TrustHFL: An Efficient Aggregation Method for Trustworthy Hierarchical Federated Learning

Sisi Zhou, Yuxiang Chen, Wei Liang, *Senior Member, IEEE*, Kuanching Li, *Senior Member, IEEE,* and Weizhi Meng, *Senior Member, IEEE*

***Abstract*—The hierarchical federated learning framework significantly reduces communication burdens on central servers; however, the integration of edge servers introduces potential risks such as Single Point of Failure (SPOF) and ongoing challenges like imbalanced data distribution. To tackle these challenges, we propose TrustHFL, an innovative aggregation method designed for secure hierarchical federated learning. TrustHFL enhances training efficiency by employing group training that clusters clients with similar data distribution characteristics. Inside each cluster, synchronous aggregation is implemented, while asynchronous aggregation is utilised between clusters to alleviate delays from bottleneck clients. We also introduce a robust access control mechanism for secure interactions between clients and edge servers, ensuring data privacy and system integrity. Moreover, our design favours off-chain computation and training, limiting on-chain storage to essential information and thereby minimising both storage and computational demands on the blockchain, ultimately enhancing training efficiency. Extensive experimental results demonstrate that the proposed method accelerates convergence speed and enhances model accuracy. Compared to existing classical federated learning methods, the model accuracy is improved by an average of 1.98% under various data distribution scenarios, while the time required to achieve the same accuracy is reduced by an average of 65.54%.**

***Index Terms*—Hierarchical Federated Learning, Blockchain, Access Control, Synchronous Aggregation, Asynchronous Aggregation.**

I. Introduction

FEDERATED learning (FL) is a privacy-preserving distributed machine learning framework, where participants train models on their local devices, enabling decentralised data storage and processing, thus improving the privacy of local data. It has been widely applied in privacy-sensitive domains, such as personalised recommendation systems, smart cities, and intelligent healthcare [1] [2]. However, traditional FL is a typical single-layer centralised architecture that relies on a central server to generate or update global model parameters, resulting in issues such as a single point of failure, privacy leakage, and performance bottlenecks [3].

To effectively reduce the communication overhead of the central server, some studies [4] [5] [6]have introduced edge computing into FL, constructing a hierarchical federated learning (HFL) framework. Although the HFL framework is more suitable for large-scale cross-domain collaborative applications [7], it still faces some issues, as shown in Figure 1.

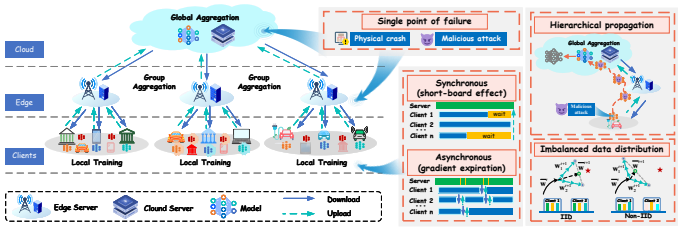


Fig. 1. Problems and Challenges of Hierarchical Federated Learning Framework.

- **Data Distribution.** Non-independent and identically distributed (NonIID) data across different devices will affect the model's performance.
- **Centralized Architecture.** The trustworthiness of the HFL framework is fundamentally constrained by a centralised trust system, which inherently has a single point of failure. In addition, including edge servers increases the points of failure, and the effects of attacks can propagate hierarchically, making defense more challenging.
- **Model Aggregation.** Synchronous aggregation [8] and asynchronous aggregation [9] are two well known aggregation methods. Synchronous aggregation may encounter bottleneck clients, which can slow down the entire process. While asynchronous aggregation does not require clients to wait for each other, it may lead to stale gradients, resulting in slower or unstable convergence. HFL, involving many devices and a broader scope, further exacerbates the limitations of synchronous and asynchronous methods.

To address the issue of imbalanced data distribution, a common solution is to implement local training schemes on the client side [10]. However, this does not improve the average performance of the federated learning model. Another approach involves modifying the model aggregation method

This work was supported in part by the Joint Key Project of National Natural Science Foundation of China under Grant U2468205, the National Natural Science Foundation of China under Grant 62202156 and Grant 62472168, in part by the Hunan Provincial Key Research and Development Program under Grant 2024AQ2028, in part by the Hunan Provincial Natural Science Foundation of China under Grants 2025JJ50399 and 2024JJ6220, and in part by the Research Foundation of Education Bureau of Hunan Province, China under Grants 23B0487 and 23B0449.

Corresponding authors: Y. Chen, K. Li and W.Liang

Sisi Zhou, Yuxiang Chen, and Kuanching Li are with the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, Hunan Engineering Research Center of Software Design and Services of Intelligent Connected Vehicles, Xiangtan 411201, China. E-mail: sisizhou@mail.hnust.edu.cn, chenyuxiang@hnust.edu.cn, aliric@hnust.edu.cn

Wei Liang is with the University of South China, Hengyang, 421001, China. Email: wliang@hnust.edu.cn

Weizhi Meng is with the School of Computing and Communications, Lancaster University, United Kingdom. Email: weizhi.meng@ieee.org

on the server, such as normalising local model updates [11] or introducing adaptive momentum [12], though these methods lack universality. Additionally, some approaches attempt to solve the imbalanced data distribution problem by sharing data [13], which contradicts the privacy-preserving intention of FL. Therefore, resolving the issue of imbalanced data distribution among FL clients remains a key research focus.

To address the issue of a single point of failure, a common approach is to leverage blockchain technology to provide a decentralised trust mechanism [14] [15]. Li *et al.* [16] and Peng *et al.* [17] have focused on storing model updates on-chain to mitigate the issue of a single point of failure in central servers. However, these studies have not considered the impact of consensus mechanisms on training efficiency or the storage pressures arising from excessive redundant data. Subsequently, a framework that integrates dual-layer blockchain with federated learning [18] to improve training efficiency emerged. However, this architecture faces new challenges related to the overhead of cross-chain interactions and privacy leakage issues.

Although there are many common privacy methods for the data itself, such as homomorphic encryption [19], secure multi-party computation [20], and differential privacy [21], both homomorphic encryption and secure multi-party computation have high computational complexity and communication overhead, making them difficult to meet the demands of large-scale data training. Differential privacy can resist inference attacks but significantly reduces model accuracy. On the other hand, for model updates, consensus mechanisms can be used for model selection [22], enhancing system security, but they come with considerable overhead. Additionally, distance-based detection methods [23] have a single evaluation type, making them unsuitable for direct application in HFL frameworks [24].

Moreover, the aforementioned approaches predominantly rely on synchronous FL, which requires all clients to complete local model training before updating the global model, resulting in bottleneck clients that can reduce training efficiency. Although existing studies [25] have proposed asynchronous FL to address this issue, these approaches can result in stale models, which negatively impact convergence performance. Therefore, finding ways to enhance model training efficiency and reduce communication overhead while protecting data privacy remains a critical challenge for the sustainable development of HFL.

This work develops a HFL framework with awareness of data distribution using technologies such as blockchain, access control, and smart contracts. The key contributions of this work are as follows.

- We proposed a group aggregation method based on local data distribution and device training efficiency, enabling each group to learn more comprehensive features while reducing the global broadcasting of local data.
- We designed a hybrid aggregation approach that combines synchronous aggregation within groups and asynchronous aggregation between groups, effectively alleviating the impact of bottleneck clients and outdated models on convergence speed and accuracy. Additionally, inter-group aggregation takes into account the influence of the previous round's global model, compensating for information loss in individual group models due to imbalanced data distribution.
- We proposed an efficient and secure model storage and access control mechanism incorporating blockchain technology to address a single point of failure. It utilises smart contracts for authentication and grouping, preventing unauthorised clients from stealing private information. Additionally, the training and model interaction processes are strategically designed to occur off-chain, reducing storage and communication overhead while improving training efficiency.
- We conducted a security analysis and performed experiments on various models and datasets. The experiments indicate that the proposed method improves model accuracy by an average of 1.98% across different data distribution scenarios, while the time required to achieve the same accuracy is reduced by an average of 65.54%.

The remainder of this article is organised as follows. Section 2 reviews the related work, while Section 3 concisely introduces the preliminaries and problem formulation. Section 4 then elaborates on the proposed method. Section 5 compares the experimental results with traditional FL methods and provides analysis, and finally, concluding remarks and future research directions are given in Section 6.

II. Related Work

FL is a distributed machine learning framework. McMahan *et al.* [8] demonstrated through experiments that FL effectively addresses data privacy and system performance requirements, showing great potential in applications involving sensitive data. However, challenges remain in terms of communication overhead, training efficiency, and security.

