Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.048431

ARTICLE

A New Solution to Intrusion Detection Systems Based on Improved Federated-Learning Chain

Chunhui Li[1,*] **and Hua Jiang**[2]

[1]School of Computer and Electronic Information, Guangxi University, Nanning, 530000, China

[2]Cyber Security and Information Center, Guangxi University, Nanning, 530000, China

*Corresponding Author: Chunhui Li. Email: 2113391018@st.gxu.edu.cn

Received: 07 December 2023 Accepted: 18 April 2024 Published: 20 June 2024

ABSTRACT

In the context of enterprise systems, intrusion detection (ID) emerges as a critical element driving the digital transformation of enterprises. With systems spanning various sectors of enterprises geographically dispersed, the necessity for seamless information exchange has surged significantly. The existing cross-domain solutions are challenged by such issues as insufficient security, high communication overhead, and a lack of effective update mechanisms, rendering them less feasible for prolonged application on resource-limited devices. This study proposes a new cross-domain collaboration scheme based on federated chains to streamline the server-side workload. Within this framework, individual nodes solely engage in training local data and subsequently amalgamate the final model employing a federated learning algorithm to uphold enterprise systems with efficiency and security. To curtail the resource utilization of blockchains and deter malicious nodes, a node administration module predicated on the workload paradigm is introduced, enabling the release of surplus resources in response to variations in a node's contribution metric. Upon encountering an intrusion, the system triggers an alert and logs the characteristics of the breach, facilitating a comprehensive global update across all nodes for collective defense. Experimental results across multiple scenarios have verified the security and effectiveness of the proposed solution, with no loss of its recognition accuracy.

KEYWORDS

Cross-domain collaboration; blockchain; federated learning; contribution value; node management; release slack resources

1 Introduction

Data exchange between database systems is often hindered by various factors, including incongruent data format standards, privacy apprehensions, and intricate procedures, resulting in data silos within organizations. However, data should not be isolated in different departments within the same organization, but should be shared and integrated. Nonetheless, data segregation within different divisions of an enterprise is counterproductive; instead, data should be collaboratively shared and integrated to prevent isolation from external communication networks [1]. To address this issue, Google [2] has proposed a new artificial intelligence model based on federated learning (FL), which

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

can protect the data in distributed terminals from not being leaked, and the model can be trained and updated safely. Countries worldwide are enhancing their data security and privacy protection. In 2018, the European Union officially implemented the General Data Protection Regulation (GDPR) [3], one of the successful reliable legislations in this regard, which prohibits enterprises from using vague language or lengthy privacy policies related to gaining access to data. However, traditional FL algorithms are troubled by five major problems [4]: Single points of failures [5], lack of trust between nodes [6], poisoning attacks [7], lack of corresponding incentives, and model ownership issues. Consequently, FL still carries the risk of information security during the uploading and downloading processes due to single central servers [8]. These problems can lead to mutual distrust between terminals and nodes, forming a major challenge hindering the development of federated learning. Blockchain technology, leveraging cryptography for data validation and storage, offers a decentralized alternative to remediate these issues.

In the context of blockchain technology, the need to strike a balance between computational efficiency and storage optimization has emerged as a critical and formidable challenge. To address this pressing issue, the present investigation introduces a novel lightweight FL chain framework, denoted as MLchain, which is capable of facilitating node management within the framework of traditional Blockchain-based Federated Learning (BFL). Furthermore, this research integrates incentive mechanisms and proof-of-work principles into the blockchain structure in order to enhance the operational efficiency of the FL chain. Concurrently, a management component is devised to address the presence of low-contributing nodes, thereby mitigating the potential disruption caused by inactive or unreliable nodes to the learning process.

In this research endeavor, the dataset is partitioned into separate entities, namely a test set and a sample set, designated for model training and recognition tasks, utilizing a fusion of FL and blockchain. The robustness and security of the proposed framework are rigorously evaluated against a spectrum of attack vectors. Subsequent to the experimentation phase, various FL algorithms are substituted with alternative counterparts, with meticulous documentation of their corresponding parameters, in an endeavor to curtail computational burden and communication overheads. The end objective is to furnish a practical solution that can be effectively deployed within Intrusion Detection Systems (IDS). To evaluate the efficacy of the proposed approach, a realistic use-case scenario is meticulously outlined, juxtaposing this novel paradigm against incumbent algorithms with respect to temporal efficiency, computational requisites, and storage footprint.

The primary contributions of this research are outlined as follows:

1) An MLchain scheme is proposed utilizing the FL framework to maintain the original accuracy and balance the computation and storage overheads effectively. Moreover, various categories of malicious attack scenarios have been formulated to evaluate the resilience and anti-attack capabilities of the MLchain scheme.
2) In order to enhance the learning efficiency of the model, a Proof of Work (PoW) mechanism is incorporated. The PoW value diminishes after a specified number of iterations, triggering early warning processing by the node management function if it falls below the threshold. Subsequently, the node with the maximum value serves as the central service node responsible for parameter reception and model integration, while other nodes engage in local training and related tasks.
3) This study introduces the function modules of IDS within the system and validates the proposed scheme through real-world scenario testing. A comparative analysis is conducted

to evaluate the time efficiency of the MLchain scheme as opposed to the traditional FL framework.

2 Related Work

2.1 Blockchain

The participants within the blockchain network are commonly referred to as miners, with their primary responsibility being the generation, validation, and propagation of block transactions. Utilizing cryptography and smart contract technology, each transaction record is rendered immutable, ensuring the integrity and sequential nature of the blockchain [9]. The scope of blockchain technology extends beyond digital currencies, with its potential utility encompassing the construction of secure systems within virtual environments [10].

The operational efficacy of blockchain technology rests upon the seamless alignment of ledger data and the secure execution of block operations, including addition, deletion, and modification. Nonetheless, the consensus mechanism imposes substantial hardware computing and storage requisites on miner nodes. Notably, a single Bitcoin transaction reportedly consumes an amount of energy capable of powering a typical household for a fortnight. As of early October 2022, the sheer size of the Bitcoin ledger had swollen to 500 GB, imposing significant computing and storage burdens on newly integrated nodes [11]. It is worth noting that the meager computing capabilities and limited storage capacities prevalent across most enterprises serve as impediments to the widespread adoption of conventional blockchain technology across diverse domains [12].

In recent years, the realm of consensus algorithms has witnessed a proliferation of advancements and refinements. These innovations include the Equity-Based Proof of Stake (PoS) consensus protocol [13], Synergistic Multiple Proof (SMP) applied to Industrial Internet of Things scenario [14], a novel transaction settlement blockchain that uses Tangle technology based on DAG structure internally for transaction consensus [15], the Proof of Elapsed Work and Luck (PoEWAL) mechanism [16], a Trust-based Proof of Work (PoW) model [17], and the Proof of Block and Trade (PoBT), a lightweight consensus protocol designed for the Internet of Things based on Hyperledger Fabric framework. These novel algorithms offer heightened efficiencies, particularly in terms of reduced computational overhead, in contrast to traditional algorithms [18].

The immutable nature of the ledger necessitates a continuous expansion of its capacity over time. Consequently, nodes with limited storage capabilities may struggle to accommodate the burgeoning blockchain ledger, thus impeding their ability to partake in the maintenance of the blockchain. The dwindling number of active miners may predispose nodes with ample storage capacity to secure more block rewards, potentially culminating in a diminished level of decentralization [12]. Subsequently, ongoing blockchain research endeavors are exploring enhanced storage optimization strategies, with four primary models identified to streamline storage utilization: Historical transaction data deletion [14,19,20], block compression techniques [21], slice-based storage methodologies [22], and the new Local Transaction Architecture approach [18]. Each of these strategies harbors distinct advantages and drawbacks, with their effectiveness contingent upon judicious selection based on storage capacity and processing efficiency to ensure optimal outcomes and mitigate excessive computational overhead.

2.2 Federated Learning

The fundamental concept of FL revolves around decentralizing user data storage, shifting the primary responsibility of the central server to aggregating model parameters from all clients, and

updating the final global model locally on the server. Throughout this process, the global model undergoes optimization [23].

As shown in Fig. 1, it can be seen that FL needs to go through four stages: Broadcast, model aggregation, model upload and download. In the traditional FL approach utilizing the FedSGD algorithm, the central server required frequent uploads for training, demanding constant and stable communication with the devices. McMahan et al. [24] proposed an improved FedAvg algorithm, which stacked as many training iterations as possible for each data upload operation, effectively reducing unnecessary communication overhead. Li et al. [25] proposed the MOON algorithm, which selects a powerful eigenvalue from the global model for training iteration to further improve the effectiveness of FL algorithm. Acar DAE [26] and Li et al. [27] added fitting values to the loss function to improve the energy efficiency of the model aggregation operation.

