

Received 11 October 2023, accepted 16 November 2023, date of publication 21 November 2023, date of current version 30 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3335603

RESEARCH ARTICLE

A Blockchain-Based Auditable Semi-Asynchronous Federated Learning for Heterogeneous Clients

QIAN ZHUOHAO¹, MUHAMMAD FIRDAUS¹, (Graduate Student Member, IEEE),
SIWAN NOH¹, AND KYUNG-HYUNE RHEE², (Member, IEEE)

¹Department of Information Security, Pukyong National University, Busan 48513, Republic of Korea

²Division of Computer Engineering, Pukyong National University, Busan 48513, Republic of Korea

Corresponding author: Kyung-Hyune Rhee (khrhee@pknu.ac.kr)


This work was supported by the Ministry of Science and ICT (MSIT), South Korea, through the Special Research and Development Zone Development Project—Development of Research and Development Innovation Valley Support Program, Supervised by the Innovation Foundation, under Grant 2023-DD-RD-0152.

ABSTRACT Federated learning (FL) is a privacy-preserving approach in Artificial Intelligence (AI) that involves exchanging intermediate training parameters instead of raw data, thereby avoiding privacy breaches and promoting effective data collaboration. However, FL still faces several unresolved challenges, including trust issues among participants because traditional FL lacks a mutual consensus auditing mechanism. Another challenge is that when the number of participating nodes is large and resources are heterogeneous; this can lead to low efficiency. To overcome these challenges, we propose a Blockchain-based Auditable Semi-Asynchronous Federated Learning (BASA-FL) system. BASA-FL includes a smart contract that coordinates and records the FL exchange process, enabling the ability to trace and audit the behavior of participating workers. In addition, we proposed an efficient semi-asynchronous approach in blockchain-based distributed FL as the main contribution to addressing heterogeneous problems. We designed a method to quantify worker contributions and distribute rewards based on their contributions. We used a multi-index comprehensive evaluation to motivate workers to maintain high-quality and efficient participation in FL tasks. We conducted several simulations to evaluate the effectiveness of the semi-asynchronous mode, the reliability of the audit mechanism, and the contribution quantification strategy.

INDEX TERMS Semi-asynchronous FL, blockchain, auditability, contribution quantification.

I. INTRODUCTION

In recent years, machine learning has significantly improved in various fields, such as image recognition, natural language processing, and recommendation systems. These successes were closely linked to the availability of large amounts of data. Federated data has been a topic of interest for many years, from federated database system (FDBS) [1] technology in the 1990s to federated cloud computing (FC) [2]. However, while federated data allow for collaboration among multiple parties, it also poses a significant risk of privacy

The associate editor coordinating the review of this manuscript and approving it for publication was Yiming Tang .

breaches. Federated learning emerged as a solution to this issue after Google proposed the FedAvg [3] algorithm in 2016 because it offers a balance between data collaboration and privacy protection. The core concept of FL is that federated participants train AI models locally on their own data and then exchange model parameters or gradients for collaboration rather than exchanging raw data, thus preserving privacy. Google implemented FL in its Gboard mobile keyboard [4] and deployed it on multiple mobile devices. This is known as the cross-device FL method. This method inspired the emergence of transaction-based FL task crowdsourcing [5], where individuals or organizations seek to acquire a specific AI model but lack the necessary data to

train it. Therefore, they can recruit workers with data to train the model collaboratively. In this way, workers can engage in data trading while ensuring their privacy. The emergence of these markets further facilitates the circulation and utilization of data, thereby driving the advancement of AI. Figure 1 illustrates a data trading platform based on federated learning.

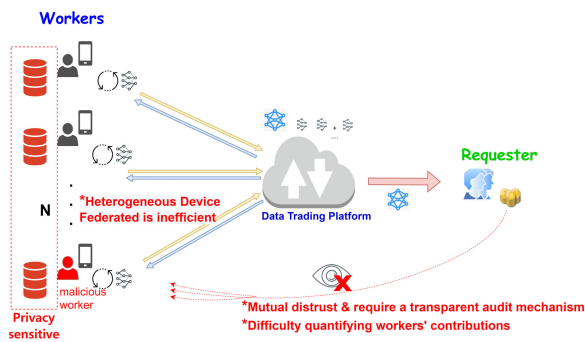


FIGURE 1. A data trading platform based on federated learning.

Inspired by the success of Gboard, FL utilization has been promoted across a myriad of applications. In healthcare, Xu et al. [6] investigated how FL can address the challenges posed by fragmented and private healthcare data, particularly in the context of biomedical applications. Their exploration involved connecting diverse data sources while safeguarding privacy. Furthermore, Firdaus and Rhee [7] employed FL to address the challenges in existing vehicular networks by enhancing traffic prediction accuracy, user privacy, and data security. In the financial realm, federated learning can be integrated with the open banking concept, facilitating decentralized data ownership and promoting the development of a privacy-preserving model in open banking data marketplaces [8]. Additionally, FL finds applications in the manufacturing industry, fostering a collaborative approach to data analysis and model training that results in improved operational efficiency, higher product quality, and compliance with regulations while maintaining data privacy and security [9].

However, FL still faces certain difficulties and challenges. First, there is the Byzantine problem among the participating parties. Traditional FL systems lack an open and fair audit mechanism. In addition, it is difficult to determine how to reasonably calculate participants' contributions and motivate them to maintain reliable training. Second, the participants in FL are distributed across various locations, and their device's computing power and communication environments are heterogeneous. This heterogeneity can cause user dropouts and significantly affect the FL convergence efficiency [10]. To address these challenges, Xie et al. proposed the concept of asynchronous FL [11], which allows each worker to make different training progress in each round, allowing stale local models to be aggregated while updating the global model. The asynchronous method means that the server will update a new global model immediately after receiving the user model update. Hence, there were no problems with idle

or lost workers. Thus, it improved the staleness problem and designed the stale function as a dynamic aggregation parameter, which can reduce the aggregation weight of the aging update, making the algorithm faster than the existing FL algorithm (e.g., FedAvg). However, the asynchronous mode increases the number of global iterations and results in higher communication costs. In addition, because each round only aggregates the characteristics of a single device, it can lead to unstable convergence and perform poorly on non-iid data. Therefore, in order to bridge the gap between the existing problems and solutions, we integrate blockchain and a semi-asynchronous FL approach to design a fair and trusted audit mechanism with reliable communication costs by considering the nature of heterogeneous clients or workers. The main contributions of this study are as follows:

- We design a blockchain-based Auditable Semi-Asynchronous Federated Learning (BASA-FL) system that executes distributed federated learning tasks based on blockchain smart contracts. BASA-FL introduces a public audit and incentive mechanism to solve the problem of Byzantine nodes in trustless federated learning crowdsourcing scenarios.
- Our system supports a semi-asynchronous federated learning mode that balances model exchange frequency and round duration, which can resist device heterogeneity and improve efficiency. This is a novel approach compared to existing research on distributed FL combined with blockchain.
- We provide an efficient multi-metric evaluation method to quantify worker contributions under semi-asynchronous conditions.
- We conduct multiple simulations to verify the performance and reliability of the BASA-FL mechanism through simulation results and analysis.

The remainder of this paper is organized as follows. Section II reviews previous studies in this field. A comprehensive overview of the background knowledge pertinent to the components of FL and blockchain is presented in Section III. Our proposed Blockchain-based Auditable Semi-Asynchronous Federated Learning (BASA-FL) system is introduced in Section IV. The role of smart contracts in the proposed model is explored in Section V. The numerical results are discussed in Section VI, followed by conclusions in Section VII.

II. RELATED WORKS

The traditional FL with a client-server architecture uses a centralized server to aggregate the global model. However, it is prone to single-point failures that lead to system collapse [12]. Moreover, a centralized server lacks a publicly transparent audit mechanism; thus, it cannot ensure fair decisions [13]. To address these problems, Majid et al. [14] proposed a decentralized structure to store the global model and local updates in the blockchain to address single points of failure and track malicious behavior. Other similar

studies [15], [16], [17], [18], [19], [20], [21] also deprecated the central aggregator, where model aggregation occurs in a trusted community in the blockchain. In addition, some studies [15], [16], [17], [19], [20] further used the trusted endorsement of the blockchain and the peer-to-peer transaction mechanism to design an incentive strategy that can effectively encourage the enthusiasm of workers to participate in FL, thereby improving training quality. Thus, blockchain can be used as an ideal complement to FL.

Blockchain has been used differently depending on the application scenario. Lu et al. [16], Feng et al. [22], and Majeed and Hong [14] utilized permissioned or private chain structures to design an industrial IoT paradigm combined with edge computing. Moreover, their work is classified as a blockchain architecture design, which has the advantage of optimizing the system and developing targeted strategies. However, they are not suitable for building public markets [23].

Building an FL trading market using public blockchain is a promising research direction. The emergence of Ethereum has improved the scalability of blockchain technology and marked the beginning of the “blockchain 2.0” era. It has also secured a solid position as infrastructure for various applications. Ethereum’s Turing completeness and flexible smart contracts allow the deployment of federated learning algorithms on the blockchain [24]. Cai et al. [25] proposed two collaborative protocols (2cp) that use Ethereum smart contracts to coordinate multiparty joint machine learning in a trustless environment and distribute fair shares to trainers. Mugunthan et al. [26] proposed using Ethereum smart contracts to build federated learning accountability in BlockFLow. This study aims to prevent user collusion by coordinating smart contracts.

Research on blockchain-based federated learning must address device heterogeneity. In real-world cross-device federated learning (FL) environments, there are significant differences in the networking, power supply, storage, and device computing power between different activity states. This can lead to the loss of some workers in the collaborative learning process and negatively affect efficiency [27]. One solution to this issue is the use of asynchronous algorithms that can fully utilize the intermittent client computing power [28]. Liu et al. [29] and Feng et al. [22] discussed a combined architecture of blockchains and asynchronous federated learning. However, although an asynchronous algorithm can speed up convergence, it also increases the frequency of the model exchange, resulting in higher communication costs and poorer convergence performance under non-iid settings [30]. Semi-asynchronous FL is a hybrid approach that combines synchronous and asynchronous methods. The aggregation server stores the first models to arrive and aggregates them based on a specific time or number. Late-arriving models may either be used in the next training round or discarded depending on their staleness. Semi-asynchronous FL has a lower aggregation frequency than asynchronous FL but a higher frequency than

classical FL [31]. For example, in the semi-synchronous mode designed by Shi et al. [32], aggregation is implemented in the cache model according to a given time window. Ma et al. [30] performed aggregation based on the number M of models arriving at the server successively and designed an algorithm to determine the optimal Wu et al. [33] introduced a cache-based lag tolerance mechanism on the aggregation server and divided all workers into three categories: latest, deprecated, and tolerable. Our study builds on the semi-asynchronous FL algorithm proposed by Ma et al. [30] with the number of arrivals as the aggregate window. We combine this with blockchain technology to design a distributed semi-asynchronous FL protocol coordinated by smart contracts. Our goal is to provide an FL data trading system with a transparent audit mechanism and to optimize the system’s efficiency in heterogeneous equipment environments.

