electronics

Article

BCAFL: A Blockchain-Based Framework for Asynchronous Federated Learning Protection

Jian Yun *,†, Yusheng Lu † and **Xinyu Liu**

College of Computer Science and Engineering, Dalian Minzu University, Dalian 116600, China; duture@foxmail.com (Y.L.); lxy365335877@163.com (X.L.)
* Correspondence: yunjianm@163.com
† These authors contributed equally to this work.

Abstract: The existing asynchronous federated learning methods have effectively addressed the issue of low training efficiency in synchronous methods. However, due to the centralized trust model constraints, they often need to pay more attention to the incentives for participating parties. Additionally, handling low-quality model providers is relatively uniform, leading to poor distributed training results. This paper introduces a blockchain-based asynchronous federated learning protection framework (BCAFL). It introduces model validation and incentive mechanisms to encourage party contributions. Moreover, BCAFL tailors matching contribution cumulative strategies for participants in different states to optimally utilize their resource advantages. In order to address the challenge of malicious party poisoning attacks, a multi-party verification dynamic aggregation factor and filter mechanism are introduced to enhance the global model's reliability. Through simulation verification, it is proven that BCAFL ensures the reliability and efficiency of asynchronous collaborative learning and enhances the model's attack resistance capabilities. With training on the MNIST handwritten dataset, BCAFL achieved an accuracy of approximately 90% in 20 rounds. Compared to the existing advanced methods, BCAFL reduces the accuracy loss by 20% when subjected to data poisoning attacks.

Keywords: blockchain; federated learning; asynchronous training; value of contribution; incentive mechanism; aggregation factor; filtering mechanism

Citation: Yun, J.; Lu, Y.; Liu, X. BCAFL: A Blockchain-Based Framework for Asynchronous Federated Learning Protection. *Electronics* **2023**, *12*, 4214. https://doi.org/10.3390/electronics12204214

Academic Editor: Andrei Kelarev

Received: 31 August 2023
Revised: 4 October 2023
Accepted: 10 October 2023
Published: 11 October 2023

1. Introduction

Machine learning is an intelligent technology that trains computer systems using a vast dataset. It enables them to learn from existing data, discover patterns, build models, and autonomously make predictions, classifications, or decisions. It is widely recognized that machine learning model training requires substantial data support. However, due to concerns related to regulations, economic utility, and data privacy, data holders are often reluctant to actively contribute their raw data, forming isolated "Data Islands", severely hindering machine learning development.

Federated learning (FL) [1], a collaborative machine learning framework pioneered by Google, addresses this challenge. It leverages local private data from participating nodes for model training and sends local model updates, instead of raw data, to a central server for aggregation. FL preserves user privacy by not directly exposing sensitive individual data, making it a promising solution to tackle data isolation and privacy leakage. The existing literature indicates that federated learning has wide-ranging potential applications in intelligent healthcare [2], edge networks [3], autonomous driving [4], and smart cities [5].

However, despite the numerous advantages introduced by federated learning, several challenges still require resolution. The primary challenge lies in the high stability requirements for the central server in traditional federated learning. A compromise or failure of the central server can severely disrupt the entire training task. Moreover, traditional federated

learning employs synchronous aggregation, necessitating the submission of local model updates from all participating parties before global model aggregation. Differences in computational capabilities and local data scales among participants may reduce efficiency. Specifically, early contributors must wait for the laggard nodes' updates before performing model aggregation, potentially causing delays in the overall training process.

Additionally, malicious nodes uploading untrustworthy local models could initiate poisoning attacks [6], posing a severe security threat to federated learning. In addition to these issues, the absence of a proper incentive mechanism can deter most data owners from engaging in federated learning. Under the current paradigms, data providers need more motivation to share their local data as they might only partially benefit from the outcomes of federated learning. Finding methods incentivizing data owners to participate actively in such a scenario becomes crucial.

In 2008, Satoshi Nakamoto introduced the concept of blockchain through the release of Bitcoin [7], marking the birth of blockchain technology. In recent years, blockchain technology has made significant progress in decentralized finance [8]. As blockchain evolves, its unique technical characteristics have become increasingly prominent. These characteristics include decentralization, immutability, and traceability [9], which offer viable solutions to address issues encountered in improving traditional federated learning. The research [10] has indicated that blockchain can potentially replace the model aggregation mechanism reliant on central nodes in traditional federated learning, enhancing its reliability and security. It can also mitigate the risks of single points of failure and malicious attacks. Furthermore, the immutability of blockchain ensures the legitimacy of the historical global model. The global model's change process becomes tamper-resistant by storing the aggregation records of model parameters in the blockchain's distributed ledger, thereby bolstering the model's trustworthiness. Most importantly, the traceability feature of blockchain facilitates the monitoring and auditing of the federated learning process. By storing the training history of models on the blockchain, any changes to the model parameters can be traced back to their origins. This not only improves the transparency of model training but also aids in detecting and preventing malicious attacks. Consequently, blockchain technology holds significant promise for advancing the field of federated learning.

Nonetheless, asynchronous federated learning faces challenges, too. Frequent aggregation may result in some participants submitting global models based on outdated versions, potentially leading to lower precision. Dealing with these low-quality models is paramount, and various strategies, examples of which are assigning distinct weights during aggregation or implementing anomaly detection techniques to exclude anomalous models, were proposed in the existing research. However, these methods need more efficiency and stability, necessitating further exploration and optimization.

In order to tackle the issues mentioned above, this paper presents a novel blockchain-based asynchronous federated learning solution, making the following contributions:

- The design of a consortium blockchain-based federated learning framework, BCAFL. This framework not only enhances the reliability of asynchronous federated learning but also meets the requirements of communication efficiency. Additionally, a consensus algorithm, C-Raft, tailored for asynchronous federated learning, is designed to improve training efficiency.
- The introduction of a participant contribution calculation method. The corresponding contribution strategies are devised based on the varying states of participants in the consensus process. Furthermore, an incentive mechanism based on contributions is introduced, effectively motivating more participants to engage in federated learning.
- The proposal of a dynamic aggregation factor based on multi-party validation. This aggregation factor effectively addresses the slow convergence issue of the global model in asynchronous scenarios, thus, enhancing the efficiency of global model training.

Section 1 of the article introduces the relevant research on the existing asynchronous federated learning; Section 2 explains the foundational theoretical knowledge of blockchain and federated learning; Section 3 provides detailed insights into the BCAFL system ar-

chitecture and module design; Section 4 demonstrates the effectiveness of the proposed solution through simulation and empirical validation. Finally, Section 5 summarizes the work presented in the paper.

2. Related Work

In federated learning, participant nodes keep their data locally and construct machine learning models by exchanging local training parameters. With the increasing significance of privacy concerns and the growing demand for improved efficiency in industrial applications of federated learning, more and more scholars are exploring emerging directions in the field [11].

In the early stages of the federated learning research, the primary focus was on synchronous federated learning. During this period, some scholars had already begun to explore integrating federated knowledge with blockchain technology. Kang et al. [12] designed a multi-weight subjective logic model to compute reputation values and assign them to participants, using participant reputation as a metric for measuring the reliability and trustworthiness of the mobile devices. They achieved secure reputation management for employees with undeniable and tamper-proof characteristics through a decentralized approach using blockchain technology. Li et al. [13] introduced the blockchain-based BFLC framework for global model storage and local model exchange. They designed an efficient committee consensus mechanism that minimizes the potential for malicious attacks. Simulated experiments demonstrated the effectiveness and security of the BFLC framework. Peng et al. [14] proposed VFChain, a verifiable and auditable federated learning framework based on a blockchain system. They stored aggregated models in a committee that could be securely rotated within the blockchain. They also introduced a novel blockchain authentication data structure to enhance the search efficiency of verifiable evidence.

The blockchain aspect of the articles mentioned above primarily relies on a consensus algorithm called the committee-based approach. As the integration between blockchain and federated learning becomes more seamless, novel consensus algorithms better suited for federated learning are proposed. Qu et al. [15] introduced a novel energy-recovering consensus algorithm that reinvested energy wasted on the meaningless challenges of PoW into federated learning. They also proposed a data trading mechanism based on reverse games to prevent training data leakage, along with a privacy-preserving model verification mechanism to validate the accuracy of the training models. Muhammad Shayan et al. [16] presented a fully decentralized multi-party machine learning peer-to-peer (P2P) approach called Biscotti. It utilized blockchain and cryptographic primitives to coordinate privacy-preserving ML processes among peer clients.

