Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.063630

ARTICLE

# Asynchronous Tiered Federated Learning Storage Scheme Based on Blockchain and IPFS

**Tianyu Li[1], Dezhi Han[1], Jiatao Li[1] and Kuan-Ching Li[2,*]**

[1]College of Information Engineering, Shanghai Maritime University, Shanghai, 201306, China
[2]Department of Computer Science and Information Engineering, Providence University, Taichung City, 43301, Taiwan
*Corresponding Author: Kuan-Ching Li. Email: kuancli@pu.edu.tw

Received: 20 January 2025; Accepted: 31 March 2025; Published: 19 May 2025

**ABSTRACT:** As is known, centralized federated learning faces risks of a single point of failure and privacy breaches, and blockchain-based federated learning frameworks can address these challenges to a certain extent in recent works. However, malicious clients may still illegally access the blockchain to upload malicious data or steal on-chain data. In addition, blockchain-based federated training suffers from a heavy storage burden and excessive network communication overhead. To address these issues, we propose an asynchronous, tiered federated learning storage scheme based on blockchain and IPFS. It manages the execution of federated learning tasks through smart contracts deployed on the blockchain, decentralizing the entire training process. Additionally, the scheme employs a secure and efficient blockchain-based asynchronous tiered architecture, integrating attribute-based access control technology for resource exchange between the clients and the blockchain network. It dynamically manages access control policies during training and adopts a hybrid data storage strategy combining blockchain and IPFS. Experiments with multiple sets of image classification tasks are conducted, indicating that the storage strategy used in this scheme saves nearly 50 percent of the communication overhead and significantly reduces the on-chain storage burden compared to the traditional blockchain-only storage strategy. In terms of training effectiveness, it maintains similar accuracy as centralized training and minimizes the probability of being attacked.

**KEYWORDS:** Federated learning; blockchain; access control; secure storage strategy; IPFS

## 1 Introduction

With the technological advancements in computing and communication technologies, mobile communication equipment and Internet of Things (IoT) devices can generate huge amounts of data suitable for model training. However, these data often have privacy restrictions on usage, making it impossible to aggregate them for model training. The concept of "Federated Learning" was first introduced by McMahan et al., which allows training data to be distributed across different local devices [1]. Participating parties no longer need to share the raw training data but can collectively train a global model by sharing their local model. However, synchronous federated learning algorithms suffer from issues such as inconsistent communication efficiency among participants and imbalanced computational speed [1–3], and the central server has to wait for the slowest device to respond before proceeding to the next round of training. Moreover, a large number of clients upload their model parameters to the server, causing the transmission bottleneck.

To address the aforementioned challenges, Xie et al. have proposed asynchronous federated optimization methods based on traditional federated learning algorithms [4,5]. Chai et al. have introduced

asynchronous federated training methods based on a hierarchical approach, where clients are divided into different layers based on response latency and asynchronously communicate with the server [6,7]. This hierarchical mechanism better adapts to the device heterogeneity and enhances the overall performance of federated training.

However, existing federated learning algorithms are designed assuming the central aggregator is secure. In practical scenarios, a malicious central aggregator may steal or tamper with model data, leading to privacy breaches or disruption of normal training. Furthermore, malicious participants can launch poisoning attacks by uploading malicious model parameters or gradients, introducing a backdoor into the model, and causing the global model to train in a direction unfavorable for convergence.

To address the security issues as mentioned above, Lyu et al. proposed differential privacy [8], while Truex et al. introduced a fully homomorphic encryption algorithm [9]. However, these methods suffer from issues such as slow speed and lack of decentralization due to the introduction of noise or extensive use of encryption algorithms. In recent years, an increasing number of researchers have utilized blockchain to manage the federated learning model training process and data storage management. For instance, Goel et al. leveraged blockchain technology to store CNN models by storing each layer of the model in blocks of the blockchain. The blockchain's hash chain structure was utilized to provide tamper resistance and secure verification for the model [10]. Mondal et al. employed smart contracts in blockchain to achieve distributed machine learning, eliminating centralization and utilizing differential privacy techniques to counter poisoning attacks [11]. Feng et al. proposed a blockchain-based asynchronous federated learning framework that optimized the control of block generation rate to reduce the latency of federated learning. It dynamically adjusted the training frequency of asynchronous federated learning to prevent frequent uploads of local models, which could overload blockchain transactions [12].

In practical IoT scenarios such as IoT, mobile edge networks, computer vision, and finance, collaborative federated training is a holistic process that includes local training, device interactions, and model aggregation on the server. Security vulnerabilities can arise at any point. At the algorithm level, as a widely used technique, federated learning typically requires a framework that strikes a balance between model convergence speed, model accuracy, cross-client fairness, communication efficiency, and security [13]. Meanwhile, due to the collaborative nature of the training process, federated learning faces security issues at the management level as well. Wang et al. proposed a blockchain-based secure data aggregation strategy for edge computing-enabled IoT [14]. They also proposed a blockchain-based heterogeneous hierarchical trust assessment strategy for 5G-ITS using joint deep learning techniques [15].

In addition, access control technology [16,17], as an information security mechanism, is widely used in managing access permissions for resources or information on the blockchain because it can be efficiently deployed on the blockchain through smart contracts [18–20]. For example, Li et al. implemented medical device supply chain management through access control smart contracts [21]. Jiang et al. proposed a blockchain-based trusted model evaluation framework for deep learning and applied it in mobile object segmentation and designed access control policies to manage operations on blockchain [22].

One of the challenges faced by blockchain-based federated learning is the heavy burden of on-chain storage when dealing with a large-scale model. Large-scale models possess more robust representational capabilities and significantly improve performance in various tasks such as natural language processing, computer vision, and speech recognition, however, they impose significant communication and storage burdens on federated training. To address this problem, Wang et al. utilized ASTORE to alleviate system storage overhead [23]. Jiang et al. employed IPFS as an auxiliary storage tool for model files in the blockchain system to alleviate the on-chain storage burden [22]. Introducing IPFS as an additional distributed storage

service can share part of the data storage pressure for the blockchain, providing new possibilities for solving the storage performance bottleneck problem of blockchain-based federated training.

Currently, two main issues have not been effectively addressed in existing research on blockchain-based federated learning. Firstly, the increasing number of clients and the complexity of models lead to a heavier storage burden on the blockchain and a surge in communication volume. Secondly, the security of communication between the clients and the nodes on the blockchain has not been adequately considered in previous work. To fill this research gap, we propose a storage scheme for asynchronous tiered federated learning based on blockchain and IPFS, which is not dependent on specific blockchain platform architecture, making it highly scalable and suitable for various research and production applications. At the technical level, the strategy employed in our scheme differs from the existing work. The proposed blockchain-based federated learning scheme demonstrates good storage performance, high communication efficiency, and secure privacy protection by access control policies. The main contributions and innovations are as follows:

1. We apply the asynchronous tiered federation learning aggregation algorithm to the blockchain framework and design a layered blockchain-based architecture that ensures decentralization and security of the asynchronous federation aggregation process.
2. We design an access control model based on asynchronous tiered federated learning and deploy it in smart contracts to achieve decentralized permission management of clients, which has not been proposed in any other research.
3. This scheme designs and implements a hybrid on-chain and off-chain storage strategy for federated learning with the help of IPFS distributed file system, which is safe, efficient, and effectively reduces the storage burden and communication overhead of the blockchain in the process of federated training.
4. The proposed scheme applies to any deep learning model and any blockchain platform, demonstrating robust scalability. We deploy the scheme on the Hyperledger Fabric framework and showcase its full functionality. Finally, we conduct experiments on multiple datasets and settings to validate and analyze this scheme's federated training effectiveness, storage performance, communication overhead, and security.
5. We compare this scheme with several other federated learning frameworks, and our security analysis demonstrates the effectiveness of the proposed scheme in countering model inference attacks, backdoor attacks, illegal client connection attacks, data theft, data forgery, and identity forgery attacks.

The remainder of this work is organized as follows. In Section 2, we present the background knowledge for this work. In Section 3, we provide a detailed description of the overall architecture, operational steps, and module designs of the proposed scheme. In Section 4, we present the necessary experimental procedures and results, analyzing them from storage performance, communication performance, and security perspectives. Finally, concluding remarks and future research works are given in Section 5.

## 2 Preliminaries

In this section, we present some background knowledge and necessary concepts for the proposed research.

### *2.1 Asynchronous Tiered Federated Learning*

The most used method in federated learning is FedAvg, where the training approach involves randomly selecting a subset of clients to participate in aggregation during each round of training and synchronously waiting for responses from all selected clients [1]. However, FedAvg encounters the straggler issue when dealing with large-scale edge devices due to limitations in computational and communication capabilities,

making it challenging to achieve global synchronization among clients and thus resulting in slow training speed [4].

To address the abovementioned issues, Chai et al. proposed a layered asynchronous training approach to mitigate the heterogeneous latency issue among different devices [6,7]. The strategy based on layered federated training is that the central server allocates clients to different logical layers based on their response latency before formal training. During each round of training, the central server only needs to wait for clients with similar response speeds within the same layer, effectively improving training efficiency and accelerating the convergence of the global model.

