Journal Pre-proof

Dual-blockchain based multi-layer grouping federated learning scheme for heterogeneous data in industrial IoT

Xin Wang, Haoji Zhang, Haoyu Wu, Hongnian Yu

PII: S2096-7209(24)00008-3

DOI: https://doi.org/10.1016/j.bcra.2024.100195

Reference: BCRA 100195

To appear in: *Blockchain: Research and Applications*

Received Date: 27 August 2023

Revised Date: 11 February 2024

Accepted Date: 22 February 2024

Please cite this article as: X. Wang, H. Zhang, H. Wu, H. Yu, Dual-blockchain based multi-layer grouping federated learning scheme for heterogeneous data in industrial IoT, *Blockchain: Research and Applications* (2024), doi: https://doi.org/10.1016/j.bcra.2024.100195.

Xin Wang: Conceptualization, Methodology, Software.

Haoji Zhang: Data curation, Writing-Original draft preparation.

Haoyu Wu: Visualization, Investigation.

Hongnian Yu: Validation, Resources

Dual-Blockchain based Multi-layer Grouping Federated Learning Scheme for Heterogeneous Data in Industrial IoT

Xin Wang[1,2], Haoji Zhang[1,*], Haoyu Wu[1], Hongnian Yu[2]
[1] School of Electronic Information and Artificial Intelligence, Shaanxi University of Science & Technology, Xi 'an, 710021, China
[2] School of Engineering and the Built Environment, Edinburgh Napier University, Edinburgh, EH10 5DT, United Kingdom

*Corresponding Author: Haoji Zhang. Email: yolohj0812@163.com

***Abstract*—Federated Learning (FL) allows data owners to train neural networks together without sharing local data, allowing the Industrial Internet of Things (IIoT) to share a variety of data. However, traditional federated learning frameworks suffer from data heterogeneity and outdated models. To address these issues, this paper proposed a dual-blockchain based multi-layer grouping federated learning architecture (BMFL). BMFL divides the participant groups based on the training tasks, then realizes the model training combining synchronous and asynchronous through the multi-layer grouping structure, and uses the model blockchain to record the characteristic tags of the global model, allowing group-manners to extract the model based on the feature requirements and solving the problem of data heterogeneity. In addition, to protect the privacy of the model gradient parameters and manage the key, the global model is stored in ciphertext, and the chameleon hash algorithm is used to perform the modification and management of the encrypted key on the key blockchain while keeping the block header hash unchanged. Finally, we evaluate the performance of BMFL on different public datasets and verify the practicality of the scheme with real fault dataset. The experimental results show that the proposed BMFL exhibits more stable and accurate convergence behavior than the classic FL algorithm, and the key revocation overhead time is reasonable.**

***Index Terms*—Federal Learning, multi-level grouping, editable Blockchain, data Heterogeneity, Industrial Internet of Things (IIOT)**

I. INTRODUCTION

THE Industrial Internet of Things (IIoT) has become a prevailing trend in modern industrial development and its utilization has led to the rapid development of the industrial sector. The IoT enables the connection of various smart devices and facilitates rapid and efficient applications. Compared with traditional manual production, automated manufacturing equipment generates a significant amount of data while improving productivity. Smart factories utilize automated machines to detect and collect vast amounts of complex real-time industrial fault data. In particular, industrial big data analytics based on artificial intelligence technology offers significant advantages, such as enhancing equipment performance, enabling predictive maintenance, and minimizing downtime due to failures [1].

Existing research focuses on privacy-sensitive data protection mechanisms in industrial big data [2-3]. Federated learning (FL) [4] as a distributed machine learning method, participating devices first use original data for local training, and then upload the model gradients to the central server, which is responsible for aggregating these models and updating them. Since only the model gradient parameters and not the original data are shared during the FL process, the data privacy of participants can be effectively protected. With the constant development of FL, it is gradually widely used in solving privacy protection problems in edge computing and industrial IoT [5-7].

Industrial big data has the characteristics of multi-source and heterogeneity, and it poses challenges to data processing and analysis compared to general data. When using classical FL in IoT scenarios involving a large number of devices, there is a problem of training accuracy degradation, the discrepancy is mainly caused by factors such as data heterogeneity (uneven distribution of local training data) and device heterogeneity (hardware resource gap). For the challenges of reduced training accuracy and wasted communication resources, FL framework research has changed from synchronous FL [8] to asynchronous federated learning (AFL) [9]. AFL avoids the device idle state and solves the concern of unreliable devices, and in the AFL framework, the global aggregation is executed as soon as the server receives the local model, thus reducing the accuracy of the original stale local model to the global model after aggregation. In [10], the AFL scheme is proposed to improve the training efficiency of heterogeneous IoT devices under unstable communication networks. In [11], to limit the number of devices trained simultaneously in the AFL network, a cache with a weighted averaging mechanism is introduced to reduce the impact of stale models and improve the aggregation efficiency. However, none of the above AFL schemes to solve the heterogeneity problem consider dealing with the stale gradient uploaded by the client and how the local client selects the feature-matched global model when the global model is updated.

Blockchain [12] technology serves as a public, digitized, distributed ledger built on a peer-to-peer network structure, and it has been applied to many network scenarios [13-16], in which cryptography technology can provide secure data sharing and storage, which ensures data authorization and authentication, and manages new participants from transaction records [17]. Therefore, it can be combined with federated learning to effectively implement decentralized storage or replace the central FL server to aggregate model parameters. In [18], the

authors propose an adaptive framework for FL and blockchain that is able to treat the trust of participants as individual probabilities to calculate the trust values of different networks. In [19], the authors integrate blockchain and FL to develop an Industry 4.0 cognitive computing distributed platform to solve the data island problem. The above methods all store model parameters on the blockchain, ensuring the auditability of the training process and the sharing of models. However, once the attacker steals the model parameters on the chain, then the original data may be leaked.

Therefore, in order to solve the problems of asynchronous global model aggregation and local data privacy protection under the FL framework in IIoT, this paper organically combines the blockchain and FL, proposes the BMFL architecture. First, participant groups are divided according to tasks, synchronously aggregates models within the group, asynchronously aggregates the group with the global. To solve the problem of asynchronous global model update and download due to the grouping structure, a Modelchain is used to record the characteristic model. In addition, the global model is encrypted and stored in IPFS. Considering the key management, this paper introduces the Keychain and uses the chameleon hash function to complete the update and modification design of the key on the blockchain, while ensuring that the block header hash unchanged, ultimately realizing the sharing of the global model and the privacy protection of local data. In this way, BMFL supports global model sharing and local data privacy protection.

The main contributions are summarized as follows.

• To address the problem of data heterogeneity affecting model training accuracy in IIoT, this scheme improves the horizontal FL structure and realizes the asynchronous aggregation of global models by multi-layer grouping for local clients, so that the local data can satisfy the privacy protection without degrading the training accuracy of global models.

• Combining blockchain and IPFS to achieve model characterized record and privacy protection, this paper proposes a model tag matching method. Model tags are selected as transactions to be uploaded to the Modelchain, in which models are stored in ciphertext form. In order to obtain as many intergroup features as possible, the group management extracts the models that satisfy the feature requirements by tags on the Modelchain.

• An efficient and fine-grained key update management method is proposed based on the chameleon hash function. When a member of the group exits training, the key of the encrypted global model is modified in the key chain. The update and re-storage of the key information on the Keychain is completed, while the hash value remains unchanged.

The remainder of this paper is organized as follows. Related work is reviewed in Section II. In Section III, the specific architecture and design goals of multi-layer federated learning and the editable blockchain model sharing scheme are described in detail. Section IV presents the experimental results and analysis. Section V presents the conclusions and next steps of this paper.