In addition, it is crucial to design a suitable incentive mechanism in FL [34] that can mobilize the enthusiasm of all parties involved and maintain the sustainability of the system [35]. Designing incentive strategies based on worker contributions is one of the most common methods, and the way to evaluate contributions can be divided into two categories: data quality and data quantity [27]. Because FL completes the learning process by passing intermediate parameters under latent data assumptions, reasonably quantifying worker contributions is challenging. The Shapley value [36] is a classic concept in cooperative game theory and one of the methods used to calculate the quality of data. Although the Shapley value is a useful method for quantifying the quality of data in FL, its high calculation cost and exponential time complexity render it difficult to use in scenarios with many participating workers. To evaluate the contribution of the data quantity, Qu et al. [37], Liu et al. [29], and others proved that the size of the dataset can directly affect the final model accuracy. However, supervising workers to reflect on the correct data quantity in FL scenarios is difficult. Although the Trusted Execution Environment (TEE) technology has been proposed as a solution to this problem, its application is limited. Therefore, the use of data quality to evaluate worker contributions is not robust in actual cross-device FL environments. Our purpose is to design a contribution calculation method suitable for the semi-asynchronous mode and distribute rewards according to worker contributions to motivate workers to provide high-quality data and maintain stable training provisions. Furthermore, our system guarantees the transparency and credibility of the audit and distribution processes through blockchain.

III. PRELIMINARY

A. FEDERATED LEARNING

The concept of FL is based on the principles of Distributed Machine Learning (DML), as outlined in a survey by Verbraeken et al. [38]. At its core, both Federated Learning and DML involve collaborative training by exchanging

intermediate parameters such as model parameters or gradients during model training. However, whereas DML focuses on using computer clusters to train large-scale machine learning models and address issues such as computational complexity, data scale, and model size, FL was first proposed by Google [3] with the goal of leveraging data distributed across multiple mobile devices to perform machine learning while addressing user concerns over data privacy.

FL has several distinct characteristics compared to DML. First, the computing nodes in FL are sourced from the recruited users who have full autonomy over their participation, meaning that they can choose to stop computing, communicate, and withdraw from the learning process at any time. Second, the data of the computing nodes in FL are independently generated; thus, they may exhibit different distribution characteristics (non-iid) and have varying data volumes. Third, computing nodes are situated in varying communication environments or use different equipment, resulting in an unstable participation process in joint learning and the risk of disconnection at any point. Finally, as the computing nodes are typically located in different geographical locations, the communication cost is significantly higher than that of DML.

To address the challenges associated with FL, Google proposed an optimization algorithm called FedAvg [3]. The FedAvg algorithm is defined as follows: the number of local data users participating in FL for a terminal server is represented by I , where D represents the dataset of the i th local data user and the volume of the dataset is $D_i = |D_i|$. Then, the objective function can be written as

$$f(\omega) = \sum_{i=1}^I \frac{D_i}{D} F_k(\omega) \quad (1)$$

where $F_k(\omega) = \frac{1}{D_i} \sum_{j \in \mathcal{D}_i} f_j(\omega) = g((x_j, y_j), \omega)$

At the beginning of the T round, the terminal server calculates the current global model parameter ω^T and distributes it to each local data client. The clients use their local data to generate a gradient descent based on the ω^T model, and after E rounds of iterations, the local models ω_i are obtained.

$$\omega \leftarrow \omega - \eta \nabla g(\omega; b) \quad (2)$$

After collecting all the local models ω_i from the clients, the terminal server aggregates ω_i and generates a global model for the next iteration ω^{T+1} based on the following equation:

$$\omega^{T+1} \leftarrow \sum_{i=1}^I \frac{D_i}{D} \omega_i \quad (3)$$

In this representation, $D = \sum_{i \in U} D_i$ denotes the sum of the data volumes of all local data users participating in training, and η represents the learning rate. In each training round, a subset of local data users, represented by $I \ll U$ (where U is the total number of local data users), participated in the model training by utilizing the stochastic gradient descent (SGD) [43] as the local optimization method.

The FedAvg algorithm requires users to iterate multiple rounds of Stochastic Gradient Descent (SGD) locally to update the model parameters before performing average aggregation. It reduces the number of transmissions by increasing the number of local calculations, thus optimizing the communication costs. In addition, in the FedAvg algorithm, the average calculation is used to aggregate all submitted local models in each round, and the aggregated model is unified as the shared starting model for the next iteration of each worker. This method has been shown to exhibit good convergence properties and is robust to non-i.i.d. data distributions.

1) SYNCHRONOUS AND ASYNCHRONOUS FL

FL algorithms can be classified into synchronous, asynchronous, and semi-asynchronous modes, based on different aggregation rules. The FedAvg algorithm is considered synchronous. In this method, the worker's local iteration and corresponding model aggregation must be performed simultaneously. All the workers were trained using the same starting model. After aggregation, the global model is shared as the initial model for each worker to train in the next round. However, this mode can cause problems, such as fast computing workers spending too much time waiting for delayed workers or losing them, especially in heterogeneous equipment environments, which can negatively impact the efficiency of FL, as illustrated in Figure 2.

However, asynchronous algorithms differ from synchronous algorithms in that they allow each worker to be at a different training progress in each round, allowing stale local models to be aggregated while updating the global model [11]. As shown in Figure 2, no limitations exist when aggregation occurs in a pure asynchronous approach. Once the server receives a model uploaded by a worker, it updates the corresponding global model. Therefore, there were no problems with idle or lost workers. However, pure asynchronous mode increases the number of global iterations, resulting in higher communication costs. In addition, because each round only aggregates the characteristics of a single device, it can lead to unstable convergence and perform poorly on non-iid data.

2) SEMI-ASYNCHRONOUS FL

The semi-asynchronous mode combines the benefits of both synchronous and asynchronous methods while avoiding their respective drawbacks. In this approach, a predefined aggregation condition was established (i.e., RoundDuration or UploadNum). Then, the worker models that meet this condition are aggregated synchronously, whereas those that fall outside are queued as stale models for subsequent aggregation rounds. For example, Ma et al. [30] proposed the FedSA algorithm, which is based on the order of arrival of local updates in the server queue and utilizes a specified number M as the threshold for model aggregation. Models uploaded beyond this threshold were queued for aggregation in future rounds and treated as stale versions. Algorithm 1

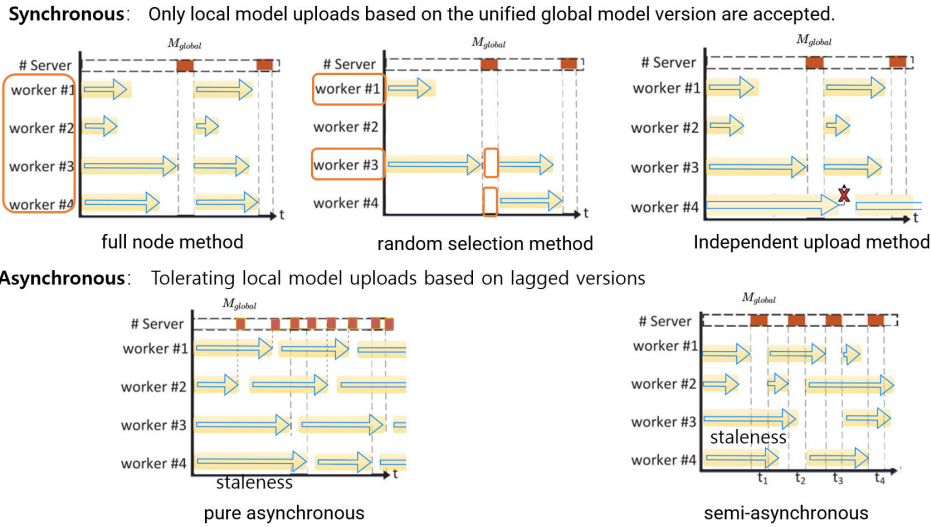


FIGURE 2. Aggregation process of sync, async and semi-async.

describes the detailed process of the semi-synchronous federated optimization (FedSA) approach.

The semi-asynchronous FL mode balances the three key elements of round time, iteration frequency, and convergence. Ma et al. [30] proposed an efficient algorithm for determining the optimal value of M that minimizes the training time, given the communication budget. In addition, the algorithm is extended to dynamic and multi-task scenarios. As a result, the FedSA algorithm has a superior convergence performance compared to pure asynchronous algorithms on non-iid data. Furthermore, compared with synchronous algorithms, its round waiting time depends on the objective transmission status of the worker, and the condition M is dynamically optimized through the algorithm, resulting in faster convergence. Stale models are incorporated into the next round of aggregation, which prevents worker loss and improves the generalization of the final model.

B. BLOCKCHAIN

A blockchain is essentially a distributed shared ledger, database, and storage structure. As shown in Figure 3, a blockchain is a chain structure that is connected by a hash. Taking Bitcoin [39] as an example, the block header stores the version number, hash value of the previous block (parent hash), timestamp, Merkle root, difficulty value, and a random number. The block body contains transaction information. The transactions are stored in a Merkle tree structure [40]. The hash pointer can uniquely identify the block and connect each block so that each piece of data in the block can be traced back to the source, and the timestamp ensures the order of the block. The chain is maintained at all nodes of the blockchain system. It uses a distributed node consensus algorithm to generate and update data and cryptography to ensure the security of data transmission and access. Therefore, the blockchain has the characteristics of decentralization,

Algorithm 1 Semi-Asynchronous Federated Learning(FedSA) [30]

```

1  $T = 0$ 
2 while  $F(\omega^T) - F(\omega^*) > \varepsilon$  do
3   Processing at Each Worker  $u_i$ 
4   if Receive  $\omega^T$  from the server then
5     Update local model  $\omega_i^\tau \leftarrow \omega_i^T - \eta \nabla g(\omega, b)$ 
6      $\tau = T$ 
7     Upload local model  $\omega_i^\tau$ 
8   Processing at parameter server
9    $U^T = \emptyset$ 
10  while  $|U^T| < M$  do
11    Receive local model  $\omega_i^\tau$  from worker  $u_i$ 
12     $U^T = U^T \cup \{u_i\}$ 
13  Update global model
14   $\omega^{T+1} \leftarrow (1 - \sum_{u_i \in U^T} \frac{D_i}{D})\omega^T + \sum_{u_i \in U^T} \frac{D_i}{D}\omega_i^\tau$ 
15  for each  $u_i \in U$  do
16    if  $u_i \in U^T$  or  $\tau_i^T > \tau^0$  then
17      Distribute update model  $\omega$  and learning
18      rate  $\eta_i$ 
19   $T = T + 1$ 
20 return the final global model  $\omega^T$ 

```

immutability, traceability, collective maintenance, openness, and transparency [41].

The decentralized nature of blockchain technology allows it to operate without the need for a central authority, instead relying on consensus among the participating nodes. This consensus mechanisms play a crucial role in ensuring that all nodes in the network agree on the validity of transactions. Different blockchains employ various consensus mechanisms, such as Proof of Work (used in Bitcoin) and Proof of Stake, to validate transactions and add new blocks to the chain.

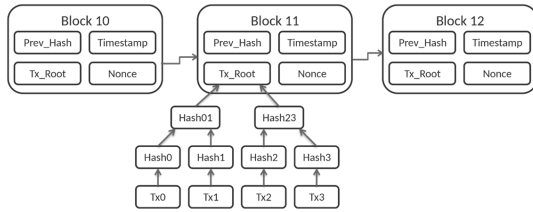


FIGURE 3. Blockchain structure.