### *2.2 Blockchain*

Blockchain is a decentralized tamper-proof ledger that is widely used and consists of a series of records called "blocks" that are linked together using specific cryptographic algorithms [24]. The blocks are maintained in a network of several mutually distrusting peer nodes, with each peer maintaining a copy of the ledger, thus eliminating the reliance on a centralized institution [25]. Blockchain platforms typically support programmable script execution. Cryptocurrencies like Bitcoin use scripts to validate transactions [26], while platforms like Hyperledger Fabric employ smart contracts as trusted decentralized applications [27]. Smart contracts can replace the central aggregator in federated learning and handle tasks such as managing model storage, exchange, and aggregation. To ensure scalability, this work investigates the architecture of mainstream blockchain platforms, as shown in Table 1.

**Table 1:** Architecture design comparison of major blockchain platforms

| Blockchain frameworks | Consensus mechanisms | Support for smart contract | Categorize | Extensibility | Segregation mechanisms |
|---|---|---|---|---|---|
| Bitcoin | Pow | No | Public | Only for digital currency | SigWit |
| Ethereum | Pow, Pos | Yes | Public | Extensible | Virtual machine |
| Hyperledger fabric | Kafka, Solo | Yes | Allied | Extensible | Docker, Channel |

Compared to various blockchain frameworks in the research results, Hyperledger Fabric is more friendly to supporting smart contracts, adopts a consortium chain-level privacy architecture, and is easy to deploy. Most importantly, due to its robust scalability, it can support smart contracts for different businesses, making it more convenient for practical applications. Considering the above factors, Hyperledger Fabric is selected as the blockchain platform to support the deployment of this scheme.

### *2.3 Attribute-Based Access Control Model*

Attribute-Based Access Control (ABAC) is a fine-grained and flexible authorization model that checks whether a user has the permissions to perform operations on an object based on entity attributes, action types, and relevant environment conditions [28]. The ABAC model consists of AS (authorization subject), AO (authorization object), EE (environment entity), and AP (authorization policy), which together form the access rule. Administrators assign specific values to these four components based on access rules, creating an access control policy, $Policy = \{AS \cup AO \cup AE \cup AP\}$. Users submit corresponding authentication information based on access rules, and the system compares it with existing access control policies to determine whether the user is authorized to perform the requested operation.

However, applying ABAC carries the risk that if the ABAC access control policies are illegitimately created, the private data protected by this authorization model may be at risk of leakage. Blockchain technology can address these risks with its anonymity, immutability, and decentralization, and it can protect and provide traceability to access control policies.

### *2.4 Model Storage Technology*

Deep learning is an important branch of machine learning, and various deep learning frameworks are popular in academia and industry, such as PyTorch, TensorFlow, and Keras. They all assist in saving complex model parameters as binary files and reloading model parameters when needed. In practical federated learning scenarios, participants cannot exchange model parameters by passing specific model variables as they would in a local setting. An effective solution is to communicate by exchanging model parameter files in a consistent format. Therefore, we need a secure and efficient model storage technology to support federated learning applications in real-world scenarios.

Common model storage technologies include data encryption, access control, blockchain, and the Inter Planetary File System (IPFS). As a decentralized, distributed, and tamper-proof digital ledger, blockchain is often used as a secure data storage platform. However, as the amount of data on the blockchain increases, it becomes essential to allocate on-chain storage space effectively to store more valuable information.

IPFS is a peer-to-peer distributed file storage system [29]. It utilizes content-based addressing. When a user initiates a file upload request, the IPFS service generates a content identifier (CID) based on the file's content hash value and returns the CID as the unique identifier for the file. IPFS offers high-performance data transmission and decentralized storage services, and the content-based addressing mechanism provides it with a certain level of tamper resistance.

## 3 Scheme Description

In this section, we present a secure and efficient scheme for federated training and data storage that ensures strict privacy protection. Based on blockchain technology, it utilizes access control techniques to validate the legitimacy of client requests, employs an asynchronous, tiered federated aggregation algorithm for federated training, and adopts an on-chain and off-chain hybrid model storage strategy to improve communication efficiency and alleviate the storage burden on the blockchain.

Assuming $N$ mutually distrusting clients $C_1, C_2, \ldots, C_N$, holding datasets $D_1, D_2, \ldots, D_N$. These $N$ clients aim to collectively train a shared global model $W^g$ without exposing their private datasets to other participants. In the case of horizontal federated learning, the training samples owned by each participant have the same feature space but different data IDs in their respective datasets.

The potential threat scenarios that this scheme may face are limited to the following situations. Curious clients may attempt to access the updated model parameters of other participants during the training process and plan attack behaviors based on them. For instance, the attribute inference attack utilizes stolen model parameters as input features to train attack models and infer the sensitive attributes of other participant's data [30]. Malicious clients may also attempt to connect with the blockchain network to carry out disruptive actions such as short-link non-poisoning backdoor attacks [31,32].

### *3.1 System Framework*

The proposed scheme consists of three main modules: the federated training module, the access control module, and the data storage module. The system framework is illustrated in Fig. 1.

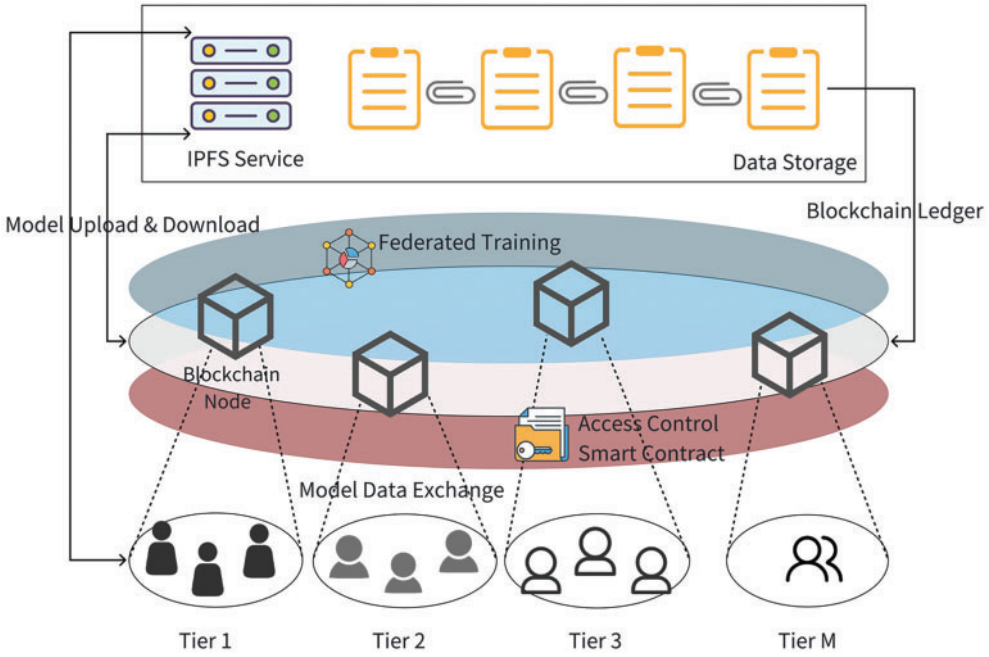


**Figure 1:** The system architecture

The federated training module is responsible for federated aggregation. Smart contracts deployed on the blockchain act as central aggregators to avoid a single point of failure. The access control module manages the interaction process between clients and the blockchain. Any requests submitted by clients to the blockchain must undergo identity verification and access control policy validation. Access privileges to the global model for different rounds are dynamically generated during the training process. The data storage module consists of the blockchain network and IPFS service, which manages client identity information, dynamic access control policies, and model resources. The data required by access control policy and CIDs of model files are stored on the blockchain network in a $(key - value)$ format, while complete model parameter files are stored on the IPFS service.

The detailed operational steps of this scheme are provided in Algorithm 1 and Fig. 2. Implementation details of each module will be discussed in subsequent sections.

**Algorithm 1:** Detailed operational steps of the proposed scheme

**Require:** Client $1 - k$, each with its private dataset $D_i$.
**Ensure:** Federated Global Model $W^g$

1. Divide clients into blockchain nodes of different communication tiers based on response latency and register legitimate identities for clients on the blockchain.
2. Generate the initial global model $w^0$ on the blockchain and store it there. Simultaneously, register the initial access control policy on the blockchain, allowing all clients to request this initial global model.
3. Each client requests the global model.
4. The blockchain network verifies the client's identity through a smart contract and checks the request's legitimacy based on the access control policy. If the request passes, the model is returned.
5. Each client starts training the global model on its private dataset and submits a request to aggregate the local model to the communication node in its corresponding logical tier.

(Continued)

**Algorithm 1 (continued)**

6. The blockchain node verifies the client's identity and the correctness of the request through the access control smart contract. If correct, it saves the submission.
7. The blockchain node checks the number of local models requested. Whenever the threshold is reached, it automatically triggers the execution of the federated aggregation smart contract, aggregates the models, obtains the global model for this round, and saves it on the blockchain.
8. After each aggregation is completed, the blockchain automatically triggers the execution of the access control smart contract and dynamically registers a new access control policy (if $round \geq 2$). Clients ready for the next round of training can access this latest global model.
9. Repeat steps 3–8 until the global model converges or the maximum number of global training rounds $T$ is reached.