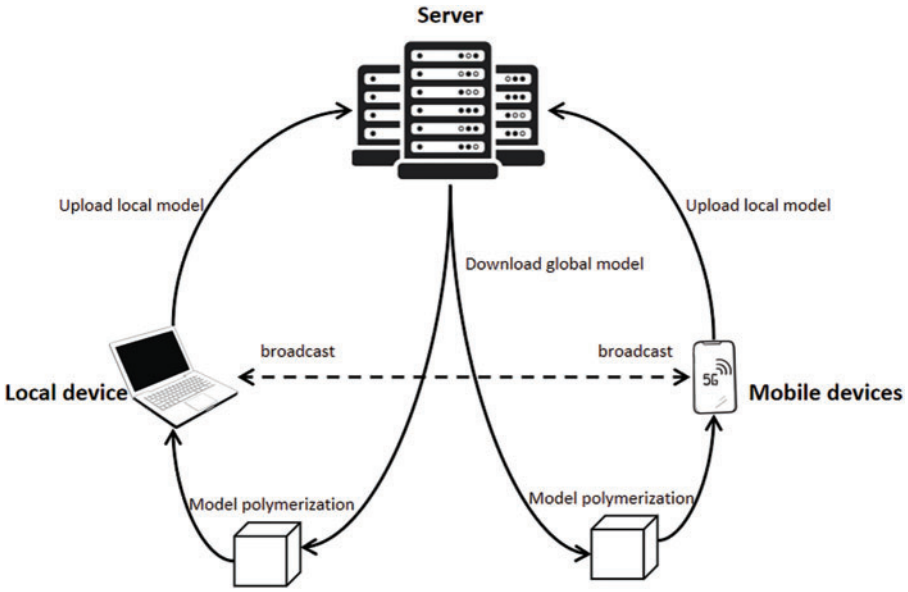

Figure 1: Federated learning process

Despite the advancements, inefficiencies permeate FL frameworks due to their reliance on single-server transport mechanisms, remote distribution of devices, and non-encrypted processing. Scholars have discerned five key flaws within FL: Vulnerabilities to single points of failure, inadequate trust among nodes, susceptibility to poisoning attacks, absence of a corresponding incentive mechanism, and unresolved model ownership concerns. Issues like information blocking, prolonged offline status, or inactivity within nodes can significantly impede training iterations, causing delays in the learning process.

Specifically, poisoning attacks present a malicious attempt to corrupt algorithms or models by transmitting adversarial data from a node to the central server, exploiting established inter-node trust mechanisms and fixed transmission protocols. For instance, in the context of model ownership, each device must locally train the model before transmitting parameters to the central server, often amplifying the global model's accuracy. Operators face challenges in maintaining this intricate learning process without ownership information and a tailored reward mechanism for individual parameters.

2.3 Block-Chain-Based Federated Learning

In recent years, numerous researchers have reviewed the framework and privacy security of blockchain and FL, respectively [28]. The current utility of blockchain within the domain of FL remains predominantly situated in the realm of exploratory research [29]. As shown in Fig. 2, the BFL principle espouses the leveraging of blockchain technology to supplant the centralized server infrastructure endemic to conventional FL frameworks, thereby effecting decentralization. Furthermore, BFL harnesses the innate security features of blockchain to remediate challenges such as data contamination and breaches that plague the FL landscape. Over the past few years, the BFL framework has increasingly garnered attention as a burgeoning area of research, with emphasis placed on enhancing computational efficiency and storage capacities. Nevertheless, the practical deployment of BFL within a specific application context has yet to materialize, warranting further exploration and investigation within the scientific community.

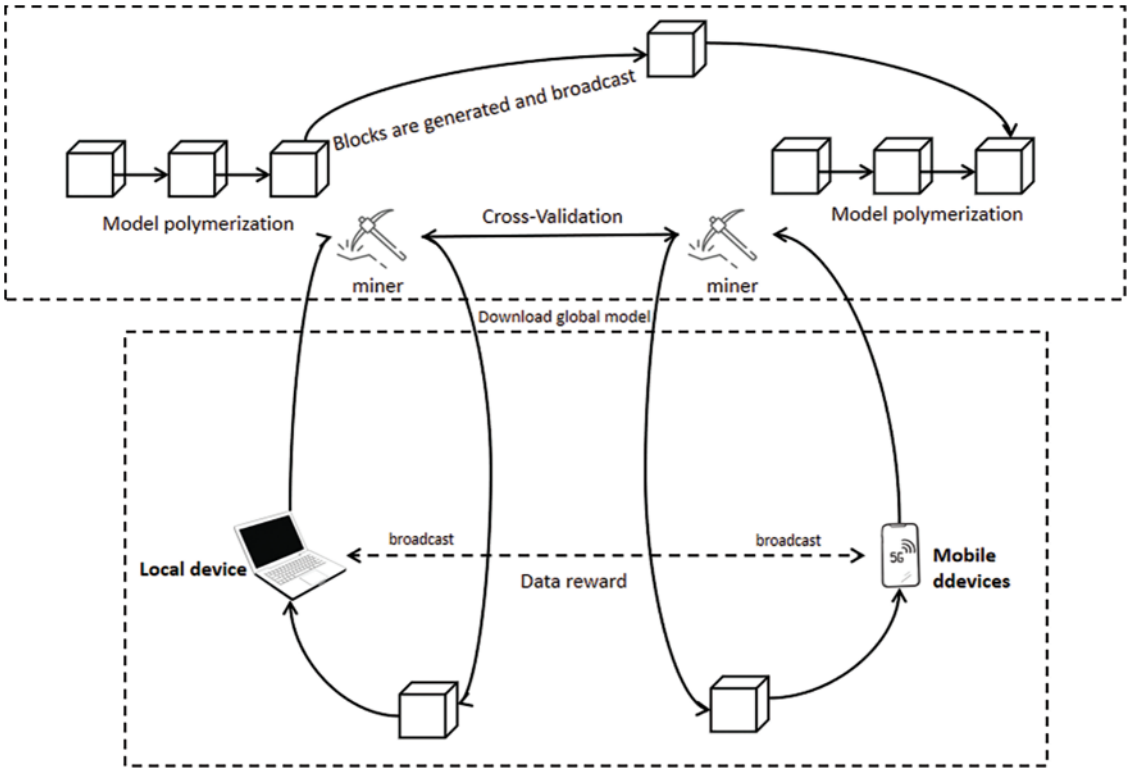

Figure 2: Typical block-chain-based federated learning framework

Majeed et al. [30] used the original FL paradigm in conjunction with blockchain technology to build an alliance chain framework, seamlessly integrating blockchain mechanisms within the process of FL operations related to data uploading and downloading. This fusion ensures the fortification of secure interactions within untrusted environments. Kim et al. [31] proposed the BFL framework, instrumental in conceptualizing the PoW principle to instate an incentive mechanism within FL setups. Upon task initiation, the node exhibiting the highest workload from the previous round of model amalgamation assumes the mantle of a central node, with other nodes engaging in local training and parameter uploads. The central node bears the onus of parameter aggregation, model integration, workload calculation, and block generation. Wang et al. [32] applied the Federated Chain

Framework to a Multi-mode Transformer task for experiments, where computational sacrifices were strategically deployed to markedly enhance the security protocols of the overarching framework. This novel approach assigns blockchain a pivotal role in delineating FL model ownership and bolstering the intricacies of device-mediated parameter uploads.

In light of these findings, the study posits that an increased allocation of computational resources towards the FL training process is indispensable, thereby warranting a sharp focus on minimizing computational and storage overheads inherent to blockchain integration.

2.4 Intrusion Detection System

Intrusion detection (ID) is a crucial security measure for safeguarding user data against malicious traffic attacks. Early Intrusion Detection Systems (IDS) rely on signature database matching or rule filtering for proactive fortification. The development of artificial intelligence has led to the creation of numerous machine learning algorithms for IDS, including k-means, SVM, and naive Bayes [33]. Some researchers also advocate for the use of deep learning [34], which has shown promise in detecting new attack methods like 0-day vulnerabilities and aiding in security diagnosis.

Mahoney et al. [35] introduced artificial intelligence into the network intrusion detection system (NIDS) and made the model learn the protocol vocabulary from the link layer to the application layer to detect unknown attacks. With the rapid development of artificial intelligence technology in recent years, a variety of deep neural networks have been introduced into different NIDS, and typical ones include deep neural networks (DNN) [36], long short-term memory neural networks (LSTM) [37], multi-layer perceptron (MLP) [38], etc. Although intelligent NIDS has a high recognition rate of traditional malicious traffic, the existing models are still carried out by traditional centralized servers, and it is difficult to form a global model for joint defense.