A. Hierarchical Federated Learning

Hierarchical model aggregation can effectively alleviate the communication pressure on the central server in FL and enhance flexibility. Chai *et al.* [26] proposed a HFL system based on node training speeds, but it did not address security issues. Luo *et al.* [27] is a HFL system associated with edge computing that primarily focuses on network bandwidth. Liu *et al.* [28] optimised data distribution distance, emphasising resource allocation yet without considering security issues. To balance data skew among mobile nodes, Mhaisen *et al.* [29] proposed minimising distribution distances between edge nodes to mitigate accuracy degradation caused by imbalanced data. However, research on HFL remains insufficient, particularly in terms of studies that comprehensively address security and data distribution imbalance issues.

B. Model Aggregation

Existing FL approaches typically employ synchronous aggregation, which can lead to unnecessary waiting due to bottleneck clients. Additionally, when a large number of clients simultaneously upload gradient parameters, communication congestion can occur. To ensure successful collaborative

training even with partial node failures, Zheng *et al.* [30] proposed asynchronous communication using asynchronous gradient descent; however, delayed gradients can affect model accuracy. Jang *et al.* [31] incorporated pruning strategies into FL, effectively reducing communication overhead at the cost of model precision. Lu *et al.* [32] utilised an adaptive gradient compression algorithm to mitigate privacy leakage risks, yet client training states remain imbalanced. Lu *et al.* [33] constructed a decentralised asynchronous FL framework, but they still employed synchronous updates for the global model. Based on the aforementioned facts, research on asynchronous FL remains insufficient, particularly regarding the balance among model reliability, accuracy, and training efficiency.

C. Security in Federated Learning

Utilising blockchain technology to replace a central server for constructing a trustworthy FL framework is an innovative solution. Kim *et al.* [34] proposed a general architecture combining blockchain and FL, one of the earlier studies in this field. It enhances the credibility of the global model by cross-validating local model update parameters, although the consensus mechanism incurs significant overhead. Subsequently, Li *et al.* [35] addressed the cost issue of blockchain by designing different block structures for storage to reduce the storage burden on participating nodes. To further address data privacy concerns, Lu *et al.* [36] applied differential privacy to perturb local training data and constructed a group consensus for validating the quality of training to ensure the credibility of the local and global models stored on the blockchain. Ji *et al.* [37] established a consensus committee based on each node's contribution to verify participant identities and assess model quality. Wang *et al.* [38] proposed the Sym-CS-HFL framework, which integrates symmetric homomorphic encryption and a local adaptive aggregation scheme to reduce reliance on asymmetric keys, simplify the encryption process, and lower computational overhead. However, all these approaches still need to resolve issues related to storage overhead and efficiency.

III. Preliminaries and Problem Formulation

A. Federated Learning

FL typically involves model distribution, local training, and model aggregation, followed by global updating. Assuming there are K clients participating in the FL task, let the local dataset of the k-th client (where $k = 1, 2, \ldots, K$) be denoted as D_k, with n_k samples. The global dataset for FL is represented as the union of the datasets from all clients, denoted as $D = \cup_{k=1}^{K} D_k$.

Let the local model of the k-th client in the t-th round be represented as $\mathbf{w}_k^{(t)}$, and the global model as $\overline{\mathbf{w}}^{(t)}$. The local objective function for the k-th client is defined as:

$$F_k\left(\mathbf{w}\right) = \frac{1}{n_k^{(t)}} \sum_{\xi \in D_k^{(t)}} f\left(\mathbf{w}_k^{(t)}, \xi_k^{(t)}\right), \tag{1}$$

where $f\left(\mathbf{w}_k^{(t)}, \xi_k^{(t)}\right)$ is the loss function for a single sample $\xi_k^{(t)}$ and the local model weights $\mathbf{w}_k^{(t)}$. Typically, model training employs a gradient descent algorithm, which updates the model parameters in the direction that most rapidly reduces the loss. Thus, the local model update is given by:

$$\begin{aligned} \mathbf{w}_k^{(t)} &= \mathbf{w}_k^{(t-1)} - \eta_k^{(t)} g_k^{(t)}\left(\mathbf{w}_k^{(t)}, D_k^{(t)}\right) \\ &= \mathbf{w}_k^{(t-1)} - \eta_k^{(t)} \frac{1}{\left|D_k^{(t)}\right|} \sum_{\xi \in D_k^{(t)}} \nabla f\left(\mathbf{w}_k^{(t)}, \xi_k^{(t)}\right), \end{aligned} \tag{2}$$

where $g_k^{(t)}\left(\mathbf{w}_k^{(t)}, D_k^{(t)}\right)$ the local gradient for the k-th client represents the learning rate for the local model iteration in the t-th round. In the case of weighted averaging, the global model $\overline{\mathbf{w}}^{(t)}$ is given by:

$$\begin{aligned} \overline{\mathbf{w}}^{(t)} &= \overline{\mathbf{w}}^{(t-1)} - \eta^{(t)} \nabla \overline{\mathbf{g}}^{(t)} \\ &= \overline{\mathbf{w}}^{(t-1)} - \eta^{(t)} \sum_{k=1}^{K} \frac{1}{K} \mathbf{g}_k^{(t)}\left(\mathbf{w}_k^{(t)}, D_k^{(t)}\right), \end{aligned} \tag{3}$$

where $\overline{\mathbf{g}}^{(t)}$ is the global gradient, $\eta^{(t)}$ representing the global learning rate.

B. Data Distribution Issue

In FL, data is stored across various local devices, leading to significant variations in data type, quantity, and quality, as well as differing distribution characteristics and correlations.

Let $(D = \cup_{k=1}^{K} D_k \sim P)$ represents the global dataset for FL, where $\mathcal{P}$ denotes the data distributions for the global dataset. Let $\mathcal{P}_k$ represent the data distributions for the client dataset. Based on the summary by Kairouz *et al.* [39], the NonIID data distribution for clients includes the following situations:

- Feature distribution skew. This refers to the case where the feature distributions of two clients are different, i.e., $\mathcal{P}_k(x) \neq \mathcal{P}_{k'}(x)$, while the label distributions remain the same, i.e., $\mathcal{P}_k(y \mid x) = \mathcal{P}_{k'}(y \mid x)$.
- Label distribution skew: This refers to the situation where the label distributions of two clients are different, i.e., $\mathcal{P}_k(y) \neq \mathcal{P}_{k'}(y)$, while the feature distributions remain the same, i.e., $\mathcal{P}_k(x \mid y) = \mathcal{P}_{k'}(x \mid y)$.
- Same label, different features: this occurs when two clients have the same label but different features, *i.e.*, $x_k^i \neq x_{k'}^j$, while $y_k^i = y_{k'}^j$.
- Same features, different labels: this occurs when two clients have the same features but different labels, *i.e.*, $x_k^i = x_{k'}^j$, while $y_k^i \neq y_{k'}^j$.
- Quantity skew or unbalancedness: this refers to an imbalance in the amount of data available to different clients, *i.e.*, $n_k \neq n_{k'}$.

It is usual practice to simulate non-IID data among different clients in experiments on FL.

C. Problem Definition

Hierarchical model aggregation is an emerging FL framework that extends traditional single-layer FL, effectively reducing the load on the central server. However, the issue of data distribution imbalance among distributed nodes requires

attention. TrustHFL considers grouping clients for training and group aggregation to accelerate the training process while ensuring security.

In a HFL scenario involving K clients and E edge servers, assuming that the data of clients is typically associated with the geographical area and the attributes of the clients themselves. The edge servers $E = \{e_1, e_2, ..., e_E\}$are divided into S groups, with one edge server randomly selected as the edge aggregator from each group. The $K = \{k_1, k_2, ..., k_K\}$ clients are classified into $S = \{G_1, G_2, ..., G_S\}$ groups based on data distribution and training efficiency, with each group corresponding to a specific edge server group. Let $epoch_k$ represents local training epochs, $epoch_G$ edge aggregations epochs.

In the proposed TrustHFL framework, our goal can be articulated as maximising the diversity of data distributions associated with edge nodes while mitigating the impact of bottleneck clients on the overall training process. To achieve a more balanced data distribution for each edge aggregator, we aim to associate distributed nodes with different data distributions with the same edge aggregator to supplement the imbalanced data.

Let $e_{Agg} \in \{0, 1\}$ denote whether the edge server e is selected as the edge aggregator, where $e_{Agg} = 1$ indicates that it is an edge aggregator and $e_{Agg} = 0$ indicates that it is not. Let $connec_k^e \in \{0, 1\}$ denote whether client $k(k \in K)$ is associated with edge aggregator e, with $connec_k^e = 1$ indicating that it is associated and $connec_k^e = 0$ not associated.