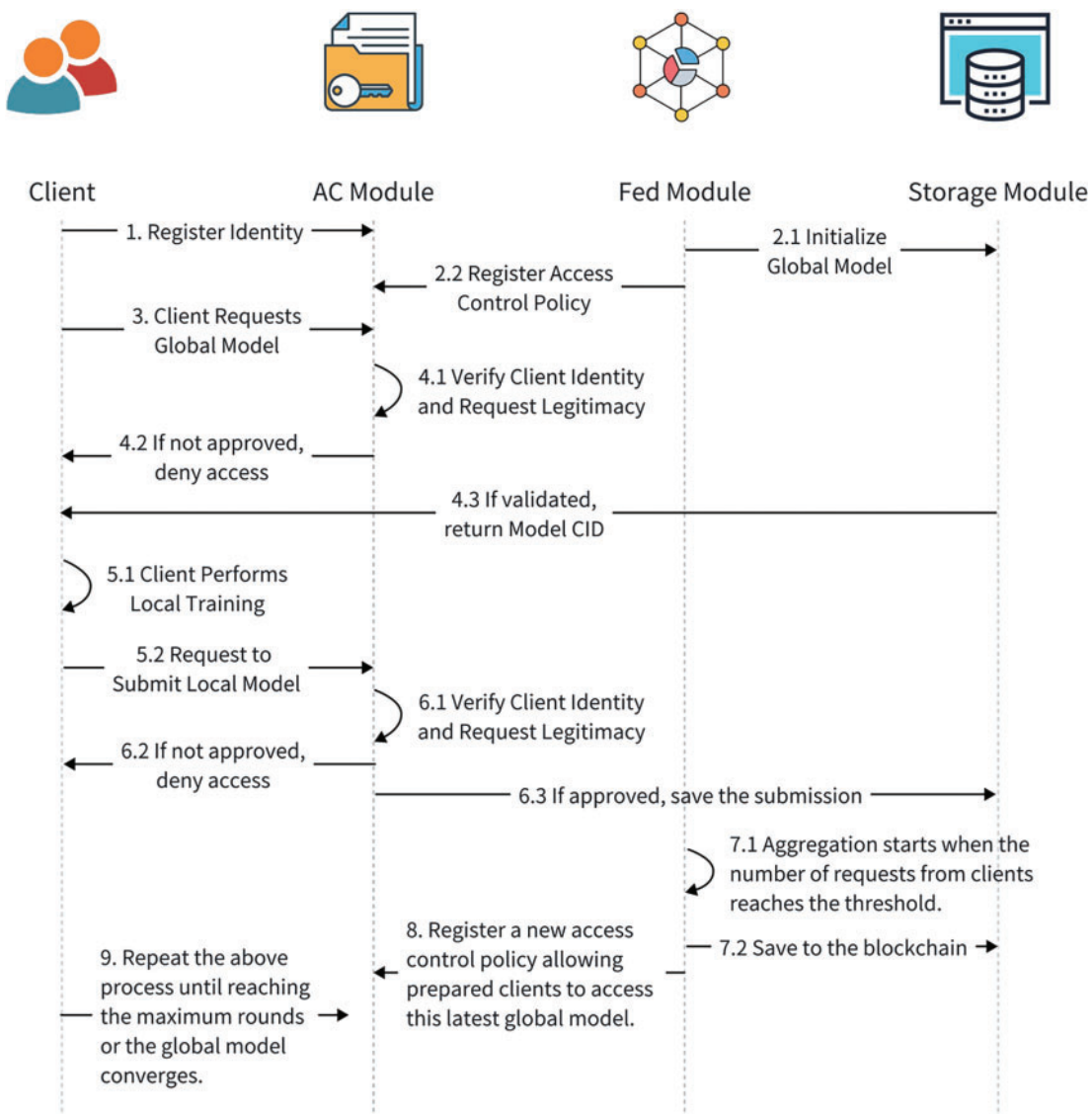


**Figure 2:** The detailed implementation and the algorithm depicted in Algorithm 1

### 3.2 Blockchain-Based Asynchronous Tiered Federated Training

In the federated training module, model aggregation is managed by the federated aggregation smart contract. This contract implements a decentralized, asynchronous tiered federated aggregation algorithm. Before training begins, each client is assigned to a different logical communication layer based on their response latency. Considering that each node on the blockchain needs to store a copy of the data, in scenarios with many clients participating in training, it can impose a significant storage burden on the blockchain. To address this, we reduce the number of blockchain nodes and maintain one blockchain node participating in consensus for each logical communication layer, instead of assigning each client to a consensus node.

In each training round, clients complete local training and submit their local models to the blockchain. When the number of submissions from clients in the same layer reaches the set threshold, it automatically triggers the execution of the federated aggregation smart contract. This smart contract implements both intra-layer aggregation and inter-layer aggregation processes. Clients participating in aggregation within each round belong to the same layer and utilize Eq. (1).

$$w_{tier_m} = \sum_{k=1}^{C} \frac{n_k}{N_m} w_k^{t+1} \tag{1}$$

To perform intra-layer aggregation to obtain the updated model for this layer, the intra-layer aggregation algorithm within the same layer is the Federated Averaging (FedAvg) algorithm introduced in Section 2.1. After updating the layer model for the current communication layer, Eq. (2) is applied.

$$w^{t+1} = \sum_{m=1}^{M} \frac{T_{tier_{M+1-m}}}{T} w_{tier_m} \tag{2}$$

To perform cross-layer global model aggregation and obtain the global model $w^{t+1}$, the weight corresponding to the layer $m$ during the global model aggregation is $T_{tier_{M+1-m}}/T$, signifying that layers with more frequent updates have smaller corresponding weights during aggregation, while layers with lower update frequencies have greater weights during aggregation. This approach helps prevent the global model from biasing towards the logic layers that respond faster [6]. The complete algorithm process of the federated training module is described in Algorithm 2.

**Algorithm 2:** Asynchronous tiered federated training

**Require:** $C$ denotes the number of clients in one round. $tier_m$ is the number of updates of tier $m$. $T$ is the current global rounds, $T = T_{tier_1} + T_{tier_2} + \ldots + T_{tier_M}$. Each client gets the initial global model $w^0$.
**Ensure:** Trained global model $W^g$.

```
for each training round do
    for each client k in parallel do
        n_k = |D_k|
        w_k^{t+1} ← LocalTraining(w_k^t, l)
        C = C + 1
    end for
    if c ≥ threshold then
        N_m = Σ_{k=1}^{C} n_k
        w_{tier_m} ← TieredAggregation(w_k^{t+1}, C)
        T_{tier_m} = T_{tier_m} + 1
```

(Continued)

**Algorithm 2 (continued)**

$C = 1$
**end if**
$w^{t+1} \leftarrow CrossTierAggregation\left(w_{tier_m}, M\right)$
**end for**
**return** global weight $w^T$

***LocalTraining*** $\left(w_k^t, l\right)$
$w_k^{t+1} = w_k^t - \eta \nabla l\left(w^t\right)$
**return** $w_k^{t+1}$

***TieredAggregation*** $\left(w_k^{t+1}, C\right)$
$w_{tier_m} = \sum_{k=1}^{C} \frac{n_k}{N_m} w_k^{t+1}$
**return** $w_{tier_m}^{t+1}$

***CrossTierAggregation*** $\left(w_{tier_m}, M\right)$
$w^{t+1} = \sum_{m=1}^{M} \frac{T_{tier_{M+1-m}}}{T} w_{tier_m}$
**return** $w^{t+1}$

Unlike other federated learning frameworks, the federated training process in this scheme is not led by a central server. Instead, it is orchestrated by smart contracts deployed on the blockchain acting as central aggregators, passively waiting for client requests, achieving an utterly decentralized management architecture. Model data is transmitted between clients and the blockchain network, and its security is maintained by the blockchain and access control smart contracts.

### *3.3 Access Control*

In our scheme, the interaction between clients and the blockchain network is divided into two categories: (1) To submit local models to the blockchain network. (2) To request the aggregated global model. The above processes correspond to security vulnerabilities that are twofold: (1) Unauthorized malicious clients may upload malicious local models to disrupt the entire federated training process. (2) The updated global model is stored on the blockchain. And the layer aggregation models for each layer and the local models submitted by each client are saved. Clients may access model data other than the global model, leading to privacy leaks.

To securely share model resources, this scheme uses access control to manage access requests from clients to the blockchain.

#### *3.3.1 Definition of Access Control Model*

The scheme adopts an Attribute-Based Access Control model (ABAC) to achieve fine-grained and flexible access control. The basic concepts of ABAC have been introduced in Section 2.3. In our scheme, the user entity (AS) represents the client, the object entity (AO) represents model resources, and the permission operation (AP) represents a client's upload or access operation on-chain model resources. The definitions of attributes for each entity are provided below.