II. Related Work

2.1. *Data heterogeneity*

Existing FL frameworks usually have highly heterogeneous hardware and Non-Independent-Identically Distributed (Non-IID) data. These heterogeneous and unbalanced data will seriously affect the training effect of federated learning [20]. Researchers have proposed various FL methods to optimize the convergence speed and improve the accuracy of training. Li T et al. [21] proposed a framework FedProx to solve heterogeneity in federated networks, which allows each participating device to perform variable workloads, correcting the gradient drift due to heterogeneous data by computing inexact solutions with proximal terms. Chen Y et al. [22] proposed an asynchronous online federated learning (ASO-Fed) framework, considering the unrealistic assumptions made by FedAvg on heterogeneous devices, in which edge devices perform online learning using a continuous stream of local data to asynchronously update the central model, this can cope with the different computing loads of heterogeneous edge devices and the challenges related to backward edge devices. He et al. [23] proposed a federated learning framework MOON in 2021, which uses the similarity between model representations to correct the local training of the parties to perform comparative learning at the model level. Zhang et al. [24] proposed a FL framework called FedSens, which combines reinforcement learning strategies to perform high-quality updates on the local client, solving the problem of each health data collected by individual devices in an abnormal health detection (AHD) system, improving the accuracy of the global model. Duan et al. [25] proposed a clustered semi-asynchronous federated learning (CSAFL) framework, which groups devices with similar data distribution for asynchronous communication, effectively mitigating the laggard effect and improving the accuracy of AFL. Although AFL has inherent advantages over synchronous FL because model aggregation can be performed without waiting for stragglers, the reliability of the grouping basis in the AFL framework in the above scheme is difficult to guarantee. The scheme design and performance comparison with the literature is shown in Table 1.

Table1.

Comparative analysis of heterogeneity problems

Reference	Characteristics	Advantages	Disadvantages
Li T[21]	It relies on a proximal term to help stabilize the method and improve the convergence behavior of federated learning	High training accuracy	High computation overhead and No consideration for data privacy.
Chen Y[22]	A decay coefficient balancing the previous and the current model.	Converging fast and preventing user dropouts	High communication efficiency
Duan M[25]	A clustered semi-asynchronous federated learning	Mitigating the straggler effect and improving the accuracy of the AFL	The basis for grouping is unclear and low privacy protection

2.2. *Privacy-preserving federated learning*

During the FL model training and aggregation process, although the medium of transmission between participants and servers is mostly model parameters or gradients, current FL schemes are not completely secure due to the fact that centralized servers can be attacked or leak models during model parameter uploads, thus exposing sensitive information about the training data through inference attacks [26]. In addition, current FL are difficult to effectively prevent poisoning attacks by malicious nodes [27]. To further improve the security of IoT data, Wei et al. [28] proposed a user-level differential privacy (UDP) method to achieve local differential privacy at different privacy-preserving levels by changing the variance of the artificial noise process. Bonawitz et al. [29] proposed an efficient and privacy-preserving aggregation protocol that combines key agreement protocol and secret sharing to ensure the privacy of gradients and allow users to exit. Zero Knowledge Proof (ZKP) is a cryptographic technique that is also widely used for user identity or data verification. For example, Guan et al. [30] proposed BlockMaze, a ZKP-based privacy-assured account model blockchain that protects account balances and information about interactions between traders. Blockchain acts as a distributed ledger that stores historical operations and prevents information from being tampered with. Recently, work combining FL with blockchain has been proposed in scenarios that are distributed and require privacy protection. Xue et al. [31] proposed an efficient privacy-preserving and traceable federated learning framework (PPTFL), using hierarchical aggregation federated learning (HAFL) and combining blockchain and IPFS to achieve secure data sharing in IoT, which is efficient in real-world applications with large numbers of users and high-dimensional gradients. Chen J et al. [32] applied deep learning, blockchain, and full homographic encryption (FHE) to an in-vehicle self-organizing network and proposed a decentralized privacy-preserving deep learning model (DPDL), which effectively protected the network privacy and trustworthiness. The comparison table between the scheme design and performance and the literature is shown in Table 2.

Table 2.
Comparative analysis of privacy-preserving mechanisms

Reference	Characteristics	Advantages	Disadvantages
Wei[28]	Each user adds noise to the parameters based on differential privacy	High efficiency	Adding noise reduces the accuracy of model
Bona[29]	All users collaborate to protect privacy	Good accuracy and low computational overhead	High communication overhead and vulnerability to drop-out users
Xue[31]	All users protect privacy and make the parameters traceable and tamper-proof	Better accuracy and low communication	Keys are not managed and the existence of user revocation
Chen J[32]	All nodes fully homomorphic encrypt local updates and aggregate ciphertexts	High security and credibility	High computation and communication overhead

In summary, the FL of the differential privacy mechanism in the above method will protect the data while affecting the model quality, although HE has a sound mathematical theory to ensure the privacy and security of encrypted data, the huge amount of calculation in the encryption and decryption process will occupy the equipment in IIoT resources, ZKP can verify the authenticity of user information without accessing specific data. However, the assertion proof process requires multiple interactive verifications by the user, which affects the user experience [33]. Among existing distributed training solutions, few solutions take into account both data heterogeneity and client data privacy.

Therefore, a single privacy protection technology cannot solve the local data privacy problem well, and a framework based on the combination of FL and blockchain technology can be used to solve the problem of global model storage and accurate delivery on the one hand, and on the other hand can realize the client effective management.

III. Multi-layer Grouped Federation Learning Model

This section designs a multi-layer grouping federated learning framework that combines synchronous and asynchronous, and then detail its components.

3.1. *Network Structure*

The structure of the multi-layer federated learning network is shown in Fig.1, which consists of three main roles: manufacturers, group managers (GM), and global server. Manufacturers complete the update of the local model and group model through communication with the GM. When the group model reaches preliminary convergence, the group management sends the group model to the global server, and the global server asynchronously aggregates the models sent by each group management. Each component is described in detail in the following sections.

***Manufacturers*:** Manufacturers have raw industrial failure data to predict possible failures of industrial equipment and provide maintenance recommendations. Specifically, in order to effectively reduce convergence difficulties caused by data heterogeneity, each manufacturer ($Party_i$) will be divided into groups according to the training task, which is determined by the adopted datasets. Because this paper uses the strip surface defect dataset for experiments, the $Party_i$ are categorized into groups according to the types of defects in the strip images they have, and the same manufacturer can belong to more than one group. Local model training is performed by the FedAvg aggregation algorithm used locally by the manufacturers.

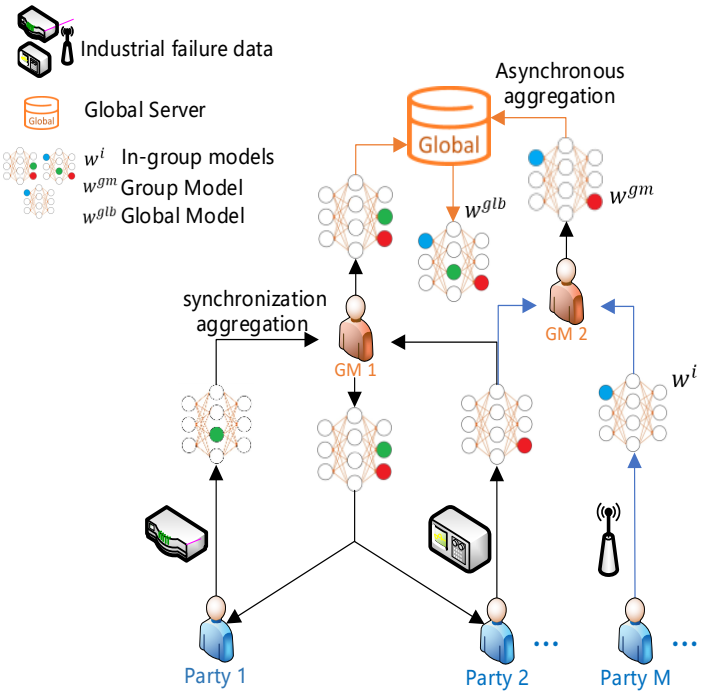

Fig. 1. Multi-layer federation learning network structure

***GM*:** The group elects the GM by consensus algorithm in different groups generated by multiple manufacturers. GM communicates with the manufacturer to complete the update of the local and group models, and when the group model reaches the convergence target, the GM sends the group model to the global server. During the federated learning process GM can request the global model from any node with different feature distributions on the blockchain on demand.