Let $|D_{\dot{k}}|$ be the size of the training data for the client k, $|D_e|$ be the sum of the training data sizes for the nodes associated with the edge aggregator, and S_{ij} representing the diversity of training data distributions between nodes and edge aggregators. Additionally, the size of the training data is considered to ensure a more balanced data distribution for each edge aggregator.

IV. Proposed Method

A. Framework Overview

This work proposes an efficient aggregation method for trustworthy hierarchical federated learning, which is primarily divided into three parts: client clustering and grouping, a combination of synchronous and asynchronous aggregation, and secure interaction with access control. Client clustering groups clients based on data distribution and training efficiency, clustering those with low similarity in data distribution but high similarity in training efficiency to ensure diversity in the data samples within each group. The hybrid synchronous and asynchronous aggregation method primarily performs synchronous aggregation within groups to maintain data sample diversity and reduce data loss, while asynchronous updates are executed between groups to ensure training efficiency. For secure storage and access control, blockchain technology is introduced, utilising smart contracts for group partitioning, access control, and the updating and querying of data. The overall framework is shown in Figure 2, with the specific steps as follows:

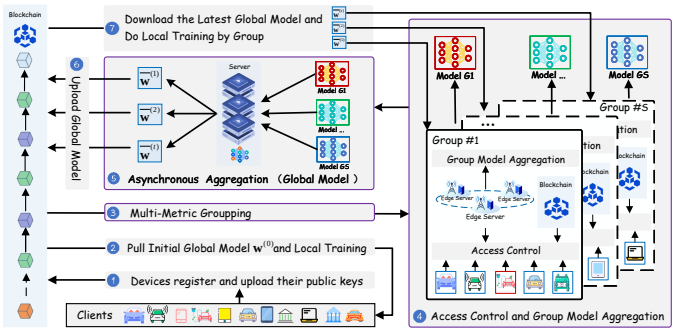


Fig. 2. The Framework of the TrustHFL.

- **Step 1**: Devices register and upload their public keys to the blockchain for access control verification.
- **Step 2**: Authorised devices pull the model for local training and submit their training efficiency along with the updated local model.
- **Step 3**: Groups are formed based on a composite metric derived from local data distribution and device training efficiency.
- **Step 4**: Aggregation edge servers obtain local model updates from each device through the access control mechanism and perform synchronous aggregation.
- **Step 5**: Asynchronous aggregation is conducted between groups, where each group submits its model to combine with the latest global model to produce the new global model.
- **Step 6**: The global model parameters for the current round are uploaded to the blockchain.
- **Step 7**: Groups that submitted updates pull the aggregated global model and initiate a new round of local training.

Repeat Steps 2–7 until the model reaches the target accuracy or the specified number of training rounds is completed.

Overall, this approach considers both the similarity of client data distribution and computational capacity to learn as much as possible from each device, alleviating the issue of data imbalance. Grouping reduces communication overhead and minimises unnecessary broadcasting of local information across the network. The synchronous intra-group and asynchronous inter-group aggregation scheme enhances training efficiency and model performance. Additionally, blockchain technology mitigates single points of failure, ensuring robust communication connections.

Notably, model aggregation within groups is accomplished via an access control mechanism, offloading intensive training and model storage to off-chain processes, while the blockchain handles core lightweight metadata storage and contract operations, ensuring the efficiency of federated learning and the performance of the model. The complete procedure of TrustHFL is described in Algorithm 1.

B. Clustering and Grouping of Clients

The proposed method, TrustHFL, groups clients based on their data distributions and training efficiencies. During the preparation phase, clients pull the initial global model for

Algorithm 1 TrustHFL Training

Input: K (Number of training clients), D_k (Local Dataset), $\mathbf{w}^{(0)}$ (Initialized global model), T_{Cloud} (Maximum global training rounds), acc_{target} (Desired model accuracy)
Output: Global model $\overline{\mathbf{w}}^{(T)}$
// Initialize global model
$t \leftarrow 0$
$\overline{\mathbf{w}}^{(t)} \leftarrow \mathbf{w}^{(0)}$
for each $k \in K$ **do**
  k Register and Upload Public Key to Blockchain
end for
// Training Loop
while $t \leq T_{Cloud}$ **do**
  for each $k \in K$ **do**
    if k has Access Control **then**
      k Do Local Training $g_k^{(t)}(\mathbf{w}_k^{(t)}, D_k^{(t)})$
      k Upload Local Model $\mathbf{w}_k$ and Local Training Iterations $epoch_k$
    end if
  end for
  $Groups \leftarrow$ **Algorithm 2: Client Device Grouping**
  for each $G_s \in Groups$ **do**
    if $\overline{\mathbf{w}}_{G_s}^{(t)}$ **then**
      $\overline{\mathbf{w}}^{(t)} \leftarrow \alpha \overline{\mathbf{w}}_{G_s}^{(t)} + (1-\alpha)\left(\frac{1}{2}\sum_{i=1}^{2} \mathbf{w}^{(t-1)}\right)$
      Upload Group Model to Blockchain
      $t \leftarrow t+1$
    end if
    if $acc_{global} \geq acc_{target}$ **then**
      break
    end if
  end for
end while
Return global model $\overline{\mathbf{w}}^{(T)}$

pre-training and submit their respective data distributions and training efficiencies.

Similarity in device computing capabilities. Each client is required to upload their device's CPU clock frequency. Let c_k represent the number of CPU cycles required by the client k to process a single data sample; this data can be measured offline [40] and is known in advance. Since all samples $\xi_k(x_k^i, y_k^i)$ have the same size (*i.e.*, in bits), the number of CPU cycles required to perform one local iteration on the client k is $c_k D_k$. Let f_k denote the CPU cycle frequency of the client k. Let $\mathbf{F}_{cpu}$ represent the vector of CPU cycle frequencies. By combining the number of clusters S, the clients are sorted in ascending order of their CPU cycle frequencies. The computational capability similarity between client k_i and client k_j is given by:

$$\text{simil}_{\text{effic}}(k_i, k_j) = \begin{cases} 0, & k_i \in S_G, \text{ and } k_j \notin S_G \\ 1, & k_i \in S_G, \text{ and } k_j \in S_G \end{cases}. \quad (4)$$

Client data distribution similarity. To address the issue of low efficiency in calculating model gradient similarity in clustering FL, represented by FMTL [41], this work draws on the ideas from PFA [42] and utilises the sparsity of neural networks to reflect client data distributions.

Specifically, a CNN model typically consists of convolution (Conv) layers, pooling layers, batch normalisation (BN) layers, ReLU activation functions, and fully connected (FC) layers, each representing a distinct type of mathematical operation. By inputting an image into the CNN, the output of each layer can be viewed as a feature map with a shape of H×W×C, where H, W, and C represent the height, width, and number of channels in the feature map, respectively. After processing by the BN and ReLU layers, corresponding BN output feature maps and ReLU output feature maps are generated. The ReLU layer converts all negative inputs to zero, resulting in a highly sparse output. Given a ReLU output feature map, sparsity for a specific channel is defined as the ratio of the number of zero elements in the matrix. In this way, we can compute the sparsity value for each channel, with the entire feature map corresponding to a sparsity vector.

Furthermore, in each local training round, the client k trains a local model $\mathbf{w}_k$ using their local dataset $|D_k = \{\xi_k(x_k^i, y_k^i)\}(k \in K, 1 \leq i \leq n_k$. Feature maps are extracted from the ReLU layer of $\mathbf{w}_k$ during this process. In the follow-up phase, the q channels $(c_1, c_2, ..., c_q)$ are randomly selected, and their corresponding client sparsity is extracted. The client k extracts the feature matrix matrix$F_h(x_k^i)$ from the output of the selected channel c_h for a given the input x_k^i, and calculates the channel features as follows:

$$feature_h(x_k^i) = \frac{zero(\text{matrix}F_h(x_k^i))}{H \times W}, \quad (5)$$

where $zero(\cdot)$ is a function that calculates the number of zeros in a matrix, H and W are the height and width of the feature matrix, respectively. By averaging the features of all samples, the features of the selected channel c_h on the dataset D_k are calculated as follows:

$$feature_h(\mathcal{D}_k) = \frac{1}{n_k}\sum_{i=1}^{n_k} feature_h(x_k^i). \quad (6)$$

Client k form feature vectors $Fea_k = (feature_1(\mathcal{D}_k), feature_2(\mathcal{D}_k), ..., feature_n(\mathcal{D}_k))$ during the local training process. Each client generates a feature vector that reflects the local data distribution. The top layer uses cosine similarity as a metric to calculate the similarity between vectors. Given the feature vectors Fea_i and Fea_j of clients k_i and k_j, the distribution similarity is computed as follows:

$$\text{simil}_{\text{distr}}(k_i, k_j) = \frac{Fea_i \cdot Fea_j}{\|Fea_i\| \times \|Fea_j\|}. \quad (7)$$

To ensure the efficiency of FL, the feature vector Fea_z of the z client is randomly selected, and its similarity with the feature vectors of other clients is calculated, generating a final similarity vector:

$$(simil_{distr}(k_z, k_1), \ldots, simil_{distr}(k_z, k_K)). \quad (8)$$

The principle for selecting the final training devices is to choose clients with high data similarity values (*i.e.*, significant differences in data distribution) from the devices with similar computational capacities for training. The clustering similarity calculation is completed through smart contracts. The clustering process runs in parallel with the local training of clients, thus avoiding any bottlenecks. While we group clients with similar efficiency into the same cluster, the aggregation process within the cluster remains synchronous. Therefore, a timeout mechanism is implemented to prevent continuous waiting for stragglers. The complete process of client device grouping is described in Algorithm 2.