Hence, it maintains transparency of transactions, where All transactions are recorded on a public ledger that anyone can access and verify [42]. However, while the transaction details are transparent, the participants' identities can be kept pseudonymous, enhancing privacy. Blockchain technology encompasses various types of networks, each tailored to specific needs. Public blockchains, like Bitcoin, offer open access to anyone without requiring permission, ensuring high decentralization through consensus mechanisms. On the other hand, private blockchains restrict access to authorized participants, allowing for controlled decentralization, and are often managed by a central authority or consortium. Consortium blockchains strike a balance, being shared among a select group of known nodes or organizations, providing a semi-open environment with partial decentralization [43].

Ethereum [44] differs from Bitcoin and can be described as a distributed-state machine. Ethereum Virtual Machine (EVM) is a crucial part of the Ethereum architecture, which acts as a “world computer,” provides an operating environment in a decentralized system, and has the characteristics of “Turing completeness.” A smart contract is a code that can be compiled and executed in EVM. A smart contract refers to code compiled by the developer and executed in the EVM. When a smart contract is deployed, its code is packaged as a contract-type transaction maintained in a blockchain that cannot be altered. The concept of ‘contract’ is regarded as “autonomous agents” in the execution environment of Ethereum. It has an independent account and automatically executes a piece of logic after receiving a transaction. The piece of logic code can interact with the “state” information stored in the blockchain and implement state transitions. Thus, developers can design applications in a decentralized environment through smart contracts, which improves the scalability of the blockchain.

IV. PROPOSED SYSTEM

A. SYSTEM OVERVIEW

This section provides an overview of our proposed Blockchain-based Auditable Semi-Asynchronous Federated Learning (BASA-FL) system, which serves as a trust-agnostic platform for participation in the FL crowdsourcing marketplace. Our system leverages blockchain smart contracts to facilitate the coordination and endorsement of the FL process, thereby establishing trust among all

the parties involved. Moreover, we address the issue of heterogeneous participating nodes by proposing a system design that integrates the semi-asynchronous mode, which balances the tradeoff between communication cost and convergence rate.

1) ROLE

Our system comprises three participants: requesters, workers, and validators.

- The requester is defined as an institution or individual who wishes to complete the training of a specific model. They require an AI model for a specific task but lack the necessary data to train the model. As such, they initiate the FL task and provide remuneration for its completion.
- The workers are defined as data owners. They possess data that meets the requester's requirements and has a certain degree of privacy. By participating in FL, they aid the requester in fulfilling the specified requirements and receive corresponding compensation from the requester.
- The validators are defined as trusted third-party audit groups elected by the blockchain. They evaluate and audit the accuracy of the local model submitted by each worker during the FL training process, flag malicious submitters, and ultimately submit a form to evaluate worker scores and receiving corresponding compensation from the requester.

2) ADVERSARIAL MODEL

We identify three types of malicious actors in FL: cheating requesters, malicious workers, and infected validators. To address these malicious behaviors, we designed the following solutions:

a: CHEATING REQUESTER

- The requester breaches the contract when paying the task remuneration. To mitigate this, we use smart contracts to automatically distribute profits after satisfying the designed logic conditions.
- The requester provides false evaluation results for each worker for profit. To address this, we entrust model quality evaluation to a third-party VRF committee and use smart contracts to endorse the entire process.
- The requester server has a risk of a single point of failure. To mitigate this, we have designed a distributed FL protocol coordinated by smart contracts, which does not require a centralized aggregator and allows each worker to aggregate models locally.

b: MALICIOUS WORKER

- Workers are malicious or lazy to train. To address this, we use the blockchain to provide a traceable environment and use a third-party committee to review the model quality of worker updates during the training process. For low-quality updates, the verifier compares workers based on τ rounds, excludes and penalizes abnormal workers whose accuracy is extreme outliers,

and assigns lower contribution scores to low-quality workers within the range. For high-latency updates, the smart contract endorses and manages worker submission time, submission version, and other information and designs statistical data as indicators for evaluating worker contribution.

c: INFECTED VALIDATORS

- Sybil attack of the validator, the attacker joins the task as the validator and the worker simultaneously by forging an alias. To mitigate this, we chose to use Chainlink VRF nonces for validator selection, which provides an integrity-verifiable source of randomness for smart contracts to ensure fairness.
- The verifier takes the score calculated by others and free rides to cheat the reward. We prevent this behavior with an encrypted commit scheme: Divided into two phases, commit and reveal [45].

3) NOTATION AND AGGREGATION ALGORITHM DESCRIPTION

The protocol adopts a semi-asynchronous FL mode, and this subsection describes the FL aggregation algorithms in detail.

TABLE 1. Describes the notation.

Notation	Description
T	Number of global epochs
τ_i^n	When worker i submits at the n th time, the version number of the global model initialized by the training
i	Worker's number
U	The set of workers $\{u_1, u_2, \dots, u_i\}$
U^T	The workers participating the global updating in round T , $\{u_1^T, u_2^T, \dots, u_{ U^T }^T\} \in U^T$
\mathcal{D}_i	The local dataset on worker u_i
D_i	The size of \mathcal{D}_i
M	The specified number of the aggregation per round
t	The maximum waiting time of blocks
ω^T	The global model in round T
ω_i^τ	The global model by worker u_i corresponds to the τ stale version
acc_i^τ	The model accuracy by ω_i^τ corresponds to the τ stale version
$s_i^n(T - \tau)$	Staleness function
R_i^n	The softmaxatio of the worker i base τ round
M_v	Submit Evaluation Manifests from validator V_v
CID_i^τ	ω_i^τ is stored in IPFS corresponding to the generated unique content identifier CID
$enc(k, P_k^i)$	When security data exchanges, encrypt the AES key k used for the data with the receiver's RSA public key
$\mathbb{H}^{D_{eval}, k}$	Hash of IPFS data, also regarded as CID

The semi-synchronous learning process is illustrated in Figure 4. Referring to the method proposed by the FedSA [30], each round aggregates a fixed number of M model parameters based on the arrival order. Furthermore, arriving versions other than M are queued in the next round of M as stale versions.

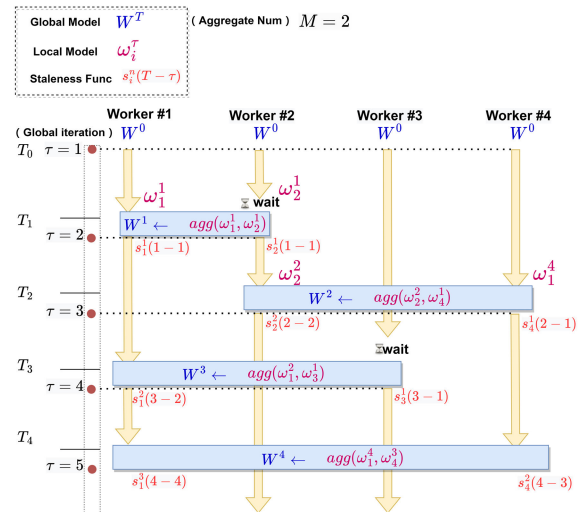


FIGURE 4. Semiasynchronous federated learning process.

FedAvg has exceptional advantages in terms of non-*iid* resistance and convergence accuracy, and for its asynchronous mode, it has advantages in terms of worker computing power utilization efficiency. The FedSA algorithm incorporates the benefits of both synchronous and asynchronous modes and can be optimized to an ideal state. For example, in the traditional client-server structure of FL, the algorithm is defined as follows:

The server assigns a machine learning task. In accordance with the task description, data provider workers willing to participate were called upon to start FL. The initial model parameters, denoted by ω^0 , are distributed to each worker, and the global round is set to T_0 . The role of each worker was defined as follows:

$$\omega_i^\tau \leftarrow \omega^T - \eta_i \nabla g(\omega^T; d_{ij}) \quad (4)$$

Minimize the Loss function with the local data $d(i, j)$ to obtain an optimal vector and go through multiple local iterations.

Subsequently, upload the local model update ω_i^τ and mark the initial model version τ . This version number τ is described as the label of the global model version used by the worker to start local model training. (Because the synchronization process forces all nodes to be trained on the same version of the global model, there is no difference in τ . However, the asynchronous mode is different, allowing the aggregation of the uploaded model training based on an older global version of the model.)

The server aggregates M models according to the arrival order of local parameter submissions. Calculated as follows:

$$\omega^T \leftarrow \left(1 - \sum_{u_i \in U^T} \frac{D_i}{D} \right) \omega^{T-1} + \sum_{u_i \in U^T} \frac{D_i}{D} \omega_i^T \quad (5)$$

After averaging the local model parameters, the global model parameter ω^T was updated, and the global model of this round, version T, was returned to the corresponding

M workers. The worker who receives the global model immediately starts the next iteration. The two processes of local training and global aggregation were repeated until the global model converged to the preset conditions.

The algorithm we use allows for stale version models when synchronizing averaging (if models submitted after M are considered stale versions and placed at the front of the next round); thus, it can be generalized as a mixed mode about M -conditional synchronization in asynchronous.

As shown in Figure 4, there are model updates from different τ versions of the $\text{agg}()$ function. Therefore, the concept of staleness refers to the interval $(T - \tau)$ of the global and local versions during aggregation.

Numerous studies have proven that large staleness can lead to training glitches and affect convergence performance. Xie et al. [11] proposed defining a staleness function to dynamically reduce the aggregation weight of stale models.

Contant : $s(T - \tau) = 1$

Polynomial : $s_a(T - \tau) = (T - \tau + 1)^{-a_t}$

$$\text{Hinge} : s_{a,b}(T - \tau) = \begin{cases} 1 & \text{if } T - \tau \leq b \\ \frac{1}{a(T - \tau) + 1} & \text{otherwise} \end{cases} \quad (6)$$

Drawing inspiration from this, we employ the staleness function to count the staleness caused by workers as a metric for evaluating worker contributions. In addition, owing to the negative effects of staleness on training, we designed a staleness threshold function in the smart contract-based aggregation protocol to exclude model submissions beyond an acceptable range.

B. SYSTEM ARCHITECTURE

We constructed a data-trading platform based on FL, as shown in Figure 5. The requester provides task details and deploys a task contract. Workers, who are data providers, interact with smart contracts to submit local models trained using their own data. Validators act as third parties to audit the quality of the worker-submitted models and calculate the contribution score of each worker. Thus, the requester, workers, and validators jointly complete an FL crowdsourcing task under the coordination of smart contracts. Finally, the requester obtains the AI model after FL, the worker receives a reward share based on their contribution score, and validators receive work rewards. The interaction flow of the protocol is illustrated in Figure 6.

The requester deploys the contract to the Ethereum blockchain and specifies the task-related information, including (task description, target performance/target rounds, number of workers and validators, and payment amount). The federated task was then divided into five steps and connected the steps by the EVM events mechanism.

Step 1: After the Requester deploys the contract, it enters the recruitment stage. Interested workers join the task by invoking the contract's $\text{signUp}()$ function and submitting

a deposit to the contract. Simultaneously, using the verifiable random source provided by the Chainlink VRF [46], several nodes are randomly selected as the validators group among the willing nodes. The selected validators must also submit a deposit to a smart contract to prevent malicious audit behavior.

Step 2: After the recruitment is completed, an event notification is generated, and the initialization stage begins. The requester calls the $\text{setGenesis}()$ function, passing the initialization values, which include these status parameters: the genesis model's CID, the number of aggregation models per round, the maximum time per round, the staleness threshold, the key, and CID about securely sharing validation dataset.