***Global*:** The global server collects the models sent by each group management, aggregates them according to weights to obtain the global model, and uploads the global model information to the blockchain. The global server uses asynchronous aggregation for model collection and aggregation.

We assume that the federated learning scenario consists of multiple geographically distant regions. An arbitrary number of global servers can be used as global servers for each region. Let M participant groups exist in each independent region as similarly geographically distributed participant units, and the model communication between each group can be independent and asynchronous. Let $P_1, \cdots, P_N$ be the N participants present in each group. Participant P_i provides the local dataset D^i containing several samples $\{x, y\}$. The goal of federated learning is to learn an in-group prediction model w^{grp} as a generalized model with generalization properties using an in-group central server through the dataset $D \triangleq \cup_{i \in N} D^i$ provided by all participants. Each participant then trained the in-group prediction model w^{grp} with local private data D^i to obtain the private prediction model w^i available to the participants. The training objective of the in-group prediction model w^{grp} should be Eq. (1), and the in-group participant model converges uniformly as follows:

$$\arg\min L(w^{grp}) = \sum_{i=1}^{N} \frac{|D^i|}{|D|} L_i(w^i) \quad (1)$$

where $L_i(w^i) = E_{(x,y)\sim D^i}[l_i(w^i;(x,y))]$ is the empirical loss function of the participant P_i.

3.2. *The multi-layer federation structure*

The multi-layer federation structure classifies the models obtained from different levels of node training into three categories: local models, group models, and global models, which have different training objectives and aggregation methods.

***Local Models*:** The local model is used for the prediction task, and its training goal is to have a high accuracy rate and low loss value when the manufacturer uses the feature sample set. The update of the local model depends on the shared data set within the group and the local model within the group, and due to the problem of local each participant's devices computing power difference, to simplify the calculation, let the network structure of each participant's personalized model training be consistent with the adoption of FedAvg, Thus, its learning goal is defined as:

$$\arg\min L_{loc}(w^i) \leftarrow CrossEntropyLoss(F_{w_t^i}(x), y) \quad (2)$$

that minimizes the cross-entropy loss function $L_{loc}(w)$ of the local model w^i.

***Group Models*:** To update local models and perform prediction tasks with stronger generalization requirements, the goal is to characterize the models in the group and reduce the impact of data drift on the models. The update of the group model is subject to both the local model of the group and the global model of the server, so its aggregation algorithm faces the challenge of heterogeneity. To solve this problem, we add proximal terms [21] to the local subproblem to effectively limit the impact of the local update of the variables. Considering that the dataset used by the group model during the training process contains some of the datasets of the group's participants, the direction of the correction converges to that of the global model's distance from the group model.

$$\min h_i(w; w^t) = F_i(w) + \frac{\mu}{2}\|w - w^t\|^2 \quad (3)$$

Eq. (3) is the addition of a proximal term to the group objective function, where device i uses a local solver to approximate the minimization objective h_i, $\|w - w^t\|^2$ is the distance between the local and global models, and μ is the penalty strength of the correction term.

Based on the above design, the learning objectives of the group model are defined in Eq. (4):

$$\arg\min L_{gm}(w) \leftarrow CrossEntropyLoss\left(F_{w_{gm}}(x), y\right) + \lambda\|w^{glb} - w^{gm}\|^2 \quad (4)$$

$L_{gm}(w)$ is the cross-entropy loss function derived from the group model trained on the shared data within the group plus the distance correction to the global model. Eq. (4) is the new loss function of the group model, $\|w^{glb} - w^{gm}\|^2$ is the distance between the local model and the global model as a correction term to the original loss function to reduce the degree of bias caused by the heterogeneous data to the model, λ is the penalty strength of the correction term.

***Global Model*:** The aggregation goal of the global model is to extract features that are as common among all group models as much as possible and store them on the blockchain for easy extraction by GM. Each update of the global model is immediately trained with the global shared samples and the group shared samples, and the new global model will be uploaded to the chain after training. To make the global model have stronger common feature extraction capabilities, the feature extraction capability of the global model is enhanced by data augmentation or augmentation. Since the global aggregation algorithm is asynchronous in the multi-layer federated learning design, there is no need to consider communication costs and synchronization issues between group nodes. When the global model receives the model sent by any GM, it first performs data amplification on the group shared samples to generate positive sample pairs, and data augmentation on the global shared samples to generate negative sample pairs. The distance between the same part of the group features and global features decreases, and the distance between the global features and different parts of the group features increases. In this paper, we follow the global objective function in MOON [23] and define the global model aggregation learning objective as Eq. (5):

$$\arg\min L_{glob} \leftarrow L_{sup} + L_{con} \tag{5}$$

L_{sup} is the loss term of the group model in model training, and L_{con} is an additional loss term defined to measure the distance between the global model and group model features. The model-contrastive loss L_{con} is defined as Eq. (6):

$$L_{con} \leftarrow -log \frac{\exp(sim(r,r_{gm})/\tau)}{\exp(sim(r,r_{gm})/\tau)+\exp(sim(r,r_{gm-})/\tau)} \tag{6}$$

where r is the global model feature, r_{gm} is the group model feature, r_{gm-} is the last round feature of the group model, $sim(.)$ is the cosine similarity function and τ is the temperature coefficient. The meaning of L_{con} measures the similarity between the global model features and group model features.

3.3. *Blockchain Multi-layer Federation*

In order to solve the problem of differential update of the global model caused by asynchronous aggregation of group models. In this section, we design the data tags to annotate the global model of different features, and combine it with blockchain technology and IPFS [34] for storage. An adaptive global model sharing scheme is proposed, allowing GM to adaptively extract characteristic global models from the blockchain through data labels.

In this paper, we use Proof-of-Work (PoW) [35] as the consensus mechanism in the blockchain to select an appropriate GM. In each training iteration, the miner receives the global model and other information from the adjacent servers, the GM receives the local model from the manufacturers, and the blockchain cross-validates this information and is finally stored in the distributed ledger as hash index. the blockchain rewards both GMs and miners according to the workload, with the reward for GMs being linearly correlated with the training activity. This approach can incentivize miners and GMs to participate actively in blockchain mining and FL. Blockchain can also use other consensus algorithms, such as byzantine fault tolerance (PBFT), which may require more upfront preparation to reach consensus among miners and different GMs.

The global server asynchronously aggregates the group models sent by the GM and generates a new global model after each effective aggregation. The scheme uses IPFS to store these global models, when the global models are uploaded to the IPFS system, the unique corresponding model will be returned. Hash index through which the original data can be retrieved.

This scheme uses a Modelchain to store the data tags of the models and the model index value hash index returned by the IPFS system so that the global models can be shared accurately.

The Modelchain consists of multiple blocks Block_i, and each Block_i can be divided into two parts: the block body and the block header. The specific structure of model Block_i is shown in Fig.2. The block header contains the index value *Index*、representing the block identity, the timestamp *Time*, The hash value *PreHash* of the previous block, random *Nonce* value, and the *Merkle Tree*. The block body contains the entire Merkle tree with the content of the model transaction tx_{cj}, containing the model adaptive tag and the corresponding *hash index* of the global model.