Furthermore, the consensus algorithms presented in the following articles offer innovative solutions for integrating federated learning and blockchain. Zhang et al. [17] assessed worker reliability by calculating reputation values based on model quality parameters. They employed blockchain to store historical reputation values, ensuring tamper resistance and non-repudiation. P. Ramanan et al. [18] introduced BAFFLE, a non-aggregate, blockchain-driven federated learning approach. BAFFLE leveraged intelligent contracts to coordinate model aggregation and update tasks in federated learning, partitioning the global parameter space into different blocks and using scoring and bidding strategies to improve the computational performance. Umer Majeed et al. [19] allocated separate channels for learning each global model in a blockchain network, storing global models as Merkle Patricia trees. However, this method had limitations, as users relied on the integrity of their respective edge devices. Lu et al. [20] proposed a secure data-sharing architecture based on blockchain authorization, integrating federated learning into the consensus process of a permission blockchain, allowing consensus computations to be used simultaneously for federated training. Pokhrel and Choi [4] presented a framework that captured the impact of controllable networks and BFL parameters on system performance. They conducted a

rigorous analysis of oVML system dynamics to quantify end-to-end latency and provide valuable insights for deriving optimal block arrival rates.

However, these articles are based on synchronous federated learning and suffer from the "Buckets effect", where the aggregation time for each round depends on the participant who sends their local model latest. Chen et al. [21] proposed a novel heterogeneous semi-asynchronous federated learning mechanism called HSA-FL. They allocated different training intensities to clients based on their heterogeneous communication and computational capabilities. They also designed two aggregation rules: adaptive update and fixed adaptive, effectively reducing training time and improving training accuracy. Ma et al. [22] introduced a semi-asynchronous federated learning mechanism, analyzing the quantitative relationship between convergence boundaries and various factors. They deployed adaptive learning rates for employees based on relative participation frequencies, addressing three challenges: edge heterogeneity, non-IID data, and communication resource constraints. Wu et al. [23] presented a semi-asynchronous FL protocol called SAFA to address issues such as low round efficiency and low convergence rates in federated learning under extreme conditions. However, this article did not integrate blockchain technology and had limited capabilities in handling single-point failures.

Pure asynchronous federated learning holds a significant advantage regarding global model aggregation efficiency but also introduces some new challenges. Zhu et al. [24] proposed an outdated compensation algorithm to mitigate model staleness in asynchronous federated learning. They also developed an online client selection algorithm to minimize training latency without the prior knowledge of channel conditions or local computation states, ensuring model accuracy and training efficiency. Zhou et al. [25] introduced a two-stage weighted asynchronous federated learning with adaptive learning rates, utilizing delayed gradients to alleviate the impact of non-IID data. While the methods mentioned above propose solutions for asynchronous federated learning, they do not incorporate blockchain technology, which poses certain security risks. Mondal et al. [26] presented an anomaly detection protocol to minimize the risk of data poisoning attacks in asynchronous federated learning. They used gradient clipping to limit the effects of model poisoning attacks further and designed a new protocol to prevent premature convergence in heterogeneous learning environments. Feng et al. [27] used an entropy-weighted approach to measure the quality of model updates. They determined the proportion and allowed local update delays in global aggregation through score design. Please refer to Table 1 for a comparison between this proposal and the existing proposal.

Table 1. The comprehensive situation of the existing schemes.

Work	Synchronize	Combined with Blockchain	Resist Data Poisoning	Incentive Mechanism	Consensus Protocol
[12]	Sync	Yes	Yes	Yes	Committee
[13,14]	Sync	Yes	No	Yes	Committee
[15]	Sync	Yes	No	Yes	PoFL
[16]	Sync	Yes	No	No	PoFL
[17]	Sync	Yes	No	No	PoR
[18]	Sync	Yes	No	No	PoA
[19]	Sync	Yes	No	No	N/A
[20]	Sync	Yes	No	No	PoQ
[4]	Sync	Yes	Yes	No	PBFT/PoW
[21–23]	Semi-Async	No	No	No	-
[24]	Async	No	No	No	-
[25]	Async	No	No	No	-
[26]	Async	Yes	Yes	No	Depends on fabric
[27]	Async	Yes	Yes	No	PoW
The Proposed	Async	Yes	Yes	Yes	C-Raft

The studies above have introduced methods to overcome challenges in federated learning, such as convergence, privacy, training efficiency, and heterogeneous environments. However, these methods still have limitations, such as overlooking incentives in asynchronous scenarios and offering limited approaches to handling low-quality model contributors. Therefore, this paper introduces the BCAFL method, which combines blockchain and federated learning to address model verification, incentives, and inefficient aggregation challenges, thereby enhancing model aggregation efficiency and system robustness.

3. Preliminary Knowledge

This chapter succinctly elucidates the fundamental concepts of blockchain, consensus algorithms, and federated learning to ensure readers have a transparent background understanding of the subsequent content.

3.1. Blockchain

The essence of blockchain is a distributed ledger that combines the advantages of various technologies such as cryptography, peer-to-peer communication, and distributed storage. It achieves trustworthiness in transactions between parties in trustless scenarios. Blockchain can be categorized into public, consortium, and private chains [28], depending on the use case and requirements.

Blockchain's consensus algorithms dictate the rules for achieving consensus in distributed systems, ensuring that nodes can reach a consistent opinion, despite issues such as node failures, network latency, and communication errors. The proof of work [29] (PoW) consensus mechanism offers high security and fairness. However, the proof of stake [30] (PoS), represented by PoS, competes for blockchain's accounting rights based on the amount and duration of coin holding. While it addresses the energy consumption issues of PoW, it introduces challenges such as the "nothing-at-stake" problem, long-range attacks, and short-range attacks [31].

The delegated proof of stake [32] (DPoS) is an improved PoS algorithm with low energy consumption and high performance. Nodes vote to elect a specific number of representative nodes based on their coin holdings. Elected representative nodes are responsible for maintaining the stability and security of the network. However, because the number of representative nodes is typically small compared to the overall network, it poses a centralization risk.

The consortium chains control node participation through admission mechanisms and primarily uses consensus algorithms such as Byzantine fault tolerance [33] (BFT), crash fault tolerance [34] (CFT), and derived algorithms. Practical Byzantine fault tolerance (PBFT) is a BFT-based consensus algorithm that supports fault tolerance and can tolerate no more than one-third of Byzantine nodes among the total network nodes.

The raft algorithm [35], as a typical CFT algorithm, can ensure consensus consistency even when less than half of the nodes in the network fail or experience network failures. It sets three consensus roles: leader, candidate, and follower, decomposing the consensus problem into leader election, log replication, and safety.

- Leader Election: Raft allocates a continuous sequence of terms; at the start of each term, a leader is elected. If a follower does not receive a heartbeat from the leader within a certain period, it transitions to a candidate. It initiates a new round of leader election to ensure robustness in consensus algorithms.
- Log Replication: The leader receives client requests and appends them as log entries to its log. Once a log entry is committed, the leader notifies the followers and allows them to replicate these log entries locally, thus, maintaining consistency.
- Security: Raft ensures that only nodes with the latest log are eligible to become leaders, thereby preventing outdated leaders from creating partitioned copies.

By integrating blockchain technology, end-to-end encryption can be achieved during federated learning for secure, privacy-preserving model parameter exchange and updates. In blockchain-based asynchronous federated learning frameworks, blockchain applications

provide higher levels of network security for local device collaboration and foster multi-party participation in collaborative research.

3.2. Federated Learning

Federated learning enables multiple terminal computing devices to conduct model training locally, uploading information containing local model updates to a central server while ensuring that the raw data remain local, achieving "available but not visible" data. Let us break down the provided mathematical expressions and summarize the federated learning process:

Suppose there are N federated learning participants, each with a local dataset n_k consisting of m samples, where $n_k = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\} (1 \leq k \leq N)$. Here, x_i represents an input sample, and y_i is the corresponding label. Participant P_k's local training parameters for one iteration in federated learning are denoted as $w_k = \{w_1, w_2, \ldots, w_m\}$, and the local loss function for participant P_k is $Loss_k(w_k, n_k)$.

The objective function for the federated learning model training is defined as follows:

$$minF(W_G) = \frac{1}{N} \sum_{k=1}^{N} \frac{n_k}{n} Loss_k(w_k, n_k) \tag{1}$$

where $F(W_G)$ represents the global objective function, W_G stands for the global model parameters, n represents the total number of local data samples across all participants, and $n = \sum_{k=1}^{N} n_k$. The process of federated learning can be summarized as follows:

Step 1: Initialization of the Global Model:

Set hyperparameters for federated learning, including the number of iterations, local training iteration count, learning rate, etc. Broadcast the initial global model W_G^0 to all participants.