The decentralized paradigm of IDS is a frontier direction in the exploration stage. Empirical studies undertaken by Tang et al. [39] have demonstrated that distributed learning endows performance ratios akin to those of centralized deep learning models while fostering enhanced scalability and resilience. The conception of Distributed Intrusion Detection Systems (DIDS) was first proposed and developed by CERIAS of Purdue University in the United States [40], thereby fraught with the burgeoning blend of artificial intelligence methodologies and Intrusion Detection, which has metamorphosed into a burgeoning research terrain within the domain of cyber security [41].

3 Introduction of System and Concept

3.1 Introduction of System Framework

The system performs two tests sequentially to detect attacks. Firstly, it uses feature matching methods to determine the type of attack, and then employs machine learning to train the model to further identify and confirm the attack. In other words, the IDS continuously monitors network data and immediately captures abnormal data. Upon examination, the system first analyzes the URL to determine whether it is a malicious attack. Upon detection of abnormal data, an alarm is immediately triggered to evaluate the security implications of the data packet. The IDS in this study uses the SVM machine learning algorithm. The general process of artificial intelligence involves three steps: Data preprocessing, feature extraction, and classification, which are not discussed in detail here, as shown in Fig. 3.

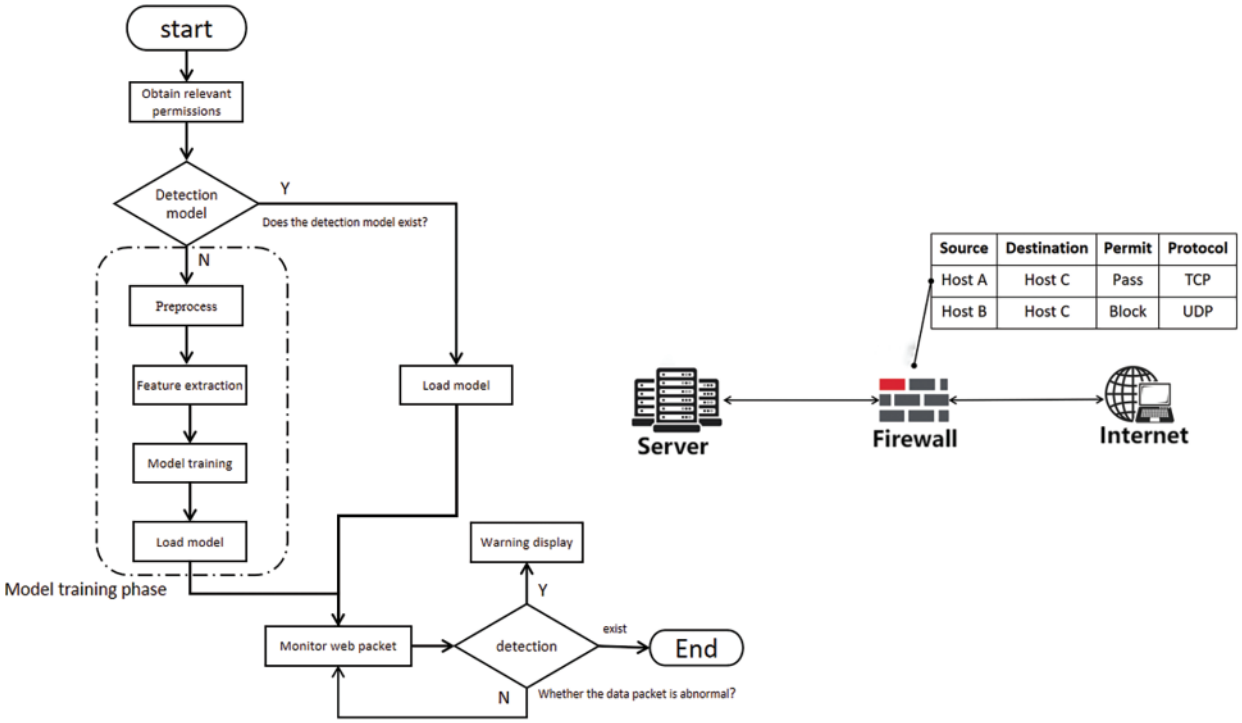

Figure 3: The process of intrusion detection system

The system initially seeks authorization for port monitoring and subsequently validates the availability of the model file. Upon confirmation that all requisite components are prepared, the system commences monitoring network data from the designated port while instantiating the model file for preliminary identification. Furthermore, the device maintains a local signature database containing malicious URL. If an exception occurs, the network packet is immediately captured, and the nature of the attack mode is determined by cross-referencing both the signature database and model inference. In instances where malicious attacks are detected, the firewall disables IP addresses. The system continues to monitor.

The model training phase in the process chart can be replaced with the FL framework. In this setup, the client deploys the machine learning algorithm environment locally to train its own data, eliminating the necessity of establishing a dedicated connection for data transmission and minimizing the risk of data leakage. As shown in Fig. 1, the traditional FL framework consists of a central server and client architecture, known as the CS pattern. The client sends the trained local model parameters to the central server, which collects and amalgamates these parameters to generate a global model. Notably, the central server's computational burden appears manageable, leading to the proposition of the MLchain framework in this study.

Fig. 4 shows that candidate node pools are selected from the client base, consisting of nodes equipped with computing and storage capabilities integrated into the FL network. Each node is responsible for model training and parameters uploading based on its local dataset, maintaining a block data ledger. If a node is selected as the central node, the node receives and integrates parameters uploaded from other nodes, and updates the blockchain with new blocks. This decentralized approach encourages effective resolution of data silo quandaries, allowing each node to contribute optimally.

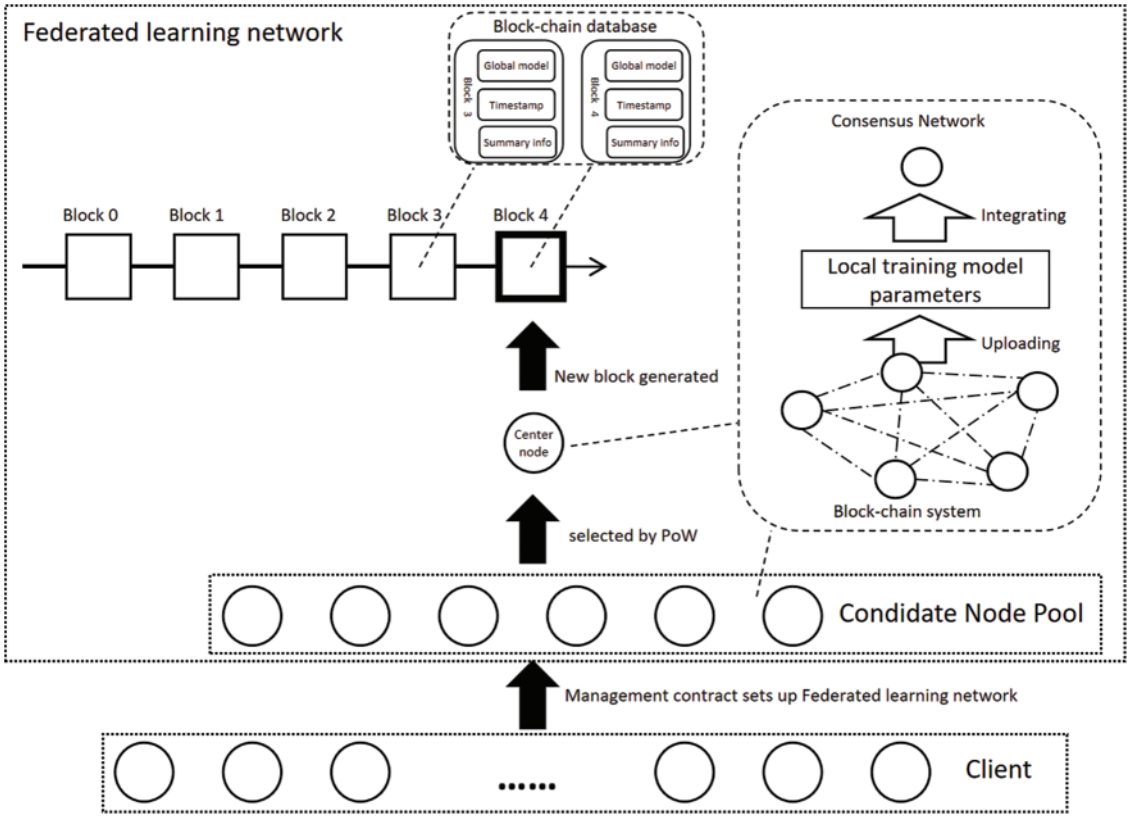

Figure 4: Lightweight federated learning chain framework that supports node management

Finally, Fig. 3 showcases the intrusion detection system's continuous network surveillance post the global model acquisition to fortify system security.