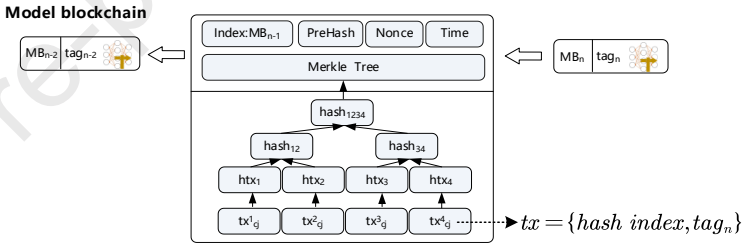

Fig. 2. Modelchain block structure

After the server completes the training of the global model, it first encrypts the global model with the AES key of GM and stores the encrypted model in the IPFS system, which returns a hash index of the corresponding location to the server, which stores the hash index and the tag provided by the GM in the Modelchain. When the group model needs to be updated, the latest adapted global model is found in the Modelchain using tag information, and the target model is obtained according to the corresponding hash index.

***Adaptive tag*:** Due to the asynchronous updates between the global model and the group model can cause the global model to be affected by the stale group model, we design a form of data tags to solve this problem. After the global model aggregates the group of models, The global model will add the tag of the group model to the global model tag after aggregating the group model, so that when other users need to download the global model, the global model suitable for the prediction task can be automatically matched based on the global model tag and times tamp.

The adaptive tag structure is shown in Fig.3. At the beginning of training, the server generates 16 bits tag encoding rules for mainstream tasks, of which the first 4 bits represent the attributes of the task (such as structured data classification, image recognition, speech recognition, semantic segmentation, etc.), 4-7 bit are the coding of the model used, 8-15 bit are the coding of the specific prediction task, The tag encoding rules

are published to all participating groups and distributed to each participant by the group management GM.

Parameters	Task Properties	Model code	Predictive task code
Length (Bytes)	0~3	4~7	8~15

16-bit tag encoding rules

Fig. 3. Schematic diagram of tag structure

The BMFL system architecture is shown in Fig.4. Each participant needs to code the prediction task first. After completing the local training, it sends the tag along with the local model to the group management GM, it receives the local model and tag from all the participants and then sorts all the tags according to the weight of the dataset size of the participants in the group. GM uses the global consensus DCT discrete cosine transform compression algorithm to compress the group model weight into a low-dimensional matrix and encodes it as the summary of group model features $byte_{GM}^{mtx}$, which merges the group tag tag and the summary of group model features $byte_{GM}^{mtx}$ into group tag $tag^{GM} = \{tag_1||tag_1||\ldots||tag_n||byte_{GM}^{mtx}\}$.

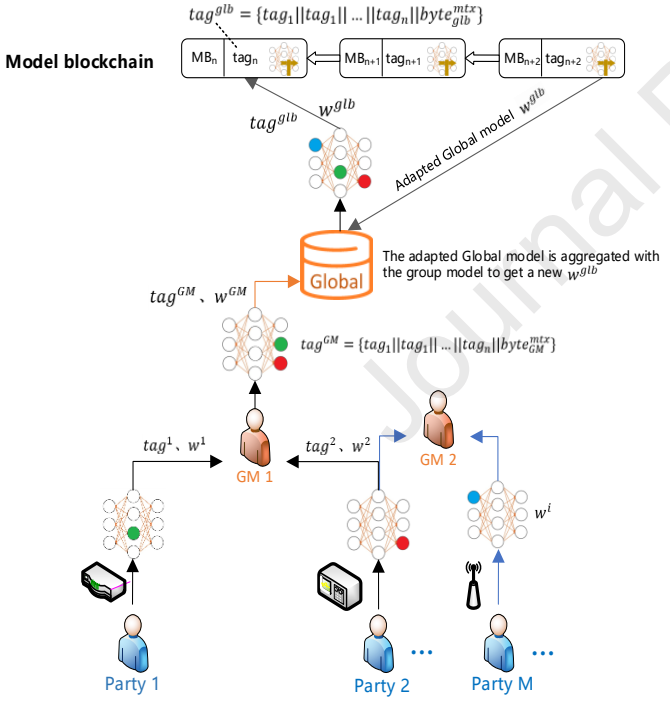

Fig. 4. Multi-layer federal blockchain system architecture

When the GM communicates with the global server, the group model w^{gm} and the group tag tag^{gm} are sent to the global server together. The server searches for a suitable global model on the Modelchain according to the weight from the largest to the smallest, and trains it with the group model to obtain a new global model w^{glb}. The server also uses the compression algorithm to generate the summary $byte_{glb}^{mtx}$ and merges the tags of the original global model with the tags of the group model. The tags are reordered with reference to the contribution value of the group management GM to obtain the new $tag^{glb} = \{tag_1||tag_1||\ldots||tag_n||byte_{glb}^{mtx}\}$, then the server uploads the new global model tag tag^{glb} and hash index corresponding to the encrypted global model in IPFS. When GM needs to use the global model update, the desired target global model is located on the Modelchain according to the matching degree of tag^{GM} and tag^{glb} with the sequence of timestamps.

The entire architecture's model training and blockchain upload process can be divided into an initialization phase and four training phases.

Initialization phase: When training starts, the global server creates a Modelchain and generates tag encoding rules, and broadcasts them to all GMs along with the initial global model w_0^{glb}.

(1) Global model aggregation phase: Whenever the global server receives w^{GM} with the corresponding tag^{GM} sent by a GM, it weights and aggregates w^{GM} to obtain a new w^{glb}. The global server uses the AES key of GM to encrypt to obtain $C_{w^{glb}}$, store $C_{w^{glb}}$ in the IPFS and obtain the storage location hash index. Finally, the global server uploads the hash index and tag^{glb} to the Modelchain. The AES key used for the encryption model and stored on the editable Keychain using the RSA algorithm in Section 3.4 to prevent key leakage.

(2) Group model optimization phase: GM can download the global model on Modelchain to optimize the group model, and the hash-index of the most matching $C_{w^{glb}}$ will be obtained according to the degree of matching between tag^{GM} and tag^{glb}. The decrypted AES key can be obtained by applying to the server, Algorithm1 describes the (1) and (2) phase in detail.

Algorithm 1: Multi-layer federal GM to Global

Input: Global shared data sets D, batch size b, learning rate η, number of global epoch E^{glb}, number of group epoch E^{gm}, hyper-parameter μ, temperature τ

Output: The final model w_n^{glb}

1 Server executes:
2 Find suitable global model w_n^{glb} in the Modelchain by tag_t^{GM}
3 initialize $w_0^{glb}, (GMID, w_t^{GM}, w_{t-1}^{GM}, tag_t^{GM})$
4 **for** epoch $e = 1,2,\ldots,E^{glb}$
5 **for** each batch $b = \{x, y\}$ of D **do**
6 $L_{sup} \leftarrow CrossEntropyLoss(F_{w_n^{glb}}(x), y)$
7 $r \leftarrow R_{w_n^{glb}}(x)$
8 $r_{gm} \leftarrow R_{w_t^{GM}}(x)$
9 $r_{gm-} \leftarrow R_{w_{t-1}^{GM}}(x)$
10 $L_{con} \leftarrow -log \frac{\exp(sim(r, r_{gm})/\tau)}{\exp(sim(r, r_{gm})/\tau) + \exp(sim(r, r_{gm-})/\tau)}$
11 $L \leftarrow L_{sup} + \mu L_{con}$
12 $w_n^{glb} \leftarrow w_n^{glb} - \eta \nabla \ell$
13 **end for**
14 **end for**
15 send w_n^{glb} on the Modelchain
GM optimizes the group model:
1 Find suitable global model w_n^{glb} in the Modelchain by tag_t^{GM}
2 **for** epoch $e = 1,2,\ldots,E^{gm}$
3 **for** each batch $b = \{x, y\}$ of D^{GM} **do**