Step 2: Local Model Training:

In the t-th iteration, participants retrieve the previous global model W_G^{t-1} and compute gradients using optimization algorithms such as stochastic gradient descent (SGD):

$$\nabla g(w_G^{t-1}; n_k) = \frac{1}{n_k} \sum_{j=1}^{n_k} \frac{\partial Loss(f(w_G^{t-1}, x_j), y_j)}{\partial w_G^{t-1}} \tag{2}$$

Update local models using the computed gradients:

$$w_k^t = w_G^{t-1} - \eta \nabla g(w_G^{t-1}; n_k) \tag{3}$$

Here, $Loss(f(w_G^{t-1}, x_j), y_j)$ represents the loss function, and $\nabla g(w_G^{t-1}; n_k)$ is the gradient of the local model.

Step 3: Global Model Aggregation

Participants train their local models using their respective datasets n_k, resulting in local models w_k^t.

The central server collects the local models and aggregates them to generate a new global model W_G^t, often achieved through methods such as federated averaging [36]:

$$W_G^t = Aggregate(w_1^t, w_2^t, \ldots, w_N^t) \tag{4}$$

Step 4: Broadcast and Next Iteration:

All participants receive the broadcast of the new global model W_G^t. The participants iterate further with the updated global model until they reach the maximum number of iterations or achieve model convergence.

4. Solution Design

4.1. System Architecture

Unlike the traditional centralized model update approach, this method does not wait for all participants to upload their local models in asynchronous federated learning. Instead, it immediately updates the global model upon receiving valid local models. However, this asynchronous approach also introduces particular challenges, with two main challenges being low efficiency in global model training and privacy–security risks.

Challenge 1. *Dependency on central server stability aggregating the global model relies on the stability of the central server. The entire asynchronous federated learning task would fail if the central server is attacked and becomes incapacitated.*

Challenge 2. *Due to the asynchronous nature and quality issues, asynchronous federated learning faces issues such as unsynchronized global model clocks, low-quality local models, and even the risk of maliciously poisoned local model aggregations. If local models are directly aggregated with the previous version of the global model, the convergence speed of the global model significantly decreases.*

The following section introduces the design strategies for each component, as presented in the text:

Blockchain: Blockchain technology plays a pivotal role. It is utilized to record key information during the global model iteration process and the accumulation of participants' contributions. Each block is linked to the previous block through a pointer, forming an immutable chain structure. The block mainly includes the following recorded content:

- Task Info: The task details are exclusively stored in the genesis block. It includes the task publisher's ID, reward function based on contribution values, contribution collateralization function (proportional to local data volume), maximum iteration count (T), and timestamp.
- Term Info: Details about the leader's ID, term number, and transaction buffer pool status.
- Model Info: Information about the local model provider, training time, IPFS address, validation set, global model IPFS address and hash value.
- Contribution Updates: Additional contribution values from the model provider and leader's appended contribution values.
- Other Parameters: Parameters such as the public keys of all participants, contribution factor α, filtering coefficient σ, and adjustment factor β.

C-Raft Consensus Algorithm: Consensus algorithms favor selecting participants with higher contribution levels when electing leaders. This preference is rooted in enhancing the efficiency and performance of the entire federated learning system. However, this inclination presents a challenge. As leaders, participants must invest more computational resources to execute model aggregation and parameter updates. Consequently, this might slow down the progress of their local model training. In order to address this issue, the paper proposes the following strategy: When a participant is chosen as a leader, they temporarily suspend their local model training and focus on tasks such as global model aggregation. This design ensures that leaders can dedicate their computational consensus efforts to model aggregation and parameter updates while preventing highly contributive participants from consistently engaging in the model aggregation process. As a result, the strategy ensures stability and diversity in the federated learning ecosystem.

Transactions: Transactions are data records generated during the communication process among nodes within a blockchain. In this context, transactions encompass training information and validation details related to model aggregation and verification. This paper's transactions include data pertinent to model aggregation and validation. The leader node maintains a transaction pool as a buffer for pending models awaiting aggregation. Conversely, follower nodes maintain a buffer queue, temporarily storing validation requests.

Contribution Values: Contribution values represent the contributions of each participant to the federated learning task. This paper draws inspiration from the conceptual framework of the PoS mechanism and regards contribution values as a form of PoS-like tokens. The role of contribution values as a prerequisite for participation in the training task is paramount. In this study, the design of contribution values considers different consensus statuses, ensuring a judicious distribution across varying circumstances. Furthermore, contribution values play a significant role in subsequent stages, including incentive mechanisms, model aggregation, and consensus elections.

Participants: Asynchronous federated training underscores the immediate global model aggregation upon receiving a valid local model. In the proposed consensus-based asynchronous federated learning (BCAFL), each participant serves as a trainer in federated learning and an active participant in the blockchain consensus—participants within the consensus algorithm alternate between leader and follower roles. A participant maintains only the most recent global model when in the leader state. Conversely, when in the follower state, the participant locally manages three models: the latest global model, the training model, and the locally trained model obtained after training. The training model corresponds to a historical version of the global model. The newest global model is employed to validate the local models' pending aggregation. Notably, to minimize communication resource consumption, all models in BCAFL are stored in IPFS [37]. During the communication process, only the model addresses are forwarded. Refer to Figure 1 for the specific mechanism.

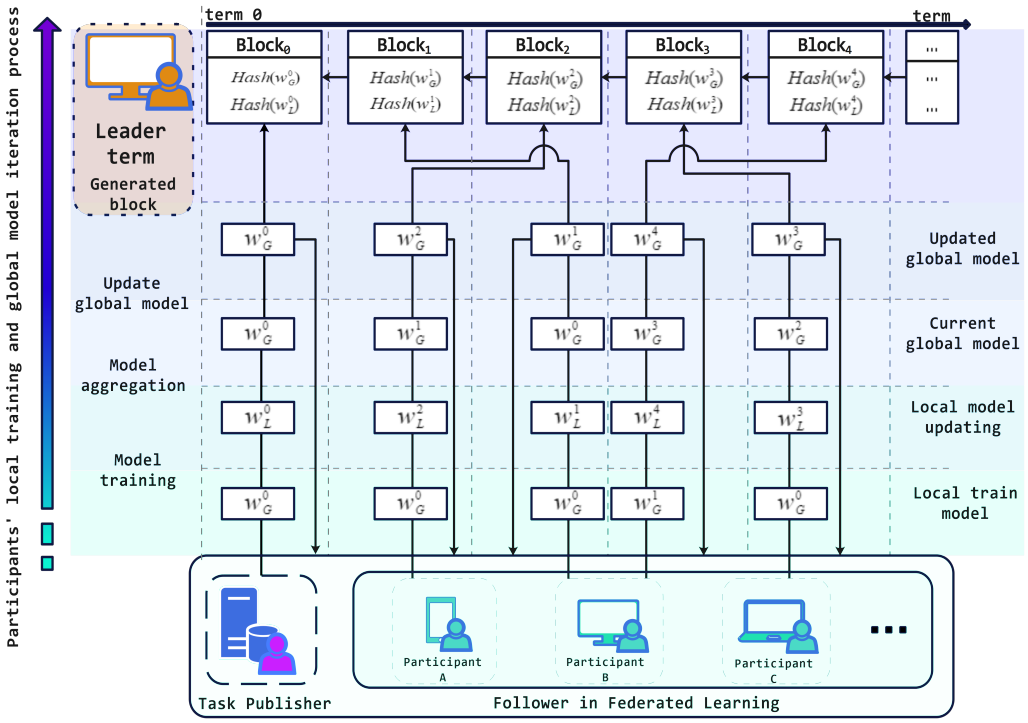


Figure 1. Participants' local training and global model iteration process.

Filtering Mechanism: This mechanism primarily addresses exceptional scenarios. The fundamental concept is to disregard the current update and cease the global model aggregation iteration when the disparity between the local training precision of the participating parties and the multi-party evaluation becomes exceedingly substantial. This precautionary step is taken to maintain the integrity of the collaborative process.

Aggregation Factor: The aggregation factor proposed in this paper comprehensively incorporates several pivotal elements, encompassing the local training precision of the individual model contributors, the contributions of other participating parties, and the validation

outcomes. By taking these factors into account collectively, the aggregation factor configuration ensures the aggregation process's reliability for the global model, maintaining its established integrity throughout.

By incorporating blockchain technology and consensus algorithms to select leader nodes instead of centralized servers, the decentralized replacement is facilitated, and the iterative process of the global model through block-based recording is enabled. Simultaneously, an incentive mechanism based on contribution value is established to motivate participants. This encourages engagement and mitigates malicious leaders' adverse impact on federated learning tasks, thereby addressing Challenge 1.