Step 3: The training state begins. Each worker queries the initial model parameters through $\text{ViewGenesis}()$ and then uses their own data to train the model parameters. The process is as follows:

$$\omega_i^\tau \leftarrow \omega^T - \eta_i \nabla g(\omega^T; d_{i,j}) \quad (7)$$

After the worker completes training, it stores the trained model parameters on IPFS and calls the $\text{addModelUpdate}()$ method to submit the CID value of the model. Each call triggers a round calculation function to mark the version information. When the number of submissions M or the maximum round-time condition is reached in the contract, the time parameter is updated, leading to the next round. Worker nodes that successfully join this endorsement round must obtain a global model of the corresponding round.

The contract provides the $\text{ViewUpdates}()$ interface, which allows workers to query the submit set $\{[CID_1, \dots, CID_i]_T, [CID_1, \dots, CID_i]_{T-1}, \dots\}$ of all rounds within the authority. Authority refers to the latest round that does not exceed one's own participation. Then, based on their own degree of lag, workers can find the set of required rounds and calculate and update them independently.

$$\omega^T \leftarrow \left(1 - \sum_{u_i \in U^T} \frac{D_i}{D}\right) \omega^{T-1} + \sum_{u_i \in U^T} \frac{D_i}{D} \omega_i^T \quad (8)$$

Thus, the $\text{addModelUpdate}()$ function is used to upload the model, and the $\text{viewUpdates}()$ function is used to view the submitted set and aggregate the model independently. These two processes are repeated to continue the FL task. Simultaneously, a third-party audit is added to the FL process. The validators calculated the accuracy of all the models and audited outlier models. Here, we exclude outlier accuracies grouped by τ . (The box plot method was used to exclude abnormal models [47]). When validators find outliers, they mark the outlier worker as malicious in the contract. Subsequently, the worker's submission is rejected and the deposit is deducted as punishment.

Step 4: The above process will continue to iterate. The smart contract establishes the current round by adjusting and calculating the relationship of the time parameters (including the initial timestamp , maxNumUpdates ,

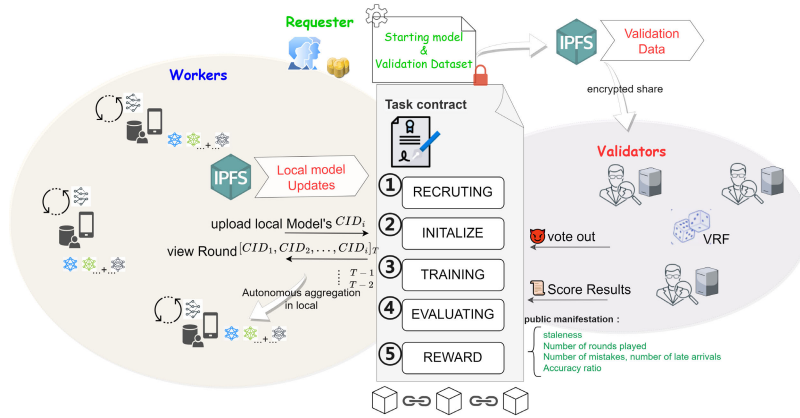


FIGURE 5. BASA-FL system architecture.

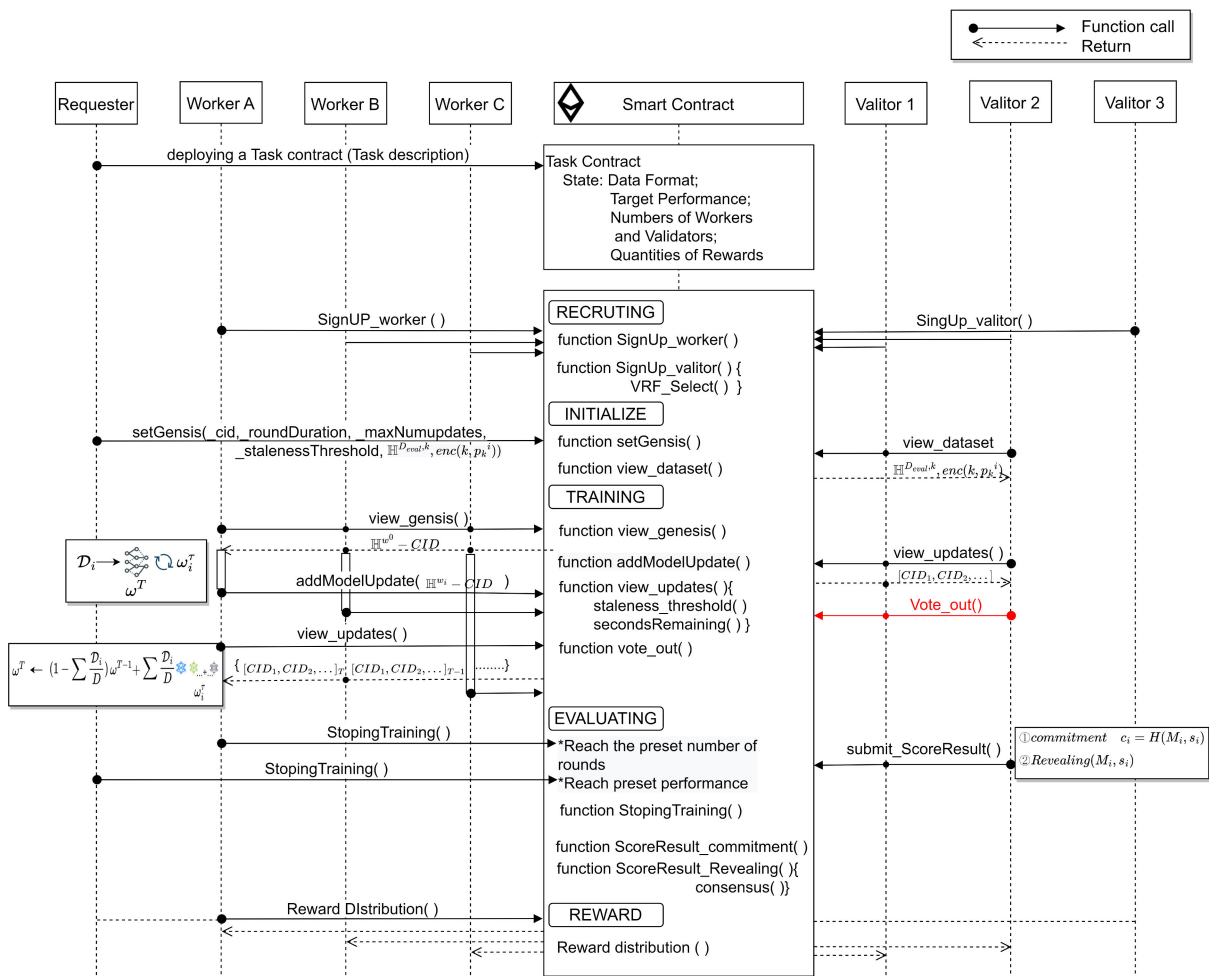


FIGURE 6. TBASA-FL workflow.

roundDuration, timeSkipped, etc.). When the global round reaches the preset number, some workers can actively call the stopTraining() function to stop the iterative learning. Alternatively, if the accuracy of the output model reaches the requester’s expectations, the requester

can actively call the stopTraining() function to stop.

After the training was stopped, the process of validators scoring workers and agreeing on the results began. Finally, the scores were used to distribute the rewards of each node.

The method for calculating the contribution score is explained in detail in Chapter 5. We outsource the evaluation work to the validators, as proposed by Lu et al. [45]. They proved that under the incentive method, the selected validators would submit the correct scoring results because malicious behavior would result in a loss of deposits. Additionally, to prevent validators from peeking at the results submitted by others and free-riding for profit, the methods of commit and reveal were used.

Step 5: Finally, the consensus scoring result is obtained, and the reward is distributed to each worker according to the proportion of the score.

V. SMART CONTRACT-BASED BASA-FL

Traditional FL [3] controls the model exchange through a central aggregator, which carries the risk of a single point of failure and does not provide an auditing mechanism to guarantee trustworthiness [22], [48]. Therefore, we combined blockchain technology to design a distributed FL protocol based on smart contracts, providing model integrity and identity proof relying on cryptography. Smart contracts are self-executing agreements encoded with specific terms and conditions, designed to operate on blockchain technology. These contracts automatically execute and enforce their pre-defined rules once triggering conditions are met, eliminating the need for intermediaries. Initially conceptualized by Nick Szabo [49], they have gained prominence with the advent of platforms like Ethereum [44]. Furthermore, smart contracts offer transparency, security, and efficiency. They can handle various transactions, from asset transfers to complex multi-party agreements. Additionally, a smart contract endorses the intermediate FL process to form an open audit mechanism (inspired by [30] and [50]).

A. STATUS PARAMETER

To effectively coordinate the FL process and record the behavior of the participating parties, we designed a set of status parameters, as outlined in Table 2. These parameters include the number of model aggregations per round M , the maximum duration of each round, the initial value of the model's CID , and the creation timestamp. The requester initializes these values, and the smart contract uses them to govern the coordination of the FL process. Furthermore, we leverage the mapping data type within a smart contract to provide high-performance query functionality. To this end, we established mappings between participant addresses and the CID and the round number of their submissions, thus enabling the recording and querying of participants' behavior. By designing appropriate contract functions, we can facilitate serverless distributed aggregation, access control, and traceability of participant behavior.

B. FUNCTIONAL DESIGN

The entire FL processes are supported by three main functions: `setGenesis()`, `addModeupdate()`, and `viewUpdates()`. The detailed design of the contract

TABLE 2. Status Parameter.

	Value Type	
<code>Request</code>	address	- The requester address
<code>roundDuration</code>	unit256	- The maximum time per round
<code>maxNumUpdates</code>	unit256	- The aggregate number M per round
<code>genesis</code>	bytes32	- CID of initial model
<code>genesisTimestamp</code>	unit256	- Timestamp of initial upload
<code>stalenessThreshold</code>	unit256	- Expected staleness threshold
<code>timeSkipped</code>	unit256	- Remaining time accumulator
Mapping		
<code>updatesInRound</code>	Round_num	→ [cid_1,cid_2,...]
<code>updateRound</code>	cid_	→ Round_num
<code>updatesFromAddress</code>	address	→ [cid_1,cid2,...]
<code>uploadRound</code>	address	→ [Round1,...,Round4...]
<code>updateStaleness</code>	address	→ [staleness1,staleness2,...]
<code>punish</code>	address	→ number of punishments

function is shown in Algorithms 2. Federation training begins after the requester calls `setGenesis()` to initialize the parameters. Subsequently, two processes are conducted: updating the local model and aggregating the global model. In the local model-update phase, the `addModeupdate()` function provides the interface of the submitting model to the worker. In contrast to the traditional server aggregation method, we allow workers to independently query the local model submitted in each round through the `viewUpdates()` interface to complete autonomous aggregation in the global model aggregation phase. Moreover, we designed the corresponding access control according to the processing logic of the semi-asynchronous FL. In this sense, the condition for changing the global round in a semi-asynchronous FL depends on whether the number of workers received meets `maxNumUpdates` or the time reaches `roundDuration`. Therefore, by designing the adjustment and calculation of time parameters, such as `timeElapsed`, `timeSkipped`, and `genesisTimestamp`, the round at each moment can be determined to control the FL process.