3.2 Formulaic Expression

Simulating multiple client devices within a virtual machine environment, while configuring routing and forwarding functionalities along with local training algorithms, is a crucial aspect of distributed computing systems. In this context, a total of S samples are involved, constituting the client node set $N = \{N_1, N_2, N_3, \ldots, N_n\}$. Here, D_i is the data owned by the ith node and D is the sum of data owned by all nodes. The set of $D = \bigcup_{i=1}^{N_n} D_i.M = \{M_1, M_2, M_3, \ldots, M_m\}$ represents a blockchain or a pool of candidate nodes. Simply put, node N_i upload the trained parameters to the selected central node $M_j.w_k^l$ is the weight trained by the D_k data at the l iteration, W_r is the global model value for round$r$, aiming to minimize the loss function $f(W)$. in all iterative processes beyond the initial step, the previous iteration's weights are required to consolidate the model, with the gradient $\nabla f(W_r)$ being computed by the local vector ∇w_k^r and the global update vector ∇W_r, as depicted by Eqs. (1)–(4).

$$\nabla w_k^r = w_k^r - w_k^{r-1} \tag{1}$$

$$\nabla W_r = W_r - W_{r-1} \tag{2}$$

$$f(W) = \frac{1}{S} \sum_{i=1}^{N_n} \sum_{D_k \in D_i} f_k(W) \tag{3}$$

$$W_r = W_{r-1} + \sum_{i=1}^{N} \frac{N_n}{S} \left(w_i^R - w_i^{(R-1)} \right) \tag{4}$$

Client N_i uploads weight ω_i^l and gradient $(\nabla f_k(W_r))_{S_k \in S_i}$ to central node M_j, and M_j complete the integration of models. Subsequently, the model, timestamp and summary information are packaged to generate a new block. Other clients can download the model from the new block as a local model update or continue operations to train the next round of new models.

The alliance chain appears to lack an incentive mechanism. The central node is only responsible for tasks with relatively lower computational intensity compared to other nodes, presenting an opportunity for positive reinforcement in rewarding the candidate node pool that successfully trains a quality model, thereby fostering a consensual framework within the federated learning network.

For instance, PoW is a consensus mechanism that introduces the concept of incentive mechanism and proof of work. It sets a computation threshold according to the number of nodes, and the threshold directly affects the difficulty of computation. Nodes are required to leverage the hash function to compute a number that keeps getting closer to the threshold, and the formula is Hash(Hash(nonce)) $\leq$ lim it, in which nonceis the random number and limit is the target threshold. This method poses a brute force mathematical challenge where diverse values of limit are tested until meeting the threshold requirement. The main disadvantage of this mechanism is that the more times the threshold is set, the more computing resources are required. Non-central nodes need to upload parameters repeatedly, which inevitably generates a large amount of communication overhead.

Traditional FL adopts the FedSGD algorithm necessitating individualized training result submissions to the server. Conversely, the enhanced FedAVG protocol streamlines operations by consolidating numerous training iterations per round before submission. For such a scenario, the server is initialized, and nodes $S_1, S_2, \ldots, S_n$ participate in FL. The formula is $H_i = \sum_{r=1}^{R} \nabla w_r^i$. R denotes final iteration number, r represents current iteration number, and W_r is the global model value of the round r of the algorithm, w_r^k is the model trained by D_k data in the r round, H_i marks the history record of node D_i.

Each iteration of the algorithm entails the randomized selection of fixed data from D for round r training. When r falls within the three intervals of $r = 1$, $1 < r < R$ and $r = R$, specific tasks are executed as follows:

1. When $r = 1$, an initial scenario unfolds where no node exhibits a contributory role in the first iteration. The candidate node is trained locally and the parameters are uploaded to a central node that will be randomly selected, and then the contribution value of other nodes is calculated after the aggregation model is completed.
2. When $1 < r < R$, the central node selection is intricately tied to the contribution values of the local model from the preceding round. The central node performs a series of operations such as receiving model parameters and aggregating models. The generated model acts as the next loop until it ends.
3. When $r = R$, completing the final round generates a new block for the final global model to join the consortium chain.

The structure and algorithmic framework have been elucidated in the preceding discussion. The MLchain, as delineated in Algorithm 1 of this investigation, is the focal point of analysis. In order

to provide a more nuanced delineation of the roles of the central node and the common node, it is imperative to underscore that the algorithm embarks on an iteration with the initialization model W_0. It is crucial to highlight that the model remains unincorporated in block 0 owing to the generation of block 0 in the inaugural round.

Algorithm 1: The training process of client N_i at the l time in lightweight federated learning chain framework that supports node management

Input: Client node $N = \{N_1, N_2, N_3, \ldots, N_n\}$;
Output: New global model

① Central node M_j is randomly selected from N, essentially born from the node with the highest contribution value of proof of work;

② Central node M_j broadcasts information to node pool N;

③ Client N receiving the broadcast sends identity information to M_j;

④ M_j adds the responding node to candidate node pool M and waits for the preset timestamp before sending the task start request;

⑤ All candidate nodes download the latest upload model from the blockchain system;

⑥ Candidate nodes except M_j confirm the corresponding training parameters, such as local data $D = \{D_1, D_2, D_3, \ldots, D_n\}$, iteration number R, loss function $f(W)$, learning rate and so on of each node;

⑦ Candidate nodes except M_j begin to train local model w_i^{r-1}, compare whether the current iteration number meets $1 < r < R$. If the inequality is satisfied, the model parameters are uploaded to M_j for the next step, otherwise go back to ⑥ to continue the local training;

⑧ M_j receives model parameters for evaluation, calculates node work proof contribution value and model integration operation;

⑨ M_j packages the aggregate model, node number of the model, completion time and other information into blocks, and then uploads it to the blockchain system;

⑩ The proof-of-work contributions of all candidate nodes are reduced proportionally.

3.3 Node Management Component

The node management component is responsible for clearing the nodes whose work proof contribution value is about to return to zero or the long-term accumulation of invalid blocks in the candidate node pool. The system process is introduced in detail in Section 3.2. The node management component will manage nodes according to the incentive mechanism and corresponding contribution values in ①⑧⑩.

The term "contribution value" denotes the node's impact coefficient in the model parameters supplied by the complete candidate node pool, with the node's utilization of computational and storage resources reflected in said parameters. The determination of the contribution value involves the employment of the Shapley value algorithm, widely utilized in financial contexts to aid in investment decision-making. An inherent benefit of this algorithm is enabling investors to garner a deeper comprehension of the risks and advantages associated with a given investment portfolio, empowering them to make more informed investment choices.

Candidate node pool $M = \{M_1, M_2, M_3, \ldots, M_n\}$ has n nodes, and $\mathrm{I} = \{\mathrm{I}_1, \mathrm{I}_2, \mathrm{I}_3, \ldots, \mathrm{I}_n\}$ represents the input variable of these nodes, which is usually also the output variable of the previous iteration. $\mathrm{O} = \{\mathrm{O}_1, \mathrm{O}_2, \mathrm{O}_3, \ldots, \mathrm{O}_n\}$ represents the output variable of these nodes. The Shapley value SV_k of a

certain node $M_k, K \in \{1, 2, \ldots, n\}$, $SV_k = \sum_{i=1}^{n} \frac{\left|\frac{\partial O_i}{\partial I_k}\right|}{\sum_{j=1}^{n}\left|\frac{\partial O_i}{\partial I_j}\right|}$. The magnitude of the Shapley value directly correlates with the impact of node model parameters on the global model output.

The decrement of the contributive value at step ⑩ aligns more closely with the concept of equitable competition across all nodes.

In addition, the central node could potentially engage in malicious behavior, such as maliciously modifying parameter information or forging contribution information. Therefore, any deviations in the parameter data or contribution levels of neighboring blocks may signify nefarious intentions by the service node. This study references an algorithm [42] based on similarity judgement that is used to mitigate the effects caused by Sybil attacks.

As shown in Fig. 5, where the dotted line represents the update vector submitted by the client node, the solid line represents the aggregated model, θ represents the angle made by the update vector of the Sybil node, γ represents the angle made by the update vector of the honest client and the Sybil node, and $\theta < \gamma$ implies that there will be more similarity between the Sybil nodes than between the Sybil node and the other nodes, the more similar the nodes are, the higher the likelihood that they are malicious.

