science, George Washington University, Washington, DC. Her current research interests focus on privacy- aware computing, wireless and mobile security, smart cities, and algorithm design and analysis. She has served on the editorial boards of several technical publications and the Technical Program Committees of various professional conferences/ workshops. She has also chaired several international conferences. She worked as a program director for the U.S. National Science Foundation (NSF) from April to October 2006 (full time), and from April 2008 to May 2010 (part time).