Throughout this process, ensuring security remains paramount, and thus, a security verification mechanism is introduced. Through digital signatures, hash validation, and similar methods, the legitimacy and trustworthiness of locally submitted models by participants are ensured, effectively preventing the occurrence of malevolent behaviors. Furthermore, the introduction of the concept of an aggregation factor is implemented to manage the aggregation process of local models intricately. This factor determines the aggregation weight of each local model within the global model. This strategy is designed to uphold the convergence rate of the global model, effectively countering Challenge 2.

Therefore, the holistic approach presented in this study integrates blockchain, consensus algorithms, incentive structures, security verification, and the aggregation factor to create a robust framework that addresses the outlined challenges and fosters an environment conducive to efficient federated learning.

4.2. Workflow

The workflow of BCAFL is illustrated in Figure 2. In asynchronous federated learning, considering the involvement of N total nodes in the training process, denoted as P_k, with each participating entity possessing a local dataset of size n_k, the procedure for aggregating a complete round of the global model can be outlined as follows:

Step 1: The task initiator initiates an asynchronous federated learning task, and the potential participants submit registration applications. These applications include information about their local dataset size and historical cumulative contribution value. The task initiator reviews the applications, selects eligible participants based on predefined criteria, and deducts a particular contribution score as collateral. Subsequently, employing a consensus algorithm, a leader is determined, and this leader publishes the initial training information to the genesis block.

Step 2: Participant P_k acquires the latest global model, denoted as w_G^{t-1}, and verifies its hash. Employing the stochastic gradient descent (SGD) optimization algorithm, P_k trains its local dataset.

Step 3: Each participant collects their local model parameters w_k^τ, local training accuracy $A_k(w_k^\tau)$, model hash $Hash(w_k^\tau)$, and training time. This compilation is then signed using a private key. Participants who have completed their local training store the verified model w_k^τ on the InterPlanetary File System (IPFS). Subsequently, they asynchronously transmit the local model update information to the leader for model validation. In this context, the model training time is assumed to be reliable, achieved through the time-consuming proof mechanisms established under Intel SGX [38] trusted hardware technology.

Step 4: The leader compares training duration and data sizes, confirming training authenticity. Once confirmed, the aggregation request is added to the transaction pool and converted into a verification request before being broadcast to other participants. Upon receipt, the participants verify the legality of the signature and retrieve the corresponding local model from IPFS.

Step 5: Using their local dataset, participants verify the legality of the obtained local model w_k^τ. The verification results are signed using the participant's private key and forwarded to the leader.

Step 6: After collecting more than $\frac{2N}{3}$ verification results (excluding P_k and the leader), the leader calculates the aggregation factor and initiates a filtering mechanism. The un-

filtered local updates are then subject to aggregation, yielding the new global model w_G^t for the next round. The leader packages all the valid verifications, the new global model, contribution updates, and consensus-related information into a block broadcast to all the participants for verification.

Step 7: The leader appends the valid block to the blockchain and uploads the new global model to IPFS. If the leader reaches the specified aggregation count, they proactively conclude their term and trigger a new leader election.

Step 8: The participants retrieve the new global model for the subsequent round and commence training anew from Step 2. This process continues until the maximum iteration count is reached or convergence is achieved.

Step 9: Upon the conclusion of the federated learning task, the task initiator allocates rewards to participants based on their accumulated contribution scores.

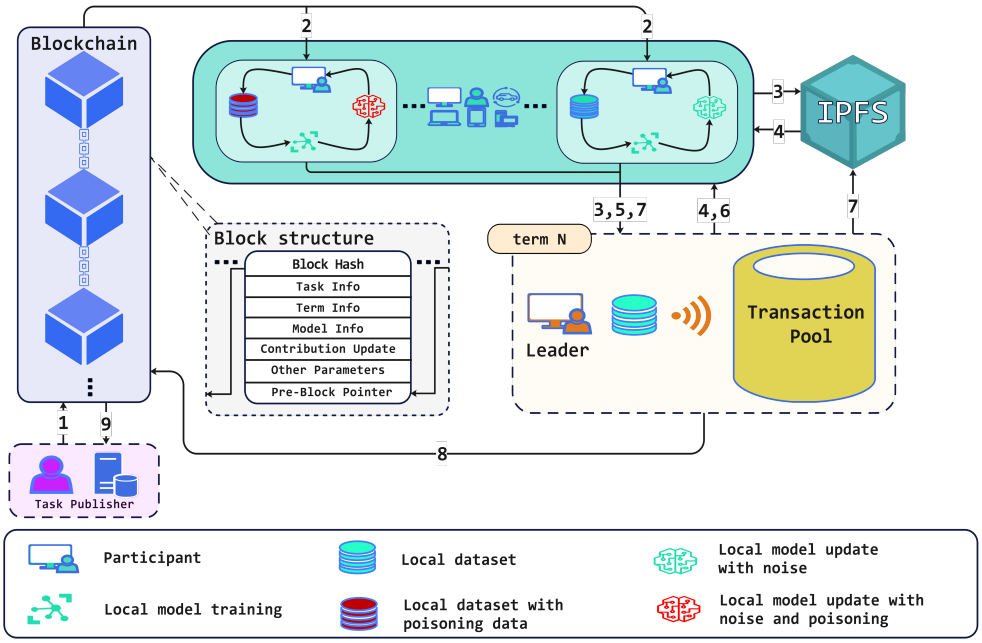


Figure 2. The overall flow of the BCAFL.

4.3. Participants Contribute Values

The participants play a pivotal role in the collaborative model training process in federated learning. They serve as providers of data resources and offer valuable computational resources for federated learning tasks. However, due to the inherent inequality in the local data sample sizes and computational resources among participants, it becomes crucial to establish an appropriate mechanism for quantifying their resource contributions.

Given that each participant in federated learning is treated as a network node within a blockchain context, where consensus among blockchain nodes is required, the participants must also undergo role transitions based on consensus algorithms. The proposed contribution calculation method in this paper takes into account how participants contribute differently to asynchronous federated learning tasks based on their roles in the consensus process. Specifically, the leaders are primarily responsible for tasks such as request forwarding, global model aggregation, and block generation. Meanwhile, the participants are mainly responsible for local model training and model verification tasks.

Each participant can spend extended periods in either a leader or follower state within the consensus process. When a participant is in a leader state, their primary contribution lies in their communication resources. Conversely, when participants transition to a follower state, their contributions revolve around the local data and computational resources, while their contribution in terms of communication resources diminishes. Therefore, considering

these two distinct scenarios, different contribution calculation methods are established for the participants in these two consensus states.

Contribution Value of Leader: The primary responsibilities of a leader include providing aggregation and communication services to followers. Therefore, calculating the leader's contribution value involves obtaining a certain proportion of the followers' contributions. For instance, if a participant P_k sets a commission coefficient σ in advance, after the aggregation, the contribution value obtained by P_k is denoted as $C(w_k^\tau)$. Then, the leader P_j's contribution value $C_j(w_k^\tau)$ for this aggregation can be expressed as:

$$C_j(w_k^\tau) = \sigma C(w_k^\tau) \tag{5}$$

Based on this strategy, leaders, driven by rational choices, prioritize nodes with higher commission coefficients for verification and aggregation.

Contribution Value of Followers: Given the variations in resources among participants, the calculation of contribution value considers relative data contribution and comprehensive evaluation parameters. The comprehensive evaluation parameter considers the provider's training accuracy, other participants' contributions, and model verification accuracy. The contribution value calculation formula is as follows:

$$C(w_k^\tau) = \frac{1}{1 + e^{-\alpha \cdot \frac{n_k}{\hat{n}} \cdot A_{final}(w_k^\tau)}} \tag{6}$$

Here, $C(w_k^\tau)$ represents the contribution value obtained by the provider P_k of the current legitimate local model. Alpha serves as a modulating factor to control the contribution value's growth rate. n_k and $\hat{n}$ represent the local data volume of P_k and the maximum number of samples among participating parties, respectively. $A_{final}(w_k^\tau)$ is the comprehensive evaluation parameter for the legitimate local model w_k^τ aggregated by participating parties P_k in their t-th participation in the global model. The contribution value of participating parties increases with the increase in their number of legitimate aggregations.

$$C_k^\tau = C_k^{\tau-1} + C(w_k^\tau) - C_j^\tau \tag{7}$$

According to the calculation method of contribution value, we can understand that the cumulative contribution of participants during the leader's period is relatively slow. This is to avoid excessive centralized model aggregation computation. In this way, we can ensure that the cumulative contribution of different participants is reasonably balanced throughout the federated learning process. The participants with high contributions will not continuously aggregate models and ignore the importance of local training but instead have the opportunity to actively participate in local model training at appropriate times, thus, maintaining the diversity of local models.