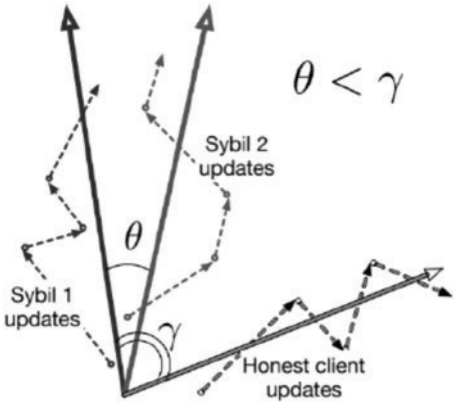

Figure 5: Schematic diagram of similarity judgement algorithm

Upon completion of step ⑧, the node management component is activated to evaluate the contribution value of each node for similarity. Nodes exhibiting excessive similarity within a group are disqualified from being selected as central nodes to safeguard against potential Sybil attacks. Any node with a contribution value consistently below a specific threshold over an extended period faces elimination from the candidate node pool. This process ensures that newly added nodes to the candidate pool can actively engage in the learning procedure. Furthermore, information within the new block specifies the origin of a node's parameters, linking the model of the block to a specific node. Subsequent to the completion of step ⑨, the node management component initiates a cleanup protocol when the blockchain exceeds a designated length, resulting in the enforced removal of the initial chain section to optimize storage space.

4 Experiment and Analysis

4.1 Preparatory Work

This study uses two classical datasets in different fields for training and testing. One is the dataset for intrusion detection, NSL-KDD [43]. This data is a typical big data with few features, with a data volume of about 5 million, of which 4 million are intrusion behavior data and the rest are normal user data. The training set contains 22 attack types, and the test set contains 17 attack types.

The other is the handwriting recognition data MNIST [44], an image data widely used in machine learning and deep learning experiments, published by the National Institute of Standards and Technology (NIST). The training set of the dataset contains 60000 images of 0–9 handwritten digital picture files, and the test set contains 10000 images.

The paper discusses traditional FL, which is divided into Horizontal Federated Learning (HFL), Vertical Federated Learning (VFL), and Federated Transfer Learning (FTL). In order to unify the standard, this study adopts two datasets for experiments: VFL adopts the NSL-KDD dataset which has fewer features and a large number of samples, while HFL adopts the MNIST dataset which is aligned with both the sample data and the feature vectors.

The evaluation indexes of IDS functions are often carried out by confusion matrix and attack rate, which vividly represent the advantages and disadvantages of the classification model. It consists of Recall rate, Precision, F1 and AttackRate, as shown in Eqs. (5)–(8).

Recall rate is a measure of the system's ability to correctly identify and classify attacks correctly. It is calculated as the ratio of true positive predictions (TP) to the sum of true positive and false negative predictions (FN). Therefore, a higher recall rate indicates that the system is better at detecting intrusions:

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

Precision, on the other hand, evaluates the system's ability to correctly identify non-attacks or normal behavior. It is calculated as the ratio of true negative predictions (TN) to the sum of true negative and false positive predictions (FP). A higher precision indicates a lower false positive rate:

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

The F1 Score combines both precision and recall into a single metric, offering a balance between the two. It is the harmonic mean of precision and recall, providing a comprehensive assessment of the model's performance:

$$F1 = 2 * \frac{Recall * Precision}{Recall + Precision} \tag{7}$$

Lastly, the AttackRate measures the proportion of attacks correctly identified by the system, giving insights into the system's effectiveness in detecting intrusions:

$$AttackRate = \frac{Count(Attack_{after})}{m * Count(Attack_{before})} \times 100\% \tag{8}$$

The higher the score of F1, the better the detection performance; F1 = 1.0 indicates that all adversarial cases can be detected without introducing any false negatives; the attack rate indicates

the influence effect in backdoor attack scenarios; the higher the attack rate indicates the worse the security of the system. The symbolic function descriptor is shown in Table 1.

Table 1: Symbolic function descriptor

Symbol	Description
TP (True Positives)	Correctly detected positive instances
FN (False Negatives)	Exception records successfully detect the attack type
FP (False Positives)	Normal records that are detected as abnormal
TN (True Negative)	Exception records are successfully detected as normal
Count ()	Counting function
$Attack_{before}$	Before replacing a label value
$Attack_{after}$	After replacing a label value
m	The number of malicious nodes

The experiments experimentation was conducted using an Intel® CoreTM i7-10700 CPU processor, 32 GB of RAM, and an NVIDIA GeForce RTX 2060 graphics card. Six virtual machines with the same system configuration were set up for the experiment. The experimental environment was configured within the system: Four nodes engaged in model training, one node managed central server responsibilities, and another node oversaw the blockchain database. Additionally, in certain specialized scenarios, extra cloud servers were deployed for network testing.

The training model for FL utilized a two-layer convolutional neural network. This is a basic form of convolutional neural network that consists of two convolutional layers. The network structure extracted input data features through convolutional operations and combined them into higher-level abstract features through cascade processing of multiple layers of neurons. Ultimately, the model output the prediction results.

Specific hyperparameters were set as follows: Each batch encompassed 30 data pieces; the local learning rate was set to 0.005, and the number of local training rounds was set to 3. The maximum number of training rounds was fixed at 100, and all clients participated in each round of aggregation in the centralized federated learning algorithm. The number of global iterations was determined to be 20. Unless explicitly stated, any additional parameters were retained as indicated in the original paper.

To reflect potential heterogeneity in the system, the computational speed of half of the clients was reduced by a factor of two compared to the other half. To ensure reliable and stable experimental results, each result was independently evaluated five times.

4.2 Experiment and Safety Analysis in Multiple Scenarios

The detection system uses the classical support vector machine (SVM) algorithm to identify network flow. This algorithm is a strong classifier, which can ensure the maximization of the region classification and select the model with excellent performance. The SVM in this section establishes a 5-layer classifier on KDD dataset, sets default algorithm parameters, imports training data to build a kernel function, maps low-dimensional data to high-dimensional space, and finds the optimal classification surface. The KDD dataset is divided into training set and test set, and their proportions are 0.85 and 0.15, respectively.

Three groups of comparison experiments are set up here: The first group is the classic SVM in the field; the second group is neural network model based on traditional BFL; and the third group is neural network model based on the MLchain.

(1) In the common scenario

In the experimental environment without any attack type, the results of SVM, BFL and MLchain experiments are shown in the Fig. 6.

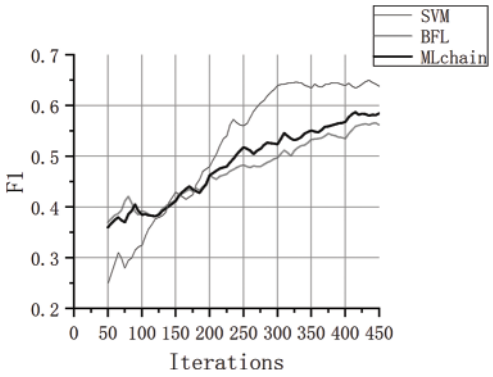

Figure 6: The convergence effect of SVM, BFL and MLchain

Although the introduction of blockchain technology does not affect model recognition accuracy, it will sacrifice a small portion of computing resources to obtain verifiable security and traceable functionality. In a typical scenario, upon reaching 450 iterations, all three sets of experiments achieve an accuracy of approximately 90%, with negligible differences not exceeding 1%.

The SVM model converges and tends to be stable after only 300 iterations, and becomes the best model in the three groups of experiments. The latter two require more iterations to complete convergence. When the number of iterations is specified within 125, the initial detection effect of the BFL model is significantly better than that of the MLchain model, but both outperform the SVM model. This discrepancy arises due to the initially limited computing and storage resources in the blockchain environment, with shortcomings becoming apparent in later stages characterized by enhanced resource deployment and accrued communication costs. In contrast to the conventional BFL framework, MLchain exhibits a performance enhancement ranging between 7% and 12%.

In conclusion, in the non-interference scenario, the final accuracy of the three groups of experiments is not much different, and the addition of blockchain will indeed affect the convergence speed to a certain extent. However, the node management component will gradually play a role in the middle and late iteration process.

(2) Stability experiment of three groups of models in the attack scenario

Malicious nodes are set to carry out backdoor attacks and poison attacks on the three groups of models in turn. The implementation process is that the malicious node adds data that can reduce the accuracy by forging the identity. Backdoor attacks and poison attacks are called directional attacks and non-directed attacks, respectively. Directional attacks refer to replacing a constant value of a label with another constant, while non-directional attacks refer to replacing a constant with a random value.