$L_{gm} \leftarrow CrossEntropyLoss\left(F_{w_t^{gm}}(x), y\right) + \lambda\|w^{glb} - w^{gm}\|^2$
$w_t^{gm} \leftarrow w_t^{gm} - \eta\nabla\ell_{gm}$
end for
end for
return w_t^{gm} to party in group

(3) Local model training phase: Participating node $Party_i$ preprocesses and annotates the local fault image dataset, which is stored in local dataset D^i. The local model $w_i^t$of the round t round of $Party_i$ is trained by the participating nodes through SGD using the local dataset D^i and $w_{t-1}^i (t > 0)$, and when the loss value $L_i(w)$ satisfies the condition $L_i(w) < 1$ and $acc > 0.8$ or the training round has reached the maximum round $t = t_{max}$, the training is stopped and the local model $w_{t\_last}^i$ of the last round is sent to GM. The local model $w_{t\_last}^i$. Meanwhile, according to the characteristics of the local dataset D^i, the participant uses the local model with tag and sends them together with the model $w_{t\_last}^i$ to the GM.

(4) Group model training phase: After the group management GM receives the local models $w_{t\_last}^{i\in U}$ and tag sent by all group participants, it aggregates all local models on average to obtain w^{GM} and generates the corresponding tag^{GM}. Finally, w^{GM} is sent to the global server along with tag^{GM}. Algorithm 2 describes the (3) and (4) phase in detail.

Algorithm 2: Multi-layer federal Party to GM

Input: Global shared data sets D, batchsize b, learing rate η, number of local epoch E^{loc}, hyper-parameter μ, temperature τ, $N \leftarrow |P_i|_{i\in group^{GMID}}$

Output: The final model w_t^{gm}

GM executes:
initialize w^0
for $t = 0,1,\dots,T-1$ **do**
  for $i = 1,2,\dots,N$ **in parallel do**
    send the group model w_t^{gm} to P_i
    $w_e^i \leftarrow$ **PartyLocalTraining**(e, w^i)
  $w_t^{gm} \leftarrow \frac{1}{N}\sum_{i=1}^{N}\frac{|D^i|}{|D^{GM}|}w_t^{GM}$
return w_t^{gm} and calculate tag_t^{GM}as $\{tag^1||\dots||tag^N||byte_{mtx}^i||pos\}$
PartyLocalTraining(e, w^i):
$w_e^i \leftarrow w^i$
for epoch $e = 1,2,\dots,E^{loc}$
  for each batch $\boldsymbol{b} = \{x, y\}$ of D^i **do**
  $L_{loc} \leftarrow CrossEntropyLoss\left(F_{w_e^i}(x), y\right)$
  $w_e^i \leftarrow w_e^i - \eta\nabla\ell_{loc}$
return w_e^i to GM

3.4 *Editable Keychain*

Considering the issues of key management and data validation of the global model, we apply cryptography technology to achieve efficient model encryption. In addition, a key chain is introduced to store and manage GM's AES keys. However, in IIoT scenarios, there are situations such as a change of GM or manufacturers leaving FL training, Therefore, the information on the keychain needs to be modified to achieve changes in GM access control rights. In order to solve the above problems, this section combines the chameleon hash function [36] to design a Keychain with the editable attributes.

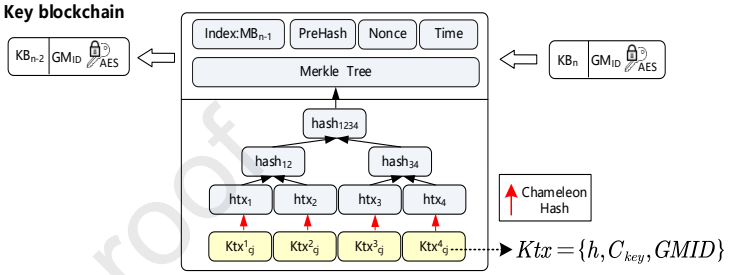

Fig. 5. Keychain block structure

Each Block$_i$ in the Keychain can be divided into two parts, the block body and the block header, however the PreHash of the Keychain is designed using the chameleon hash. The specific structure of the Keychain Block$_i$ is shown in Fig.5. The block header contains the block identity index value *Index*, a timestamp *Time*, the hash value *PreHash* of the previous block, a random *Nonce* and the *Merkle Tree*. the block body contains the entire Merkle Tree, and the key transaction Ktx_{cj} in the block body contains the chameleon hash, the encrypted AES key C_{key}, and the AES key belongs to the identity sequence number GMID of the GM.

The Keychain contains four main phases: (1) GM identity registration, (2) GM key uploading, (3) GM key modification, and (4) user key request. The four steps are described in detail below.

(1) **GM identity registration:** GM generates a pair of keys through the RSA key generation function $KeyGen_{RSA}(pp) \rightarrow (pk, sk)$, and secretly saves its own private key sk. The GM generates its own AES key through AES key generation function $KeyGen_{AES}(pp) \rightarrow key$, and secretly saves the key. GM sends a key registration application to the server, the private key is used to sign its own GMID, $SIG(GMID, sk) \rightarrow \sigma$, and the signature result σ is sent to the server. The server verifies the correctness of the signature $VER(\sigma, pk) \rightarrow GMID$, and if the verification is successful, the server uses the server private key sk_{sever} to sign the identity serial number SID and the GM's identity serial number GMID, $SIG(GMID, SID, sk_{sever}) \rightarrow \sigma'$ and returns the result to the GM. After GM verifies the correctness of the signature, $VER(\sigma', pk_{sever}) \rightarrow (GMID, SID)$, the server and GM send signatures to each other to achieve two-way verification and establish a secure and trusted channel with the server.

(2) **GM key upload Keychain:** GM generates the chameleon hash public key y_p and trapdoor key x_s, and broadcasts the key y_p. GM uses public key pk_{server} to encrypt its own AES key

$ENC_{RSA}(key, pk_{server}) \rightarrow C_{key}$, and generates chameleon hash random number $CH\_update((C_{key}||GMID), x_s, h) \rightarrow (r, s)$ to the server. The server verifies whether the GM identity is valid, and cancels the key upload request if it does not pass the identity verification. Otherwise, it sends $m = \{C_{key}, GMID\}$ and (r, s) to the miner, which generates the chameleon hash $CH(m, y_p, r, s) \rightarrow h$ and uploads the key transaction $Ktx = \{h, C_{key}, GMID\}$ as the content in the block.

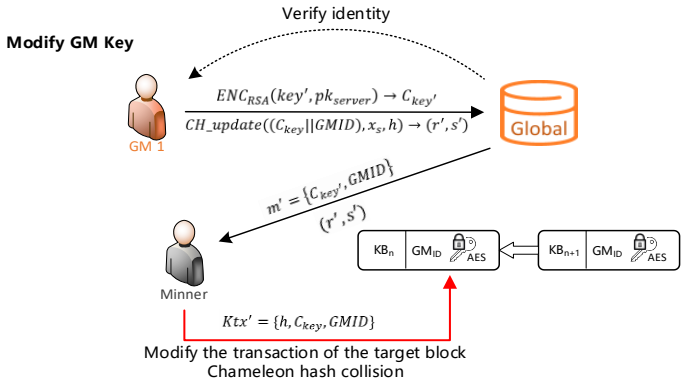

Fig. 6. Keychain Chameleon Hash Modification

(3) **GM key modification:** As shown in Fig.6, when GM needs to change access control authority, it can change its own key ciphertext C_{key} in the Keychain. GM first generates a new AES key through the AES key generation function $KeyGen_{AES}(pp) \rightarrow key'$ and encrypts the new AES key $ENC_{RSA}(key', pk_{server}) \rightarrow C_{key'}$. GM calculates the new random $CH\_update((C_{key'}||GMID), x_s, h) \rightarrow (r', s')$ using the trapdoor key x_s with the chameleon hash function, and sends a random (r', s') to the server. The server verifies whether the GM identity is valid and cancels the key modification request if it does not pass the identity verification. Otherwise, $m' = \{C_{key'}, GMID\}$ and (r', s') will be handed over to the miner, who verifies the chameleon hash, $CH(m', y_p, r', s') \rightarrow h$ and replaces the key transaction $Ktx' = \{h, C_{key'}, GMID\}$ in the block to complete the modification of the AES key on the Keychain.