**Definition 1.** *$Client$ defines the attributes of the client entity. Client = {ClientID, ClientTier, TierRound, GlobalRound }. $ClientID$ is the identifier of the client participating in federated training, $ClientTier$ indicates*

*the client's communication layer, $TierRound$ represents the number of updates in the client's communication layer (also the number of updates for the client) and $GlobalRound$ is the global round.*

**Definition 2.** *$Model$ defines the attributes of the model resources saved on-chain. Model = {ModelID, ModelType, $ModelTier$, $TierRound$, $GlobalRound$, $ModelEncryAddr$, $Timestamp$}. $ModelID$ is the unique index of the on-chain model resource, $ModelType$ denotes the type of model resource, categorized as client local model, layer aggregation model, and global aggregation model. If it is not a global model, the $ModelType$ identifies the model's layer, and $TierRound$ explains the communication rounds of that layer. $GlobalRound$ represents the global update round, $ModelEncryAddr$ indicates the IPFS storage index of the encrypted model, and Timestamp is a timestamp recording the time when the model is saved to the blockchain.*

*3.3.2 Federated Secure Access Policies*

The access control policy in this scheme is called the Federated Secure Access Policy (FSAP), where $FSAP = \{AC, AM\}$, $AC = \{Client.ClientTier, Client.GlobalRound\}$, $AM = \{Model.ModelID\}$, $AC$ and $AM$ are the necessary attributes involved in the access control verification for $Client$ and $Model$ respectively. The unique identifier calculation for FSAP is $FSAP\_ID = SHA256\{AC.ClientTier + AM.ModelID\}$. Access control policies are recorded on the blockchain network in the form $(FSAP\_ID, FSAP)$.

Before exchanging model data between clients and the blockchain network, identity legitimacy verification and access control request verification are required. Initially, an access control request $FSAPRequest = \{Client, Model, OP\}$ is generated, where $Client$ represents the requesting client, $Model$ denotes the model the client is requesting to operate on, and $OP$ signifies the client's requested operation, which can be model upload or model access. Subsequently, we define $FSAPRequestID = SHA256\{FSAPRequest.Client.ClientTier + FSAPRequest.Model.ModelID\}$, and validated clients must check if this index exists in the blockchain state database. If $FSAPRequestID$ is found, access is granted for subsequent operations. Otherwise, it indicates that the client does not have permission for this operation, and access is denied.

To achieve decentralized dynamic management of client access to on-chain resources, a smart contract named the Federal Secure Access Contract (FSAPC) is designed, which includes the following methods:

$AddValidClient$: Add legitimate client identities participating in federated learning to the blockchain ledger.

$CheckClient$: Retrieve client information from the blockchain state database to verify legitimacy.

$AddPolicy$: Add access control policies to the blockchain ledger.

$QueryPolicy$: Retrieve access control policies from the blockchain state database.

$CheckValid$: Generate access control requests for the client's current access operation and verify the validity of the request.

Algorithms 3–5 describe the main methods in this smart contract. $APIstub.PutState(k, v)$ denotes uploading index $k$ and its corresponding value $v$ to the blockchain ledger, $APIstub.GetState(k)$ indicates retrieving index $k$ from the blockchain state database, and SHA-256 is a commonly used hash function in the SHA-2 hash family.

**Algorithm 3:** AddPolicy(): Adding access control policy to the blockchain

**Require:** $AC$, $AM$, $APIstub$
**Ensure:** OK or Error
1. **if** $AC = NullorAM = Null$ **then**
2. **return** Error("Invalid input!")
3. **end if**
4. $FSAP\_ID \leftarrow SHA256\,(AC.ClientTier + AM.ModelID)$
5. $Respose \leftarrow APIstub.GetState\,(FSAP\_ID)$
6. **if** $Response \neq Null$ **then**
7. **return** Error("Policy already exists!")
8. **end**
9. $FSAP \leftarrow \{AC, AM\}$
10. $APIstub.PutState\,(FSAP\_ID, FSAP)$
11. **return** Success($FSAP$)

**Algorithm 4:** QueryPolicy(): Querying access control policy from the blockchain

**Require:** $AC$, $AM$, $APIstub$
**Ensure:** $FSAP$ policy or Error
1. **if** $AC = NullorAM = Null$ **then**
2. **return** Error("Invalid input!")
3. **end if**
4. $FSAP\_ID \leftarrow SHA256\,(AC.ClientTier + AM.ModelID)$
5. $FSAP \leftarrow APIstub.GetState\,(FSAP\_ID)$
6. **if** $FSAP = Null$ **then**
7. **return** Error("Policy does not exist!")
8. **end if**
9. **return** Success($FSAP$)

**Algorithm 5:** CheckValid(): Checking whether a client's request is valid or not

**Require:** $Client$, $Model$, $OP$, $APIstub$
**Ensure:** OK or Error
1. **if** $AC = NullorModel = Null$ **then**
2. **return** Error("Invalid input!")
3. **end if**
4. $Client \leftarrow APIstub.GetState\,(Client.ClientID)$
5. **if** $ClientID = Null$ **then**
6. **return** Error("Invalid Client!")
7. **end if**
8. **if** $OP$ is uploading local model and $Model.ModelType$ is local model **then**
9. **return** Success("OK")
10. **end if**
11. **return** Error("Wrong model type, access is denied!")

### *3.4 Data Storage Management*

As the round of federated training increases, the data storage burden on the blockchain will intensify and throughput will also be constrained. This scheme employs a combined on-chain and off-chain storage strategy to reduce communication costs and storage burdens. The storage of client's identity information and access control policies has been discussed in previous Section 3.3.2, and this section focuses on the exchange and storage strategy for model data.

In this work, IPFS is utilized as off-chain auxiliary storage service to store actual model files, generating a unique CID for each file. The CID returned by the IPFS service is encrypted and saved on blockchain as the attribute $Model.ModelEncryAddr$ for model resources. Model resources are stored in the form $(ModelID, Model)$, where $ModelID$ is derived by hashing essential description information of the model resource, $ModelID = SHA256\{ModelType + ModelTier + TierRound + GlobalRound\}$. These four attributes uniquely identify a model resource, and the SHA256 algorithm exhibits strong collision resistance, making it nearly impossible to find or fabricate other data with the same hash result. Thus, the $ModelID$ generated using this method is unique and can serve as the unique identifier for model resources.

In this storage strategy, clients and the blockchain no longer exchange actual model files. Clients upload the encrypted CID of their local model as part of the $FSAPRequest$ to the blockchain, completing the submission of their local model. The blockchain responds to client requests by providing the CID of the global model. Fig. 3 illustrates the data flow corresponding to the above storage process.

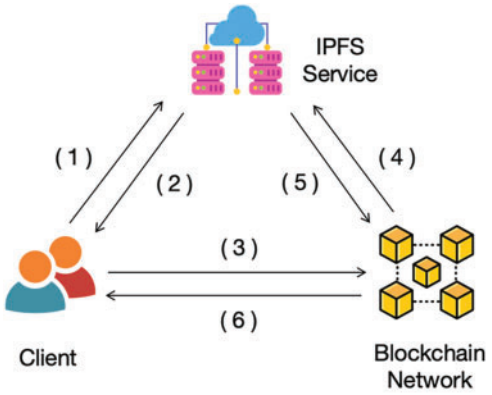


**Figure 3:** Data flow of the proposed scheme, showing the data interactions between components

Explanation of each step in Fig. 3 is as follows:

1. The client uploads a local model file to IPFS service.
2. IPFS service returns the corresponding CID.
3. Client sends local model identifier to blockchain network.
4. Blockchain sends CID to IPFS service.
5. IPFS service returns the corresponding model file.
6. Blockchain returns a global model identifier.

To achieve the decentralization and efficient management of the storage process described above, a Model Storage Smart Contract (MSC) is designed with the following main methods:

$AddModel$ ( ): Add model resources to the data storage module.

$RequestModel$ ( ): Retrieve model resources on the blockchain network.

Algorithms 6 and 7 provide algorithmic descriptions of the two methods mentioned above.

**Algorithm 6:** AddModel(): Uploading a model resource to the blockchain

**Require:** Model file, Model attribute information
**Ensure:** OK or Error
1. $CID \leftarrow$Upload model file to IPFS
2. **if** $CID = Null$ **then**
3. **return** Error("Error while uploading model file to IPFS.")
4. **end if**
5. $Model \leftarrow$Model attribute information
6. $Model.ModelEncryAddr \leftarrow Encrypt(CID)$
7. $Model.Timestamp \leftarrow Linux.Time.Now()$
8. $ModelID \leftarrow SHA256(ModelType + ModelTier + TierRound + GlobalRound)$
9. $APIstub.PutState(ModelID, Model)$
10. **return** Success("OK")

**Algorithm 7:** RequestModel(): Searching for a model record in the blockchain

**Require:** Model attribute information
**Ensure:** Encrypted IPFS CID or Error
1. $ModelID \leftarrow SHA256(ModelType + ModelTier + TierRound + GlobalRound)$
2. $Model \leftarrow APIstub.GetState(ModelID)$
3. **if** $Model = Null$ **then**
4. **return** Error("Error in model information!")
5. **end if**
6. **return** Success(Model.ModelEncryAddr)

## 4 Experiment and Analysis

As experimentation, we deployed the Hyperledger Fabric blockchain platform [27], utilized Golang for smart contract programming, employed Node.js for client applications, and executed all client nodes in separate Docker containers. All experiments were conducted on Linux servers equipped with Intel(R) Xeon(R) Silver 4214R CPU and RTX 3080Ti GPU, with 12 CPU cores and 45 GB RAM. The setup involved 40 clients distributed across 4 nodes on the blockchain network, each with 10 clients.