Algorithm 2 Client Device Grouping

Input: K (Number of training clients), S (Number of clusters for grouping clients)
Output: Groups (Grouped clients based on data distribution and training efficiency)
for each client $k \in K$ **do**
  GetDataDistribution from client k
  GetTrainingEfficiency from client k
end for
CalculateCompositeMetric
SortClientsByMetric(K)
Groups ← ClusterDevices(K, S)
return Groups

C. The Combination of Synchronous and Asynchronous Aggregation

The global model update methods in FL can be categorised into synchronous, pseudo-asynchronous, and asynchronous approaches [1]. In synchronous aggregation, the server waits for all clients to submit their local model updates before aggregating the global model. To mitigate delays caused by bottleneck clients in synchronous FL, pseudo-asynchronous FL is introduced. This allows efficient clients to download the global model from the central server during idle periods, conduct local training, and submit updated local models. The central server periodically updates the global model by aggregating newly uploaded local models. However, this approach may result in underutilization of data from bottleneck clients, affecting model prediction accuracy. Consequently, research has shifted towards asynchronous FL. In asynchronous FL, any client can download the global model from the central server during idle times, perform training, and upload the local model. The central server updates the global model immediately upon receiving local models.

In this research work, we combine the advantages of synchronous and asynchronous aggregation by employing intra-group synchronous aggregation and inter-group asynchronous aggregation, as shown in Figure 3.

After completing grouping, clients submit model updates to the intra-group aggregation edge server for synchronous aggregation, reducing sample loss. To enhance training efficiency, inter-group asynchronous aggregation is used for global model aggregation. To address the impact of uneven data distribution on the global model, historical global models are utilised to compensate for information loss in individual group models. The global model aggregation is performed iteratively, and the aggregation formula is given as follows:

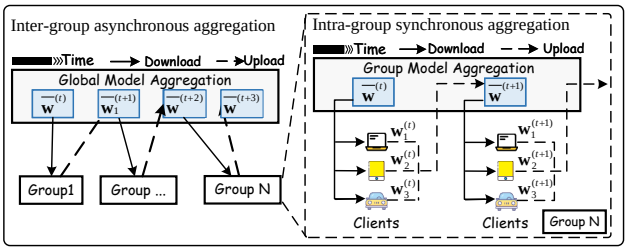


Fig. 3. Synchronous Intra-Group and Asynchronous Inter-Group Aggregation.

$$\overline{\mathbf{w}}^{(t)} = \alpha \overline{\mathbf{w}}_{G_s}^{(t)} + (1-\alpha)\frac{1}{2}\sum_{i=1}^{2}\mathbf{w}^{(t-1)}, \tag{9}$$

where α denotes the aggregation weight.

D. Secure Interaction with Access Control

After the grouping has been completed, a random edge server is selected as the aggregation server for each round within the group. Clients submit their local model updates to the aggregation server through a blockchain access control mechanism, as illustrated in Figure 4. It is worth noting that different aggregation servers may have varying data states, which can lead to consistency issues during the model update process. In our approach, the global model is shared at the edge server level in each round, thereby reducing consistency problems arising from model discrepancies. Random selection helps balance the load across edge servers, decreases the likelihood of attackers predicting the aggregation server, thereby enhancing the system's security and improving its robustness.

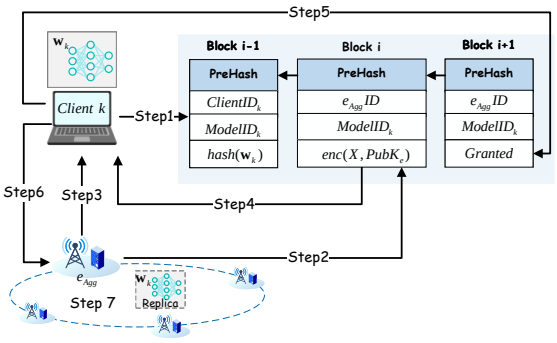


Fig. 4. The Workflow of the Secure Interaction with Access Control.

- **Step 1**: Client k uploads key information about the local model, which mainly includes the client identifier $ClientID_k$, the model identifier $ModelID_k$, and the hash code $hash(\mathbf{w}_k)$.
- **Step 2**: The aggregation server e_{Agg}, acting as the requester, generates a random number X, and then sends the message $enc(X, PubK_k)$ to the blockchain.

- **Step 3**: The aggregation server e_{Agg} sends the message $(e_{Agg}ID, ModleID_k, enc(X, \Pr iK_e))$ to the client k through off-chain communication.
- **Step 4**: Client k uses their public key and private key $PriK_k$ to decrypt $enc(X, \Pr iK_e)$ and $enc(X, PubK_k)$.
- **Step 5**: Client k verifies the identity of the aggregation server e_{Agg} by comparing the two values of X. Once the verification is successful, the aggregation server e_{Agg} is granted access, and client k submits its updates $(e_{\text{Agg}}ID, ModleID_k, Granted)$ on-chain.
- **Step 6**: Once approved, client k sends the model $enc(\mathbf{w}_k, PubK_e)$ to the aggregation server e_{Agg} through an encrypted off-chain channel.
- **Step 7:** The aggregation server e_{Agg} saves the model as a copy through authorisation and broadcasts it within the aggregation server group.

It is essential to note that the replication mechanism introduced in Step 7 achieves fault tolerance and load balancing without incurring additional communication costs. Authorised clients can store the model locally as a copy, effectively mitigating the single point of failure of the server. To balance the communication overhead of model transmission, the model requester calculates the number of copies based on the hash code of its identifier and performs a modulus operation on the total number of available copies.

V. Security Analysis

The access control mechanism in TrustHFL effectively defends against common attacks, as this mechanism primarily focuses on ensuring the security of model storage and access.

- If an attacker attempts to Bypass Access Control by directly requesting the model from all clients without submitting the request to the blockchain, the request will be invalid. Since the access control in TrustHFL is implemented using blockchain technology, all requests and interactions must be recorded and verified on the blockchain. If attackers do not submit their requests through the blockchain, they cannot obtain legitimate access rights. Therefore, the bypasser's requests are invalid.
- In a Sybil attack, an attacker tries to access the model using multiple identities. However, the uniqueness of blockchain certificates prevents attackers from replicating multiple certificates. Even if they create numerous fake identities, they cannot pass the identity verification. Any requests from these fake identities will not be processed, making the Sybil attack unsuccessful.
- If an attacker conducts a Man-in-the-Middle Attack, attempting to interfere with and manipulate communication between the model owner and the requester, they may try to impersonate the requester to steal the model. However, attackers cannot forge or tamper with the data on the blockchain. Additionally, the random number verification proposed in Step 5 can detect replay attacks.
- In the case of a model tampering attack, where the attacker, posing as the model owner, modifies the local model and sends it to the requester, TrustHFL can detect such attacks when the requester verifies the model's integrity using the hash code on the blockchain.

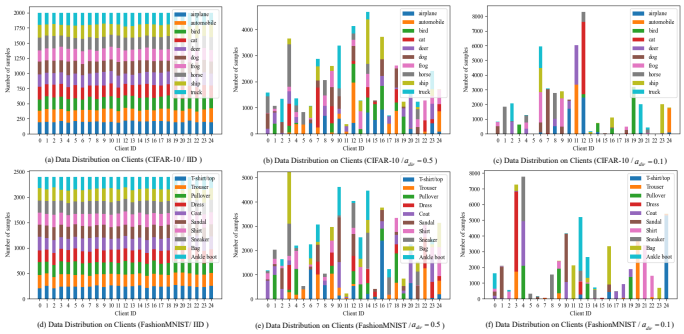


Fig. 5. IID Data Distribution and Dirichlet Simulation of NonIID Data Distribution. Panels (a) to (c) illustrate the data distribution results for the CIFAR-10 dataset, and panels (d) to (f) present the data distribution results for the FashionMNIST dataset.