We used the status parameter to record the corresponding version of the model submitted by each worker, evaluated the staleness through the dissimilarity between two adjacent submissions, and designed the staleness threshold function. In addition, when the worker calls `addModelupdate()` but exceeds the staleness threshold, it is recorded as a negative contribution.

C. CALLING PROCESS

The interactive process of semi-asynchronous FL based on smart contracts blockchain is shown in Algorithm 3. The requester initializes the parameters based on the circumstances required to start the training task, obtains a satisfactory global model, and automatically distributes the rewards after stopping the training process by calling `stopTraining()`. The workers then run the AI model calculations in parallel and repeat the two steps of local model update and global model aggregation. Here, the aggregation

Algorithm 2 Semi-Asynchronous Federated Learning Smart Contract (SAFL-SC)

Statuses Variable:
address: Requester;
uint: totalRound; roundDuration; maxNumUpdates; genesisTimestamp; stalenessThreshold; timeSkipped;
bytes: genesis;
mapping(address => bytes[]): updatesFromAddress;
mapping(uint => bytes[]): updatesInRound;
mapping(bytes => uint): updateRound;
mapping(address => uint[]): uploadRound; updateStaleness;
mapping(address => uint): punish;

```

1 func setGenesis(_cid, _roundDuration, _maxNumUpdates, _stalenessThreshold):
2   if msg.sender is not Requester or genesis is not 0 then
3     send a msg “Not the registered evaluator” or “Genesis has already been set”
4     return Require Failed;
5   Initialize : genesis, roundDuration, maxNumUpdates, stalenessThreshold
6   genesisTimestamp ← now
7 func currentRound( ):
8   timeElapsed ← timeSkipped + now - genesisTimestamp;
9   round ← 1 + (timeElapsed / roundDuration);
10  return round;
11 func secondRemaining():
12  timeElapsed ← timeSkipped + now - genesisTimestamp;
13  remaining ← roundDuration - (timeElapsed % roundDuration);
14  return remaining
15 func addModeupdate(_cid):
16  _round ← currentRound();
17  staleness ← _round - uploadRound[_address][-1] - 1;
18  storage updateStaleness[_address].push(staleness);
19  if updateRound [ cid in updateFromAddress ] == _round then
20    send.msg “Already added an update for this round”
21    return Require Failed;
22  if staleness >= stalenessThreshold then
23    storage punish[msg.sender] += 1;
24    storage uploadRound[msg.sender].push(_round);
25    send.msg “exceed the staleness threshold, Please get the latest round of updates to start over”
26    return Require Failed;
27  storage updatesInRound[_round].push(_cid);
28  storage updatesFromAddress[msg.sender].push(_cid);
29  storage updateRound[_cid] ← _round;
30  storage uploadRound[msg.sender].push(_round);
31  if maxNumUpdates > 0 && updatesInRound[_round].length >= maxNumUpdates then
32    storage timeSkipped += secondsRemaining();
33 func viewUpdates(_round):
34  if _round > uploadRound[_address][-1]: then
35    send.msg “the viewed round Exceeded your permission”
36    return Require Failed;
37  return updatesInRound[_round];
38 func checkJoinRound():
39  return (uploadRound[msg.sender][-1]);
40 func stopTraining():
41  if msg.address == requester or currentRound >= totalRound then
42    return Distribute rewards to workers;

```

Algorithm 3 Calling Process. The Requester Defines TotalRound, roundDuration, maxNumUpdates, stalenessThreshold, ω^0, α According to Task Requirements and Recruitment Situation

```

1 Requester executes :
2   sc.setGenesis(cid( $w^0$ ),roundDuration,
3   maxNumUpdates,stalenessThreshold);
4   get global Model;
5   sc.stopTraining();
6
7 Client executes // parallel:
8    $\mathcal{W} \leftarrow sc.genesis$ ;
9    $\tau = 0$ ;
10  while  $T \geq TotalRound$  or Task interruption do
11     $T \leftarrow sc.currentRound()$ ;
12    update local model  $\omega^\tau \leftarrow \mathcal{W} - \eta \nabla g(\omega, b)$ 
13    sc.addModelUpate(cid( $\omega^\tau$ ))
14    if Successful Transactions or Receive “exceed
        staleness threshold...” then
15       $R \leftarrow sc.checkJoinRound$ ;
16      for  $i$  in  $[\tau, R]$  do
17         $\mathcal{W}^i[ ] \leftarrow sc.viewUpdates(i)$ 
18        aggregate
19         $\mathcal{W} \leftarrow (1 - \alpha)\mathcal{W} + \alpha \sum \mathcal{W}^i[ ]$ 
20       $\tau \leftarrow R$ ;
21    else
22      wait Round change and upload again;
23      addModelUpate(cid( $\omega^\tau$ );
24      for  $i$  in  $[\tau, R]$  do
25         $\mathcal{W}^i[ ] \leftarrow sc.viewUpdates(i)$ 
26        aggregate
27         $\mathcal{W} \leftarrow (1 - \alpha)\mathcal{W} + \alpha \sum \mathcal{W}^i[ ]$ 
28       $\tau \leftarrow R$ ;
29    sc.stopTraining()

```

process is different from the traditional aggregator method (i.e., FedAvg [3]) because it implements local autonomous aggregation.

In semi-asynchronous mode [33], the aggregation progress of each worker and the missing updates are different. In our BASA-FL, we use `checkJoinRound()` and `viewUpdates()` to return the individual progress and query the updates of each missing round, respectively. Thus, the model training and aggregation processes are repeated until the calculated value is returned by the contract and the `currentRound()` function satisfies the requester’s preset requirements of `totalRound`. Consequently, the workers can call `stopTraining()` and obtain shared rewards based on their contributions.

Figure 7 illustrates the blockchain-based FL in synchronous and semi-asynchronous modes to reflect the advantages of BASA FL. Figure 7 shows that when the round aggregation number M is three, the synchronous mode

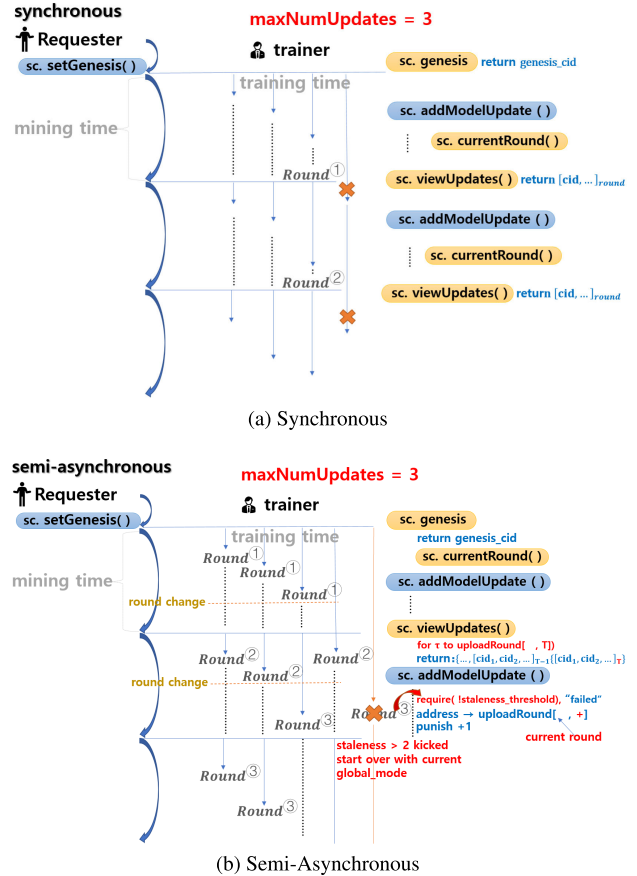


FIGURE 7. Calling process.

(i.e., represented by Figure 7 (a)) does not receive stale model updates. In contrast, the semi-asynchronous mode (i.e., represented by Figure 7 (b)) accepts the stale model updates and counts them in the next round. Therefore, the semi-asynchronous mode does not waste any worker computing resources and enhances efficiency under heterogeneous conditions.

VI. AUDIT AND EVALUATION IN SEMI-FL

A. τ -BASED AUDIT

Our system uses a third-party validator to audit the models submitted by workers and locate the Byzantine nodes to ensure training quality. The general method for defending the Byzantine nodes is to compare the dissimilarities between the submitted models in the same round to discover outliers. However, we consider that the global round in semi-asynchronous mode contains an obsolete model of asynchronous aggregation that is far from the current model. Hence, it is difficult to distinguish between stale and malicious models in each round T . Therefore, we use τ rounds as the reference unit to compare workers under the same starting model with each other, calculate their accuracy, and use boxplots to locate outliers beyond Q1-1.5QR (the details are described in Figure 10 in Section VII). The

corresponding workers are eliminated from the smart contract and their deposits are penalized.

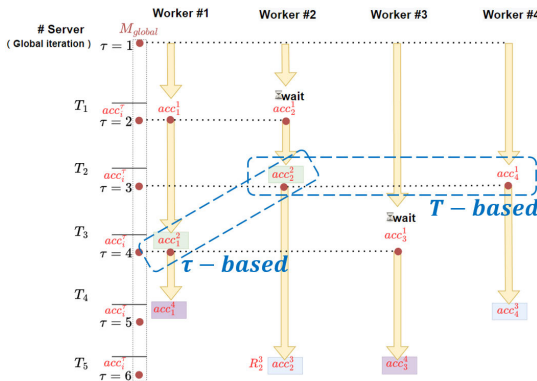


FIGURE 8. τ -based audits.

B. CONTRIBUTION QUANTIFICATION

In the data-trading platform, we need to reasonably quantify the contribution of workers and distribute the rewards paid by requesters to workers according to the contribution ratio. The costs paid by the workers are divided into data resources and computing power supplies. However, in FL, raw data are not revealed for privacy protection, and only the intermediate model parameters are shared instead of the original data. The parameters submitted by the worker are repeatedly aggregated with other parameters for convergence; thus, quantifying the worker’s contribution is another challenge.

In contrast, synchronous systems evaluate the worker’s contribution by comparing the model accuracy between workers in units of rounds or by calculating the marginal contribution (i.e., Shapley value) of workers in the current round. However, we argue that these contribution evaluations are unsuitable for semi-asynchronous systems because they actively disrupt the balance of participation among workers and shorten the round time. Hence, workers who frequently submit to with imbalance manner inherently have a large share, deviating from the requester to appeal for high-quality data to train the model for the primary purpose.

Therefore, we designed a quantitative method to balance the quality and computing power contribution using the following three metric functions for semi-asynchronous FL:

1) QUALITY METRICS FUNCTIONS

$$qu = \text{Softmax} \left(\frac{e^{acc_i}}{\sum_{u=1}^U e^{acc_u}} \right) \quad (9)$$

2) STALENESS METRICS FUNCTIONS

$$st = \log_a \left(\frac{a}{(T - \tau) + 1} \right) \quad (10)$$

$a \rightarrow \text{stalenessThreshold}$

3) PARTICIPATION METRICS FUNCTIONS

$$pa = \log_{Round}(Participation) \quad (11)$$

Thus, when each worker completes the task, given three metrics: $X_1 \leftarrow \overline{qu}$, $X_2 \leftarrow \overline{st}$, $X_3 \leftarrow pa$, the combined score for each worker was calculated using uniform metric weighting.