4.4. C-Raft Consensus Algorithm

In order to further enhance the iterative efficiency of asynchronous federated learning and improve system reliability, this paper innovatively introduces the C-Raft consensus algorithm. Constructed upon the raft algorithm and comprehensively addressing the three critical sub-problems of raft, namely, leader election, log replication, and security, the C-Raft algorithm also incorporates the following innovative improvements to suit the asynchronous federated learning process better, the process of the C-Raft consensus algorithm shown in Figure 3:

Appoint candidate: While respecting raft's original leader election method, this paper proposes a wheel selection method based on contribution value weights. This method assigns weights to each participant proportional to their contribution values and selects candidates with probabilities based on these weights. The weight calculation formula is as follows:

$$C_{weight} = \frac{C_k}{C_{total}} \tag{8}$$

In this context, C_k represents the current contribution value of each participating party, C_{total} represents the total contribution value of all nodes, and C_{weight} represents the probability of selecting P_k as a candidate.

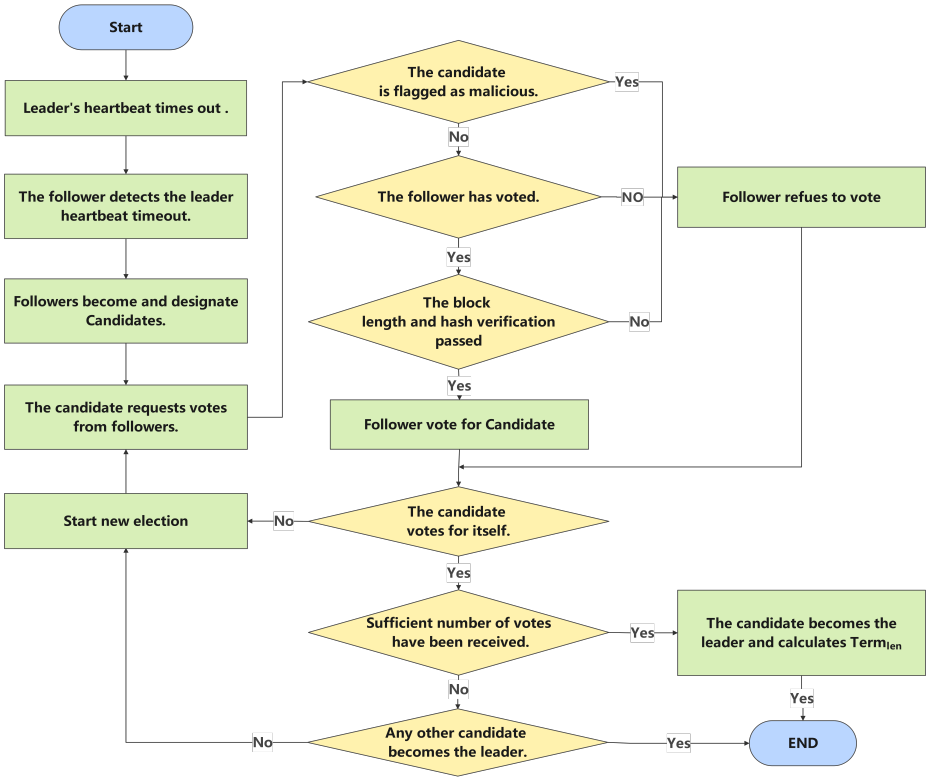


Figure 3. Process of the C-Raft consensus algorithm.

Log replication: When the leader generates a new global model, it packages the relevant parameters of this round's aggregation and transaction pool into a block and performs log synchronization. The follower verifies the accuracy of the information in the block based on verifying the log term number and identifier. If the block verification fails, the log sync is rejected. If more than $\frac{N}{2}$ participants reject the leader, it is considered that the global model aggregation has failed. When the cumulative failure count for this term exceeds f times for block verification reasons, it is marked as a malicious participant, banned from participating in subsequent training tasks, and its deduction contribution value is confiscated.

Leader's active termination: The duration of the leader's term is designed to be influenced by their accumulated contribution value. Specifically, the leader voluntarily retires after generating a certain number of blocks consecutively. The formula for calculating the continuous block generation quantity is:

$$Term_{len} = max\left(2 \times \left(\hat{C} - C_k\right), 2N + 1\right) \tag{9}$$

where $\hat{C}$ is the maximum contribution value among the participants when the leader is elected, and C_k is the contribution snapshot when the leader is elected.

The formula shows that the duration of tenure for each participant at the beginning of the task depends on the total number of participants. As the task progresses, the contribution values of each node gradually increase and vary from each other. When the gap between the lowest and highest contribution values is substantial, the difference in contribution values is chosen as the duration of tenure. This strategy was devised for two reasons:

Prioritize Global Stability: The participating nodes with a significant contribution are considered resource-rich, and selecting them as leaders can help maintain the stability of the federated learning task. Therefore, this strategy increases the likelihood of these high-contributing nodes being designated leaders (Formula (8)), ensuring that the task proceeds in a stable environment.

Fully Utilize Participant Differences: The different participants differ regarding local data volume, training efficiency, and communication resources. Through the leader mechanism, some participants with relatively abundant communication resources but slightly lower model quality can contribute their communication resources by serving as leaders, accumulating their contribution values.

This design fully leverages the strengths of different participants, contributing to the rapid convergence of the federated learning model.

4.5. Dynamic Aggregation Factor and Filtering Mechanism Based on Multi-Party Verification

The study [39] reveals that the weights of local models play a crucial role, directly affecting the aggregation speed and result. Its outdated local model may lead to relatively low accuracy when computing resources are limited at a specific node. Over-reliance on the updates of this particular node may further cause a decline in the accuracy of the global model. Conversely, if a node has abundant data resources, even with an outdated local model, these models may have relatively high accuracy. In such cases, we have more reason to fully utilize the updates of these nodes to promote faster convergence of the global model. Therefore, in federated learning, it is necessary to carefully balance the contributions of each node to ensure the rational utilization of local model updates, thereby achieving optimal performance and efficient convergence of the global model.

The article [40] proposes an asynchronous federated model aggregation method based on the accuracy of local models with the aggregation formula:

$$w_G^t = \frac{w_G^{t-1} + \gamma^t \times w_k^\tau}{1 + \gamma^t} \tag{10}$$

In this paper, γ^t is the dynamic aggregation factor proposed. The local model with the poisoning attack will be validated by the committee and assigned a relatively low weight, but the specific committee's weight assignment strategy is not provided. Therefore, this paper proposes a two-dimensional dynamic aggregation factor global model aggregation method based on training accuracy and multi-party model validation. This method considers the contribution of each verification party while collecting model evaluations and can resist f antagonistic nodes (malicious participants) in federated learning with a total of $3f + 1$ participating parties. The main idea is that leaders collect the validation results of most participating parties and eliminate extreme evaluation values to calculate the aggregation factor. The main steps are as follows and all process in Algorithm 1:

Algorithm 1 Dynamic aggregation factor based on multi-party verification.

Input: T, f, M, w_G^{t-1}, A_G^{t-1}, $A_k(w_k^\tau)$, β, σ
Output: $A_{final}(w_k)$, w_G^t

for $t \leftarrow 1$ **to** T **do**
  for each leader P_j **do**
    Collect local model w_k^t, and accuracy $A_k(w_k^\tau)$.
    Broadcasts local model w_k^τ and collect verification.
    if number of validations $\geq M$ **then**
      Calculate contribution score $V_{j\to k} = C_j \cdot A_{j\to k}(w_k^\tau)$
      Calculate distances: $dist_j = \sum_{i=1,i\neq j}^{M} |V_{j\to k} - V_{i\to k}|$
      Sort distances and select smallest f values to obtain $V_{min} = [dist_1, dist_2, \ldots, dist_f]$
      Calculate aggregated validation score: $V(w_k^\tau) = \frac{1}{f}\sum_{i=1}^{f} A_{j\to k}(w_k^\tau)$
      if $\frac{V(w_k^\tau)}{A_k(w_k^\tau)|A_k(w_k^\tau) - V(w_k^\tau)| + \varepsilon} < \sigma$ **then**
        Leader refuses aggregation
      else
        Calculate Comprehensive evaluation parameter $A_{final}(w_k) = \beta V(w_k^\tau) + (1-\beta)A_k(w_k^\tau)$
        Calculate aggregation factor: $\gamma^t = \frac{A_{final}(w_k^\tau)}{A_k(w_k^\tau)}$
        Update global model: $w_G^t = \frac{w_G^{t-1} + \gamma^t \times w_k^\tau}{1+\gamma^t}$
      end if
    end if
  end for
end for

Step 1: When the number of validations for a local model to be aggregated in a transaction pool reaches $M(M \geq 2f+1)$, the leader obtains all the evaluations and calculates the validation score.

$$V_{j\to k} = C_j \cdot A_{j\to k}(w_k^\tau) \tag{11}$$

where C_j is the contribution value of the verification party P_j, $A_{j\to k}(w_k^\tau)$ is the validation result provided by P_j to the model contributor Pk for the model w_k^τ.