In summary, TrustHFL can ensure the security of model storage and access to a significant extent.

VI. Experimental Evaluation and Discussion

In this subsection, we describe the experimental environment used to conduct the evaluations and experiments.

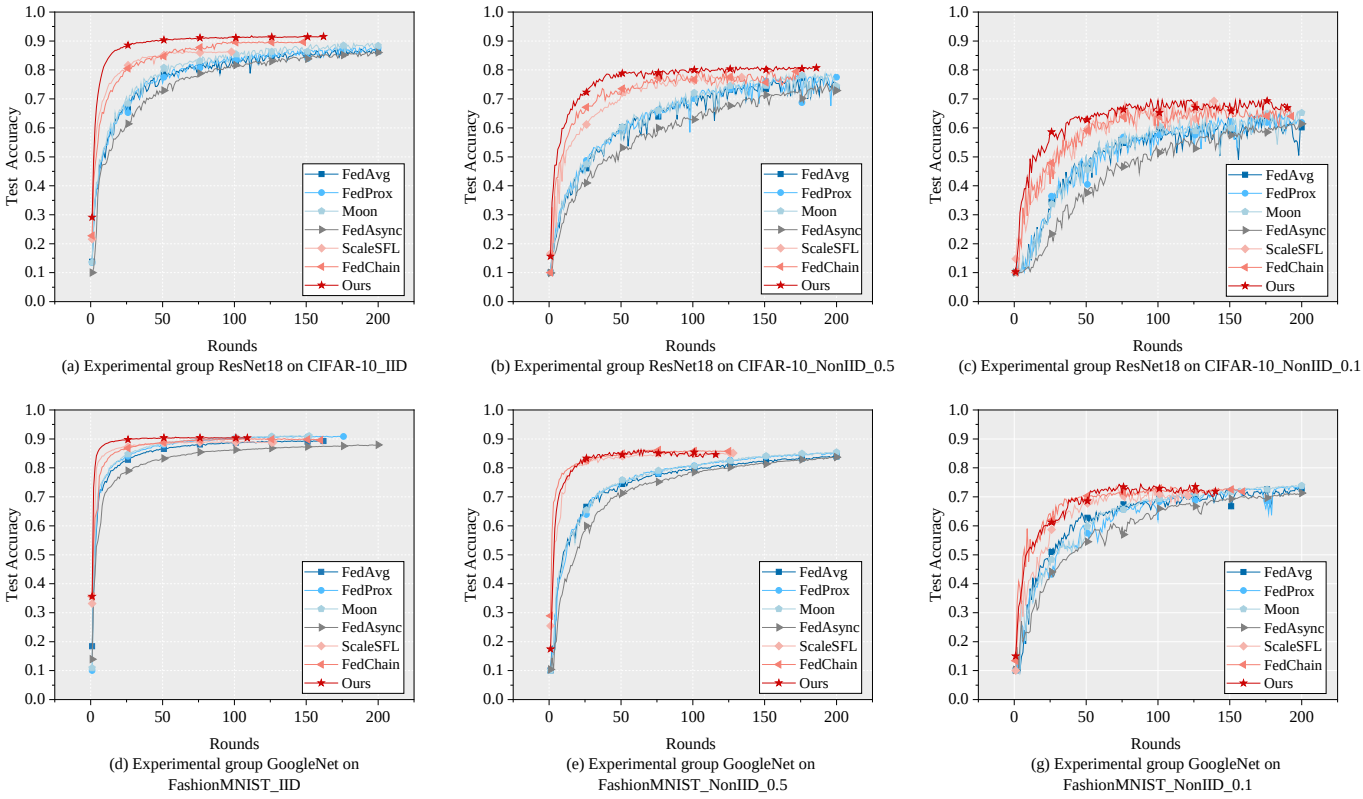
The experimental environment consists of 16 vCPU Intel(R) Xeon(R) Platinum 8481C, 80G memory, 1T storage, one NVIDIA GeForce RTX 4090D GPU card with 24G memory, running Ubuntu 20.04 operating system. The software environment includes Python 3.7.1, PyTorch 1.13.0, CUDA 11.7, and Hyperledger Fabric 2.2.

Experimentations were conducted using the ResNet18 and GoogleNet models, while the evaluation process utilised the CIFAR-10 and FashionMNIST datasets. The experimental setup included two types of data distribution: IID and NonIID. In the IID scenario, clients were randomly assigned datasets of equal size. In the NonIID scenario, a commonly used Dirichlet distribution was employed to simulate the NonIID data distribution among clients. As shown in Figure 5, a smaller probability distribution parameter a_{dir} indicates higher data heterogeneity and size bias. The NonIID distribution better reflects real-world scenarios compared to the IID distribution.

Considering the sample categories of the dataset, this research sets the Dirichlet probability distribution parameter a_{dir} for the NonIID configuration to 0.5 and 0.1. To note that, the Dirichlet probability distribution only allows for imbalanced categories; the number of samples per client may be the same or different, without a guarantee. If the goal is to "intentionally" make the categories and quantities different among clients, this would fall under a "dual imbalance" partition, which this paper does not address. Additionally, the NonIID data partitioning in this paper ensures that each client receives data. The baseline to the comparative experiments in evaluation and comparison are FedAvg [8], FedProx [43], MOON [44], FedAsync [45], ScaleSFL [46], and FedChain [47].

A. Performance of Federated Learning

The performance comparison of FL is primarily based on accuracy, training time, and convergence efficiency. The experiment is set up with 25 clients and 200 global rounds, with



(a) Experimental group ResNet18 on CIFAR-10_IID
(b) Experimental group ResNet18 on CIFAR-10_NonIID_0.5
(c) Experimental group ResNet18 on CIFAR-10_NonIID_0.1
(d) Experimental group GoogleNet on FashionMNIST_IID
(e) Experimental group GoogleNet on FashionMNIST_NonIID_0.5
(g) Experimental group GoogleNet on FashionMNIST_NonIID_0.1

Fig. 6. The Test Accuracy of Global Model

TABLE I
THE BEST TEST ACCURACY OF GLOBAL MODEL

Models/Datasets	ResNet18_CIFAR-10			GoogleNet_FashionMNIST		
Settings	IID	$a_{dir} = 0.5$	$a_{dir} = 0.1$	IID	$a_{dir} = 0.5$	$a_{dir} = 0.1$
FedAvg	0.8723	0.7780	0.6462	0.8939	0.8412	0.7339
FedProx	0.8781	0.7860	0.6453	0.9095	0.8523	0.7376
MOON	0.8916	0.7874	0.6693	**0.9112**	0.8538	0.7390
FedAsync	0.8615	0.7527	0.6237	0.8793	0.8369	0.7133
ScaleSFL	0.8641	0.7841	0.6629	0.8919	0.8573	0.7377
FedChain	0.8978	0.8092	0.6792	0.9009	0.8617	0.7347
Ours	**0.9058**	**0.8113**	**0.6829**	0.9058	**0.8622**	**0.7436**

an early stopping condition of no improvement in validation set performance over 50 rounds.

The proposed method, TrustHFL, demonstrates excellent overall performance. As shown in Table I and Figure 6, the best accuracy of the models decreases in Non-IID settings compared to IID settings, indicating a significant impact of data distribution on accuracy. Specifically, in the ResNet18_CIFAR-10 experimental group, our method achieves an average improvement in the best model accuracy of 2.82%, 2.84%, and 2.92% over all baseline methods in the IID, NonIID($a_{dir} = 0.5$), and NonIID($a_{dir} = 0.1$) scenarios, respectively. In the GoogleNet_FashionMNIST experimental group, the improvements are 0.80%, 1.26%, and 1.22% in the same scenarios. This demonstrates that our method has a distinct advantage in NonIID settings, primarily due to the use of grouped training. By clustering clients with lower similarity in data distribution into groups, we ensure diversity within each group's data distribution, thereby enhancing model performance.