(4) **User request key:** When other nodes obtain an encrypted global model from the IPFS system, they need the AES key of the GM participating in the training to decrypt the model. Other GMs locate the key transaction Ktx issued by the target GM on the Keychain through GMID, obtain the C_{key} and establish a trusted channel with the server. The server decrypts C_{key} to obtain $DEC_{RSA}(C_{key}, sk_{server}) \rightarrow key$ and sends it to the GM requesting the key, which extracts the corresponding hash index through the Modelchain and obtains $C_{w^{glb}}$ in IPFS, and uses the AES key to decrypt the ciphertext of the global model $DEC_{AES}(C_{w^{glb}}, key) \rightarrow w^{glb}$ to obtain the target global model.

Significantly, the chameleon hash function in steps (2)(3) can only edit the blockchain if the GM holds the trapdoor key. This edit will also be recorded and is undeniable. Therefore, BMFL ensures that the block header hash on the Keychain remains unchanged while modifying and editing the AES key.

IV. Performance Evaluation

In this section, we will demonstrate the performance of the proposed BMFL scheme through comparative experiments. In addition, the key upload blockchain time overhead based on different hash algorithms is tested to evaluate the effectiveness of the editable blockchain. First, we obtain the testing accuracy of BMFL through extensive experiments on representative datasets. The experiment was deployed in Windows 10 OS with AMD Ryzen5 5600X 6-Core Processor@3.70GHz CPU, 16GB of RAM on board and NVIDIA GeForce RTX3070 8G GPU. in Python 3.8.12, PyTorch 1.10.0, and CUDA 10.0.

4.1. *BMFL Performance Analysis*

4.1.1. *Performance of BFML on general datasets*

(1) This paper selects four real data sets widely used for data classification, MNIST, EMNIST, CIFAR-10, and CIFAR-100, to evaluate BMFL, and compares it with two classic algorithms, FedAvg and FedProx, to demonstrate the convergence process of local and global training and test accuracy. In order to better characterize the impact of this scheme on convergence that effectively solves the problem of heterogeneity, the Non-IID settings that we adopt is Dirichlet: Label distribution on each device follows the Dirichlet distribution. First, common hyperparameters are set under different data sets. The network structure uses ResNet18, the global training rounds are 30 rounds, the local training rounds are 5 rounds, the number of participants is 5, the batch size is set to 16, and the learning rate η is set to 0.001 and uses stochastic gradient descent with momentum.

As shown in Fig.7, the iterations end after 150 rounds of local training among 5 participants and 30 rounds of global training. With the simple handwritten digit picture MNIST dataset, three aggregation algorithms all locally reached convergence after 50 rounds of iterations, and the final global accuracy of BMFL reached 99.46%; On the derived EMNIST dataset, the global accuracy of BMFL reaches 95.69%, and it basically reaches convergence after 50 rounds of local training iterations. The other two algorithms also have good accuracy, but the convergence process is very unstable. On the CIFAR-10 dataset, a small data set for recognizing universal objects, the BMFL global model accuracy finally reached 81.42%. Due to the previous division of the dataset into non-IID, the training accuracy of FedAvg was only able to reach 70.04%, and the convergence process was unstable. The accuracy of FedProx $(\mu = 1)$ was on par with our scheme. On the CIFAR-100 dataset, which has a variety of label types, the accuracy of BMFL and the other two aggregation algorithms decreased. The final accuracy of the BMFL global model reached 60.05%. Compared to FedAvg, the accuracy increased by about 20%, and compared to FedProx, the improvement was about 5%. Furthermore, the convergence process of our scheme was relatively stable with almost no oscillation. Therefore, when facing heterogeneous data in public datasets, BMFL has better test accuracy and stable convergence, which proves the effectiveness of our scheme.

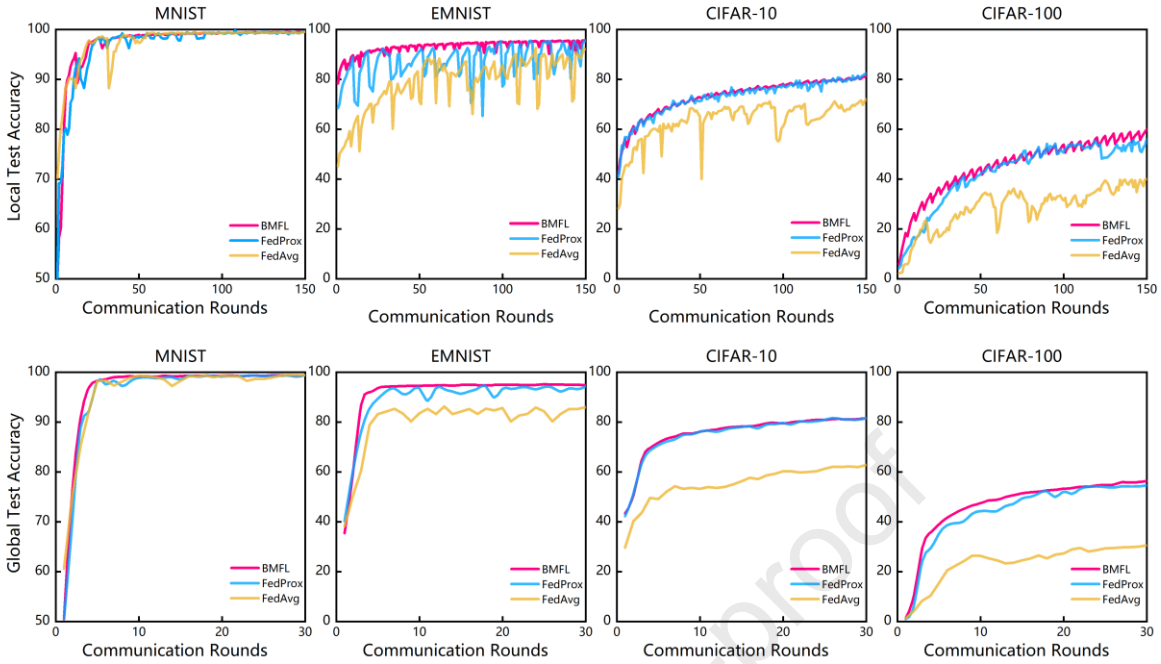

Fig. 7. Comparison of accuracy between BMFL and different FL algorithms

(2) We compare the test accuracy of BMFL with three synchronous and asynchronous FL schemes under the above four datasets. Among them, FedHiSyn [37] uses a hierarchical synchronous model update, FedAT [38] uses a semi-asynchronous FL model update. SCAFFOLD [39] solves the client-drift when local clients update synchronously by adding correction terms. Here we only focus on the situation where all devices participate and there is no disconnection, and the Non-IID is set to Dirichlet (0.8). As shown in Table 3, BMFL is not as effective as the other three schemes on the EMNIST dataset, but the accuracy in the remaining three datasets is on average 0.14%, 1.49% and 1.38% higher. It shows that it has better performance under most data sets. BMFL can help train high-accuracy models by grouping local clients, combining synchronous and asynchronous training methods, and matching feature-based models on the model blockchain. In comparison to these three solutions for heterogeneity issues, this paper also considers the privacy protection of local data.