The federated learning training on convolutional neural networks performed in this research utilized the MNIST and CIFAR-10 standard datasets. MNIST comprises 60,000 training images of size $28 * 28$ and 10,000 test images, with 1 input channel and images categorized into 10 classes [33]. CIFAR-10 includes 50,000 training images of size $32 * 32$ and 10,000 test images, with 3 input channels and images categorized into 10 classes [34]. We randomly divide the data based on the data sample index. The local training settings were that local epoch set to 3, local batch size set to 10, and all clients using the same learning rate across different datasets.

### 4.1 Experimental Procedure

This subsection demonstrates the key steps and functionalities of the experiments. As shown in Fig. 4 the smart contracts executed in the experiment, method names, and input parameters are marked with blue underlines, while the experiment results are marked with red underlines.

```
root@jtli-ubuntu:/home/gopath/src/github.com/FSAP/client/nodejs# node ./invoke
.js fsapc QueryPolicy '{"AC":{"ClientTier":2,"GlobalRound":89},"AM":{"ModelID"
:"bb128b310c994e98a1f05317d62349191f38fa7cce5fb952a5f2a212314b8551"},"AE":0}'
Wallet path: /home/gopath/src/github.com/FSAP/client/nodejs/wallet
fsapc QueryPolicy {"AC":{"ClientTier":2,"GlobalRound":89},"AM":{"ModelID":"bb1
28b310c994e98a1f05317d62349191f38fa7cce5fb952a5f2a212314b8551"},"AE":0}
Transaction has been submit, result is: 19199811b086075b2ee2356ae1942c7a6bcc7f
1fffd6cc3f0a142101172e5243
```

**Figure 4:** QueryPolicy execution result

Fig. 4 illustrates the query result of an FSAP access control policy on the blockchain, obtained by invoking the $QueryPolicy()$ method in the access control module smart contract. The meaning of the input parameters is whether a certain client with communication layer 2 and the current global round being 89, has permission to access the model indexed by the $ModelID$ shown in Fig. 4. $ModelID$ is derived from $SHA256\{ModelType{:}1, ModelTier{:}-1, TierRound{:}-1, GlobalRound{:}89\}$, representing the global model aggregated after the round 89 of federated training. Since a new access control policy is dynamically registered after each round of federated aggregation, allowing legitimate clients to access the global model, this query successfully returns the $FSAP\_ID$ of the access control policy.

Fig. 5 displays the verification result of a client's request initiated on the blockchain. The client information and model information are the same as shown in Fig. 3. As described in Section 3.3.2, this client has undergone identity validation, operation type validation and access control policy validation through the $CheckValid()$ method, returning a success message, indicating that the client's request has been validated through the access control module.

```
root@jtli-ubuntu:/home/gopath/src/github.com/FSAP/client/nodejs# node ./invoke
.js fsapc CheckValid '{"Client":{"ClientID":23,"ClientTier":2,"TierRound":11,"
GlobalRound":89},"Model":{"ModelType":1,"ModelTier":-1,"TierRound":-1,"GlobalR
ound":89,"ModelEncryAddr":"","Timestamp":0},"OP":{"OPType":0}}'
Wallet path: /home/gopath/src/github.com/FSAP/client/nodejs/wallet
fsapc CheckValid {"Client":{"ClientID":23,"ClientTier":2,"TierRound":11,"Globa
lRound":89},"Model":{"ModelType":1,"ModelTier":-1,"TierRound":-1,"GlobalRound"
:89,"ModelEncryAddr":"","Timestamp":0},"OP":{"OPType":0}}
Transaction has been submit, result is: OK
```

**Figure 5:** CheckValid execution result

Fig. 6 shows the result of adding model resource records to the blockchain ledger, obtained by calling the *AddModel*() method in the Model Storage Smart contract. The input parameters indicate adding the local model submitted by a certain client in communication layer 3 to the blockchain, with the client's current update round being 23 and the current global round being 90. Upon completion of the method, a success message is returned.

```
root@jtli-ubuntu:/home/gopath/src/github.com/FSAP/client/nodejs# node ./invoke
.js store AddModel '{"ModelType":-1,"ModelTier":3,"TierRound":23,"GlobalRound"
:90,"ModelEncryAddr":"CSSLXx0LwtOgVqX/L8ZTSAEH3TbMr7wGhoLeQFsnXdPlEHJ9qjsvlbxn
CFlb7iub","Timestamp":1697363262}'
Wallet path: /home/gopath/src/github.com/FSAP/client/nodejs/wallet
store AddModel {"ModelType":-1,"ModelTier":3,"TierRound":23,"GlobalRound":90,"
ModelEncryAddr":"CSSLXx0LwtOgVqX/L8ZTSAEH3TbMr7wGhoLeQFsnXdPlEHJ9qjsvlbxnCFlb7
iub","Timestamp":1697363262}
Transaction has been submit, result is: The model information has been on-chai
n successfully!
```

**Figure 6:** AddModel execution result

Fig. 7 illustrates the result of querying the global model for round 89 by executing the method *RequestModel*(). Upon successful querying, the *Model.ModelEncryAddr* is returned.

```
root@jtli-ubuntu:/home/gopath/src/github.com/FSAP/client/nodejs# node ./invoke
.js store RequestModel '{"ModelType":1,"ModelTier":-1,"TierRound":-1,"GlobalRo
und":89,"ModelEncryAddr":"","Timestamp":0}'
Wallet path: /home/gopath/src/github.com/FSAP/client/nodejs/wallet
store RequestModel {"ModelType":1,"ModelTier":-1,"TierRound":-1,"GlobalRound":
89,"ModelEncryAddr":"","Timestamp":0}
Transaction has been submit, result is: RpEjkNKHtGcaG5+XMZBGE1kjtukEX5Q3uJAzbZ
1muEHFqYk3pjPv7QeMeiMWzebQ
```

**Figure 7:** RequestModel execution result

Figs. 8 and 9 depict the results of an unauthorized access attempt. This unauthorized access was a random probe from a curious client. A client in communication layer 1 attempted to access a model in the global training round 92. The $AM.ModelID$ is computed from $SHA256\{ModelType: -1, ModelTier: 1, TierRound: 19, GlobalRound: 76\}$, where $ModelType$ indicates a request for the client's local model. Since access permissions are not registered for any local models on the chain, and only access permissions for global models are dynamically generated during training, this query returned an error message indicating that the access control policy does not exist, and the client was informed that access was denied.

Fig. 10a,b and Table 2 demonstrate the federated training results of our experiments, including the accuracy rate and global elapsed time of training. The experimental results show that the blockchain-based asynchronous tiered federated learning algorithm used in this scheme achieves similar accuracy rates as centralized training. Compared to off-chain training without blockchain, the main time loss in this scheme is due to the time consuming for clients to exchange model data with the blockchain network, the execution time of multiple smart contracts and the consensus verification time of various nodes of the blockchain network. Among them, the node consensus time consumption is not within the scope of this work. We will provide a detailed analysis of the time overhead of this scheme in subsequent Section 4.2.2.

### *4.2 Performance of the Scheme*

#### *4.2.1 Storage Performance*

To verify the advantages of the collaborative storage strategy in this scheme, we compared multiple strategies based on the performance metric of the storage space occupied by model resources saved on-chain. Assuming the number of client requests submitted per round is $C$, the number of on-chain nodes is $N$, and the total number of clients is $K$.

Our storage strategy is to reasonably compress the number of on-chain nodes based on the federated training hierarchy. Nodes on blockchain must store $N\,(C+2)$ copies of model resources. However, if node compression is not applied, $K\,(C+2)$ copies would need to be stored, and the difference in storage volume lies in the gap between $N$ and $K$. Therefore, although the relationship between blockchain networks and clients is not fixed in practical applications, we still recommend establishing a reasonable mapping between the number of clients and the number of blockchain nodes to save storage space while maintaining sufficient on-chain consensus.

```
root@jtli-ubuntu:/home/gopath/src/github.com/FSAP/client/nodejs# node ./invoke
.js fsapc QueryPolicy '{"AC":{"ClientTier":1,"GlobalRound":92},"AM":{"ModelID"
:"d01c09f7fc38542b320b5c6f26f774d1d857f5bffb0e374ddba2dc5112b68bc2"},"AE":0}'
Wallet path: /home/gopath/src/github.com/FSAP/client/nodejs/wallet
fsapc QueryPolicy {"AC":{"ClientTier":1,"GlobalRound":92},"AM":{"ModelID":"d01
c09f7fc38542b320b5c6f26f774d1d857f5bffb0e374ddba2dc5112b68bc2"},"AE":0}
2023-11-01T06:38:36.201Z - warn: [DiscoveryEndorsementHandler]: _build_endorse
_group_member >> G0:0 - endorsement failed - Error: Policy does not exist!
2023-11-01T06:38:36.202Z - warn: [DiscoveryEndorsementHandler]: _build_endorse
_group_member >> G1:1 - endorsement failed - Error: Policy does not exist!
2023-11-01T06:38:36.205Z - warn: [DiscoveryEndorsementHandler]: _build_endorse
_group_member >> G0:0 - endorsement failed - Error: Policy does not exist!
2023-11-01T06:38:36.206Z - warn: [DiscoveryEndorsementHandler]: _build_endorse
_group_member >> G1:1 - endorsement failed - Error: Policy does not exist!
2023-11-01T06:38:36.206Z - error: [DiscoveryEndorsementHandler]: _endorse - en
dorsement failed::Error: Endorsement has failed
```