In the attack scenario, the SVM model exhibits a lack of resistance to attacks, making it susceptible to label replacement and consequent accuracy degradation. Consequently, testing the security of the

SVM model yields insignificant results. As the KDD dataset comprises a vast number of records but a limited variety of around 40 feature types, the impact of malicious attacks is relatively minimal. Therefore, additional experimentation was conducted using the MNIST dataset. In this experiment, the correct rate and attack rate of BFL and MLchain are specifically counted, as shown in the Table 2, where P denotes the accuracy rate and AR represents the vulnerability rate.

Table 2: Stability test of the model

Data	Attack pattern	m	BFL		MLchain	
			P(%)	AR(%)	P(%)	AR(%)
NSL-KDD	No-ops	0	96.8	—	96.5	—
	Backdoor	1	96.5	66.6	96.7	44.4
		2	96.8	72	96.5	72
		5	96.2	83.33	96.5	66.6
	Poison	—	92.7	—	96.5	—
MNIST	No-ops	0	88.4	—	88	—
	Backdoor	1	83.2	23.33	86.4	5.1
		2	76.8	34.5	80.1	15.5
		5	69.6	99	74.6	98.9
	Poison	—	65	—	75.1	—

Regardless of accuracy and attack rate, MLchain model is more robust than BFL. As the number of malicious nodes increases, the intensity of attacks rises, leading to heightened attack rates and diminished accuracy. Given the abundance of KDD data and limited features, it proves challenging to elucidate the impact of backdoor attacks. Only poison attacks can effectively reveal disparities within the model, where MLchain consistently upholds stable accuracy, unlike BFL, which experiences a decline in accuracy. Analysis of the MNIST dataset conspicuously exposes differentials between the two model groups. Whether subjected to backdoor or poison attacks, escalating attack rates correlate with declining accuracies. Despite demonstrating commendable resilience against attacks, MLchain registers notably high attack rates under the backdoor assault of five malicious nodes, suggesting potential enhancement opportunities for the scheme. It is essential to ensure that the number of nodes matches or exceeds that of malicious nodes to achieve optimal defense effectiveness.

(3) Stability experiments of three groups of models in the Sybil attack scenario

The central node may exhibit malicious behavior, such as modifying parameter information or forging contribution information. In this experiment, a malicious node joined the federated learning process by copying its own backdoor attack dataset and forging its identity. The experimental results are shown in Fig. 7.

As can be seen from the experimental results, when the number of malicious nodes reaches 2, the attack rate on the benchmark algorithm FedAVG is nearly 100%, indicating that at this time the malicious nodes have fully achieved their attack purpose and successfully left the backdoor. The similarity-based judgement proposed in this study can achieve almost 100% defense effect when more than 2 identical malicious nodes appear. And when only 1 malicious node appears, the attack rate reaches 100%. This phenomenon is chiefly attributed to the mutual resemblance among the bona

fide nodes which wield an insignificantly minimal influence during the model fusion process, in stark juxtaposition to the distinctive malevolent node, Sybil's unique dissimilarity from other network nodes results in its disproportionately elevated impact within the aggregation process. Nevertheless, a critical flaw is detected in this methodology when disingenuous nodes are sparse and eschew the application of duplicative Sybil stratagems, as such instances may adversely affect the overall efficacy of the training protocol.

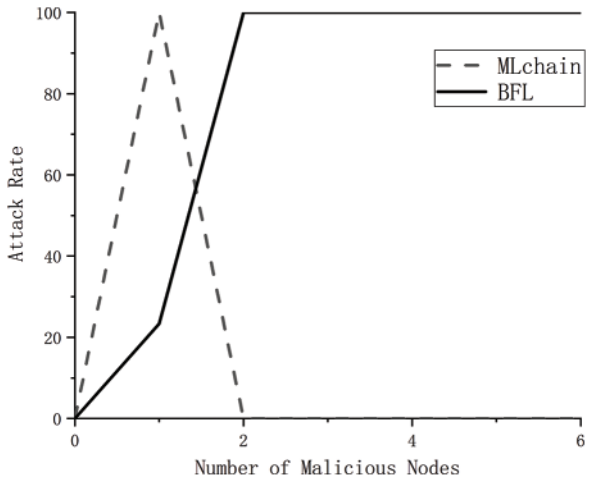

Figure 7: Impact of malicious nodes number on attack rate in Sybil attack scenario

To summarize, the similarity judgment approach propounded herein offers robust resilience against attacks, notwithstanding its diminished effectiveness in scenarios featuring a singular malicious node. It is noteworthy to acknowledge that the incursion rate during singular node vulnerabilities does not scale significantly, thereby implying the meager viability of the proposed scheme in such limited circumstances.

4.3 Federated Learning Algorithm

In order to reduce the computational load and communication overhead of the FLs process, this summary mainly compares the popular traditional FL algorithms, such as FedSGD, FedAvg, FedProx, etc. The MNIST dataset was used to train the model in the same environment, and the accuracy, training time and calculation time ratio were recorded.

Defined as the fraction of time a specific device expends in executing computational tasks relative to the total duration of the collective computing activities, the calculation time ratio serves as a pivotal metric for evaluating the device's engagement in computational endeavors. Enhanced algorithmic computational efficiency is correlated with decreased training periods and heightened share of computational time. A detailed exposition juxtaposing prevalent algorithms is illustrated in Table 3.

As can be seen from the table, FedRep demonstrates a minimum 15% increase in accuracy compared to alternative algorithms, attributed to the distinctive substitution method it employs to prioritize precision. However, under conditions where data lacks independence and uniform distribution, the client encounters challenges wherein newly discovered feature vectors fail to perform satisfactorily in the test set. It is evidently unreasonable to regard the local model as the definitive global model, thereby presenting a key challenge that necessitates resolution and validation within

Federated Learning (FL) environments. Consequently, to enhance the algorithm's rationality and robustness, numerous scholars integrate fitting values into conventional loss functions to harmonize model aggregation rationale. For instance, a loss component is incorporated into FedAvg to gauge the disparity between model predictions and actual targets.

Table 3: Common FL algorithms

Algorithm	Precision	Training time/s	Calculation time ratio/%
FedSGD	0.797	1.7689	32.6
FedAvg [24]	0.799	1.3687	37.6
FedAvg + loss [24]	0.805	1.7606	32.9
FedProx [27]	0.799	1.6289	34.3
FedRep [45]	0.974	2.1425	42.7
FedAsync [46]	0.835	1.4764	100.0

In a highly heterogeneous environment, devices face additional challenges when performing FL tasks. For example, there may be differences in the computing and communication capabilities of devices, and the data distribution may also exhibit characteristics that are not independent and equally distributed. FedProx is a generalization and reconstruction of FedAvg algorithm, with its principal innovation lying in the introduction of a proximal term to deal with heterogeneity in FL, and it has more stable and accurate convergence behavior.

FedAsync achieves a 100% efficiency ratio in computational time, attributable to the direct transmission of parameters to the server post-model aggregation, precluding the necessity for nodes to complete their uploads. Nevertheless, this necessitates the central node to sustain communication channels and computational resources, impeding the attainment of decentralization objectives.

The disparity in accuracy among algorithms aside from FedRep is marginal, approximately 1%. Not only does the FedAvg algorithm exhibit the briefest training duration, but it also excels in equipment utilization. Therefore, the MLchain framework proposed in this study is implemented based on FedAvg algorithm. This section underscores that algorithm comparability is not of paramount importance. Each algorithm is devised for a specific environment or circumstance, and a single experiment cannot fully explain the energy efficiency of the algorithm. The demonstration of an algorithm's efficacy may necessitate the creation of a highly heterogeneous environment incurring substantial costs.

5 Lightweight Blockchain Resource Management Scheme Attempt

In this section, FedAvg + PoW is introduced into the MLchain framework, and the scheme is put into the intrusion detection system for testing. In the simulation of real resource redundancy scenarios, KDD datasets are used for model training to further verify the possibility of applications of FL based on blockchain.

To streamline node management, the conventional global model within the original block is substituted with a file address (referred to as the file hash value), and the actual model file is stored in a database. This modification effectively minimizes the performance overhead arising from ledger storage in the blockchain. The current research leverages the InterPlanetary File System (IPFS) to

enable this functionality. The essence lies in organizing nodes within the network into a distributed file system while employing a more efficient hash-addressing scheme for resource indexing. Notably, IPFS, which remains an ongoing open-source initiative, lacks a direct file deletion interface, thus obviating the need to expunge historical data by simply overwriting it. Consequently, IPFS offers the capability to permanently retain the most recent files amidst frequent upload and download operations, aligning perfectly with the objectives of this investigation.