In terms of convergence speed, the proposed method clearly outperforms other methods. As shown in Table II and Figure 7, the time taken by each method to achieve the same level of accuracy, based on the lowest accuracy from each group setting in Table I, indicates that our method requires the least time in all cases. Specifically, in the ResNet18 on CIFAR-10 experimental group, under the IID, NonIID($a_{dir} = 0.5$), and NonIID($a_{dir} = 0.1$) settings, the proposed method reduced the average time compared to the baseline method by 80.91%, 75.87%, and 60.17%, respectively. In the GoogleNet on FashionMNIST experimental group, under the same settings, the proposed method reduced the average time compared to the baseline method by 77.34%, 61.16%, and 37.77%, respectively.

Faster convergence is a key factor in reducing training time. Additionally, TrustHFL groups clients with dissimilar data characteristics and similar computational capabilities into the same cluster, allowing for asynchronous aggregation across multiple groups, which enhances training efficiency. The method also takes into account the contributions of historical global models, mitigating information loss caused by uneven data distribution within individual groups.

B. Ablation Study

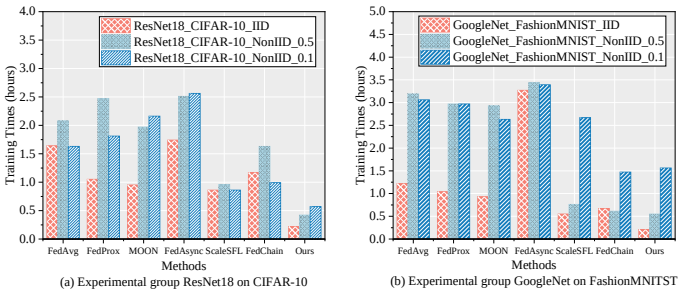
The grouping, hierarchical asynchronous approach, and blockchain module proposed in the TrustHFL method all contribute to the training efficiency and model accuracy of the system. The results of the ablation experiments, including

TABLE II
THE TIME TAKEN BY EACH METHOD TO ACHIEVE THE SAME LEVEL OF ACCURACY (TIME UNIT: HOURS)

Method	ResNet18_CIFAR-10						GoogleNet_FashionMNIST					
	IID		$a_{dir}=0.5$		$a_{dir}=0.1$		IID		$a_{dir}=0.5$		$a_{dir}=0.1$	
	acc	times	acc	times	acc	times	acc	times	acc	times	acc	times
FedAvg	0.8618	1.64	0.7529	2.08	0.6246	1.63	0.8794	1.22	0.8373	3.19	0.7144	3.06
FedProx	0.8625	1.05	0.7528	2.47	0.6244	1.81	0.8795	1.04	0.8371	2.97	0.7174	2.97
MOON	0.8632	0.95	0.7528	1.97	0.6237	2.16	0.8799	0.93	0.8372	2.93	0.7133	2.63
FedAsync	0.8615	1.74	0.7527	2.51	0.6237	2.56	0.8793	3.27	0.8369	3.44	0.7133	3.39
ScaleSFL	0.8615	0.86	0.7532	0.96	0.6247	0.86	0.8797	0.55	0.8369	0.76	0.7135	2.67
FedChain	0.8625	1.17	0.7530	1.63	0.6246	0.99	0.8798	0.67	0.8345	0.61	0.7133	1.47
Ours	0.8630	**0.22**	0.7584	**0.42**	0.6276	**0.57**	0.8824	**0.21**	0.8370	**0.55**	0.7133	**1.56**

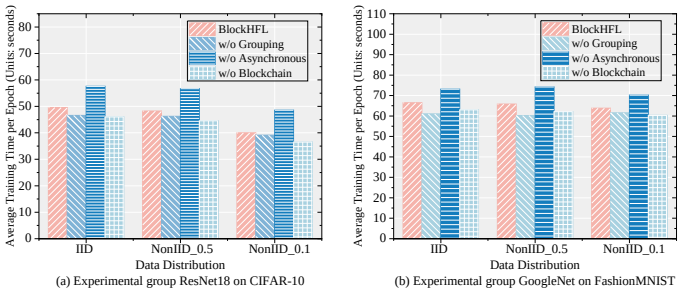
TABLE III
ABLATION STUDY: THE BEST TEST ACCURACY OF GLOBAL MODEL AND AVERAGE TRAINING TIME PER EPOCH (TIME UNIT: SECONDS)

Method	ResNet18_CIFAR-10						GoogleNet_FashionMNIST					
	IID		$a_{dir}=0.5$		$a_{dir}=0.1$		IID		$a_{dir}=0.5$		$a_{dir}=0.1$	
	acc	time	acc	time	acc	time	acc	time	acc	time	acc	time
BlockHFL	0.9058	49.59	0.8113	48.26	0.6829	40.02	0.9058	66.45	0.8622	65.84	0.7436	63.82
w/o Grouping	0.9022	46.58	0.7672	46.24	0.6592	39.12	0.8886	61.05	0.8580	60.21	0.7288	61.57
w/o Asynchronous	0.9034	57.63	0.7326	56.75	0.6307	48.59	0.8959	73.14	0.8495	74.21	0.7364	70.29
w/o Blockchain	0.9053	46.04	0.8115	44.62	0.6831	36.56	0.9052	63.09	0.8625	62.21	0.7432	60.37



(a) Experimental group ResNet18 on CIFAR-10 (b) Experimental group GoogleNet on FashionMNITST

Fig. 7. The Time Taken by Each Method to Achieve the Same Level of Accuracy



(a) Experimental group ResNet18 on CIFAR-10 (b) Experimental group GoogleNet on FashionMNIST

Fig. 8. Ablation Study:Average Training Time per Epoch

the best test accuracy of the global model and the average training time per round, are shown in Table III and Figure 8.

When the grouping module was removed, in the ResNet18 on CIFAR-10 experimental group, the model accuracy decreased by 0.36%, 4.41%, and 2.37% under IID, NonIID($a_{dir}=0.5$), and NonIID($a_{dir}=0.1$) data distribution scenarios, respectively. The average training time per round was reduced by 6.07%, 4.19%, and 2.25%, respectively. In the GoogleNet on FashionMNIST scenario, the model accuracy decreased by 1.72%, 0.42%, and 1.48% under IID, NonIID($a_{dir}=0.5$), and NonIID($a_{dir}=0.1$) conditions, respectively, while the average training time per round was reduced by 8.13%, 8.55%, and 3.53%.

When the hierarchical asynchronous module was removed, in the ResNet18 on CIFAR-10 experimental group, the model accuracy decreased by 0.24%, 7.87%, and 5.22% under IID, NonIID($a_{dir}=0.5$), and NonIID($a_{dir}=0.1$) data distribution scenarios, respectively. The average training time per round was reduced by 16.21%, 17.59%, and 21.41%, respectively. In the GoogleNet on FashionMNIST scenario, the model accuracy decreased by 0.99%, 1.27%, and 0.72% under IID, NonIID($a_{dir}=0.5$), and NonIID($a_{dir}=0.1$) conditions, respectively, while the average training time per round was reduced by 10.07%, 12.71%, and 10.14%.

When the blockchain module was removed, in the ResNet18 on CIFAR-10 experimental group, the model accuracy showed no significant changes under IID, NonIID($a_{dir}=0.5$), and NonIID($a_{dir}=0.1$) data distribution scenarios, but the average training time per round was reduced by 7.16%, 7.54%, and 8.65%, respectively. In the GoogleNet on FashionMNIST scenario, the model accuracy also showed no significant changes under IID, NonIID($a_{dir}=0.5$), and NonIID($a_{dir}=0.1$) conditions, while the average training time per round was reduced by 5.06%, 5.51%, and 5.41%.

The grouping method can effectively enhance model accuracy, especially in NonIID scenarios. Although it increases training time, this increase remains within an acceptable range. The hierarchical asynchronous approach significantly reduces training time, alleviating the impact of bottleneck clients on the system while maintaining accuracy. Additionally, the blockchain module improves the system's security.