**Figure 8:** QueryPolicy query failure

```
root@jtli-ubuntu:/home/gopath/src/github.com/FSAP/client/nodejs# node ./invoke
.js fsapc CheckValid '{"Client":{"ClientID":15,"ClientTier":1,"TierRound":28,"
GlobalRound":92},"Model":{"ModelType":-1,"ModelTier":1,"TierRound":19,"GlobalR
ound":76,"ModelEncryAddr":"","Timestamp":0},"OP":{"OPType":0}}'
Wallet path: /home/gopath/src/github.com/FSAP/client/nodejs/wallet
fsapc CheckValid {"Client":{"ClientID":15,"ClientTier":1,"TierRound":28,"Globa
lRound":92},"Model":{"ModelType":-1,"ModelTier":1,"TierRound":19,"GlobalRound"
:76,"ModelEncryAddr":"","Timestamp":0},"OP":{"OPType":0}}
2023-11-01T06:54:45.068Z - warn: [DiscoveryEndorsementHandler]: _build_endorse
_group_member >> G0:1 - endorsement failed - Error: Policy does not exist, acc
ess is denied!
2023-11-01T06:54:45.069Z - warn: [DiscoveryEndorsementHandler]: _build_endorse
_group_member >> G1:0 - endorsement failed - Error: Policy does not exist, acc
ess is denied!
2023-11-01T06:54:45.074Z - warn: [DiscoveryEndorsementHandler]: _build_endorse
_group_member >> G0:1 - endorsement failed - Error: Policy does not exist, acc
ess is denied!
2023-11-01T06:54:45.075Z - warn: [DiscoveryEndorsementHandler]: _build_endorse
_group_member >> G1:0 - endorsement failed - Error: Policy does not exist, acc
ess is denied!
2023-11-01T06:54:45.075Z - error: [DiscoveryEndorsementHandler]: _endorse - en
dorsement failed::Error: Endorsement has failed
```

**Figure 9:** CheckValid verification failure

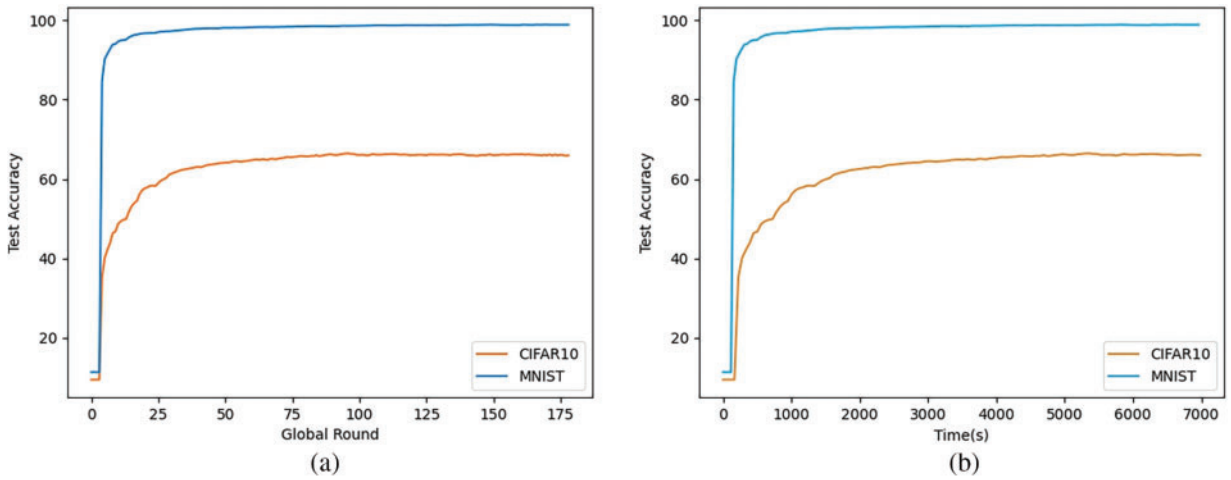


**Figure 10:** On-chain federated training performance. (**a**) On-chain federated training accuracy with global training rounds; (**b**) On-chain federated training accuracy over time

**Table 2:** Training results on two datasets

| Dataset | On-chain federated training accuracy (%) | Off-chain federated training accuracy (%) | Centralized training accuracy (%) | On-chain federated training time (s) |
|---|---|---|---|---|
| MNIST | 98.85 | 98.76 | 98.92 | 8061 |
| CIFAR10 | 66.45 | 66.94 | 69.74 | 10,374 |

For the hybrid storage strategy and the blockchain-only storage strategy, we only consider the storage space occupied by model resources, excluding client identities and dynamic access control policy data stored on the blockchain ledger. The model scale chosen for federated learning in this scheme is variable without an upper limit, and we conducted experiments using a model based on the CIFAR-10 dataset. The size of the model resource files used in this experiment is 895 KB, and according to statistics, each resource record on-chain requires storage space within 200 bytes. In the experimental setup, the global model converges around 100 rounds, and we observed the on-chain storage performance of the three storage strategies before 100 rounds during the experiment. As shown in Fig. 10a, before the accuracy reaches 0.6, we recorded the consumption of on-chain storage space for every 0.1 increase in accuracy. In the later stages of training, as the improvement in accuracy slows with the global training rounds after the accuracy reaches 0.6, we recorded the consumption of on-chain storage space for every 0.01 increase in accuracy. The graph shows that as the global model accuracy improves, the consumption of on-chain storage resources under the blockchain-only strategy increases sharply. In contrast, under the collaborative storage strategy used in this scheme, the consumption of on-chain storage resources changes very gradually. In Fig. 11b, we recorded the on-chain storage space consumption at global rounds 20, 40, 60, 80, and 100, showing that under the collaborative storage strategy, as the global rounds increase, less storage space is consumed on blockchain.

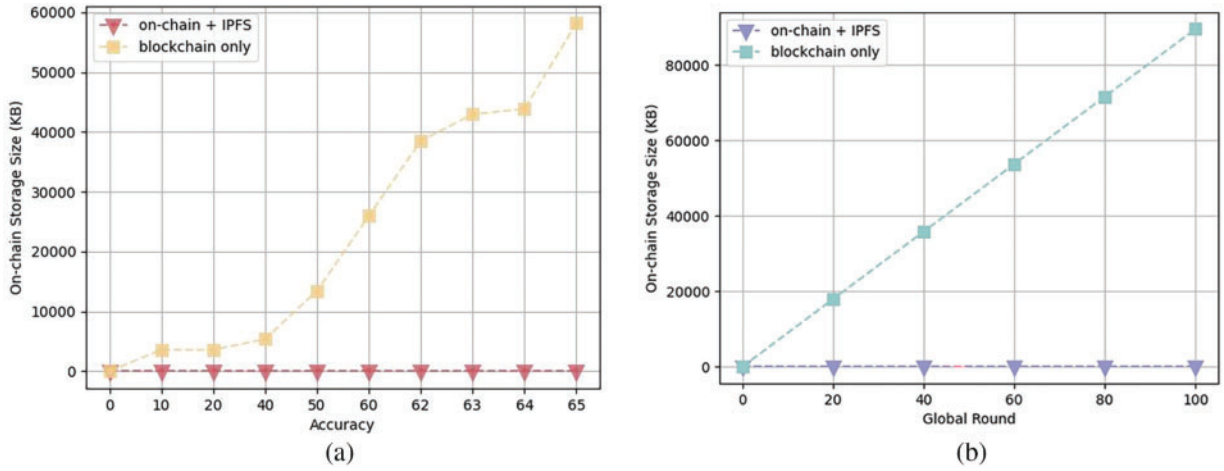


**Figure 11:** On-chain storage space consumption. (**a**) On-chain storage space consumption with accuracy changes; (**b**) On-chain storage space consumption with global rounds change

The experimental results demonstrate that with the increase in global rounds and improvement in global model accuracy, the growth of on-chain storage space under the collaborative storage strategy is significantly

lower than under the blockchain-only strategy, which indicates that the hybrid storage strategy in this scheme effectively reduces the storage burden on the blockchain, enhancing the scheme's storage performance.

*4.2.2 Communication Overhead*

To evaluate the communication overhead of the scheme, we considered the main steps in the federated training process. Specifically, apart from the necessary time spent on federated aggregation, the remaining overhead can be divided into the following three aspects: (request-response) overhead between clients and the blockchain, execution overhead of access control smart contracts, and communication overhead of model storage, including the execution of relevant smart contracts and communication between parties and the IPFS server.

We use $C$ to represent the threshold value of the number of client requests each tier of the federated training, $|T_P|$ to represent the transmission time of the request or response between clients and the blockchain network, $|T_{AC}|$ to represent the execution overhead of access control smart contracts, $|T_{SC}|$ to represent the execution overhead of model storage smart contracts, $|T_U|$ for the time overhead of uploading model files to IPFS, $|T_D|$ for the time overhead of retrieving model files from IPFS and $|T_M|$ to represent the time overhead of directly transmitting model files. In our experiments, the overheads' average time consumption was calculated as shown in Table 3.