Diverging slightly from the aforementioned approach, users acquire the file hash value by verifying the blockchain ledger, subsequently transmitting it to the interplanetary file system. Upon receipt and verification of the hash value as valid, the system furnishes the model file to the user. It is imperative to highlight that files transmitted between nodes necessitate encryption and decryption procedures before plaintext visibility can be achieved.

To ensure impartial comparison, three sets of experiments—comprising FL, BFL, and the enhanced framework delineated in this section—are performed under an equivalent number of iterations. To uphold data integrity and consistency, the experiments are conducted on virtual machines within a standardized environment, thereby mitigating the impacts of system or hardware discrepancies to the greatest extent feasible.

As shown in Fig. 8, the experimental results reveal that BFL exhibits a consistently prolonged execution duration, illustrating an augmented ratio in computation time vis-à-vis iterations. Furthermore, the discernible divergence between BFL and the comparator groups accentuates progressively over time. The juxtaposition underscores that the mere amalgamation of blockchain technology with FL fails to substantially alter the precision rate, while inducing constraints on computation and storage capacities. The integration of block chain inevitably imposes a substantial temporal overhead. Remarkably, the enhanced MLchain framework boasts a performance superiority of 10% over the conventional BFL framework. Intriguingly, the efficacy of the MLchain framework manifests more prominently with escalating iteration counts. Consequently, a strategic trade-off surfaces, wherein computational resources are leveraged to fortify resilience against cyber assaults. The decision-making process necessitates a case-by-case evaluation of the situational exigencies.

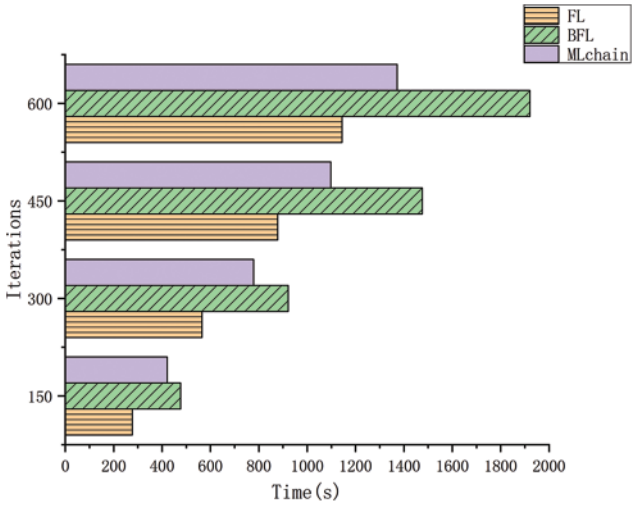

Figure 8: Comparison of convergence times for FL, BFL and final MLchain

The NSL-KDD dataset, renowned for its canonical status in intrusion detection research, is characterized by swift convergence and universal applicability across most algorithms. Consequently, a comparative analysis was conducted encompassing traditional Federated Learning protocols alongside the experimental configurations. The outcomes of the four algorithmic models are summarized in Table 4 for comprehensive review and examination.

Table 4: Comparison of experimental results for each model

Model	Accuracy	Precision	Recall	F1
SVM	99.34%	99.62%	98.81%	99.2%
FL	98.33%	98.75%	98.23%	98.5%
BFL	98.79%	98.53%	99.04%	98.8%
MLchain	99.34%	98.87%	98.77%	99.1%

The outcomes of the four sets of models demonstrate a high degree of similarity. However, the blockchain-driven federated learning framework not only upholds recognition accuracy but also safeguards user privacy, thus bearing significant relevance for practical implementations. Furthermore, the integration of blockchain technology plays a pivotal role in averting model collapse stemming from a singular device malfunction.

Conclusively, within the conventional setting, there exists scant variance in the ultimate precision levels across the four experimental sets. The incorporation of blockchain technology modestly impacts the pace of convergence while increasingly spotlighting transmission efficiency during intermediate and advanced iterations commensurate with growing data magnitudes.

6 Conclusion

This study proposes a lightweight federated learning chain framework with node management support, enhancing model accuracy through node management components and proof-of-work mechanisms. The study explores the continuous improvement of model accuracy and device utilization efficiency via an incentive mechanism and rational node management strategies. Robustness and accuracy assessments are conducted under diverse attack scenarios. To streamline the blockchain and alleviate block congestion, the InterPlanetary File System is incorporated to manage excessive blocks.

Compared with the traditional BFL model, the MLchain scheme in this study exhibits a minimum 10% enhancement, with further gains anticipated as redundant data accumulates. Empirical analyses affirm this finding. This research contributes a pioneering approach in applying blockchain technology and federated learning, particularly beneficial for applications necessitating federated learning like smart city edge security, vehicular network diagnostics, and IoT information classification and traceability. This proposed scheme not only bolsters federated learning reliability but also optimizes blockchain throughput, bearing practical and widespread utility.

However, the final scheme's performance is contingent on specific conditions and carries various limitations:

Lack of formulaic theoretical proof. Federated learning has always been a very controversial algorithm, which is embodied in the highly heterogeneous environment or the data is not independent and equally distributed.

Lack of local model assessments. In order to guarantee the precision of the model and avoid suboptimal training outcomes attributable to premature overfitting, it is imperative to ascribe distinct weighting factors to various parameters when computing the evaluation metric.

Application scope tailored to dataset availability, imposing constraints on generalizability beyond the intrusion detection system supported by the NSL-KDD dataset.

Inability to express the heterogeneity that may exist in real systems. This impediment arises from the constrained access to experimental apparatus. Should the computational velocity of one subset of users be doubled in comparison to the other subset, each outcome necessitates repetitive and independent assessment to warrant the reliability and stability of the experimental findings.

Acknowledgement: The authors gratefully acknowledge the invaluable guidance provided by the esteemed doctoral team at Xi'an Jiaotong University and the constructive feedback from anonymous reviewers. Additionally, the authors express deep gratitude to their families and significant others for their unwavering support throughout the completion of this article.

Funding Statement: This research was financially supported by the Project of National Natural Science Foundation of China under the grant titled "Research on Intermittent Fault Diagnosis of New Interconnection Networks under Comparative Model" (Approval Number: 61862003).

Author Contributions: H. Jiang conceptualized and designed the study. C. Li was responsible for data collection, analysis, interpretation of results, and drafting the manuscript. All authors critically reviewed the outcomes and authorized the final version of the manuscript.

Availability of Data and Materials: All data included in this study are available from recognized dataset websites and can be obtained by contacting the authors if required.

Conflicts of Interest: The authors declare no conflicts of interest pertaining to the research conducted and presented in this study.

References

[1] A. Zwitter and O. J. Gstrein, "Big data, privacy and COVID-19–Learning from humanitarian expertise in data protection," *Int. J. Humanitarian Act.*, vol. 5, no. 1, pp. 4, 2020. doi: 10.1186/s41018-020-00072-6.

[2] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 2, pp. 1342–1397, 2021. doi: 10.1109/COMST.2021.3058573.

[3] R. Sun, C. Li, W. Wang, E. Tong, J. Wang and J. Liu, "Research progress of blockchain-based federated learning," *J. Comput. Appl.*, vol. 42, no. 11, pp. 3413–3420, 2022.

[4] L. Li, Y. Fan, M. Tse, and K. Y. Lin, "A review of applications in federated learning," *Comput. Ind. Eng.*, vol. 149, no. 5, pp. 106854, 2020. doi: 10.1016/j.cie.2020.106854.

[5] Q. Yang, "AI and data privacy protection: The federal learning crackdown," *J. Inf. Secur. Res.*, vol. 5, no. 11, pp. 961–965, 2019.

[6] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. 2017 ACM SIGSAC Conf. Comput. Commun. Secur.*, Dallas, TX, USA, Oct. 2017, pp. 603–618. doi: 10.1145/3133956.3134012.

[7] E. Bagdasaryan, A. Veit, Y. Q. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2020, pp. 2938–2948.

[8] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020. doi: 10.1109/MSP.2020.2975749.