For the blockchain module, this work evaluates the system's throughput and latency using the open-source tool Caliper. The specific testing environment and configuration are as

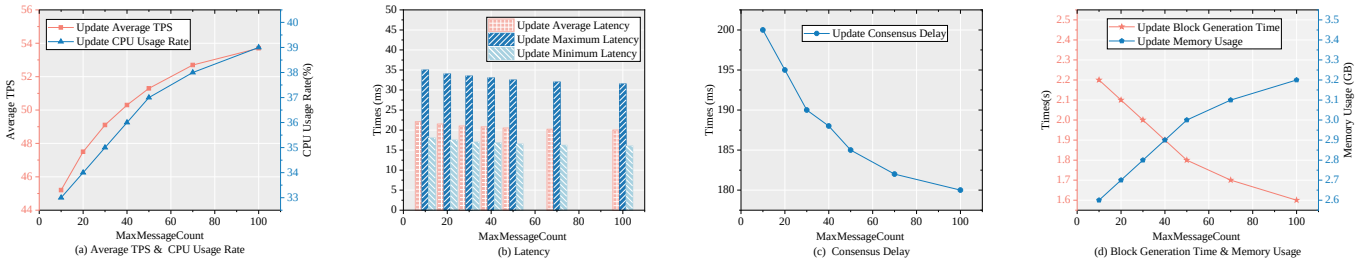


Fig. 9. The Main Performance Data of the Updated Workload

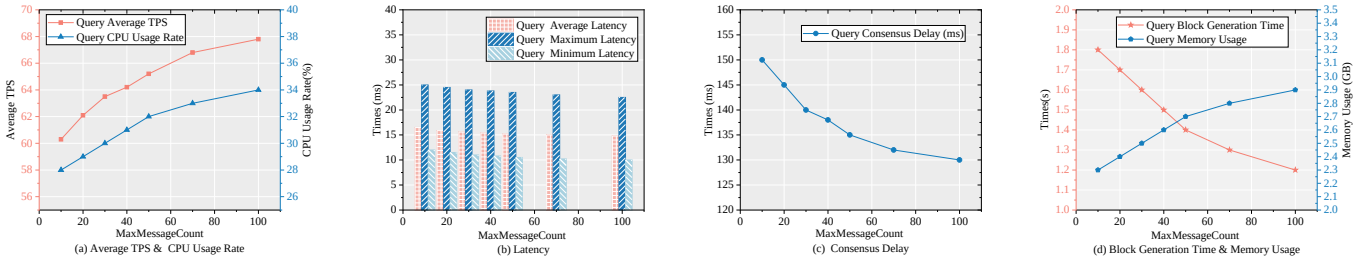


Fig. 10. The Main Performance Data of the Query Workload

follows: Blockchain Platform: Hyperledger Fabric; Testing Network: fabric-samples/test-network; Caliper Version: 0.4.2; Fabric Version: 2.2; Workload includes Query and Update; Number of Clients: 25; Test Rounds: Each MaxMessageCount setting runs three times, with each test lasting 30 seconds.

Blockchain Configuration. We adjust the block size configuration using MaxMessageCount and upload updated models, querying the latest global model's key performance data, as shown in Figure 9 and Figure 10.

Query Workload. Throughput increases with MaxMessageCount, indicating that larger block sizes can enhance the efficiency of query operations. Average transaction latency, as well as maximum and minimum latencies, decrease with the increase in MaxMessageCount, demonstrating improved efficiency in the network's handling of query requests. Resource usage, such as CPU and memory usage, slightly rises with MaxMessageCount but remains within acceptable limits.

Update Workload. Throughput also increases with MaxMessageCount, but overall it is lower than that of query operations, due to the fact that update operations require more resources and time to process data modifications. The latency for update operations is higher than for query operations, but it decreases with the increase in MaxMessageCount. Update operations utilise resources more intensively than query operations, particularly in CPU usage, due to the greater computational resources required. The success rate of update operations is slightly lower at lower MaxMessageCount settings but gradually reaches 100% as block size increases.

Notably, when MaxMessageCount is set to 100, the network achieves optimal performance for both workloads, increasing throughput and reducing latency while maintaining a high success rate and resource efficiency. This setting provides valuable insights for optimising Hyperledger Fabric networks in practical applications. However, it is essential to consider factors such as long-term network stability, resource limitations, and

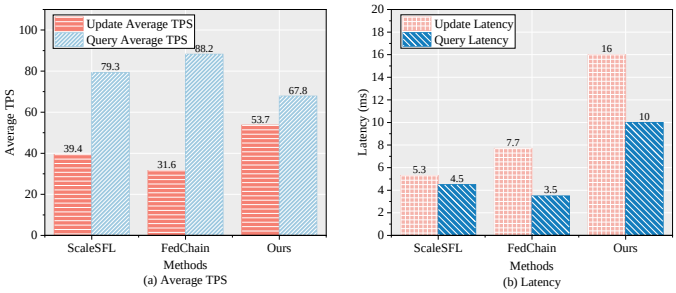


Fig. 11. The Comparison of Blockchain Throughput and Latency

business requirements to ensure the best configuration choices.

In the scenario with 25 clients, the throughput and latency of ScaleSFL, FedChain, and the method proposed in this paper, TrustHFL, are compared as shown in Figure 11.

TrustHFL has no significant improvement in throughput and latency for querying the global model, but it has the best throughput for the update upload interface. The main reason is that TrustHFL uploads a small amount of key information and uses sharding. In contrast, FedChain uploads metadata but does not implement sharding, which leads to a bottleneck. ScaleSFL has implemented a sharding-level consensus mechanism to improve efficiency. We will be the focus of further research on throughput and latency for the querying interface in the next steps.

VII. Concluding Remarks and Future Work

This paper proposes a grouping method based on feature matrix extraction through model sparsification, aggregating nodes with different data distributions to address data imbalance. Utilising technologies such as blockchain, access control, and smart contracts, we achieve synchronous intra-group and asynchronous inter-group aggregation, mitigating

the shortboard effect and preventing unauthorised client access to the model, thus ensuring data privacy and system security. Furthermore, the computational training process is designed to occur off-chain, with only key information stored on-chain, reducing the blockchain's load and ensuring training efficiency. Additionally, we consider the impact of historical global models on model performance; each global aggregation accounts for the information loss incurred by the imbalanced data distribution of the previous round's historical model to enhance model convergence speed and accuracy. Experimental results demonstrate that the proposed method improves model accuracy by an average of 1.98% across different data distribution scenarios, while the time required to achieve the same accuracy is reduced by an average of 65.54%. In future work, we will further investigate communication overhead among distributed computing nodes and cross-domain interactions. Additionally, in terms of security, we will delve deeper into defensive measures against poisoning attacks and inference attacks within the hierarchical federated learning framework, as well as lightweight privacy protection schemes for data portions, to further enhance the flexibility, security, and efficiency of TrustHFL.

REFERENCES

[1] L. Wang, H. Zhou, Y. Bao, X. Yan, G. Shen, and X. Kong, "Horizontal federated recommender system: A survey," *ACM Computing Surveys*, vol. 56, no. 9, pp. 1–42, 2024.

[2] Z. Li, Z. Hou, H. Liu, T. Li, C. Yang, Y. Wang, C. Shi, L. Xie, W. Zhang, L. Xu *et al.*, "Federated learning in large model era: Vision-language model for smart city safety operation management," in *Companion Proceedings of the ACM on Web Conference 2024*, 2024, pp. 1578–1585.

[3] M. Ye, X. Fang, B. Du, P. C. Yuen, and D. Tao, "Heterogeneous federated learning: State-of-the-art and research challenges," *ACM Computing Surveys*, vol. 56, no. 3, pp. 1–44, 2023.

[4] Z. Wang, H. Xu, J. Liu, H. Huang, C. Qiao, and Y. Zhao, "Resource-efficient federated learning with hierarchical aggregation in edge computing," in *IEEE INFOCOM 2021-IEEE conference on computer communications*. IEEE, 2021, pp. 1–10.

[5] C. Zhang, T. Zhu, H. Wu, and H. Ning, "Perml-fed: enabling personalized multi-level federated learning within heterogenous iot environments for activity recognition," *Cluster Computing*, vol. 27, no. 5, pp. 6425–6440, 2024.

[6] W. Liang, J. Xiao, Y. Chen, C. Yang, K. Xie, K.-C. Li, and B. Di Martino, "Tmhd: Twin-bridge scheduling of multi-heterogeneous dependent tasks for edge computing," *Future Generation Computer Systems*, vol. 158, pp. 60–72, 2024.

[7] X. Chen, G. Xu, X. Xu, H. Jiang, Z. Tian, and T. Ma, "Multicenter hierarchical federated learning with fault-tolerance mechanisms for resilient edge computing networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

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

[9] C. Xu, Y. Qu, Y. Xiang, and L. Gao, "Asynchronous federated learning on heterogeneous devices: A survey," *Computer Science Review*, vol. 50, p. 100595, 2023.

[10] Z. Li, Z. Lin, J. Shao, Y. Mao, and J. Zhang, "Fedcir: Client-invariant representation learning for federated non-iid features," *IEEE Transactions on Mobile Computing*, 2024.

[11] L. Zhang, Y. Luo, Y. Bai, B. Du, and L.-Y. Duan, "Federated learning for non-iid data via unified feature learning and optimization objective alignment," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 4420–4428.