VII. RESULTS AND DISCUSSION

We configured a semi-synchronous FL simulation environment¹ and designed the following simulations to verify our ideas.

- 1) In the worker’s resource (i.e., computation capability) heterogeneity setting, we compare the aggregation efficiency of synchronous, asynchronous, and semi-asynchronous modes.
- 2) The effect of τ – based auditing method.
- 3) Fairness of contribution quantification method.

We formulated a simulation model for the semi-asynchronous FL process that considers the number of participants, their submission rate, size of the dataset, number of semi-asynchronous aggregations (M), staleness threshold, and termination criteria, such as the specified number of rounds. Our simulation model was designed to mimic the real-world execution process of semi-asynchronous FL, utilizing preset parameters to determine the queueing and performing AI model training and aggregation calculations accordingly.

We chose the cifar10 dataset to complete the image classification task using the RestNet-18 model. Torchvision’s built-in resnet18 model consists of a 7×7 downsampling convolution, a max pooling layer, eight basic blocks, a global pooling layer, and a fully connected layer. The cifar10 dataset uses 50,000 images for training and 10,000 images for testing. The learning rate was configured as 0.01, the batch size was set to 32, and the local epochs were set to 3. The data distribution was classified into two settings: IID and non-IID. In the IID setting, each client has the same data distribution; meanwhile, the non-iid setting uses ‘Hetero Dirichlet,’ where the label distribution assigned to each client is offset. For instance, each worker was divided into 3 to 4 categories of 10 with different proportions. Simultaneously, the number of samples assigned to workers was disrupted.

A. SIMULATION 1: COMPARING SYNCHRONOUS, ASYNCHRONOUS, AND SEMI-ASYNCHRONOUS MODE

Synchronous, asynchronous, and semi-asynchronous training effects were simulated in iid and non-iid configurations.

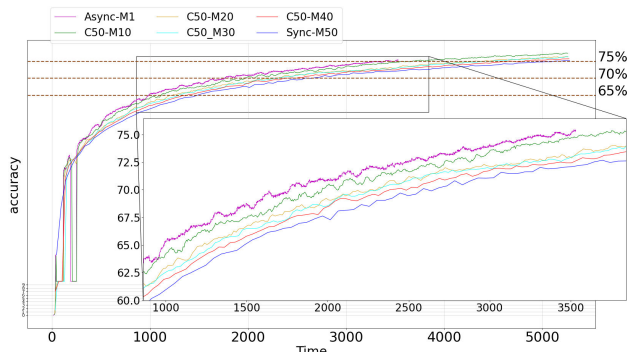
The task configuration is presented in Table 3. We set up 50 workers to participate in these tasks. Moreover, the worker’s speed was preset to a normal distribution with a mean of 30 s and a standard deviation of 10. The data were distributed to 50 workers in equal numbers but divided into

¹<https://github.com/BASAFL/BASA-FL>

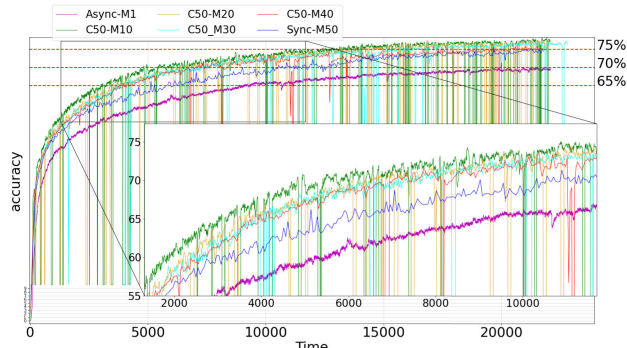
TABLE 3. Simulation 1 configuration.

Setting	
Number of clients:	50
M:	Async-1/ 10/ 20/ 30/ 40/ Sync-50
Speed:	Normalvarizte(30s,10)
Dataset Size	Similar(5000/50)

two distribution methods: iid and non-iid. We adjusted the round aggregation number M for comparison. When M is 50, it is regarded as synchronous, and when M is 1, it is considered asynchronous.



(a) iid



(b) Non-iid

FIGURE 9. accuracy vs. time (iid & Non-iid).

The results of the IID and non-IID configurations, as presented in Table 4(a) and Figure 9(a), as well as Table 4(b) and Figure 9(b), respectively, demonstrate that as the value of M (which represents the level of asynchronous) decreases, the time required to reach the desired accuracy decreases, but the number of iterations required increases. In practical FL scenarios, communication costs are high, and frequent exchanges of models can exacerbate system performance. Additionally, in the non-IID configuration, a decrease in the value of M leads to a more unstable convergence, making it challenging to achieve 75% accuracy in the *Async*– $M1$ scenario. However, while the synchronous mode offers stable convergence, it tends to be time-consuming

TABLE 4. Time consumption and number of transports.

Accuracy	65%		70%		75%	
	Time	Num of Tx	Time	Num of Tx	Time	Num of Tx
Sync	1536s	31*50	2640s	54*50	5184s	107*50
M40	1431s	41*40	2417s	70*40	4695s	137*40
M30	1330s	54*30	2338s	96*30	4450s	184*30
M20	1255s	82*20	2275s	150*20	4330s	287*20
M10	1129s	171*10	1975s	302*10	3630s	558*10
Async	988s	1656*1	1763s	2982*1	3315s	5631*1

(a) iid

Accuracy	65%		70%		75%	
	Time	Num of Tx	Time	Num of Tx	Time	Num of Tx
Sync	5184s	107*50	9072s	188*50	19728s	410*50
M40	3981s	116*40	7041s	206*40	16697s	490*40
M30	3802s	157*30	6778s	281*30	14026s	583*30
M20	3505s	232*20	6400s	425*20	14080s	937*20
M10	3025s	464*10	4830s	743*10	10875s	1674*10
Async	8860s	5498*1	20907s	12969*1	-	-

(b) Non-iid

owing to delays in each round. Therefore, adjusting the value of M in the semi-asynchronous mode allows for a balance between the round time, iteration frequency, and convergence.

B. SIMULATION 2: COMPARING THE T – BASED AND τ – BASED AUDITING METHODS

We demonstrate that the τ -based audit method in the semi-asynchronous mode is superior in terms of accuracy and effectiveness compared to the traditional T -based audit method. In addition, we demonstrated that using this method to eliminate Byzantine nodes significantly improves the quality of the final model.

TABLE 5. Simulation 2 configuration.

Setting	
Number of clients:	26
M:	20
Condition:	Run 20 global iterations
Speed:	Normal variate(32s,3)
Dataset Quality	workerA \rightarrow poisoning attack

The task configuration is presented in Table 5. All 26 workers participated in the task, and the submission speed of the workers was preset with a normal distribution, with a mean of 32 s and a standard deviation of 3. Data were distributed to 26 workers in equal numbers; however, the samples were poisoned by worker A. We compared the T -based outlier audit and the τ -based outlier audit and found that, as expected, the old normal model was mixed in the T round, and the model accuracy of worker A in the $T2$ and $T3$ rounds did not exceed $Q1 - 1.5QR$ truncation point. Therefore, it is proven that the τ -based auditing method we adopted can detect malicious worker A, and Figure 10(a)

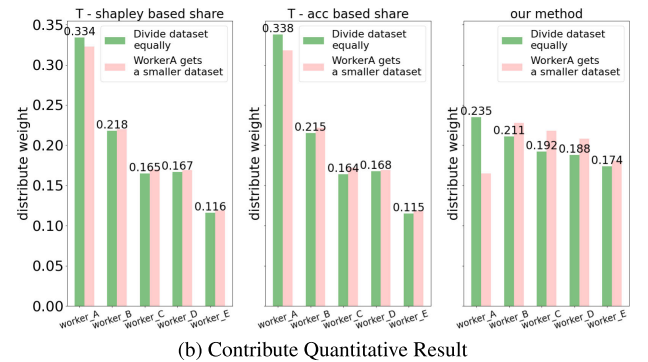
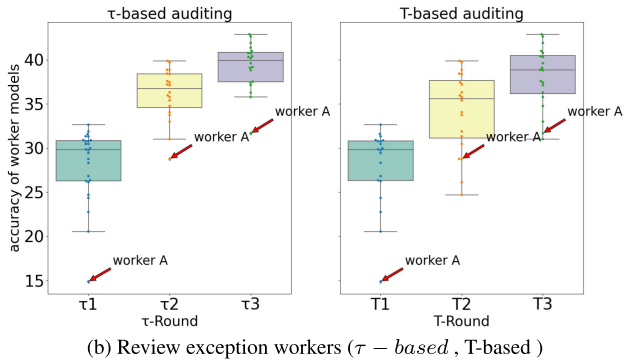
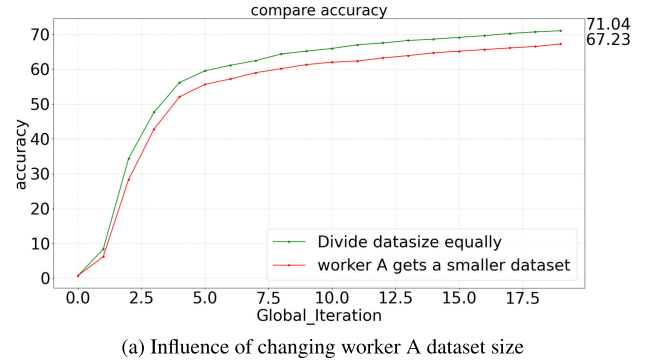
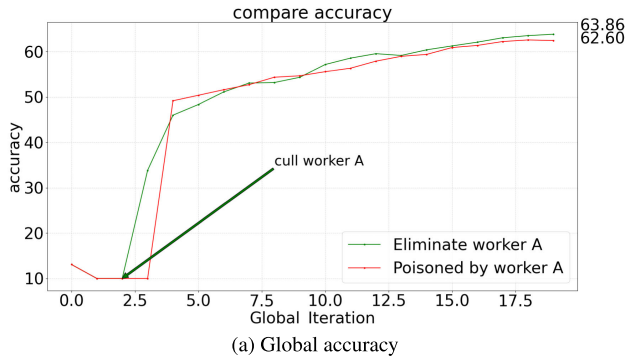


FIGURE 10. Influence of exclude malicious worker A.

shows that after excluding the model of worker A, the final accuracy increases significantly.

C. SIMULATION 3: CONTRIBUTION QUANTIFICATION METHOD

We set two extreme assumptions to compare the rationality of the calculation method for extreme cases.

Assumption 1: The most frequently updated worker A has a small dataset size.

Assumption 2: The most slowly updated worker E has a large dataset size.

TABLE 6. Simulation 3 configure.