**Table 3:** Average time consumption statistics of various overheads in the communication process

| Notation | Average time spent (s) |
|---|---|
| $\|T_P\|$ | 0.056 |
| $\|T_{AC}\|$ | 0.5588 |
| $\|T_{SC}\|$ | 0.6113 |
| $\|T_U\|$ | 0.2452 |
| $\|T_D\|$ | 0.180 |
| $\|T_M\|$ | 1.5616 |

It is assumed that client requests do not always arrive at the blockchain simultaneously in the asynchronous scenario. In each round of training, each client needs to request the global model and upload the local model, with a transmission overhead of $2C|T_P|$, each request needs to undergo access control verification, incurring an overhead of $2C|T_{AC}|$. Each round of training needs to operate $C$ local models, save a new layer model and a new global model, which incurs storage overhead of $(C+2)|T_{SC}|$. Additionally, the model files from clients and the blockchain network need to be stored on IPFS, incurring storage overhead of $(C+2)|T_U|$, each client needs to download the global model file from IPFS, and the blockchain also needs to download the local model files from each client, resulting in a total overhead of $2C|T_D|$. Therefore, the total additional overhead for one round of training in our scheme is $T = 2C|T_p| + 2C|T_{AC}| + (C+2)|T_{SC}| + (C+2)|T_U| + 2C|T_D| = 2.4461C + 2.7130\,(\text{ms})$. In comparison, the total overhead without using the collaborative storage strategy is $T' = 2C|T_M| + C|T_{AC}| + (C+2)|T_{SC}| = 4.2933C + 1.2226\ (\text{ms})$, Compared to $T$, the overhead of IPFS storage has been reduced while the overhead of direct transmission of model files between clients and the blockchain has been increased. In the case of federated learning alone without blockchain, the total overhead for one round of training is $T'' = 2C|T_M| = 3.1232C\,(\text{ms})$. Comparing the differences between $T$ and $T'$, it is evident that this scheme has significant advantages over the storage strategy solely based on the blockchain. Particularly, when federated learning adopts models with larger

parameter sizes or when the number of client nodes significantly increases, the advantage of using the storage strategy in reducing communication overhead becomes more pronounced. Additionally, the overhead of smart contracts added for security purposes in this scheme is within an acceptable range, providing the scheme with higher security guarantees.

Suppose we consider more specific scenarios, a large number of client requests may simultaneously reach the blockchain at certain moments, a situation more likely to occur when many clients participate in training. Therefore, we need to test the performance of the smart contracts designed in this scheme in high-concurrency scenarios. We set the number of concurrent transactions to be within 1000 and tested the throughput of each smart contract method at intervals of 100. The experimental results are shown in Fig. 12a,b. Based on the experimental results, it can be observed that the throughput of each smart contract method is not significantly affected by the increase in the number of concurrent transactions, indicating that the performance of the smart contracts in this scheme in high-concurrency scenarios can be considered relatively stable.

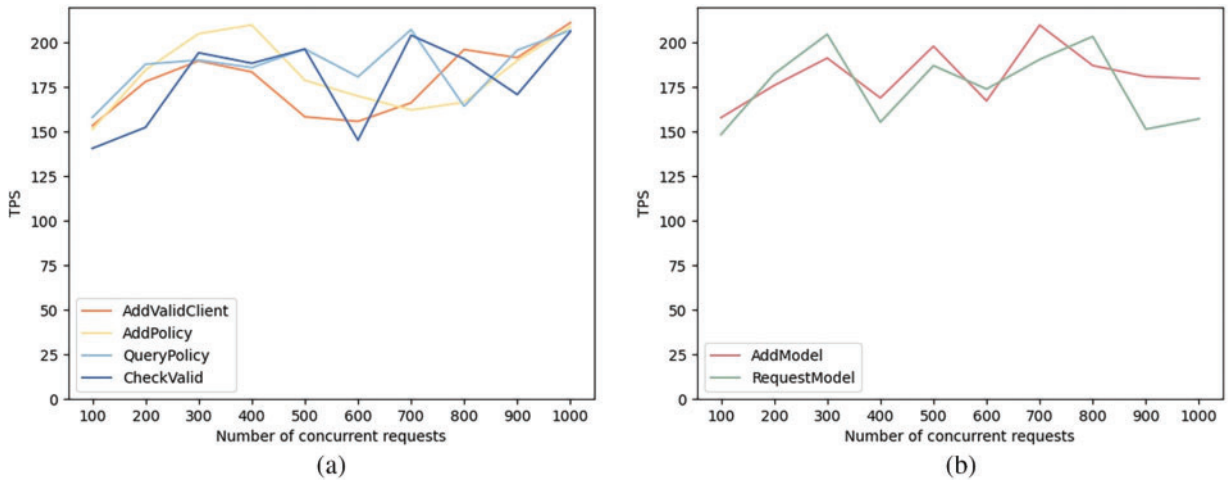


**Figure 12:** Throughput of smart contract methods at various levels of concurrency. (**a**) The throughput of each method in the Access Control smart contract at different levels of concurrency; (**b**) The throughput of each method in the Model Stor-age smart contract at different levels of concurrency

### *4.3 Security*

Table 4 provides a detailed comparison of the implementation of various security measures between this scheme and other federated learning frameworks.

**Table 4:** Security comparison of schemes

| Security | Our scheme | Other FL framework |
|---|---|---|
| Model training security | A fully decentralized scheme that strictly protects model resources to reduce the likelihood of attacks, preventing adversaries from joining the federated training process. | Utilize secure federated training algorithms. |

(Continued)

**Table 4 (continued)**

| Security | Our scheme | Other FL framework |
|---|---|---|
| Client access security | Implementing client authentication and access control for client requests using access control technology. | Not the focus of our scheme. |
| Aggregation algorithm security | Not the focus of our scheme. | Utilize methods such as differential privacy and homomorphic encryption, but they may impact the accuracy and effectiveness of model training. |
| Resource sharing security | Utilize a fair decentralized smart contract solely responsible for resource storage, balancing flexibility and security. | Traditional Federated Learning does not address this aspect. Blockchain-based Federated Learning does not employ access control mechanisms to manage on-chain resources flexibly. |

Table 5 comprehensively compare the security aspects of this scheme with other FL frameworks. Table 4 highlights this scheme's innovative perspectives and approaches towards securing federated training.

**Table 5:** Security comparison between this scheme and other federated learning frameworks

| Security | [11] | [12] | [23] | [35] | Our scheme |
|---|---|---|---|---|---|
| Model training security | ✓ | ✓ | ✓ | ✓ | ✓ |
| Client access security | × | × | × | × | ✓ |
| Aggregation algorithm security | ✓ | ✓ | × | × | × |
| Resource sharing security | × | × | ✓ | ✓ | ✓ |

The security requirements met by this program are listed below:

1. **Response to threat scenarios assumed by this scheme:** The experimental results shown in Figs. 7 and 8 demonstrate that in this experiment, illegal access to resources cannot pass the verification of the access control module, cutting off the attacker's attack chain at its root, preventing them from accessing any on-chain data sources. Therefore, attacks such as model theft or any attacks based on model parameters during the training process are ineffective against this scheme.
2. **Resource anonymity:** The true attributes of model resources are hidden in the $ModelID$, computed by SHA-256 based on the attributes of the model resources. SHA-256 is a secure one-way hash function that protects the anonymity of model resources.
3. **Traceability:** Client requests to the blockchain can be recorded in the blockchain ledger. Expressly, any model resource clients submit will be saved to the blockchain. Blockchain administrators can trace any improper behavior back to any client node based on this information.

4. **Data integrity protection:** The integrity of on-chain recorded data is maintained by the blockchain's distributed consensus mechanism, while the integrity of real model resources is maintained by IPFS's content-based hash verification and distributed storage mechanism.

### *4.4 Scalability*

As mentioned above, the proposed scheme exhibits high storage and communication performance, enabling it to accommodate many clients participating in federated learning. Moreover, this scheme is designed based on blockchain technology, independent of any specific blockchain platform, thus possessing high portability. Additionally, the scheme is fully decentralized, the smart contracts on the blockchain are also pluggable, and clients participating in federated learning can utilize any machine learning model and optimization method. Therefore, the scheme proposed in this paper demonstrates high scalability.

## 5 Summary and Outlook

To minimize untrustworthy issues during the federated training process, we propose an asynchronous tiered federated learning storage solution based on blockchain and IPFS. Experiments and analyses conducted on various datasets and settings demonstrate the efficiency, security, and scalability of our scheme. We address security concerns at the central aggregator node and along the path from clients to the aggregator in the federated training process. Leveraging smart contracts on the blockchain network for federated aggregation, the paper achieves a fully decentralized architecture for asynchronous hierarchical federated training. The access control model designed effectively enhances supervision over client access and ensures secure model resource access. Additionally, a hybrid on-chain and off-chain model storage strategy is employed to alleviate storage burdens on the blockchain network and enhance the overall communication efficiency of the solution.