Step 2: Calculate the distance between each validation score and other validation scores:

$$dist_j = \sum_{i=1,i\neq j}^{M} | V_{j\to k} - V_{i\to k} | \tag{12}$$

Considering the worst case, where all f malicious validation results are collected, select the minimum distance values among the first f to obtain $V_{min} = [dist_1, dist_2, \ldots, dist_f]$, and take the average of the corresponding validation results $A_{j\to k}$ (w_k^τ).

$$V(w_k^\tau) = \frac{1}{f}\sum_{i=1}^{f} A_{j\to k}(w_k^\tau) \tag{13}$$

Step 3: Through the training accuracy $A_k(w_k^\tau)$ of the model provider and the validation result $V(w_k^\tau)$ calculated in the previous step, the leader performs a filtering mechanism to mitigate the negative impact of extreme values:

$$\frac{V(w_k^\tau)}{A_k(w_k^\tau)|A_k(w_k^\tau) - V(w_k^\tau)| + \varepsilon} < \sigma \tag{14}$$

where ε is set to ensure that the denominator is not zero, generally set as 10^{-6}; $\sigma(0 < \sigma \leq 1)$ is the filtering coefficient, used to identify the strictness of the filtering. The closer σ is to 0, the lower the strictness of filtering, i.e., the filtering is only performed when $A_k(w_k^\tau)$ and $V(w_k^\tau)$ significantly differ. In this paper, σ is set to 1.

Step 4: Consider both local precision $A_k(w_k^\tau)$ and validation result $V(w_k^\tau)$ and assign a balance factor $\beta(0 < \beta \leq 1)$ to balance local training accuracy and validation evaluation indicators.

$$A_{final}(w_k^\tau) = \beta V(w_k^\tau) + (1-\beta)A_k(w_k^\tau) \tag{15}$$

The final step is calculating the comprehensive evaluation parameter $A_{final}(w_k^\tau)$ and comparing it with the precision ratio of the model contributors to obtain the aggregation factor for this round.

$$\gamma^t = \frac{A_{final}(w_k^\tau)}{A_k(w_k^\tau)} \tag{16}$$

Finally, we bring γ^t into the Formula 10 to obtain the new global model of this round.

5. Experimental Evaluation

5.1. Experimental Environment

The experimental hardware configuration consists of 6 workstations equipped with Intel i7-10700 and 16GB RAM, running on the Ubuntu 20.04.6 LTS system. Python3.7 is used to implement a blockchain based on the C-Raft consensus algorithm, allowing for the creation of a blockchain network within a local area network. Each node in the blockchain trains a local model and submits its weight to the leader node for broadcasting. After being verified by followers, the leader node aggregates the training results, generating a block that is broadcasted for verification and block production across the entire network.

5.2. Experimental Settings

The experimental data are from the MNIST and CIFAR-10 datasets. The MNIST dataset consists of 60,000 training and 10,000 testing samples, each of which is a 28 × 28 grayscale image representing a handwritten digit from 0 to 9. The convolutional neural network used to train the MNIST handwriting dataset consists of two 5 × 5 convolutional layers and two fully connected layers, as shown in Figure 4. The CIFAR-10 dataset contains 50,000 training samples and 10,000 testing samples, each of which is an RGB image of size 32 × 32. The labels include ten general objects such as "airplane", "dog", and "car", with approximately 6000 images per class and a total of 50,000 training images and 10,000 testing images for evaluating the performance of machine learning algorithms. The convolutional neural network used to train the CIFAR-10 dataset consists of two 5 × 5 convolutional layers and three fully connected layers, as shown in Figure 5. Each workstation runs 2–3 nodes for federated learning and consensus communication. For each node, a random selection of 750–1500 non-overlapping data samples is allocated as its local dataset. The training parameters for the two datasets, as well as the distribution of each node's dataset, are shown in Table 2, and the participating blockchain nodes for training are [5, 8, 11, 14, 17], with port numbers set from 7001 to 7017, corresponding to the number of training set samples in Table 2 one-to-one. In the aggregation precision and running time experiment, we set the value of β in Equation (15) to 0.7.

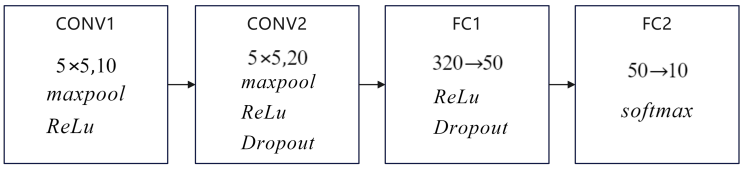


Figure 4. Convolutional Neural Network (CNN) structures on the MNIST handwritten dataset.

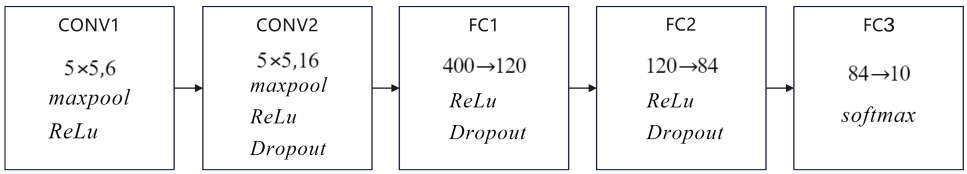


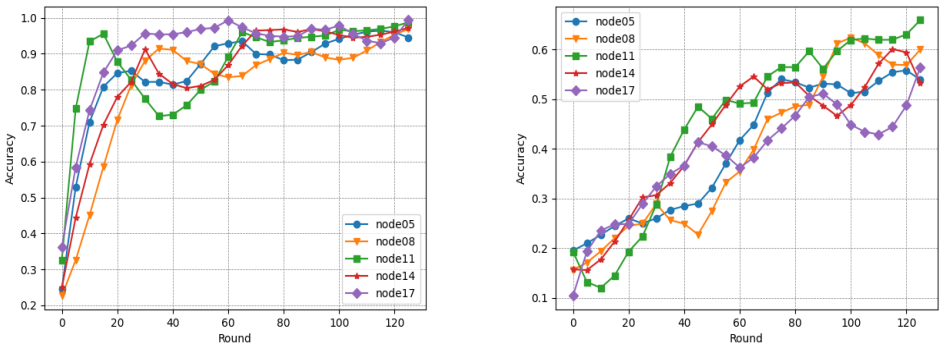
Figure 5. Convolutional Neural Network (CNN) structures on the CIFAR-10 dataset.

Table 2. Distributed model training related parameters.

	MNIST	**CIFAR-10**
Epoch	5	10
BatchSize	64	256
TrainSet	[1047, 1102, 1082, 952, 1284, 1265, 1218, 775, 924, 1383, 1137, 1213, 863, 970, 1122, 1232, 1252]	[1159, 903, 944, 1004, 1458, 897, 890, 808, 1077, 1251, 1127, 1080, 1071, 1018, 1210, 1212, 1278]

5.3. Aggregation Precision

The experimental results show the trend in global model accuracy with the number of aggregation rounds for 5-node to 17-node models. The frequency of model submission for each node was randomly assigned to simulate uneven node computing power and poor network conditions. Due to the leader not participating in each training round, there is a fluctuation in accuracy when the leader's tenure ends and the next leader takes over. Therefore, the curve shows a fluctuating upward trend. Poor computing power and network conditions increase the experiment's randomness but enhance its reference value for practical environments. As shown in Figure 6a, compared to the CIFAR-10 dataset, the MNIST handwritten dataset is lighter. Under five node numbers, the global model significantly improves before 20 training rounds, with an average accuracy of 0.84. At 60 rounds, the average accuracy is 0.9. With the CIFAR-10 dataset shown in Figure 6b, training of a five-layer convolutional neural network cannot achieve high accuracy in the early stages of training aggregation. However, the average accuracy reaches 0.56 at 80 rounds.