Assumption 1	Normal Group	Comparison Group
Number of Client	5	5
Condition	3	3
staleness threshold	4	4
speed	A : B : C : D : E = 24 : 25 : 26 : 27 : 50	A : B : C : D : E = 24 : 25 : 26 : 27 : 50
Dataset size	similar(50000/5)	A : other = 2000:10000

(a) Assumption 1 configure

Assumption 2	Normal Group	Comparison Group
Number of Client	5	5
Condition	3	3
staleness threshold	4	4
speed	A : B : C : D : E = 24 : 25 : 26 : 27 : 50	A : B : C : D : E = 24 : 25 : 26 : 27 : 50
Dataset size	similar(40000/5)	other : E = 8000:18000

(b) Assumption 2 configure

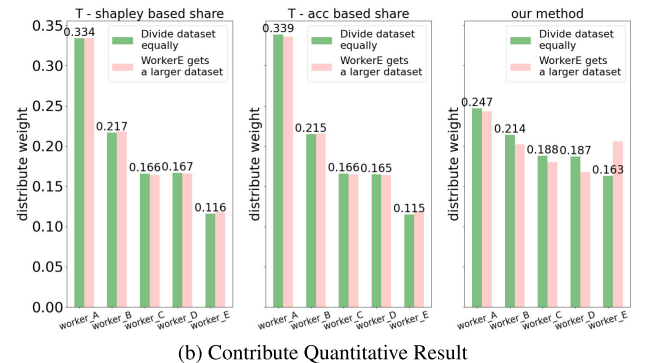
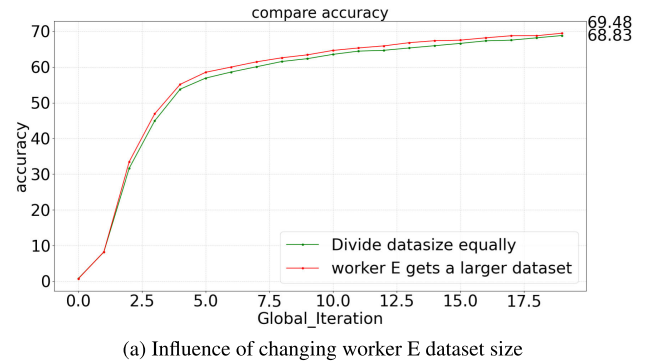


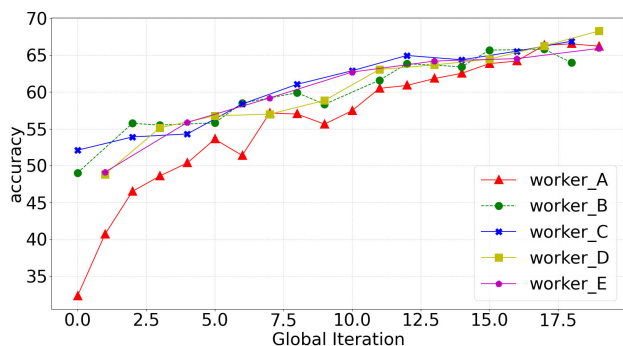
FIGURE 12. Assumption 2 result.

We compared our calculation method with the round-contribution scoring method for these two cases. Our comprehensive evaluation based on metrics balances quality

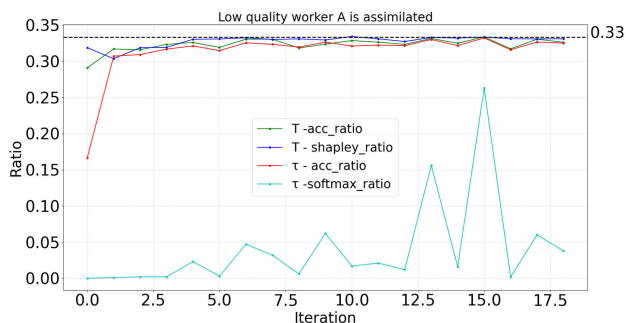
and computing speed, which is more reasonable than a round-based evaluation.

First, Figure 11(a) shows the calculation results for hypothesis one. The comprehensive calculation shared on the far-right shows the calculation results. When worker A obtains a small dataset, its reward share should be significantly reduced in our evaluation mechanism.

As shown in Figure 11(b), after worker A reduces the dataset, the accuracy of the global model is reduced by three points. This proves that the quality of the dataset held by the worker significantly affects the final requester's interests. However, in the Round-based evaluation method, worker A, who regularly participates, naturally obtains a high proportion, which weakens the incentive for data quality, as shown in the T-Shapley method and the T-acc ratio on the left in the figure. Therefore, our comprehensive contribution quantification method finds a balance between each metric, with quality as the main contribution and speed as the secondary contribution. Similarly, in Hypothesis 2, as shown in Figure 12(b), because worker E's large dataset improves the accuracy of the final model, its share in our quantitative results increases appropriately.



(a) The accuracy of each worker's local model in each round(Assumption 1 Comparison Group)



(b) Round Contribution of worker A under different calculation methods

FIGURE 13. Use softmax to differentiate quality contribution.

When calculating the proportion of worker contributions in each round, the reason we chose to bring the accurac value into the softmax function is shown in Figure 13. We found that although the dataset held by worker A is small, he continues to submit the average aggregation with other worker models, and the gap between the accuracy of the model and other

workers is decreasing, including the shapely value. Therefore, we introduce the softmax function to expand this difference, increase the proportion of data quality in the evaluation, and form a good incentive.

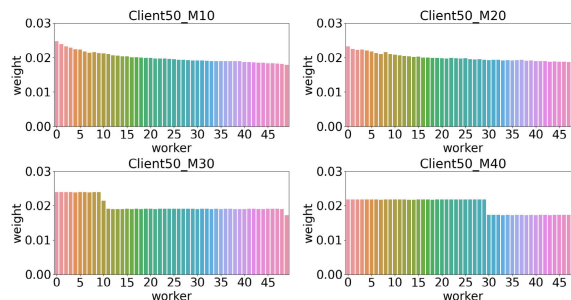


FIGURE 14. Contribute quantitative results (50 workers).

Finally, we calculated the respective contributions of 50 workers. We configured the speed as a normal distribution with a mean of 30 s and a standard deviation of 10, in line with the scenario of the heterogeneous clients we discussed. We divided the 50,000 datasets equally, and the calculation results for each worker's contributions are shown in Figure 14.

The requester actively configures the M value according to the task's response to determine the optimal strategy. However, different M values cause some workers to participate non-subjectively behind. Therefore, data quality was the main evaluation factor in our comprehensive evaluation, whereas participation (equipment computing power) was the secondary factor. The returned results were as expected, and a good balance between the metrics was found.

D. COMPARISON

In this study, we compared the proposed BASA-FL system with other systems. As illustrated in Table 7, our system incorporates an auditable mechanism that is not present in traditional FL systems. Specifically, in our BAFL-FL, we incorporate mechanisms to identify and exclude malicious workers, evaluate each worker's contributions, and adjust rewards and punishments based on those contributions, resulting in an overall improvement in FL quality.

Similarly, we compared our system with other auditable blockchain-based Federated Learning (FL) systems. The proposed system offers several key advantages. First, it is designed to operate effectively in a heterogeneous worker-resource environment and incorporates an optimal semi-asynchronous aggregation algorithm. Second, we implemented a worker reward and penalty mechanism based on their performance quality to ensure fairness across all participants. Third, we introduce a reliable contribution quantification method that encompasses two crucial aspects. The first aspect concerns the transparency of the evaluation process. For instance, in the 2cp crowdsourcing configuration [25], Alice independently utilized her dataset to execute the model accuracy calculation task, leading to a

TABLE 7. Comparison results.

	Federated Learning Method	Blockchain Type	Support for Heterogeneity		Fairness		Reliability of contribution	
			SRH	SDH	SDF	IDF	CCP	CCM
Fedavg	Sync	-	N	Y	-	-	-	-
Fedasync	Async	-	Y	M	-	-	-	-
FedSA	Semi-Async	-	Y	Y	-	-	-	-
Refiner[13]	Sync	Permissionless	N	-	Y	Y	Y	Y
2cp[9]	Sync	Permissionless	N	-	Y	Y	N	Y
BAFL[14]	Async	Permissioned	Y	-	Y	-	Y	N
our	Semi-Async	Permissionless	Y	-	Y	Y	Y	Y

¹ SRH:Support for Resource Heterogeneity; SDH:Support for Data Heterogeneity.

² SDF:Superior Data Fairness; IDF:Inferior Data Fairness.

³ CCP:consensus of the COntribution Calculation Process; CCM:Consensus of Contribution Calculation Metrics.

⁴ N:Not satisfied; Y:Satisfied; M:Moderately Satisfied -:Not considered.

lack of consensus on the computed outcomes. In contrast, our approach involves a randomized third party assigned to perform the calculation task. This inclusion enhanced the reliability of the calculation process. The second aspect focuses on the credibility of the metric sources. In our BASA-FL system, the calculation of the participant counts and staleness values is managed through smart contracts. Concurrently, a third-party consensus was employed to calculate the accuracy of the model. This dual-pronged approach ensured the reliability of the metric sources.

Nevertheless, although our BASA-FL has notable advantages over conventional FL frameworks, additional efforts are required to explore potential attacks and defenses, thereby enhancing and solidifying the robustness of the protocol. Our system identifies three types of malicious actors in FL, i.e., cheating requesters, malicious workers, and infected validators. However, other adversary attacks must be considered in further research, including poisoning and membership inference attacks. Poisoning attacks involve adversaries attempting to corrupt the global model by transmitting malicious updates during the collaborative training phase. Conversely, in membership inference attacks, adversaries aim to reverse engineer users' confidential data by observing the trained model updates. These attacks pose significant threats to the security and integrity of FL systems. Moreover, safeguarding client-sensitive data from diverse threats requires the exploration and application of various privacy techniques. These include differential privacy, homomorphic encryption, secure multiparty computation, and the utilization of a Trusted Execution Environment (TEE) [51]. Furthermore, a thorough investigation of the establishment of a dependable incentive mechanism is warranted. This study aimed to uphold fairness and inspire active client participation, contributing to the system's sustained development.

VIII. CONCLUSION

In this study, we propose a Blockchain-Based Auditable Semi-Asynchronous Federated Learning system (BASA-FL). The proposed system uses a distributed FL protocol based

on smart contracts, providing an auditable environment. To the best of our knowledge, our system represents an innovative approach incorporating blockchain and a semi-asynchronous mode to address worker heterogeneity effectively. In addition, we present a scheme for quantifying worker contributions specifically tailored to our system's semi-asynchronous nature. This mechanism identifies and excludes malicious workers, evaluates each worker's contributions, and adjusts rewards and punishments based on their contributions, resulting in an overall improvement of FL quality. Moreover, we evaluate the performance of the proposed system through extensive simulations. The implementation of an efficient multi-metric evaluation method further facilitated the quantification of worker contributions under semi-asynchronous conditions. Multiple simulations confirmed the enhanced performance and reliability of the BASA-FL mechanism, underscoring its potential to reshape collaborative AI training paradigms. We also showed that the τ -based audit in semi-asynchronous mode proves superior in terms of accuracy and effectiveness compared to the traditional T -based method. Moreover, it markedly enhances the final model quality by eliminating Byzantine nodes. Finally, we outline promising future directions, including exploring attack defense strategies, privacy-enhancing techniques, and establishing a dependable incentive mechanism.

ACKNOWLEDGMENT

(Qian Zhuohao and Muhammad Firdaus contributed equally to this work.)