[9] W. Wang, "The application prospect of blockchain smart contract in book community," in *Proc. the Sixth National Docum. Catalog. Work Sem.*, Fuzhou, China, 2019, pp. 340–348.
[10] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Serv.*, vol. 14, no. 4, pp. 352–375, 2018. doi: 10.1504/IJWGS.2018.095647.
[11] Bitinfocharts, Blockchain size, 2022. Accessed: May 28, 2023. [Online]. Available: https://bitinfocharts.com/
[12] X. Qing and D. Fan, "Survey on lightweight blockchain technology," *J. Softw.*, vol. 34, no. 1, pp. 33–49, 2023. doi: 10.13328/j.cnki.jos.006421.
[13] S. King and S. Nadal, "PPCoin: Peer-to-peer crypto-currency with proof-of-stake," *J. Self-Publ. Paper*, vol. 19, no. 1, 2012.
[14] Y. Liu, K. Wang, Y. Lin, and W. Xu, "LightChain: A lightweight blockchain system for industrial internet of things," *IEEE Trans. Ind. Inform.*, vol. 15, no. 6, pp. 3571–3581, Jun. 2019. doi: 10.1109/TII.2019.2904049.
[15] F. M. Benčić and I. Podnar Žarko, "Distributed ledger technology: Blockchain compared to directed acyclic graph," in *2018 IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Vienna, Austria, 2018, pp. 1569–1570. doi: 10.1109/ICDCS.2018.00171.
[16] Raghav, N. Andola, S. Venkatesan, and S. Verma, "PoEWAL: A lightweight consensus mechanism for blockchain in IoT," *Pervasive Mob. Comput.*, vol. 69, no. 9, pp. 101291, 2020. doi: 10.1016/j.pmcj.2020.101291.
[17] J. Huang, L. Kong, G. Chen, M. Y. Wu, X. Liu and P. Zeng, "Towards secure industrial IoT: Blockchain system with credit-based consensus mechanism," *IEEE Trans. Ind. Inform.*, vol. 15, no. 6, pp. 3680–3689, June 2019. doi: 10.1109/TII.2019.2903342.
[18] S. Biswas, K. Sharif, F. Li, S. Maharjan, S. P. Mohanty and Y. Wang, "PoBT: A lightweight consensus algorithm for scalable IoT business blockchain," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2343–2355, Mar. 2020. doi: 10.1109/JIOT.2019.2958077.
[19] C. Xu, K. Wang, G. Xu, P. Li, S. Guo and J. Luo, "Making big data open in collaborative edges: A blockchain-based framework with reduced resource requirements," in *2018 IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, 2018, pp. 1–6. doi: 10.1109/ICC.2018.8422561.
[20] C. Ehmke, F. Wessling, and C. M. Friedrich, "Proof-of-property–A lightweight and scalable blockchain protocol," in *2018 IEEE/ACM 1st Int. Work. Emerg. Trends Softw. Eng. Block. (WETSEB)*, Gothenburg, Sweden, 2018, pp. 48–51.
[21] T. Kim, J. Noh, and S. Cho, "SCC: Storage compression consensus for blockchain in lightweight IoT network," in *2019 IEEE Int. Conf. Consum. Electron. (ICCE)*, Las Vegas, NV, USA, 2019, pp. 1–4. doi: 10.1109/ICCE.2019.8662032.
[22] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang and R. P. Liu, "Survey: Sharding in blockchains," *IEEE Access*, 2020, vol. 8, pp. 14155–14181. doi: 10.1109/ACCESS.2020.2965147.
[23] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh and D. Bacon, "Federated learning: Strategies for improving communication efficiency," arXiv preprint arXiv:1610.05492, 2016.
[24] M. H. Brendan, M. Eider, R. Daniel, H. Seth, and A. A. Blaise, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int Conf. Artif. Intell. Stati.*, New York, PMLR, USA, 2017, vol. 54, pp. 1273–1282. https://arxiv.org/abs/1602.05629.
[25] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Nashville, TN, USA, 2021, pp. 10708–10717. doi: 10.1109/CVPR46437.2021.01057.
[26] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough and V. Saligrama, "Federated learning based on dynamic regularization," in *Proc. 9th Int Conf. Learning Rep.*, 2021. Accessed: May 28, 2023. [Online]. Available: http://export.arxiv.org/abs/2111.04263.
[27] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, Indio, CA, USA, Systems and Machine Learning Foundation, 2020, pp. 429–450.

[28] R. O. Ogundokun, S. Misra, R. Maskeliunas, and R. Damasevicius, "A review on federated learning and machine learning approaches: Categorization, application areas, and blockchain technology," *Information*, vol. 13, no. 5, pp. 263, 2022. doi: 10.3390/info13050263.

[29] D. Hou, J. Zhang, K. L. Man, J. Ma, and Z. Peng, "A systematic literature review of blockchain-based federated learning: architectures," in *2021 2nd Inform. Commun. Technol. Conf. (ICTC)*, Nanjing, China, 2021, pp. 302–307. doi: 10.1109/ICTC51749.2021.9441499.

[30] U. Majeed and C. S. Hong, "FLchain: Federated learning via MEC-enabled blockchain network," in *20th Asia-Pacific Netw. Operat. Manag. Symp. (APNOMS)*, Matsue, Japan, 2019, pp. 1–4. doi: 10.23919/APNOMS.2019.8892848.

[31] H. Kim, J. Park, M. Bennis, and S. L. Kim, "Blockchained on-device federated learning," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1279–1283, Jun. 2020. doi: 10.1109/LCOMM.2019.2921755.

[32] W. Wang *et al.*, "Privacy protection federated learning system based on blockchain and edge computing in mobile crowdsourcing," *Comput. Netw.*, vol. 215, no. 4, pp. 109206, 2022. doi: 10.1016/j.comnet.2022.109206.

[33] S. Wang, J. F. Balarezo, S. Kandeepan, A. Al-Hourani, K. G. Chavez and B. Rubinstein, "Machine learning in network anomaly detection: A survey," *IEEE Access*, vol. 9, pp. 152379–152396, 2021. doi: 10.1109/ACCESS.2021.3126834.

[34] F. E. Ayo, S. O. Folorunso, A. A. Abayomi-Alli, A. O. Adekunle, and J. B. Awotunde, "Network intrusion detection based on deep learning model optimized with rule-based hybrid feature selection," *Inform. Secur. J: A Global Perspect.*, vol. 29, no. 6, pp. 267–283, 2020. doi: 10.1080/19393555.2020.1767240.

[35] M. V. Mahoney and P. K. Chan, "Learning nonstationary models of normal network traffic for detecting novel attacks," in *Proc. Eighth ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD '02)*, New York, NY, USA, July 2002, pp. 376–385. doi: 10.1145/775047.775102.

[36] R. M. S. Priya *et al.*, "An effective feature engineering for DNN using hybrid PCA-GWO for intrusion detection in IoMT architecture," *Comput. Commun.*, vol. 160, no. 6, pp. 139–149, 2020. doi: 10.1016/j.comcom.2020.05.048.

[37] N. Gupta, V. Jindal, and P. Bedi, "LIO-IDS: Handling class imbalance using LSTM and improved one-vs-one technique in intrusion detection system," *Comput. Netw.*, vol. 192, no. 4, pp. 108076, 2021. doi: 10.1016/j.comnet.2021.108076.

[38] A. Rosay, F. Carlier, and P. Leroux, "MLP4NIDS: An efficient MLP-based network intrusion detection for CICIDS2017 dataset," in *Proc. Int. Conf. Mach. Learn. Netw.*, Paris, France, Dec. 3–5, 2019, vol. 12081, pp. 240–254. doi: 10.1007/978-3-030-45778-5_16.

[39] Z. Tang, H. Hu, and C. Xu, "A federated learning method for network intrusion detection," *Concurr. Comput.*, vol. 24, no. 10, pp. e6812, 2022. doi: 10.1002/cpe.6812.

[40] G. Folino and P. Sabatino, "Ensemble based collaborative and distributed intrusion detection systems: A survey," *J. Netw. Comput. Appl.*, vol. 66, no. 1, pp. 1–16, 2016. doi: 10.1016/j.jnca.2016.03.011.

[41] T. Ren, R. Jin, and Y. Luo, "Network intrusion detection algorithm that merges blockchain and federation learning," (in Chinese), *Netinf. Secur.*, vol. 21, no. 7, pp. 27–34, 2021. doi: 10.3969/j.issn.1671-1122.2021.07.004.

[42] C. Fung, J. Chris, M. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," arXiv preprint arXiv:1808. 04866, 2018.

[43] Zalando, "Fashion-MNIST," Accessed: Apr. 1, 2023. Available: https://github.com/zalandoresearch/fashion-mnist

[44] D. Dheeru and K. Taniskidou, "UCI machine learning repository," 2017. Accessed: Apr. 1, 2023. [Online]. Available: https://archive.ics.uci.edu/

[45] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *Proc. the 38th Int. Conf. Mach. Learn.*, Jul. 18–24, 2021, vol. 139, pp. 2089–2099.

[46] Y. Fraboni, R. Vidal, L. Kameni, and M. Lorenzi, "A general theory for federated optimization with asynchronous and heterogeneous clients updates," arXiv preprint arXiv:2206.10189, 2022.