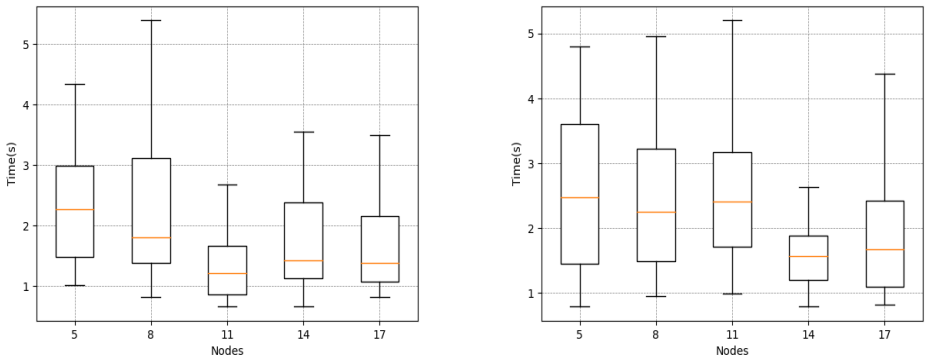
(**a**) MNIST (**b**) CIFAR-10

Figure 6. The training accuracy of global models on MNIST and CIFAR-10 datasets.

5.4. Running Time

The metrics include the time it takes for the leader to broadcast the model submitted by the provider to other follower nodes, the validation time of the follower nodes, the time for the leader to validate and aggregate the model, and the consensus time. As shown in Figure 7a, the average time indicators for the MNIST dataset under five node numbers are [2.34, 2.33, 1.49, 2.07, 2.20], while those for the CIFAR-10 dataset shown in Figure 7b

are [2.56, 2.52, 3.08, 1.99, 2.01]. As the number of nodes increases, the running time of the process does not increase with the node.



(**a**) MNIST (**b**) CIFAR-10

Figure 7. Single round aggregation time.

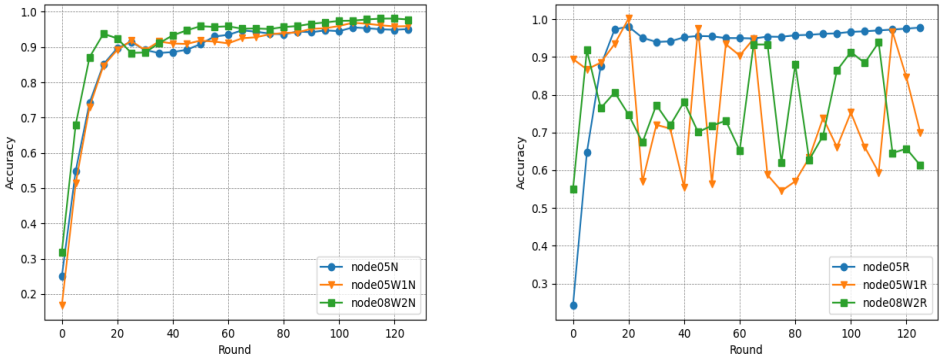
5.5. Data Poisoning

This paper proposes dynamic aggregation factor verification to prevent model providers from uploading false training models or local models with inaccuracy. Followers use local datasets to verify the models provided by the model provider. The leader analyzes the honesty of the model provider by referring to the average validation accuracy of $\frac{2}{3}$ of the followers and the claimed training accuracy of the model provider. If the condition satisfied by Formula N is met, the model provider's model aggregation request is accepted. To simulate model provider misbehavior, experiments use the actual accuracy of the model provider's training results but interfere with the model provider's provided model weights by adding a random perturbation to the weight matrix of the model with a uniform distribution interval of $[-0.5, 0.5]$, to simulate model provider cheating:

$$w_{disturbed} = w_{raw} + rand(w_{raw} \cdot shape) - 0.5 \tag{17}$$

In this experiment, we no longer simulate uneven node computing power or poor network conditions to reflect better the impact of data poisoning on the global model. Additionally, we set the value of β in Equation (15) to 1. As shown in Figure 8a, first, it can be observed that without the ideal state where simulated nodes are not in good condition, the curve does not fluctuate as much as in Figure 6a, and the curve becomes smoother. Additionally, node05N represents the case where no malicious node generally runs in a 5-node blockchain system, node05W1N represents the case where there is one malicious node in a 5-node blockchain system, and node08W2N represents the case where there are two malicious nodes in an 8-node blockchain system. As can be seen from the figure, the accuracy is 0.9 after 20 rounds in all three situations, and the behavior after 20 rounds is consistent. Due to the existence of a dynamic aggregation factor verification mechanism, the leader node selectively ignores models with a significant difference between the evaluation accuracy of the followers and the claimed accuracy of the model provider when aggregating models. This shows that in the case of less than f malicious nodes, the model aggregation is still unaffected by this scheme. The node05R represents the distributed training accuracy of the DBAFL scheme, with five nodes running without data poisoning. The node05RW1R represents the training accuracy when one node is malicious during the training with five nodes. The node08RW2R represents the training accuracy when two nodes are malicious in an eight-node scenario. As shown in Figure 8b, the average training accuracy of DBAFL after twenty rounds is 95.8%, 73.3%, and 75.8%, respectively. It can be seen that the training accuracy decreases by approximately 20% when a node engages in the data poisoning attack. On the other hand, the average accuracy of BCAFL under the same attack conditions

is 92.8%, 93.3%, and 95.1%, respectively. It can be observed that the BCAFL scheme is better at resisting data stealing attacks compared to the DBAFL scheme.



(**a**) The BCAFL Scheme (**b**) The DBAFL Scheme

Figure 8. Protection against data poisoning comparison.

6. Conclusions

This paper proposes an innovative asynchronous federated learning framework, called BCAFL, based on blockchain technology, to address the issues of system reliability, training efficiency, and participation incentives in the existing asynchronous federated learning methods. By leveraging the decentralized and tamper-proof nature of blockchain, BCAFL eliminates the dependence on a single central server, enhances the security and trustworthiness of asynchronous federated learning, and introduces model verification and incentive mechanisms to encourage active participation and ensure high-quality model updates from participants, resulting in improved collaboration willingness and contribution. To address the different states of participants in consensus, an adaptive contribution strategy is designed to maximize the resource advantages of each participant. Finally, a dynamic aggregation factor and filtering mechanism based on multi-party validation are introduced to ensure that only legitimate and high-quality model updates are incorporated into the final global model, thus, enhancing the reliability of the global model.

Through simulation and experimentation, we have confirmed that the increase in the number of participating nodes does not significantly impact the overall efficiency of the BCAFL method. Under different numbers of nodes in distributed training on the MNIST handwritten dataset, we achieved an accuracy of approximately 90% in 20 rounds, while on the CIFAR-10 dataset, with five layers of network training, we achieved an average accuracy of approximately 50% in 80 rounds. Our approach can better resist data poisoning attacks through the filtering mechanism. With a total of $3f + 1$ participants, it can fight f malicious participant nodes, ensuring global accuracy does not decline with negative attacks. Compared to the existing DBAFL method, our approach can reduce the accuracy loss by 20% during data poisoning attacks. Although DBAFL uses committee mechanisms to resist data poisoning, it needs to be explained in detail in the paper, and its actual performance may be better.

In future work, we will consider using BCAFL in non-independent distributed datasets. At the same time, we will introduce an efficient differential privacy algorithm for BCAFL, which adds adaptive noise to the local training process of the participants to prevent privacy data leakage caused by inference attacks. We will further improve the training efficiency while enhancing the privacy protection performance.