REFERENCES

- [1] A. P. Sheth and J. A. Larson, "Federated database systems for managing distributed, heterogeneous, and autonomous databases," *ACM Comput. Surveys*, vol. 22, no. 3, pp. 183–236, Sep. 1990.
- [2] D. Heimbigner and D. McLeod, "A federated architecture for information management," *ACM Trans. Inf. Syst.*, vol. 3, no. 3, pp. 253–278, Jul. 1985.
- [3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

- [4] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," 2018, *arXiv:1811.03604*.
- [5] Y. Tong, Y. Wang, and D. Shi, "Federated learning in the lens of crowdsourcing," *IEEE Data Eng. Bull.*, vol. 43, no. 3, pp. 26–36, 2020.
- [6] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *J. Healthcare Informat. Res.*, vol. 5, no. 1, pp. 1–19, Mar. 2021.
- [7] M. Firdaus and K.-H. Rhee, "A joint framework to privacy-preserving edge intelligence in vehicular networks," in *Proc. Int. Conf. Inf. Secur. Appl. Cham, Switzerland: Springer*, 2022, pp. 156–167.
- [8] G. Long, Y. Tan, J. Jiang, and C. Zhang, "Federated learning for open banking," in *Federated Learning: Privacy and Incentive*. Cham, Switzerland: Springer, 2020, pp. 240–254.
- [9] V. Hegiste, T. Legler, and M. Ruskowski, "Application of federated learning in manufacturing," 2022, *arXiv:2208.04664*.
- [10] C. Yang, Q. Wang, M. Xu, Z. Chen, K. Bian, Y. Liu, and X. Liu, "Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data," in *Proc. Web Conf.*, Apr. 2021, pp. 935–946.
- [11] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," 2019, *arXiv:1903.03934*.
- [12] C. Ma, J. Li, L. Shi, M. Ding, T. Wang, Z. Han, and H. V. Poor, "When federated learning meets blockchain: A new distributed learning paradigm," *IEEE Comput. Intell. Mag.*, vol. 17, no. 3, pp. 26–33, Aug. 2022.
- [13] C. Ma, J. Li, M. Ding, H. H. Yang, F. Shu, T. Q. S. Quek, and H. V. Poor, "On safeguarding privacy and security in the framework of federated learning," *IEEE Netw.*, vol. 34, no. 4, pp. 242–248, Jul. 2020.
- [14] U. Majeed and C. S. Hong, "FLchain: Federated learning via MEC-enabled blockchain network," in *Proc. 20th Asia-Pacific Netw. Operations Manage. Symp. (APNOMS)*, Sep. 2019, pp. 1–4.
- [15] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchain on-device federated learning," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1279–1283, Jun. 2020.
- [16] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020.
- [17] X. Bao, C. Su, Y. Xiong, W. Huang, and Y. Hu, "FLChain: A blockchain for auditable federated learning with trust and incentive," in *Proc. 5th Int. Conf. Big Data Comput. Commun. (BIGCOM)*, Aug. 2019, pp. 151–159.
- [18] P. K. Sharma, J. H. Park, and K. Cho, "Blockchain and federated learning-based distributed computing defence framework for sustainable society," *Sustain. Cities Soc.*, vol. 59, Aug. 2020, Art. no. 102220.
- [19] S. Wang, "BlockFedML: Blockchain federated machine learning systems," in *Proc. Int. Conf. Intell. Comput., Autom. Syst. (ICICAS)*, Dec. 2019, pp. 751–756.
- [20] Y. Qu, S. R. Pokhrel, S. Garg, L. Gao, and Y. Xiang, "A blockchain federated learning framework for cognitive computing in industry 4.0 networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2964–2973, Apr. 2021.
- [21] S. Otoum, I. Al Ridhawi, and H. T. Mouftah, "Blockchain-supported federated learning for trustworthy vehicular networks," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.
- [22] L. Feng, Y. Zhao, S. Guo, X. Qiu, W. Li, and P. Yu, "BAFL: A blockchain-based asynchronous federated learning framework," *IEEE Trans. Comput.*, vol. 71, no. 5, pp. 1092–1103, May 2022.
- [23] Y. ChaoQun, "Decentralized federated learning based on committees and blockchain," 2022, *arXiv:2205.11137*.
- [24] D. Li, D. Han, T.-H. Weng, Z. Zheng, H. Li, H. Liu, A. Castiglione, and K.-C. Li, "Blockchain for federated learning toward secure distributed machine learning systems: A systemic survey," *Soft Comput.*, vol. 26, no. 9, pp. 4423–4440, May 2022.
- [25] H. Cai, D. Rueckert, and J. Passerat-Palmbach, "2CP: Decentralized protocols to transparently evaluate contributivity in blockchain federated learning environments," 2020, *arXiv:2011.07516*.
- [26] V. Mugunthan, R. Rahman, and L. Kagal, "BlockFlow: An accountable and privacy-preserving solution for federated learning," 2020, *arXiv:2007.03856*.
- [27] J. Zhu, J. Cao, D. Saxena, S. Jiang, and H. Ferradi, "Blockchain-empowered federated learning: Challenges, solutions, and future directions," *ACM Comput. Surveys*, vol. 55, no. 11, pp. 1–31, Nov. 2023.
- [28] J. Wang, Z. Charles, Z. Xu, G. Joshi, H. B. McMahan, M. Al-Shedivat, G. Andrew, S. Avestimehr, K. Daly, and D. Data, "A field guide to federated optimization," 2021, *arXiv:2107.06917*.
- [29] Y. Liu, Y. Qu, C. Xu, Z. Hao, and B. Gu, "Blockchain-enabled asynchronous federated learning in edge computing," *Sensors*, vol. 21, no. 10, p. 3335, May 2021.
- [30] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3654–3672, Dec. 2021.
- [31] C. Xu, Y. Qu, Y. Xiang, and L. Gao, "Asynchronous federated learning on heterogeneous devices: A survey," 2021, *arXiv:2109.04269*.
- [32] G. Shi, L. Li, J. Wang, W. Chen, K. Ye, and C. Xu, "HySync: Hybrid federated learning with effective synchronization," in *Proc. IEEE 22nd Int. Conf. High Perform. Comput. Communications; IEEE 18th Int. Conf. Smart City; IEEE 6th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Dec. 2020, pp. 628–633.
- [33] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "SAFA: A semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Trans. Comput.*, vol. 70, no. 5, pp. 655–668, May 2021.
- [34] S. Rahmadika, M. Firdaus, S. Jang, and K.-H. Rhee, "Blockchain-enabled 5G edge networks and beyond: An intelligent cross-silo federated learning approach," *Secur. Commun. Netw.*, vol. 2021, pp. 1–14, Mar. 2021.
- [35] M. Firdaus, S. Rahmadika, and K.-H. Rhee, "Decentralized trusted data sharing management on Internet of Vehicle edge computing (IoVEC) networks using consortium blockchain," *Sensors*, vol. 21, no. 7, p. 2410, Mar. 2021.
- [36] I. Covert and S.-I. Lee, "Improving kernelshap: Practical Shapley value estimation using linear regression," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2021, pp. 3457–3465.
- [37] Y. Qu, L. Gao, T. H. Luan, Y. Xiang, S. Yu, B. Li, and G. Zheng, "Decentralized privacy using blockchain-enabled federated learning in fog computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5171–5183, Jun. 2020.
- [38] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rel, "A survey on distributed machine learning," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–33, 2020.
- [39] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Bus. Rev.*, p. 21260, 2008.
- [40] M. Szydlo, "Merkle tree traversal in log space and time," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 2004, pp. 541–554.
- [41] M. Firdaus, S. Noh, Z. Qian, and K.-H. Rhee, "BPFL: Blockchain-enabled distributed edge cluster for personalized federated learning," in *Proc. Int. Conf. Comput. Sci. Appl. Int. Conf. Ubiquitous Inf. Technol. Appl.* Cham, Switzerland: Springer, 2022, pp. 431–437.
- [42] P. Gangwani, T. Bhardwaj, A. Perez-Pons, H. Upadhyay, and L. Lagos, "On the convergence of blockchain and IoT for enhanced security," in *Artificial Intelligence in Cyber-Physical Systems*. Boca Raton, FL, USA: CRC Press, 2023, pp. 35–49.
- [43] M. Ali, H. Karimpour, and M. Tariq, "Integration of blockchain and federated learning for Internet of Things: Recent advances and future challenges," *Comput. Secur.*, vol. 108, Sep. 2021, Art. no. 102355.
- [44] V. Buterin, "A next-generation smart contract and decentralized application platform," *White Paper*, vol. 3, no. 37, pp. 1–2, 2014.
- [45] Y. Lu, Q. Tang, and G. Wang, "On enabling machine learning tasks atop public blockchains: A crowdsourcing approach," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2018, pp. 81–88.
- [46] L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, and D. Moroz, "Chainlink 2.0: Next steps in the evolution of decentralized Oracle networks," *Chainlink Labs*, 2021.
- [47] C. Xu, J. Ge, Y. Li, Y. Deng, L. Gao, M. Zhang, Y. Xiang, and X. Zheng, "SCEI: A smart-contract driven edge intelligence framework for IoT systems," 2021, *arXiv:2103.07050*.
- [48] M. Firdaus, H. T. Larasati, and K.-H. Rhee, "A secure federated learning framework using blockchain and differential privacy," in *Proc. IEEE 9th Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/IEEE 8th Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, Jun. 2022, pp. 18–23.
- [49] N. Szabo, "Formalizing and securing relationships on public networks," *1st Monday*, vol. 2, no. 9, p. 548, 1997.

- [50] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2438–2455, Sep. 2021.
- [51] M. Firdaus, S. Noh, Z. Qian, H. T. Larasati, and K.-H. Rhee, "Personalized federated learning for heterogeneous data: A distributed edge clustering approach," *Math. Biosci. Eng.*, vol. 20, no. 6, pp. 10725–10740, 2023.



QIAN ZHUOHAO received the B.S. and M.S. degrees from the Department of Information Security, Pukyong National University (PKNU), Republic of Korea, in 2021 and 2023, respectively, and the master's degree from the Laboratory of Information Security and Internet Applications (LISIA), PKNU, in 2023. His research interests include applied cryptography, privacy preservation in decentralized systems, and AI with blockchain integration.



MUHAMMAD FIRDAUS (Graduate Student Member, IEEE) received the M.S. degree from the School of Electrical Engineering and Informatics, Institut Teknologi Bandung (ITB), Indonesia, in 2019, and the Ph.D. degree from the Department of Artificial Intelligence (AI) Convergence, Pukyong National University (PKNU), Republic of Korea, in 2023. He is currently a Postdoctoral Researcher with the Laboratory of Information Security and Internet Application (LISIA), PKNU.

His research interests include applied cryptography, blockchain, federated learning, edge intelligence, vehicular networks, and security and privacy protection in wireless communications and networking.



SIWAN NOH received the M.S. and Ph.D. degrees from the Department of Information Security, Pukyong National University (PKNU), Republic of Korea, in 2018 and 2023, respectively. He is currently a Postdoctoral Researcher with the Laboratory of Information Security and Internet Application (LISIA), PKNU. His research interests include applied cryptography, communication security, blockchain, and access control.



KYUNG-HYUNE RHEE (Member, IEEE) received the M.S. and Ph.D. degrees from the Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea, in 1985 and 1992, respectively. From 1985 to 1993, he was a Senior Researcher with the Electronic and Telecommunications Research Institute (ETRI), Republic of Korea. He was also a Visiting Scholar with The University of Adelaide, The University of Tokyo, and the University of California, Irvine.

He was the Chairman of the Division of Information and Communication Technology, Colombo Plan Staff College for Technician Education, Manila, Philippines. He is currently a Professor with the Department of IT Convergence and Application Engineering, Pukyong National University, Republic of Korea. His research interests include key management and its applications, mobile communication security, and the security evaluation of cryptographic algorithms.

...