In the future, we plan to incorporate secure federated aggregation algorithms into the scheme to enhance the robustness of models against attack behaviors. Additionally, we will attempt to deploy our scheme in specific scenarios through open-source channels to observe its real-time operation.

**Acknowledgement:** The authors are grateful to all the editors and anonymous reviewers for their detailed review and insightful advice.

**Funding Statement:** This research is partially supported by the National Natural Science Foundation of China (Grant No. 52331012), the Natural Science Foundation of Shanghai Municipality (Grant No. 21ZR1426500), and the Program for Cultivation of Graduate Students' Top-notch Innovative Talents of Shanghai Maritime University (Grant No. 2023YBR007).

**Author Contributions:** The authors confirm contribution to the paper as follows: conceptualization: Tianyu Li, Dezhi Han, Kuan-Ching Li; methodology and software: Tianyu Li; funding acquisition: Dezhi Han; performance of the analysis: Jiatao Li. All authors reviewed the results and approved the final version of the manuscript.

**Availability of Data and Materials:** Not applicable.

**Ethics Approval:** Not applicable.

**Conflicts of Interest:** The authors declare no conflicts of interest to report regarding the present study.

## References


1. McMahan B, Moore E, Ramage D, Hampson S, Arcas B. Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS); 2017 Apr 20–22; Fort Lauderdale, FL, USA. p. 1273–82.
2. Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar A, Smith V. Federated optimization in heterogeneous networks. In: Proceedings of Machine Learning and Systems; 2020 Mar 2–4; Austin, TX, USA. p. 429–50.
3. Karimireddy SP, Kale S, Mohri M, Reddi S, Stich S, Suresh AT. Scaffold: stochastic controlled averaging for federated learning. In: Proceedings of the 37th International Conference on Machine Learning; 2020 Jul 13–18: Online. p. 5132–43. [cited 2025 Jan 1]. Available from: https://icml.cc/Conferences/2020.
4. Xie C, Koyejo S, Gupta I. Asynchronous federated optimization. arXiv:1903.03934. 2019.
5. Chen Y, Ning Y, Slawski M, Rangwala H. Asynchronous online federated learning for edge devices with non-IID data. In: 2020 IEEE International Conference on Big Data (Big Data); 2020 Dec 10–13; Atlanta, GA, USA. p. 15–24. doi:10.1109/bigdata50022.2020.9378161.
6. Chai Z, Chen Y, Anwar A, Zhao L, Cheng Y, Rangwala H. FedAT: a high-performance and communication-efficient federated learning system with asynchronous tiers. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis; 2021 Nov 14–19; St. Louis, MO, USA. p. 1–16. doi:10.1145/3458817.3476211.
7. Chai Z, Ali A, Zawad S, Truex S, Anwar A, Baracaldo N, et al. TiFL: a tier-based federated learning system. In: Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing; 2020 Jun 23–26; Stockholm, Sweden. p. 125–36. doi:10.1145/3369583.3392686.
8. Lyu L, Li Y, Nandakumar K, Yu J, Ma X. How to democratise and protect AI: fair and differentially private decentralised deep learning. IEEE Trans Dependable Secure Comput. 2022;19(2):1003–17. doi:10.1109/TDSC.2020.3006287.
9. Truex S, Baracaldo N, Anwar A, Steinke T, Ludwig H, Zhang R, et al. A hybrid approach to privacy-preserving federated learning. In: Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security; 2019 Nov 11–19; London, UK. p. 1–11. doi:10.1145/3338501.3357370.
10. Goel A, Agarwal A, Vatsa M, Singh R, Ratha N. DeepRing: protecting deep neural network with blockchain. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW); 2019 Jun 16–17; Long Beach, CA, USA. p. 2821–8. doi:10.1109/cvprw.2019.00341.
11. Mondal A, Virk H, Gupta D. BEAS: blockchain enabled asynchronous & secure federated machine learning. arXiv:2202.02817. 2022.
12. 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(5):1092–103. doi:10.1109/TC.2021.3072033.
13. Zhou S, Li K, Chen Y, Yang C, Liang W, Zomaya AY. TrustBCFL: mitigating data bias in IoT through blockchain-enabled federated learning. IEEE Internet Things J. 2024;11(15):25648–62. doi:10.1109/JIOT.2024.3379363.
14. Wang X, Garg S, Lin H, Kaddoum G, Hu J, Hossain MS. A secure data aggregation strategy in edge computing and blockchain-empowered internet of things. IEEE Internet Things J. 2022;9(16):14237–46. doi:10.1109/JIOT.2020.3023588.
15. Wang X, Garg S, Lin H, Kaddoum G, Hu J, Hassan MM. Heterogeneous blockchain and AI-driven hierarchical trust evaluation for 5G-enabled intelligent transportation systems. IEEE Trans Intell Transp Syst. 2023;24(2):2074–83. doi:10.1109/TITS.2021.3129417.
16. Sandhu RS, Samarati P. Access control: principle and practice. IEEE Commun Mag. 1994;32(9):40–8. doi:10.1109/35.312842.
17. Li J, Han D, Weng TH, Wu H, Li KC, Castiglione A. A secure data storage and sharing scheme for port supply chain based on blockchain and dynamic searchable encryption. Comput Stand Interfaces. 2025;91:103887. doi:10.1016/j.csi.2024.103887.
18. Liang W, Xie S, Cai J, Wang C, Hong Y, Kui X. Novel private data access control scheme suitable for mobile edge computing. China Commun. 2021;18(11):92–103. doi:10.23919/JCC.2021.11.007.

19. Liu H, Han D, Li D. Fabric-IoT: a blockchain-based access control system in IoT. IEEE Access. 2020;8:18207–18. doi:10.1109/ACCESS.2020.2968492.
20. Han D, Zhu Y, Li D, Liang W, Souri A, Li KC. A blockchain-based auditable access control system for private data in service-centric IoT environments. IEEE Trans Ind Inform. 2022;18(5):3530–40. doi:10.1109/TII.2021.3114621.
21. Li J, Han D, Wu Z, Wang J, Li KC, Castiglione A. A novel system for medical equipment supply chain traceability based on alliance chain and attribute and role access control. Future Gener Comput Syst. 2023;142:195–211. doi:10.1016/j.future.2022.12.037.
22. Jiang R, Li J, Bu W, Shen X. A blockchain-based trustworthy model evaluation framework for deep learning and its application in moving object segmentation. Sensors. 2023;23(14):6492. doi:10.3390/s23146492.
23. Wang T, Du M, Wu X, He T. An analytical framework for trusted machine learning and computer vision running with blockchain. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW); 2020 Jun 14–19; Seattle, WA, USA. p. 32–8. doi:10.1109/cvprw50498.2020.00011.
24. Liang W, Xie S, Li KC, Li X, Kui X, Zomaya AY. MC-DSC: a dynamic secure resource configuration scheme based on medical consortium blockchain. IEEE Trans Inf Forensics Secur. 2024;19:3525–38. doi:10.1109/TIFS.2024.3364370.
25. Narayanan A, Bonneau J, Felten E, Miller A, Goldfeder S. Bitcoin and cryptocurrency technologies: a comprehensive introduction. Princeton, NJ, USA: Princeton University Press; 2017.
26. Nakamoto S. Bitcoin: a peer-to-peer electronic cash system. Dhaka, Bangladesh: HN Publishing; 2008. doi:10.2139/ssrn.3440802.
27. Androulaki E, Barger A, Bortnikov V, Cachin C, Christidis K, De Caro A, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference; 2018 Apr 23–26; Porto Portugal. p. 1–15. doi:10.1145/3190508.3190538.
28. Hu VC, Kuhn DR, Ferraiolo DF, Voas J. Attribute-based access control. Computer. 2015;48(2):85–8. doi:10.1109/MC.2015.33.
29. Benet J. IPFS—content addressed, versioned, P2P file system. arXiv:1407.3561. 2014.
30. Melis L, Song C, De Cristofaro E, Shmatikov V. Exploiting unintended feature leakage in collaborative learning. In: 2019 IEEE Symposium on Security and Privacy (SP); 2019 May 19–23; San Francisco, CA, USA. p. 691–706. doi:10.1109/sp.2019.00029.
31. Clements J, Lao Y. Backdoor attacks on neural network operations. In: 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP); 2018 Nov 26–29; Anaheim, CA, USA. p. 1154–8. doi:10.1109/GlobalSIP.2018.8646335.
32. Rakin AS, He Z, Fan D. TBT: targeted neural network attack with bit Trojan. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2020 Jun 13–19; Seattle, WA, USA. p. 13195–204. doi:10.1109/cvpr42600.2020.01321.
33. LeCun Y, Cortes C. The mnist database of handwritten digits [Internet]. [cited 2025 Mar 30]. Available from: https://api.semanticscholar.org/CorpusID:60282629.
34. Krizhevsky A. Learning multiple layers of features from tiny images [Internet]. [cited 2025 Mar 30]. Available from: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.
35. Wang X, Zhao Y, Qiu C, Liu Z, Nie J, Leung VCM. InFEDge: a blockchain-based incentive mechanism in hierarchical federated learning for end-edge-cloud communications. IEEE J Sel Areas Commun. 2022;40(12):3325–42. doi:10.1109/JSAC.2022.3213323.