Author Contributions: Conceptualization, J.Y. and X.L.; methodology, Y.L.; software, Y.L.; validation, J.Y., Y.L. and X.L.; formal analysis, J.Y.; investigation, X.L.; resources, J.Y.; data curation, Y.L.; writing—original draft preparation, X.L.; writing—review and editing, J.Y.; visualization, Y.L.; supervision, J.Y.; project administration, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Fundamental Research Funds for the Central Universities grant number 04442023128.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Konecný, J.; McMahan, H.B.; Yu, F.X.; Richtárik, P.; Suresh, A.T.; Bacon, D. Federated Learning: Strategies for Improving Communication Efficiency. *arXiv* **2016**, arXiv:1610.05492. [CrossRef]
2. Brisimi, T.S.; Chen, R.; Mela, T.; Olshevsky, A.; Paschalidis, I.C.; Shi, W. Federated learning of predictive models from federated Electronic Health Records. *Int. J. Med. Inform.* **2018**, *112*, 59–67. [CrossRef] [PubMed]
3. Lim, W.Y.B.; Luong, N.C.; Hoang, D.T.; Jiao, Y.; Liang, Y.C.; Yang, Q.; Niyato, D.T.; Miao, C. Federated Learning in Mobile Edge Networks: A Comprehensive Survey. *IEEE Commun. Surv. Tutor.* **2019**, *22*, 2031–2063. [CrossRef]
4. Pokhrel, S.R.; Choi, J. Federated Learning With Blockchain for Autonomous Vehicles: Analysis and Design Challenges. *IEEE Trans. Commun.* **2020**, *68*, 4734–4746. [CrossRef]
5. Jiang, J.C.; Kantarci, B.; Oktug, S.; Soyata, T. Federated Learning in Smart City Sensing: Challenges and Opportunities. *Sensors* **2020**, *20*, 6230. [CrossRef] [PubMed]
6. Fang, M.; Cao, X.; Jia, J.; Gong, N. Local model poisoning attacks to {Byzantine-Robust} federated learning. In Proceedings of the 29th USENIX Security Symposium (USENIX Security 20), Virtual Event, 12–14 August 2020; pp. 1605–1622.
7. Vranken, H. Sustainability of bitcoin and blockchains. *Curr. Opin. Environ. Sustain.* **2017**, *28*, 1–9. [CrossRef]
8. Bonifazi, G.; Cauteruccio, F.; Corradini, E.; Marchetti, M.; Ursino, D.; Virgili, L. Applying Social Network Analysis to Model and Handle a Cross-Blockchain Ecosystem. *Electronics* **2023**, *12*, 1086. [CrossRef]
9. Zheng, Z.; Xie, S.; Dai, H.; Chen, X.; Wang, H. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress), Boston, MA, USA, 11–14 December 2017; pp. 557–564.
10. Li, J.; Shao, Y.; Wei, K.; Ding, M.; Ma, C.; Shi, L.; Han, Z.; Poor, H.V. Blockchain Assisted Decentralized Federated Learning (BLADE-FL): Performance Analysis and Resource Allocation. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 2401–2415. [CrossRef]
11. Qu, Y.; Uddin, M.P.; Gan, C.; Xiang, Y.; Gao, L.; Yearwood, J. Blockchain-enabled Federated Learning: A Survey. *ACM Comput. Surv.* **2022**, *55*, 1–35. [CrossRef]
12. Kang, J.; Xiong, Z.; Niyato, D.; Xie, S.; Zhang, J. Incentive Mechanism for Reliable Federated Learning: A Joint Optimization Approach to Combining Reputation and Contract Theory. *IEEE Internet Things J.* **2019**, *6*, 10700–10714. [CrossRef]
13. Li, Y.; Chen, C.; Liu, N.; Huang, H.; Zheng, Z.; Yan, Q. A Blockchain-Based Decentralized Federated Learning Framework with Committee Consensus. *IEEE Netw.* **2021**, *35*, 234–241. [CrossRef]
14. Peng, Z.; Xu, J.; Chu, X.; Gao, S.; Yao, Y.; Gu, R.; Tang, Y. VFChain: Enabling Verifiable and Auditable Federated Learning via Blockchain Systems. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 173–186. [CrossRef]
15. Qu, X.; Wang, S.; Hu, Q.; Cheng, X. Proof of Federated Learning: A Novel Energy-Recycling Consensus Algorithm. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 2074–2085. [CrossRef]
16. Shayan, M.; Fung, C.; Yoon, C.J.M.; Beschastnikh, I. Biscotti: A Blockchain System for Private and Secure Federated Learning. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 1513–1525. [CrossRef]
17. Zhang, Q.; Ding, Q.; Zhu, J.; Li, D. Blockchain Empowered Reliable Federated Learning by Worker Selection: A Trustworthy Reputation Evaluation Method. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Nanjing, China, 29 March 2021. [CrossRef]
18. Ramanan, P.; Nakayama, K. BAFFLE: Blockchain Based Aggregator Free Federated Learning. In Proceedings of the 2020 IEEE International Conference on Blockchain (Blockchain), Virtual Event, 2–6 November 2020.
19. Majeed, U.; Hong, C.S. FLchain: Federated Learning via MEC-enabled Blockchain Network. In Proceedings of the 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), Matsue, Japan, 18–20 September 2019.
20. Lu, Y.; Huang, X.; Dai, Y.; Maharjan, S.; Zhang, Y. Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial IoT. *IEEE Trans. Ind. Inform.* **2020**, *16*, 4177–4186. [CrossRef]
21. Chen, S.; Wang, X.; Zhou, P.; Wu, W.; Lin, W.; Wang, Z. Heterogeneous Semi-Asynchronous Federated Learning in Internet of Things: A Multi-Armed Bandit Approach. *IEEE Trans. Emerg. Top. Comput. Intell.* **2022**, *6*, 1113–1124. [CrossRef]
22. Ma, Q.; Xu, Y.; Xu, H.; Jiang, Z.; Huang, L.; Huang, H. FedSA: A Semi-Asynchronous Federated Learning Mechanism in Heterogeneous Edge Computing. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 3654–3672. [CrossRef]
23. Wu, W.; He, L.; Lin, W.; Mao, R.; Maple, C.; Jarvis, S. SAFA: A semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Trans. Comput.* **2020**, *70*, 655–668. [CrossRef]
24. Zhu, H.; Kuang, J.; Yang, M.; Qian, H. Client Selection With Staleness Compensation in Asynchronous Federated Learning. *IEEE Trans. Veh. Technol.* **2023**, *72*, 4124–4129. [CrossRef]
25. Zhou, Z.; Li, Y.; Ren, X.; Yang, S. Towards Efficient and Stable K-Asynchronous Federated Learning With Unbounded Stale Gradients on Non-IID Data. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 3291–3305. [CrossRef]

26. Mondal, A.; Virk, H.; Gupta, D. Beas: Blockchain enabled asynchronous & secure federated machine learning. *arXiv* **2022**, arXiv:2202.02817.
27. Feng, L.; Zhao, Y.; Guo, S.; Qiu, X.; Li, W.; Yu, P. BAFL: A Blockchain-Based Asynchronous Federated Learning Framework. *IEEE Trans. Comput.* **2022**, *71*, 1092–1103. [CrossRef]
28. Issa, W.; Moustafa, N.; Turnbull, B.; Sohrabi, N.; Tari, Z. Blockchain-Based Federated Learning for Securing Internet of Things: A Comprehensive Survey. *ACM Comput. Surv.* **2023**, *55*, 1–43. [CrossRef]
29. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X.; Wang, H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352–375. [CrossRef]
30. Bamakan, S.M.H.; Motavali, A.; Bondarti, A.B. A survey of blockchain consensus algorithms performance evaluation criteria. *Expert Syst. Appl.* **2020**, *154*, 113385. [CrossRef]
31. Ferdous, M.S.; Chowdhury, M.J.M.; Hoque, M.A.; Colman, A. Blockchain consensus algorithms: A survey. *arXiv* **2020**, arXiv:2001.07091.
32. Larimer, D. Delegated proof-of-stake (dpos). *Bitshare Whitepaper* **2014**, *81*, 85.
33. Zhong, W.; Yang, C.; Liang, W.; Cai, J.; Chen, L.; Liao, J.; Xiong, N. Byzantine Fault-Tolerant Consensus Algorithms: A Survey. *Electronics* **2023**, *12*, 3801. [CrossRef]
34. Yao, W.; Ye, J.; Murimi, R.; Wang, G. A survey on consortium blockchain consensus mechanisms. *arXiv* **2021**, arXiv:2102.12058.
35. Huang, D.; Ma, X.; Zhang, S. Performance analysis of the raft consensus algorithm for private blockchains. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *50*, 172–181. [CrossRef]
36. Zhou, P.; Lin, Q.; Loghin, D.; Ooi, B.C.; Wu, Y.; Yu, H. Communication-efficient Decentralized Machine Learning over Heterogeneous Networks. In Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 19–22 April 2021. [CrossRef]
37. Benet, J. Ipfs-content addressed, versioned, p2p file system. *arXiv* **2014**, arXiv:1407.3561.
38. Chen, L.; Xu, L.; Shah, N.; Gao, Z.; Lu, Y.; Shi, W. On Security Analysis of Proof-of-Elapsed-Time (PoET). In *Lecture Notes in Computer Science*; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 282–297. [CrossRef]
39. Chen, Y.; Ning, Y.; Slawski, M.; Rangwala, H. Asynchronous Online Federated Learning for Edge Devices with Non-IID Data. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020. [CrossRef]
40. Xu, C.; Qu, Y.; Luan, T.H.; Eklund, P.W.; Xiang, Y.; Gao, L. An Efficient and Reliable Asynchronous Federated Learning Scheme for Smart Public Transportation. *IEEE Trans. Veh. Technol.* **2023**, *72*, 6584–6598. [CrossRef]