[12] J. Jin, J. Ren, Y. Zhou, L. Lyu, J. Liu, and D. Dou, "Accelerated federated learning with decoupled adaptive optimization," in *International Conference on Machine Learning*. PMLR, 2022, pp. 10 298–10 322.

[13] T. Zhou, Z. Lin, J. Zhang, and D. H. Tsang, "Understanding and improving model averaging in federated learning on heterogeneous data," *IEEE Transactions on Mobile Computing*, 2024.

[14] W. Liang, S. Xie, K.-C. Li, X. Li, X. Kui, and A. Y. Zomaya, "Mc-dsc: A dynamic secure resource configuration scheme based on medical consortium blockchain," *IEEE Transactions on Information Forensics and Security*, 2024.

[15] W. Liang, Y. Liu, C. Yang, S. Xie, K. Li, and W. Susilo, "On identity, transaction, and smart contract privacy on permissioned and permissionless blockchain: A comprehensive survey," *ACM Computing Surveys*, vol. 56, no. 12, pp. 1–35, 2024.

[16] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2020.

[17] Z. Peng, J. Xu, X. Chu, S. Gao, Y. Yao, R. Gu, and Y. Tang, "Vfchain: Enabling verifiable and auditable federated learning via blockchain systems," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 173–186, 2021.

[18] L. Feng, Z. Yang, S. Guo, X. Qiu, W. Li, and P. Yu, "Two-layered blockchain architecture for federated learning over the mobile edge network," *IEEE network*, vol. 36, no. 1, pp. 45–51, 2021.

[19] R. Xiong, W. Ren, S. Zhao, J. He, Y. Ren, K.-K. R. Choo, and G. Min, "Copifl: A collusion-resistant and privacy-preserving federated learning crowdsourcing scheme using blockchain and homomorphic encryption," *Future Generation Computer Systems*, vol. 156, pp. 95–104, 2024.

[20] L. Chen, D. Xiao, Z. Yu, and M. Zhang, "Secure and efficient federated learning via novel multi-party computation and compressed sensing," *Information Sciences*, vol. 667, p. 120481, 2024.

[21] Y. Yang, B. Hui, H. Yuan, N. Gong, and Y. Cao, "{PrivateFL}: Accurate, differentially private federated learning via personalized data transformation," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 1595–1612.

[22] S. Zhou, K. Li, Y. Chen, C. Yang, W. Liang, and A. Y. Zomaya, "TrustBCFL: Mitigating data bias in IoT through blockchain-enabled federated learning," *IEEE Internet of Things Journal*, 2024.

[23] H. Zhou, Y. Zheng, H. Huang, J. Shu, and X. Jia, "Toward robust hierarchical federated learning in internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 5, pp. 5600–5614, 2023.

[24] Y. Li, X. Tao, X. Zhang, J. Liu, and J. Xu, "Privacy-preserved federated learning for autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 8423–8434, 2021.

[25] C. Xu, Y. Qu, Y. Xiang, and L. Gao, "Asynchronous federated learning on heterogeneous devices: A survey," *Computer Science Review*, vol. 50, p. 100595, 2023.

[26] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng, "Tifl: A tier-based federated learning system," in *Proceedings of the 29th international symposium on high-performance parallel and distributed computing*, 2020, pp. 125–136.

[27] S. Luo, X. Chen, Q. Wu, Z. Zhou, and S. Yu, "Hfel: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6535–6548, 2020.

[28] S. Liu, G. Yu, X. Chen, and M. Bennis, "Joint user association and resource allocation for wireless hierarchical federated learning with iid and non-iid data," *IEEE Transactions on Wireless Communications*, vol. 21, no. 10, pp. 7852–7866, 2022.

[29] N. Mhaisen, A. A. Abdellatif, A. Mohamed, A. Erbad, and M. Guizani, "Optimal user-edge assignment in hierarchical federated learning based on statistical properties and network topology constraints," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 55–66, 2021.

[30] S. Zheng, Q. Meng, T. Wang, W. Chen, N. Yu, Z.-M. Ma, and T.-Y. Liu, "Asynchronous stochastic gradient descent with delay compensation," in *International conference on machine learning*. PMLR, 2017, pp. 4120–4129.

[31] Y. Jiang, S. Wang, V. Valls, B. J. Ko, W.-H. Lee, K. K. Leung, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 12, pp. 10 374–10 386, 2022.

[32] X. Lu, Y. Liao, P. Lio, and P. Hui, "Privacy-preserving asynchronous federated learning mechanism for edge network computing," *Ieee Access*, vol. 8, pp. 48 970–48 981, 2020.

[33] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4298–4311, 2020.

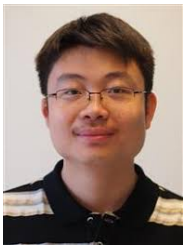
[34] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1279–1283, 2019.
[35] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2020.
[36] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2019.
[37] S. Ji, J. Zhang, Y. Zhang, Z. Han, and C. Ma, "Lafed: A lightweight authentication mechanism for blockchain-enabled federated learning system," *Future Generation Computer Systems*, vol. 145, pp. 56–67, 2023.
[38] J. Wang, W. Tian, J. Tang, X. Ye, Y. Wan, Z. Xu, and L. Chen, "Sym-cs-hfl: A secure and efficient solution for privacy-preserving heterogeneous federated learning," *Journal of Information Security and Applications*, vol. 94, p. 104253, 2025.
[39] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and trends® in machine learning*, vol. 14, no. 1–2, pp. 1–210, 2021.
[40] G. Zheng, Q. Ni, K. Navaie, and H. Pervaiz, "Semantic communication in satellite-borne edge cloud network for computation offloading," *IEEE Journal on Selected Areas in Communications*, 2024.
[41] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 8, pp. 3710–3722, 2020.
[42] B. Liu, Y. Guo, and X. Chen, "Pfa: Privacy-preserving federated adaptation for effective model personalization," in *Proceedings of the Web Conference 2021*, 2021, pp. 923–934.
[43] X. Yuan and P. Li, "On convergence of fedprox: Local dissimilarity invariant bounds, non-smoothness and beyond," *Advances in Neural Information Processing Systems*, vol. 35, pp. 10 752–10 765, 2022.
[44] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 10 713–10 722.
[45] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.
[46] E. Madill, B. Nguyen, C. K. Leung, and S. Rouhani, "Scalesfl: a sharding solution for blockchain-based federated learning," in *Proceedings of the Fourth ACM International Symposium on Blockchain and Secure Critical Infrastructure*, 2022, pp. 95–106.
[47] H. Wu, X. Zhu, and W. Hu, "A blockchain system for clustered federated learning with peer-to-peer knowledge transfer," *Proceedings of the VLDB Endowment*, vol. 17, no. 5, pp. 966–979, 2024.

Sisi Zhou received her Bachelor's and Master's degrees in 2009 and 2012, respectively, and currently pursuing a Ph.D. degree at Hunan University of Science and Technology. Her research interests include information security and privacy protection, with a focus on blockchain technology and federated learning.

Yuxiang Chen received a Ph.D. degree from Hunan University in 2021. He is currently an assistant professor at Hunan University of Science and Technology. His research interests include network monitoring, network security, big data, and AI.

Wei Liang (Senior Member, IEEE) received a Ph.D. from Hunan University in 2013, and a Postdoctoral Scholar with Lehigh University during 2014–2016. Currently, he is a Professor and the Vice President of the University of South China, while also serving as a professor at the School of Computer Science and Engineering at Hunan University of Science and Technology. His research interests focus on information security, with primary research areas including medical blockchain, data privacy protection and brain-computer interfaces, intelligent transportation and Internet of Vehicles security, as well as security management in wireless sensor networks. He has published over 200 academic papers in various international journals and major conferences, including IEEE TC, TON, TSC, TDSC, TPDS, TIFS; ACM TOIT, TOMM, Computing Surveys; as well as IEEE INFOCOM 2023, ACM SIGMOD 2024, ACM SIGMETRICS 2025, ACM CCS 2025, and DAC 2025.

Kuanching Li (Senior Member, IEEE) is a Professor at Hunan University of Science and Technology. He received his Ph.D. degree from San Pablo Main in 2001, and has published papers in high-ranked journals and conferences and served as associate and guest editor for several scientific journals. His research interests include cloud and edge computing, big data, and blockchain.



Weizhi Meng (Senior Member, IEEE) is a Full Professor in the School of Computing and Communications, Lancaster University, United Kingdom. He obtained his Ph.D. degree in Computer Science from the City University of Hong Kong. His primary research interests are blockchain technology, cyber security, and artificial intelligence in security, including intrusion detection, blockchain applications, smartphone security, biometric authentication, and IoT security.