Table 3.
Comparison of training accuracy with related work

Schemes	Dirichlet (0.8)			
	MNIST	EMNIST	CIFAR-10	CIFAR-100
BMFL(ours)	**98.15%**	**88.02%**	**80.41%**	**40.52%**
FedHiSyn[37]	97.98%	90.50%	80.14%	42.85%
FedAT[38]	97.83%	89.32%	77.80%	38.05%
SCAFFOLD[39]	98.22%	90.95%	78.99%	36.25%

4.1.2. *Performance of BFML on general datasets*

(1) To demonstrate the usefulness of BMFL, the steel surface defect dataset "SeverStal" were used to compare the accuracy of BMFL and FedAvg. Each image in the fault dataset can contain no defects, contain defects of a single class or multiple classes. We segmented the defect classes (Class Id = [1, 2, 3, 4]) and the segments for each defect class were coded as one row. As shown in Fig.8, the BMFL local model reaches convergence after 15 rounds of iterations and the accuracy is stable at about 70%, while the accuracy of FedAvg after convergence is about 40% and fluctuates significantly. The global test accuracy of BMFL is 72.09%, which is about 26% higher than FedAvg, which proves that BMFL has better results in the IIoT.

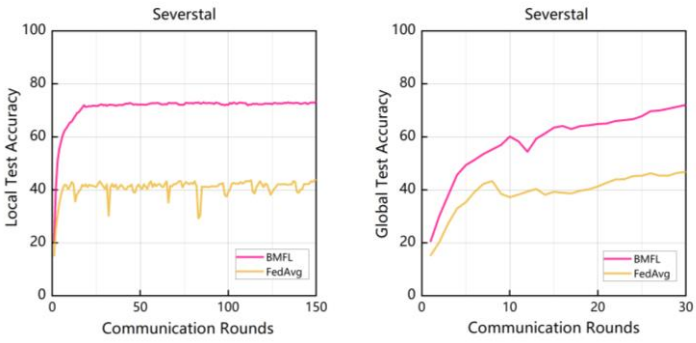

Fig. 8. Accuracy performance of BMFL on SeverStal

(2) Considering the limitations of computing power and communication resources in the IIoT, we compare the residual network ResNet18 used in this solution with the multi-layer perceptron (MLP) [40]. The hyperparameter settings are the

same as in Section 5.1.1 above. Analyze the experimental performance of these two different training networks on the BMFL architecture under the fault data set. The results are shown in Fig.9.

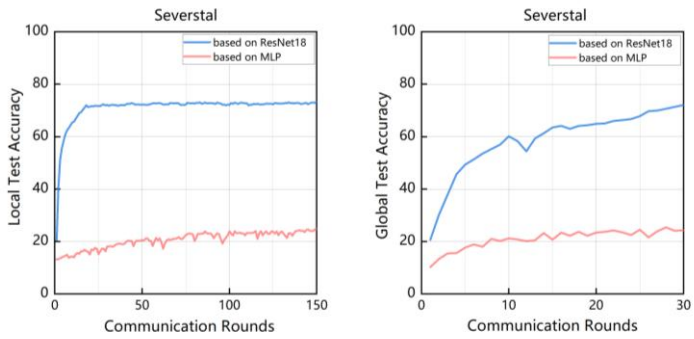

Fig. 9. Comparison of accuracy of BMFL under different network

In order to comprehensively demonstrate the performance under different training networks, under the same premise of hyperparameter settings and data set preprocessing in Section 5.1.1, the training accuracy of BMFL under the two network structures is shown in Table 4, In the MNIST dataset, which only has one image channel of grayscale images, the process of feature extraction is relatively easy. The global training effects of the two network structures both reach 95%. However, the derived EMNIST dataset causes a decrease in the training accuracy of the two models. In the CIFAR dataset, which has more categories and 3-channel RGB images, the accuracy performance of the two networks is average. However, the ResNet18 network used in our scheme outperforms MLP in terms of training results.

Table 4.
Comparison of training accuracy with different network

Network	Datasets				
	MNIST	EMNIST	CIFAR-10	CIFAR-100	Severstal
ResNet18	99.41%	94.79%	81.51%	56.37%	73.33%
MLP	96.92%	76.25%	50.40%	30.32%	24.79%

4.2. *Editable Keychain analysis*

In order to verify the feasibility of the editable Keychain, this section compares the key uploading time overhead of the commonly used hash algorithm in the blockchain and the chameleon hash algorithm. In the blockchain simulation, a Keychain is first built, a total of 10 nodes are set to be uploaded at the same time, and tested the time overhead of hash calculation for AES key with lengths of 128 bit ~ 1024 bit (ignoring communication overhead). As shown in Fig.10, as the growth of the key length hash computation overhead increases, the traditional hash operation in the key length of 1024 bit, the time overhead is within 0.14 ms, of which SHA256 time overhead is minimum. The chameleon hash algorithm reaches 10.6519 ms, the key length of 256 bit chameleon hash calculation time is only 1.86 ms. this shows that in the key length of the smaller hash operation, the time overhead of the chameleon hash is within the acceptable range. Therefore, in this scheme, the chameleon hash is used in the Keychain when there is a need for modification, and the SHA256 hash algorithm is used in the Modelchain, which has a reasonable time overhead under the premise of realizing the key management and access control functions.

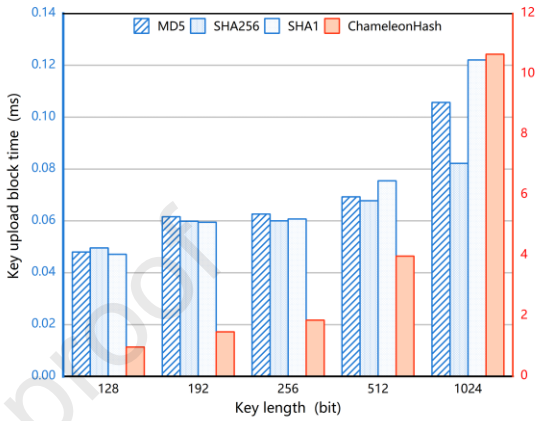

Fig. 10. Time cost for different hash of the Keychain

V. Conclusion

This paper designs a multi-layer group federation scheme based on dual-blockchain to solve the problems of data heterogeneity and privacy protection in the IIoT. Through the multi-layer grouping structure to divide the feature data according to the geographical region, the same feature model within the group is trained synchronously, the inter-group and global aggregation is trained asynchronously. In order to solve the problem of asynchronous update of the global model, a model chain is introduced to store global model labels to optimize the convergence efficiency of the group model, and the model is stored in IPFS in ciphertext. In addition, an editable Keychain is introduced, using the chameleon hash to achieve key update and management on the chain. Experimental results show that the proposed BMFL scheme can improve the accuracy of model training in heterogeneous data environments, while conducting simulations under real industrial fault datasets, significantly improving the convergence behavior under heterogeneous data compared with traditional federated learning framework. In the future, it is necessary to solve the problem of time overhead of model uploading to improve the efficiency of the overall solution.

Funding

This work was supported in part by Natural Science Basic Research Program of Shaanxi under Grant No. 2022JM-346.

References

[1] Zhao B, Fan K, Yang K, et al. Anonymous and privacy-preserving federated learning with industrial big data[J]. IEEE Transactions on Industrial Informatics, 2021, 17(9): 6314-6323.

[2] C. Yin, J. Xi, R. Sun, and J. Wang, "Location privacy protection based on differential privacy strategy for big data in the industrial internet of things," IEEE Transactions on Industrial Informatics, vol. 14, no. 8, pp. 3628–3636, 2017.

[3] Z. Zhang, M. Dong, L. Zhu, Z. Guan, R. Chen, R. Xu, and K. Ota, "Achieving privacy-friendly storage and secure statistics for smart meter data on outsourced clouds," IEEE Transactions on Cloud Computing,2017.

[4] McMahan H B, Moore E, Ramage D, et al. Federated learning of deep networks using model averaging[J]. arXiv preprint arXiv:1602.05629, 2016, 2: 2.

[5] Wang Q, Li Q, Wang K, et al. Efficient federated learning for fault diagnosis in industrial cloud-edge computing[J]. Computing, 2021, 103(10): 2319-2337.

[6] Lu Y, Huang X, Dai Y, et al. Blockchain and federated learning for privacy-preserved data sharing in industrial IoT[J]. IEEE Transactions on Industrial Informatics, 2019, 16(6): 4177-4186

[7] Z. Liu, T. Li, V. Smith, and V. Sekar, "Enhancing the privacy of federated learning with sketching," preprint arXiv:1911.01812, 2019.

[8] McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data[C]//Artificial intelligence and statistics. PMLR, 2017: 1273-1282.

[9] Xu C, Qu Y, Xiang Y, et al. Asynchronous federated learning on heterogeneous devices: A survey[J]. Computer Science Review, 2023, 50: 100595

[10] Zhao Y, Zhao J, Jiang L, et al. Privacy-preserving blockchain-based federated learning for IoT devices[J]. IEEE Internet of Things Journal, 2020, 8(3): 1817-1829.

[11] Zhou C, Tian H, Zhang H, et al. TEA-fed: time-efficient asynchronous federated learning for edge computing[C]//Proceedings of the 18th ACM International Conference on Computing Frontiers. 2021: 30-37.

[12] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf, Accessed on: Apr. 3, 2020.

[13] K. Gai, Y. Wu, L. Zhu, Z. Zhang, and M. Qiu, "Differential privacy-based blockchain for industrial internet-of-things," IEEE Trans. Ind. Informat., vol. 16, no. 6, pp. 4156–4165, Jun. 2020

[14] Xu J, Xue K, Tian H, et al. An identity management and authentication scheme based on redactable blockchain for mobile networks[J]. IEEE Transactions on Vehicular Technology, 2020, 69(6): 6688-6698

[15] K. Gai, Y. Wu, L. Zhu, L. Xu, and Y. Zhang, "Permissioned blockchain and edge computing empowered privacy-preserving smart grid networks," IEEE Internet Things J, vol. 6, no. 5, pp. 7992–8004, Oct. 2019.

[16] Liu Q, Li K. Decentration transaction method based on blockchain technology[C]//2018 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS). IEEE, 2018: 416-419.

[17] Tseng L, Wong L, Otoum S, et al. Blockchain for managing heterogeneous internet of things: A perspective architecture[J]. IEEE network, 2020, 34(1): 16-23.

[18] Otoum S, Al Ridhawi I, Mouftah H. Securing critical IoT infrastructures with blockchain-supported federated learning[J]. IEEE Internet of Things Journal, 2021, 9(4): 2592-2601.

[19] Qu Y, Pokhrel S R, Garg S, et al. A blockchained federated learning framework for cognitive computing in industry 4.0 networks[J]. IEEE Transactions on Industrial Informatics, 2020, 17(4): 2964-2973

[20] Zhao Y, Li M, Lai L, et al. Federated learning with non-iid data[J]. arXiv preprint arXiv:1806.00582, 2018.

[21] Li T, Sahu A K, Zaheer M, et al. Federated Optimization in Heterogeneous Networks [DB/OL]. https://arxiv.org/abs/1812.06127v4, 2020-04-21.

[22] Chen Y, Ning Y, Slawski M, et al. Asynchronous online federated learning for edge devices with non-iid data[C]//2020 IEEE International Conference on Big Data (Big Data). IEEE, 2020: 15-24.

[23] Li Q, He B, Song D. Model-Contrastive Federated Learning[C]//Nashville, TN, USA, 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021: 10708-10717.

[24] Zhang D Y, Kou Z, Wang D. Fedsens: A federated learning approach for smart health sensing with class imbalance in resource constrained edge computing[C]//IEEE INFOCOM 2021-IEEE Conference on Computer Communications. IEEE, 2021: 1-10.

[25] Zhang Y, Duan M, Liu D, et al. CSAFL: A clustered semi-asynchronous federated learning framework[C]//2021 International Joint Conference on Neural Networks (IJCNN). IEEE, 2021: 1-10.

[26] Melis L, Song C, De Cristofaro E, et al. Exploiting unintended feature leakage in collaborative learning[C]//2019 IEEE symposium on security and privacy (SP). IEEE, 2019: 691-706.

[27] Mothukuri V, Parizi R M, Pouriyeh S, et al. A survey on security and privacy of federated learning[J]. Future Generation Computer Systems, 2021, 115: 619-640.

[28] Wei K, Li J, Ding M, et al. User-level privacy-preserving federated learning: Analysis and performance optimization[J]. IEEE Transactions on Mobile Computing, 2021, 21(9): 3388-3401.

[29] Bonawitz K, Ivanov V, Kreuter B, et al. Practical secure aggregation for privacy-preserving machine learning[C]//proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 1175-1191.

[30] Guan Z, Wan Z, Yang Y, et al. BlockMaze: An efficient privacy-preserving account-model blockchain based on zk-SNARKs[J]. IEEE Transactions on Dependable and Secure Computing, 2020, 19(3): 1446-1463.

[31] Chen J, Xue J, Wang Y, et al. Privacy-Preserving and Traceable Federated Learning for data sharing in industrial IoT applications[J]. Expert Systems with Applications, 2023, 213: 119036.

[32] Chen J, Li K, Philip S Y. Privacy-preserving deep learning model for decentralized vanets using fully homomorphic encryption and blockchain[J]. IEEE Transactions on Intelligent Transportation Systems, 2021, 23(8): 11633-11642.

[33] Chen C, Li Y, Wu Z, et al. Privacy Computing Meets Metaverse: Necessity, Taxonomy and Challenges[J]. arXiv preprint arXiv:2304.11643, 2023.

[34] Ali M S, Dolui K, Antonelli F. IoT data privacy via blockchains and IPFS[C]//New York, USA, 7th International Conference on the Internet of Things.2017:1-7.

[35] Yun J, Goh Y, Chung J M. Analysis of mining performance based on the mathematical approach of PoW[C]//2019 International conference on electronics, information, and communication (ICEIC). IEEE, 2019: 1-2.

[36] Krawczyk H, Rabin T. Chameleon hashing and signatures[J]. Cryptology ePrint Archive, 1998.

[37] Li G, Hu Y, Zhang M, et al. FedHiSyn: A hierarchical synchronous federated learning framework for resource and data heterogeneity[C]//Proceedings of the 51st International Conference on Parallel Processing. 2022: 1-11.

[38] Chai Z, Chen Y, Anwar A, et al. FedAT: A high-performance and communication-efficient federated learning system with asynchronous tiers[C]//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2021: 1-16

[39] Karimireddy S P, Kale S, Mohri M, et al. Scaffold: Stochastic controlled averaging for federated learning[C]//International conference on machine learning. PMLR, 2020: 5132-5143.

[40] Fu A, Zhang X, Xiong N, et al. VFL: A verifiable federated learning with privacy-preserving for big data in industrial IoT[J]. IEEE Transactions on Industrial Informatics, 2020, 18(5): 3316-3326.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: