A Blockchain-Reinforced Federated Intrusion Detection Architecture for IIoT

Dingde Jiang, *Member, IEEE*, Zhihao Wang, *Student Member, IEEE*, Ye Wang, Lizhuang Tan, Jian Wang, *Senior Member, IEEE*, and Peiying Zhang, *Member, IEEE*

***Abstract*—Federated learning (FL) in Industrial IoT (IIoT) facilitates collaborative model training across distributed edge devices, ensuring data privacy and localized insights without centralized data aggregation. However, the networked parameter sharing mechanism in FL renders it vulnerable to exploitation by man-in-the-middle (MITM) attackers, potentially disrupting the model training process. To mitigate this threat, this article presents a novel blockchain-reinforced FL architecture aimed at enabling cooperative intrusion detection. Initially, FL is leveraged to aggregate all learned information from edge servers, thereby disseminating extracted attack characteristics to all participants through gradient sharing. Subsequently, a blockchain-based parameter verification scheme is introduced to safeguard against tampered local parameters affecting the global model. Clients record model parameters in smart contracts deployed on a private chain, and parameter servers verify parameter confidentiality before aggregation, ensuring only valid parameters are considered. Finally, extensive experiments are conducted using an edge IIoT cybersecurity data set comprising 61 features spanning ten protocol layers and five attacks targeting IIoT connectivity protocols. Simulation results demonstrate that the proposed scheme significantly enhances intrusion detection accuracy, achieving a threefold improvement when two-thirds of federated nodes are subjected to MITM attacks.**

***Index Terms*—Blockchain, federated learning (FL), Industrial IoT (IIoT), IIoT security, smart contract.**

Manuscript received 3 December 2023; revised 4 March 2024 and 8 April 2024; accepted 26 May 2024. Date of publication 12 June 2024; date of current version 8 August 2024. This work was supported in part by the Fund Projects under Grant 2020-JCJQ-ZD-016-11 and Grant 315197107; in part by the Open Fund of Digital Media Art, Key Laboratory of Sichuan Province under Grant 20DMAKL01; in part by the National Natural Science Foundation of China under Grant 62173345; in part by the Natural Science Foundation of Shandong Province under Grant ZR2023LZH017, Grant ZR2022LZH015, and Grant 2023QF025; in part by the Open Foundation of Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Qilu University of Technology (Shandong Academy of Sciences) under Grant 2023ZD010; in part by the Integrated Innovation of Science, Education and Industry of Qilu University of Technology under Grant 2023PX057; and in part by the Talent Project of Qilu University of Technology under Grant 2023RCKY141. *(Corresponding authors: Dingde Jiang; Lizhuang Tan).*

Dingde Jiang is with the School of Information and Communication Engineering and the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Chengdu 611731, China, also with the Laboratory of Electromagnetic Space Cognition and Intelligent Control, Beijing 100091, China, and also with the Digital Media Art, Key Laboratory of Sichuan Province, Sichuan Conservatory of Music, Chengdu 610041, China (e-mail: jiangdd@uestc.edu.cn).

Zhihao Wang is with the University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: wangzhihao1998@foxmail.com).

Ye Wang is with the School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou 450001, China (e-mail: iswangye@gs.zzu.edu.cn).

Lizhuang Tan is with the Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250013, China, and also with the Shandong Provincial Key Laboratory of Computer Networks, Shandong Fundamental Research Center for Computer Science, Jinan 250013, China (e-mail: tanlzh@sdas.org).

Jian Wang is with the College of Science, China University of Petroleum (East China), Qingdao 266580, China (e-mail: wangjiannl@upc.edu.cn).

Peiying Zhang is with the Qingdao Institute of Software, College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China, and also with the Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China (e-mail: zhangpeiying@upc.edu.cn).

Digital Object Identifier 10.1109/JIOT.2024.3406602

I. Introduction

INDUSTRIAL IoT (IIoT) represents the integration of IoT technologies within industrial and manufacturing domains, serving as a cornerstone of smart manufacturing and Industry 4.0 initiatives. Enhanced by artificial intelligence (AI), virtual reality, cloud computing, edge/fog computing, data twin technologies, IIoT offers effective, intelligent, and convenient solutions for traditional manufacturing and industry [1], [2]. Equipped with network capabilities, IIoT facilitates remote and automated connectivity for a vast array of traditional manufacturing devices. IIoT establishes a network communication channel for terminal industrial equipment, namely, sensors, robots, logistics vehicles, machine tools, etc., and high-level control devices, including automatic controller, data analyzer, information systems, etc. A key distinguishing feature is its robust data collection and analysis capabilities, enabling refined quality control, fault localization, and overall process enhancement. Besides, industry big data also can be utilized as the training sample for AI models, for further data-driven advanced applications, such as smart logistics and intelligent manufacturing. However, the proliferation of heterogeneous devices and network connections introduces significant security risks to IIoT systems, including network intrusion, data pollution, malware threats, and vulnerability exploitation [3], [4]. Therefore, the implementation of protective measures becomes significant to safeguard edge servers, mitigate privacy data leakage, and uphold manufacturing safety.

The continuously generating data and information from IIoT have facilitated abundant intelligent applications, which in turn promote the development of IIoT. The sensors or terminals generate sensitive information for transmitting, sharing and storing in IIoT, such as the product characteristics and manufacturing approaches. However, without assured data privacy, there exists a risk of severe infringement upon

2327-4662 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Green University of Bangladesh. Downloaded on January 23,2025 at 14:02:44 UTC from IEEE Xplore. Restrictions apply.

production security and intellectual property rights. Nonetheless, concerns regarding security, privacy, and regulatory compliance have constrained data sharing and value exchange. Driven by this demand, the federated learning (FL) technology emerges as the times require, which can realize deepening integration of data from different holders without transferring data to other places [5]. FL extracts and enlarges data value at the basis of protecting data security and user privacy. All participants train and maintain a local model and regularly share model parameters and synchronize with the global model. Local models process data locally and train on their respective local data sets using their own computing resources. Subsequently, model updates, typically in the form of gradients or parameters, are generated by local models. These updates are aggregated to form a new global model, constituting multiple rounds of training in FL. By preserving raw data on local devices and solely sharing model updates, FL effectively preserves user privacy. Local models ensure that sensitive data never leaves the device, addressing privacy concerns associated with centralized approaches. In this way, each local model will gain information from other participants after several federated rounds. FL has been applied in extensive scenarios, including data management [6], DDoS mitigation [7], adversarial attack defense [8], etc.

However, FL encounters inherent limitations hindering its widespread adoption in IIoT, such as privacy vulnerabilities, communication overhead, client heterogeneity, and unverified model uploads. In FL-based IIoT intrusion detection scenarios, the FL model becomes susceptible to contamination and tampering attacks. Interactions between FL clients and servers propagate across the network, exposing exploitable vulnerabilities to man-in-the-middle (MITM) attackers. An attacker can transmit intentionally tampered model parameters to the server by spoofing the client. After aggregating and distributing parameters, an attacker can propagate injected information to nodes throughout the network. These attacks may involve injecting entirely meaningless model parameters, undermining the model's training performance and detection efficacy, or could manifest as insidious manipulations, evading detection of specific attacks and creating additional attack vectors within the system. Fortunately, these security-related drawbacks of FL can be alleviated by blockchain [9], [10], and [11]. There are several distinct characteristics of blockchain, including decentralization, transaction transparency, tamper-proof ability, and multiparty consensus [12], [13]. A pivotal mechanism within blockchain is the smart contract, which digitizes traditional contracts and agreements, imbuing them with programmable logic. By loading smart contract and programming logic into blockchain, users can bring custom execution logic to services rather than only transaction. Besides, the smart contract will be distributed to every node with block synchronization, which makes all nodes follow the rules and execute the instructions [14]. FL and blockchain can complement each other in some aspects, realizing multiparty cooperative trusted network through combination. The transaction and accounting mechanism blockchain can be exploited to address the data value representation and user incentives. Furthermore, blockchain's tamper-proof nature offers a means to trace and penalize malicious actors within federated participants by storing data fingerprints in blocks.

In this article, to protect the FL parameters from being easily tampered with by intermediaries during the transmission process, we propose a blockchain-reinforced FL architecture for IIoT intrusion detection scenario. Within the edge-cloud IIoT framework, distinct edge servers collect and extract features from both normal and intrusion traffic. Each edge server hosts a federated client responsible for local deep learning model training, facilitating intelligent intrusion detection. Concurrently, a cloud server, serving as the federated parameter server (PS), consolidates parameters received from all clients to synthesize global attack features, disseminating them across all edge servers. Besides, to further ensure the security of parameter transmission, inspired by the tamper-proof mechanism, we introduce blockchain into overall architecture, connecting all cloud and edge servers by blockchain. A smart contract that can record the information abstract of the transferred parameter at the edge side. After client training and updating model, the parameter of neural network will be sent to PS and recorded into blockchain simultaneously. The PS then verifies parameters. The consistent ones are aggregated, while inconsistent ones are discarded, which mitigates message content modification and malicious node access attacks. Finally, the simulation results validate the practicality and efficacy of the proposed architecture in cooperative intrusion detection. Our contribution can be summarized as follows.

1) We introduce a blockchain-reinforced FL architecture to realize cooperative intrusion detection in IIoT, leveraging the consensus mechanism to enhance the security of the federated parameter transmission.
2) We propose a parameter aggregation algorithm and a parameter verification algorithm with smart contract, enabling secure aggregation and decentralized verification of federated parameters.
3) We validate the effectiveness, security, and overhead of the proposed algorithm based on an IIoT intrusion detection data set. Through the construction of three distinct scenarios, we evaluate intrusion detection performance relative to centralized training. We also analyze the complexity and security theoretically. The simulation results show that the proposed architecture can effectively mitigates MITM attacks, ensuring the integrity of the training process.

The main content of this article will be organized as follows. Basic concept and application of FL and blockchain are introduced in Section I. Related works are illustrated in Section II. The proposed blockchain-reinforced FL architecture for cooperative intrusion detection in IIoT is detailly illustrated in Section III. Section IV elaborated on the methodology in the proposed architecture. Simulation and results are shown in Section V. Conclusion of the whole work and future direction are given in Section VI.

II. Related Work

A. Federated Learning for IIoT

As a security-oriented big data machine learning technology, FL has extensive application scenarios, not constrained by specific areas or algorithms. Taheri et al. [15] applied FL to detect malware applications in IIoT, along with a generative

Authorized licensed use limited to: Green University of Bangladesh. Downloaded on January 23,2025 at 14:02:44 UTC from IEEE Xplore. Restrictions apply.

adversarial network to achieve a robust collaboration federated training model. FL can aggregate information from different clients or sources, which makes it an effective approach to realizing coordinated defense. Song et al. [8] proposed a framework to aggregate defense knowledge against adversarial examples from different IIoT sources. With FL, the defense capabilities against different attacks can be fused. Li et al. [7] utilized FL in DDoS mitigation and combine it with edge computing. To train a global optimized model, data sets from different distributed IIoT sources are collected without data and communication constraints. In IIoT system, there are always massive heterogeneous devices to provide different services or perform manufacturing tasks. In traditional FL architectures, affected by heterogeneous federated clients, the model aggregation speed is always restricted by the client with the worst performance. More clients or participants means higher communication latency or coverage speed. To address the problem and improve the training efficiency of overall model, Zhang et al. [6] applied deep reinforcement learning algorithm to the federated node selection process to improve the data privacy and training efficiency of the IIoT devices. The above traditional FL schemes or methods rarely consider the security of the parameter transmission process. The parameters of FL are vulnerable to MITM attacks, such as interception and tampering, when they are passed between the client and the server.

B. Blockchain for IIoT

As an emerging technology, blockchain technology can promote the security of IIoT infrastructure [16]. One of the most common applications is using the decentralization and encryption capabilities of blockchain to enhance the security of IIoT [17]. Sodhro et al. [18] employed blockchain to randomly manage keys in IIoT system, which sacrifices little communication overhead and computation bits but brings more efficiency and reliability. Blockchain technology can also be employed to secure the wireless communication of IIoT terminals [19]. The blockchain is especially helpful in some applications, such as cooperative distributed unmanned-aerial-vehicular networks [20]. With the tamper-proof nature and the distributed consensus mechanism, blockchain is also extensively exploited in privacy guarantees [21]. Tian et al. [22] exploited the smart contract of blockchain to encourage multiparty participation in machine learning.

As one of the core technologies, smart contract has the characteristics of decentralization, de-trusting, programmable and tamper-proof, making it more suitable and flexible to embed various data and properties [23]. Smart contract can realize secure and efficient data exchanging, value migration, and property management [13]. Many functionalities can be implemented on blockchain using smart contract, including intrusion detection [24], access control [25], etc. One of the most important restrictions of IIoT is the constrained resource, which means complicated and computationally intensive algorithms are not suitable to deploy on IIoT devices. Narayanan et al. [26] proposed a blockchain-based scheme for lightweight IoT device authentication and data encryption.

C. Blockchain-Based Federated Learning

FL aims to address the trusted data computing and data privacy problem, while blockchain aims to address the trusted data sharing and storage problem, which complement each other. Blockchain can be used to solve the certificate deposit problem of FL. Endeavors have been made in the application of blockchain-based FL in IIoT. The communication channels of FL are commonly unreliable, and the participations are untrustworthy, which hinders the extensive and effective application in IIoT. Therefore, many researchers focus on introducing blockchain into FL to enhance reliability and security. Zhang et al. [27] introduced a blockchain-based FL architecture for failure detection of IIoT devices, where the integrity and credibility of collected data from devices can be verified. A blockchain-based cross-device FL system is proposed in [28], which can achieve model poisoning attacks detection, fair federated training, and IIoT device reputation maintenance. Due to the limited resources and massive amounts of the terminal devices of IIoT, it is not suitable or effective to deploy FL or blockchain on them. Many researchers consider deploying the blockchain-based FL architecture on the edge server to improve efficiency [29].

One of the most effective application scenarios of blockchain-based FL in IIoT is data sharing or exchanging because FL is a data-driven approach, where data privacy or data value estimation is important. Some blockchain-enabled applications can be deployed, such as edge intelligence in space--air--ground integrated networks [30]. Lu et al. [31] proposed a blockchain-enabled secure data sharing architecture for distributed federated parties, and address the data sharing by transforming it into a machine-learning problem. Based on differential privacy and homomorphic encryption technology, Jia et al. [32] proposed multiple data protection approaches using distributed K-means clustering, distributed random forest and distributed AdaBoost algorithms, which are integrated into the built blockchain-based FL architecture. To realize device privacy protection without transferring data to other devices, Xu et al. [33] proposed a personalized blockchain reliability prediction model integrated with FL, to help users locate relatively reliable peers. Malomo et al. [34] took advantage of the security characteristics of blockchain to secure the storage for offsite digital assets along with FL. Ho Park et al. [35] introduced a proof of authentication (PoAh) approach into FL architecture to detect the DDoS attack in IoT networks. The PoAh mechanism of blockchain is utilized for data authentication and validation.

We summarize and compare some existing works as Table I. Rather than using blockchain to protect the security of FL training process, some works that integrate the blockchain and FL try to authenticate the data generated during training or motivate the clients to participate in the training [27], [31], [35]. In this case, if the training process of FL is disturbed or tampered with, the recorded data will be untrustworthy. Therefore, schemes using blockchain to secure the FL training process are proposed [28], [29], [32], [36] to verify the credibility or reputation of the clients, or to store the training parameters into blockchain. However,

Authorized licensed use limited to: Green University of Bangladesh. Downloaded on January 23,2025 at 14:02:44 UTC from IEEE Xplore. Restrictions apply.

TABLE I
SUMMARY OF RELATED WORKS

Ref.	Methodology	Securing FL with blockchain	Remarks
Zhang *et al.* [27]	Blockchain-based federated learning systems for failure detection in IIoT.	×	Blockchain is introduced for verifiable integrity of client data. A smart contract acts as an incentive mechanism to motivate clients with different data sizes and heterogeneity to participate in FL training.
Rehman *et al.* [28]	Fair and Trustworthy Cross-Device Federated Learning in IIoT.	√	Blockchain smart contract is used to maintain the device reputation of participants by compelling them to make active and honest model contributions.
Zhang *et al.* [29]	Defensive Transmission Model Based on Blockchain-Assisted Reinforced Federated Learning in IIoT.	√	The model parameters of FL are directly uploaded to the blockchain, to ensure the non-tamperability and traceability.
Lu *et al.* [31]	Blockchain-based privacy-preserved data sharing of FL.	×	The data sharing events between requesters and provides are recorded in blockchain.
Jia *et al.* [32]	Blockchain-enabled FL data protection with differential privacy and the homomorphic encryption.	√	The security protection methods are directly performed on the data from distributed nodes.
Park. *et al.* [35]	A PoAh-based FL for DDoS attack detection in IIoT.	×	The blockchain is used for authenticating the data after training without securing the FL training process. The data generated by FL may be tampered by malicious users without protection.
Kim *et al.* [36]	Blockchained On-Device Federated Learning	√	FL is deployed with on-device machine learning. Local models trained by devices or clients are uploaded and verified by blockchain without central server, which brings heavy loads.
Proposed Architecture	Blockchain-Reinforced Federated Learning Architecture for Intrusion Detection	√	Federated learning process is secured by only recording the information abstract of the local parameters into blockchain rather than the whole parameters. The server can easily verify the confidentiality of the parameters.

storing the raw parameters in the blockchain is an expensive operation, considering the parameter number of deep learning models is quite large. Therefore, we propose to only store the information abstract of the parameter of each client in the blockchain, enabling the server to quickly verify with minimal overhead whether parameters from the client have been tampered with.

III. PROPOSED SYSTEM ARCHITECTURE

A. FL-Based Intrusion Detection

In this article, the FL part follows the common PS structure [27], where the model aggregation is centralized. The application scenario of FL, namely, cooperative intrusion detection in IIoT, is horizontal FL, which is determined by the distribution characteristics of the data sets. We consider an intrusion detection scenario based on the cloud-edge collaboration [37], where edge servers act as the federated clients and the cloud server act as the parameter sever. We employ the edge servers in IIoT as the federated clients, as they capture and possess local attack and normal traffic. A centralized cloud server is utilized as the PS to aggregate the model. Each client in this structure holds its own data set and cannot access other's data. The collected user data will not be transferred through network, for protecting privacy and reducing network overhead. The neural network model of PS and clients are the same. Meanwhile, information from all edge servers will be aggregated through the PS. The features of network traffic from different edge servers have many overlaps, while the samples from different edge servers have fewer overlaps. Therefore, the network traffic data sets of clients have the same features but different traffic types. The Notations involved in the model are shown in Table II. The horizontal FL employed in this article is expressed as follows [38]:

TABLE II
SUMMARY OF NOTATIONS

Notation	Description
η	the learning rate of local models
χ	the smart contract deployed on the private chain
ϖ_i	the local blockchain wallet account of client i
b	the batch of samples
g	the gradients of local models
m_r^i	the MD5 hash value of client i at round r
C	the set of federated clients
D_i	the dataset held by client i
I_i	the data samples of client i
W_r	the weights of global model in round r
W_r^c	the weights of local model of client
X_i	the data features of client i
Y_i	the data labels of client i
$\mathbb{W}$	the set of the verified weight from C

$$X_i = X_j, Y_i = Y_j, I_i \neq I_j \quad \forall D_i, D_j, i \neq j \tag{1}$$

where D_i represents data set held by client i, X_i is the feature of D_i, Y_i means the labels of data set from client i, and I_i means the data samples from client n.

At each round of cooperative training of clients, after obtaining current global parameters from PS, the federated clients train the model only locally, to protect the privacy of the private data set. Then the updated parameters or parameters will be transferred to PS, instead of the training data set. Receiving the parameters from clients, the PS aggregates the

Authorized licensed use limited to: Green University of Bangladesh. Downloaded on January 23,2025 at 14:02:44 UTC from IEEE Xplore. Restrictions apply.

parameters following a specific aggregation strategy, which is federated averaging [38] (FedAvg) in this article. Afterward, the aggregated parameters will be sent to each client again to start another round of training. There are many advantages of PS structure. The separated design makes it possible to modulate the commonly used machine learning components, which simplifies the development and integration. As an asynchronous task model, the overall communication overhead or occupied bandwidth can be effectively saved. Another significant advantage is the elastic expansion of resources, where new clients can participate without restarting all system or training. In the PS structure, several failures of clients will not cause the disastrous result to the overhead system, which means fault tolerance and fast recovery capability. In addition, the abbreviations used in this paper are shown in Table III. The FL procedure can be described as follows.

Step 1 (Connection Establishment): The PS firstly constructs the training or modeling task, searching for participating clients. The clients that hold the corresponding data set receive the training task and establish a connection to PS. When the preset minimum client amount is reached, the federated training process is started with a participating client set C.

Step 2 (Parameter Initialization): Before the first training round, each client will initialize the model weight locally, following different strategies. The PS will randomly select a client to fetch the model weight as the initial global model weight W_0, which will be distributed to each client in C.

Step 3 (Local Model Updating): Each client c in C holds a data set, which is related to the device input or edge server data collecting. In training round r, the global model weight W_r will be distributed to all clients in C. Client c exploits the current global model W_r and the locally hold data set to train the model and compute gradients ∇g. The local model will be updated as

$$W_{r+1}^{c} = W_r - \eta \nabla g(W_r; b) \tag{2}$$

where η is the learning rate of client.

Step 4 (Centralized Aggregation): After receiving all weights W_{r+1}^{c} from clients, the PS will aggregate all parameters following the averaging strategy, expressed as:

$$W_{r+1} = \sum_{c=1}^{|C|} \frac{1}{|C|} W_{r+1}^{c} \tag{3}$$

where $|C|$ represents the number of clients.

Step 5 (Global Model Updating and Performance Evaluation): After parameter aggregation, the PS will send W_{r+1} to each client. Besides, the PS will also check whether the max training round is reached or whether the accuracy meets the requirements using a test data set. If the condition is met, FL ends. Otherwise, the system will jump to step 3 and federated training will continue.

B. Consensus Mechanism

Smart contract is a piece of code running on blockchain, relying on the unanimous recognition of implementation among participants. Smart contract manages and changes the state value stored on blockchain through preagreed code. In the state transition model, the state transition of blockchain is realized by transaction, as same as smart contract. A data filed is defined in transaction function, to pass the function parameter to the contract. Hence, a contract address is also generated for a contract instance. It has balance and can become trading objects. However, it cannot be manipulated by humans. Instead, it is deployed on the network as running programs. Individual users can interact with the smart contract by submitting a transaction to execute a certain function of the smart contract. Smart contracts can define rules like regular contracts and automatically enforce them through code. Besides, to avoid endless loops in smart contract, the GAS mechanism is introduced to guarantee that the contract can be terminated within a limited time. Every execution will consume a certain amount of gas until the prepaid gas runs out.

TABLE III
ABBREVIATION TABLE

Abbreviation	Description
ABI	Application Binary Interface
AI	Artificial Intelligence
CNN	Convolutional Neural Network
DAG	Directed Acyclic Graph
DDoS	Distributed Denial-of-Service
DHT	Distributed Hash Table
ECDSA	Elliptic Curve Digital Signature Algorithm
ETH	Ether
FL	Federated Learning
IIoT	Industrial Internet of Things
IID	Independently Identical Distribution
gRPC	Google Remote Procedure Call
MD5	Message-Digest Algorithm 5
MITM	Man in the Middle
PS	Parameter Server
PoW	Proof of Work

Each operation of the deployed smart contract is initiated as a transaction, each of which should be signed. Digital signatures are primarily used to verify the authenticity of a transaction and prevent forgery. The transaction is then packaged into blocks and broadcast to the network. To ensure that a transaction is tamper-proof, a user must sign it using their private key. Upon receiving the transaction, other nodes use the public key to verify the signature's validity, thereby determining the transaction's authenticity and origin. The digital signature algorithm (DSA) used in this article is elliptic curve DSA (ECDSA), a variant of the DSA applied to elliptic curves (ECs). To sign a transaction with ECDSA, the public and private key should be generated. Given an EC $E_p(a, b)$ and a generator point G, where the prime p specifies the finite field size, and a, b are coefficients. G generates subgroup with order n and cofactor h. The above parameters form the domain parameters, represented as (p, a, b, G, n, h). The private key d_A is selected as a random integer in the range $[0, 1, \ldots, n-1]$. The public key Q_A is calculated as $d_A \cdot G$.

Authorized licensed use limited to: Green University of Bangladesh. Downloaded on January 23,2025 at 14:02:44 UTC from IEEE Xplore. Restrictions apply.

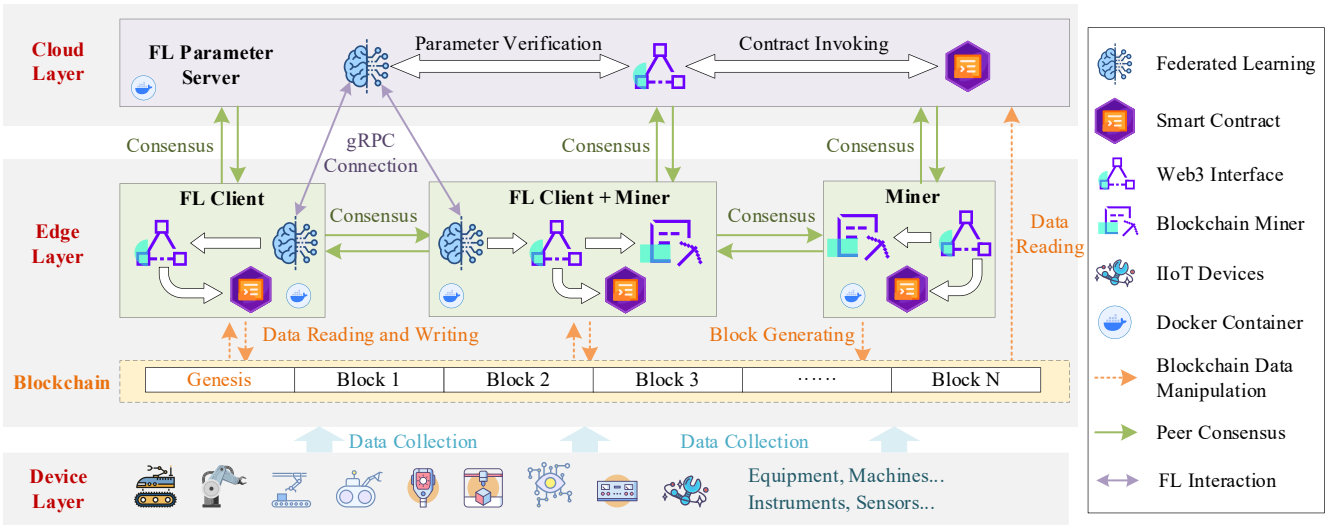

Fig. 1. Overall architecture of proposed system.

The signing of ECDSA takes the inputs of a message m and a private key d_A and outputs the signature (r, s). First, a random point P is calculated as $k \cdot G$, where $k < n$ is a random integer. Take the x-coordinate of P as r. Then calculate the hash of message as follows:

$$z = \text{hash}(m) \tag{4}$$

where the hash function could be SHA-2, SHA-256, etc. Calculate the signature proof s as

$$s = k^{-1}(z + rd_A)(\text{mod } n) \tag{5}$$

where $k^{-1}(\text{mod } n)$ represents the modular inverse. Then, the signature (r, s) is obtained, which confirming the signer knows the message z and the private key d_A. The private key d_A cannot be revealed from the signature (r, s) because of the difficulty of elliptic-curve discrete logarithm problem algorithm.

To verify an ECDSA signature, the message m, the signature (r, s), and the public key Q_A are taken as input. Similarly, the hash of m is calculated as (4). Then recover the random point P through

$$P' = \left(z \cdot s^{-1}(\text{mod } n)\right) \cdot G + \left(r \cdot s^{-1}(\text{mod } n)\right) \cdot Q_A. \tag{6}$$

Finally, the validity of the signature can be determined by comparing whether the x-coordinate of P' is as same as r.

The information of the local parameters is appended to the blockchain through smart contract, which is validated by all nodes. A consensus protocol is required to ensure all nodes agree the specific order in which the entities are appended [39]. In this article, we use Ethash algorithm of Ethereum as the Proof-of-Work (PoW) consensus mechanism, which is designed to make it application specific integrated circuit (ASIC) resistant and verifiability by lightweight clients. To realize PoW, Ethah firstly takes a seed, which is generated by the block headers up to the point where the mining begins. It basically depends on two data sets, namely, a small c pseudorandom ache with size of about 16M, and a big directed acyclic graph (DAG) data set with size of about 1G. The DAG is used as a fixed-size resource depending the nonce and block header. It is required to choose subsets of DAG to achieve memory hardness in Ethash. The larger DAG is calculated by the small cache. Miner node needs to maintain the two data sets for faster mining. But the lightweight node only needs to maintain the smaller cache for verification. Therefore, in this article, we design several types of nodes for different functional requirements and efficiency based on this principle, including the server and light client nodes only contain the small cache, and the miner node with DAG, etc., shown as Fig. 1.

C. Overall Architecture

In this article, we propose a blockchain-reinforced FL architecture for cooperative intrusion detection. The overall architecture is mainly composed of federated PS, federated clients, Ethereum Blockchain and IIoT Terminal Devices, depicted in Fig. 1.

Device Layer contains various IIoT terminal devices, including manipulators, robots, sensors, vehicles, meters, etc. Normally, the edge terminals are connected with the edge layer through wired network, such as fieldbus, Industrial Internet, or wireless network, such as LoRa, Wi-Fi, and Bluetooth. The data generated or collected by the devices is transmitted to the edge layer for further processing [40]. The data could be used to train AI models that can optimize production process. The devices receive the control instructions from the edge layer.

FL requires a deep learning runtime environment for training. Apart from the occupied storage space and memory, the training process also consumes a lot of computational power. Besides, the PoW mechanism of blockchain also requires huge computation resources, which are designed to protect the brute force attack. However, the resource-constrained IIoT devices cannot separate such a big computing power from their tasks. In addition, most devices directly connect to the edge controller rather than the edge servers, with the single network interface and non-IP protocol, which causes obstacles

Authorized licensed use limited to: Green University of Bangladesh. Downloaded on January 23,2025 at 14:02:44 UTC from IEEE Xplore. Restrictions apply.

for the network traffic analysis. Therefore, we exploit the edge servers as the federated clients and the cloud server as the federated PS.

Edge Layer is responsible for controlling the devices, receiving and processing the data from device layer. Edge Layer contains different edge servers, namely, Training Node, Training + Miner Node, and Miner Node, along with other edge servers not participating in FL. Training Node means there are only federated clients and Ethereum lightweight nodes. Because to conduct mining, the Ethereum node will first generate a DAG file, consuming about 1GB disk space. Without mining, Training Node only establishes a basic Geth environment to synchronize blocks and interact with smart contract. Training + Miner Node is fully functional, with federated client and Ethereum node, which can conduct federated training, blockchain mining, and interaction with smart contract. The resource consumption and space occupation of this node are the most. Miner Node means the node does not participate in the federated process, only mining to verify the transaction and generate blocks. Cloud Layer mainly consists of the cloud server, equipped with the federated PS and Ethereum lightweight node. The cloud server only needs to verify parameters from clients, without writing and mining. All servers linked to private blockchain follow the consensus mechanism, the Ethash. Only federated clients can write data into blockchain through invoking the recording method in smart contract. The PS only invokes the query method to verify parameters. In practice, the PS can also conduct mining to verify the parameters. The federated clients communicate with PS through a gRPC channel. Federated clients do not communicate and do not know each other's existence, which improves security and privacy.

The overall system operation workflow is depicted in Fig. 2. Procedures with a "−" indicate that there is no chronological correlation, which means they can be executed simultaneously or sequentially. For example, the steps (1-1) and (1-2) can be executed at the same time, or step (1-1) happens before or step (1-2) happens before. The PS contains FL Server and Web 3 Interface. Training Node contains FL Client and Web 3 Interface. Training and Miner Node contains FL Client, Web 3 Interface, and Miner.

1) Initialization Stage [Step (1-1)]: After gRPC connection is established, FL Server randomly selects a FL Client for initializing parameters.

Step (1-2): After the blockchain connection is established, a smart contract to realize interfaces for data writing and query is deployed through Web3 Interface. In this article, the smart contract is deployed by the PS that controls the FL process. The generated address on the blockchain is distributed to each client to interact with the contract. The deployment of a contract is also a transaction in the blockchain, which should be verified by PoW mechanism. The transaction is firstly queued in the txpool, namely, the transaction pool, which is composed by all the currently pending transactions and the transactions that are added to the queue for future processing.

Step (2-1): Receiving the parameter fetching request, the selected FL Client returns the initial parameters.

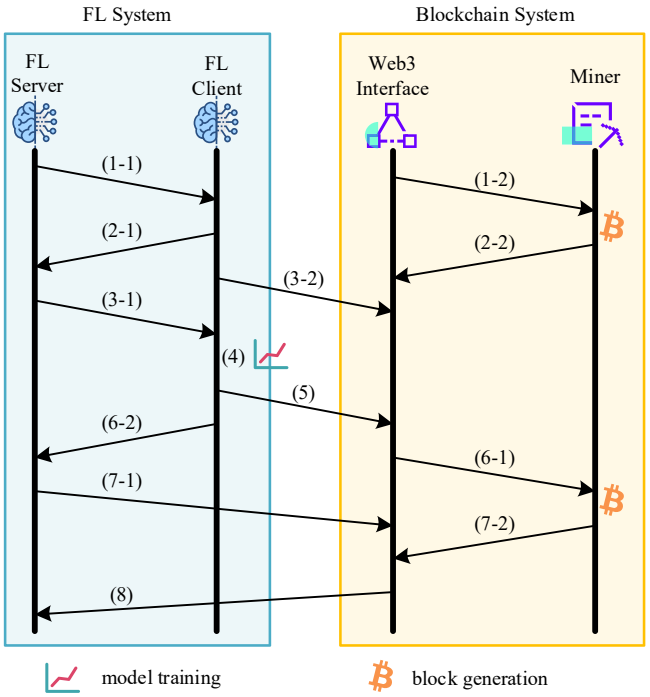

Fig. 2. Overall system operation process.

Step (2-2): Once noticing there is an unverified transaction in txpool, the miner node that preempts the bookkeeping right first starts mining for a new block. After a block is generated, the smart contract is deployed in blockchain with a globally unique address. After other nodes synchronized this block, the smart contract could be instantiated at all nodes.

2) FL Training Stage [Step (3-1)]: After receiving the initiation parameter, the first training round of FL is started. An instruction is distributed to all FL Clients to start local training.

Step (3-2): Before local update, the FL Client communicates with the blockchain through Web3 Interface, probing the status of blockchain and getting the node address to send transaction.

Step (4): Each FL Client starts several training epochs locally, in which the accuracy values of both local training and global test data set are evaluated. The training process is as same as normal deep learning models.

Step (5): After a certain number epochs of local training, the model parameter is generated and encapsulated in a gRPC message. Furthermore, the message as string type will be input to Message Digest Algorithm 5 (MD5), obtaining an information abstract with the size of 128 bits. Then this hash value is recorded to blockchain, through invoking a method in the pre-instantiated smart contract.

3) Parameter Verification Stage [Step (6-1)]: The message containing updated parameters in step (5) is sent to PS through gRPC channel.

Step (6-2): A state change of smart contract will be regarded as a transaction. Similar to step (1-2), the transaction is sent to txpool, waiting miners to get bookkeeping right.

Step (7-1): After receiving a gRPC message with parameter from federated client, the PS sends a verification request

Authorized licensed use limited to: Green University of Bangladesh. Downloaded on January 23,2025 at 14:02:44 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 Parameter Verification Algorithm

Function client_record (W_r^i, χ, ϖ_i):
  $m_r^i \leftarrow MD5(W_r^i)$
  $\chi \leftarrow \chi \cup \{< r, m_r^i >\}$
  ϖ_i starts a transaction t for state transition
  miners start to verify t
end function

Function server_verify (W_r^i, χ):
  $m_r^i \leftarrow MD5(W_r^i)$
  wait until key r is appended into χ
  if $\chi[r] = m_r^i$ **do**
    return True
  end if
  return False
end function

Function server_flush (χ, $\varpi_{|c|}$):
  for each record γ in χ **do**
    $\chi \leftarrow \chi - \{\gamma\}$
    ϖ_i starts a transaction t for state transition
  end for
  miners start to verify all t in txpool
end function

through Web3 Interface that queries the smart contract data stored in the blockchain.

Step (7-2): The transaction is verified by Miner and new block will be generated and broadcasted to all nodes.

4) Parameter Aggregation Stage [Step (8)]: After querying, once there is corresponding information in blockchain, the parameter will be utilized in parameter aggregation. Otherwise, if the timeout is exceeded and the information is still unattainable, or the Hash value is inconsistent, the parameter from that FL Client will be dropped. The parameter aggregation will be conducted only with the verified parameter. After aggregation, if the ending condition is met, FL training will be completed with the result in this round. Else, the FL training will continue and jump to step (3-1).

IV. Methodology

Usually, the message containing parameters is transported in plaintext between client and server, or secured through certain encryption and identity authentication approaches. It still can be intercepted and camouflaged to obtain key information or deceive the system. In this article, to realize the secure aggregation of federated parameter and decentralized verification, we propose a parameter aggregation algorithm based on FedAvg and a parameter verification algorithm based on blockchain.

A. Parameter Verification

The parameter verification algorithm is realized based on smart contract and Web3 interface, as illustrated in Algorithm 1. Assuming the client set is C, each of them holds a local wallet account ϖ_i to collect and pay Ether, where $i \in [0, |C|-1]$. There is one PS s, contract to a private chain, the contract address and the application binary interface(ABI) will be generated. With the Web3 interface, a deployed contract χ will be instantiated and invoked. According to the state transition model of blockchain, every operation changing the chain state is regarded as a transaction that consumes gas when conducting. A parameter recording operation and the record flushing operations are transactions, which should be called with "transact" function. And after transactions are sent, the miner is called to conduct PoW for validating the transaction and generating block. The proposed parameter verification algorithm provides three interfaces to interact with the federated applications, namely, client_record, server_flush, and server_verify. For client_record, we first calculate the MD5 value of W_r^i that represents the neural network weights of client C_i and training round r. The round number r and calculated MD5 value m_r^i compose a key-value pair, which is recorded in blockchain afterwards. Before each federated task, the history records are emptied by calling server_flush, in case of confusion. The server_verify function receives input of parameters to be verified, querying the corresponding record in smart contract to verify. If the transaction has not been verified, the algorithm will wait until verified and written. If the MD5 value is not consistent, a False value will be returned. Otherwise, a True value will be returned to inform the server the parameter is verified and secure.

B. Parameter Aggregation

Based on the FedAvg, we extend a secure parameter aggregation algorithm, illustrated in Algorithm 2. The proposed secure parameter aggregation algorithm is divided into two parts, server aggregation and client updating. For PS-side, the initial global parameter W_0^g is initialized from a random client in C. Clients conduct local training parallelly. For each participating client, local epoch number, local batch size, local learning rate, and local data set are initialized before local training. After a local training round, client i sends its local parameter W_r^i to the PS. After receiving W_r^i from client i, the PS invokes the server_verify to validate the parameter. If validated, W_r^i will be stored temporarily for global parameter aggregation. Otherwise, W_r^i will be dropped, preventing disturbing the global model.

V. Evaluation and Analysis

A. Evaluation Setup

To evaluate performance of the proposed architecture, we construct a simulating environment of FL and blockchain. We exploit the Edge-IIoTset [41], a realistic cyber security data set for IoT and IIoT intrusion detection. Essential data preprocessing steps are performed on the data set, namely, data cleaning, feature engineering based on domain knowledge, down and over sampling, label encoding, and min-max standardization. After preprocessing, there are 91 numerical features and one target column with five attack types, namely, DDoS, information gathering, injection, and malware. The features of Edge-IIoTset covers the Frame, IP, ARP, ICMP, HTTP,

Authorized licensed use limited to: Green University of Bangladesh. Downloaded on January 23,2025 at 14:02:44 UTC from IEEE Xplore. Restrictions apply.

Algorithm 2 Secure Parameter Aggregation Algorithm

initialize parameter W_0^g from a random client
for each federated round r **do**
  $\mathbb{W} \leftarrow \varnothing$
  for each client i in C **do parallelly**
    for each local training epoch **do**
      sample a batch from local dataset
      train locally and update W_{r-1}^i
    end for
    $W_r^i \leftarrow W_{r-1}^i$
    client_record (W_r^i, χ, ϖ_i)
    if server_verify(W_r^i, χ) **do**
      $\mathbb{W} \leftarrow \mathbb{W} \cup \{W_{r+1}^c\}$
    end if
  end for
  $W_r^g \leftarrow (\sum_{j=0}^{|\mathbb{W}|} \mathbb{W})/|\mathbb{W}|$
  evaluate the aggregated W_r^g
  for each client i in C **do**
    $W_r^i \leftarrow W_r^g$
  end for
end for

TCP/UDP, DNS, MQTT, Modbus/TCP layers, including flow five-tuple, HTTP request method, TCP segment length, etc.

To realize lightweight and efficient management of nodes, we exploit Docker container to establish the experiment environment. We employ Flower [42] to implement FL. The version of flwr is 1.6.0, running in Python 3.8.18. The backend of flwr is PyTorch 2.1.2 running on the PC with Intel Core i9-13900HX@2.2 GHz, 32-GB RAM, and NVIDIA GeForce RTX4060 GPU. We take Ethereum as the underlying private blockchain network. The difficulty of PoW is set to 0x20000. The smart contract is written and complied with Solidity 0.8.24. We utilize Python-Web3 library to interact with Ethereum [43]. There are several types of containers deployed, namely, the server containers equipped with federated server application and Web3 interface, the client containers equipped with federated client application and Ethereum application, and the miner containers equipped with only Ethereum application. There are two different types of clients, including the fully functional clients and the lightweight clients.

A multilayer CNN model is used as the local and global model, composed by three convolutional layers with max pooling, a fully connected layer with Dropout, and an output fully connected layer. The activation function is ReLU for input and hidden layers, and Softmax for output layer. Cross Entropy loss and Adam optimizer are used. Learning rate is set to 0.001, batch size is set to 256.

For comparison, we also let every client train standalone in the centralized manner with the same neural network model. We design three experiments with different conditions to evaluate the model performance. The first scenario employs three independently identical distribution (IID) data sets sampled from preprocessed Edge-IIoTset. The second scenario employs a resampled data set but with a client attacked. The label of data set from attacked client is tampered to simulate a MITM attack. Here, we assume that the attacker tries to disturbance the local model to make it unable to detect attacks correctly, by confusing the normal traffic and attack traffic. The labels of data set of the attacked client are set to random attack types or normal type. The third scenario uses resampled data set and with two clients attacked. Inevitably, introducing the blockchain into the system lead to additional overhead. We then measure the system performance overhead brought by the blockchain. Since we deploy the proposed architecture in a private chain, we also evaluate the monetary cost in gas for every operation. In the end, we theoretically analyze the complexity and security of the proposed architecture.

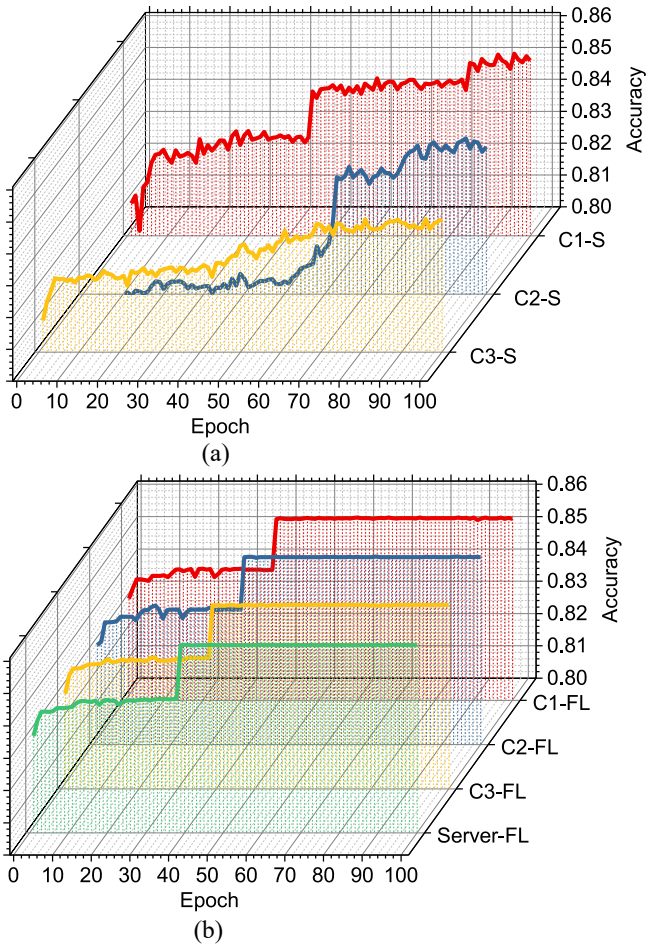

Fig. 3. Accuracy comparison between standalone and federated training when no client attacked. (a) Standalone training. (b) Federated training.

B. Effectiveness Evaluation

To verify the effectiveness of the proposed architecture, we extract three IID data sets from preprocessed Edge-IIoTset for three federated clients. First, the clients are trained centralized with own data set and the same model, the results of which is show in Fig. 3. Fig. 3(a) depicts the accuracy of centralized standalone training of three FL clients for 100 local epochs, where $Cn - S$ means that client n is training on the standalone (S) manner. From Fig. 3(a), it can be observed that the efficiency of training varies among the three clients. The accuracy of C1 experienced three stages of improvement during the training process. The accuracy of

Authorized licensed use limited to: Green University of Bangladesh. Downloaded on January 23,2025 at 14:02:44 UTC from IEEE Xplore. Restrictions apply.

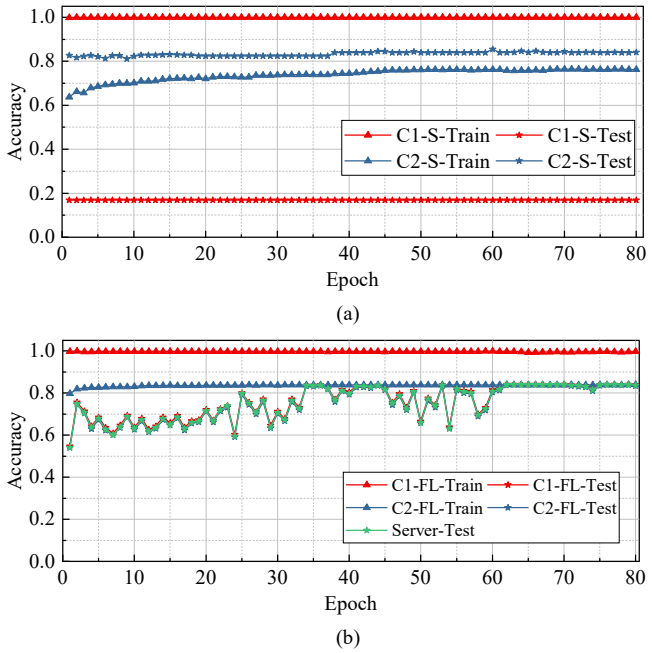

Fig. 4. Accuracy comparison between standalone and FL when one client attacked. (a) Standalone training. (b) Federated training.

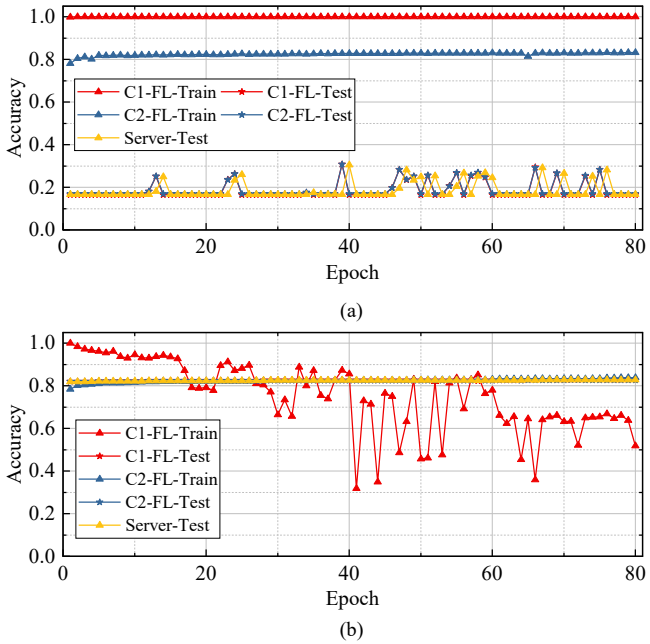

Fig. 5. Validation and test performance of proposed scheme. (a) FedAvg. (b) Proposed architecture.

C2 rapidly increases at 60 epochs and then slowly improves thereafter. However, C3 consistently improves slowly, but ultimately achieves lower accuracy performance than the other two clients. The differences in data distribution, network initialization parameters, and random batch extraction during training among the three clients result in different training efficiency. Fig. 3(b) is the accuracy results of three clients and server in FL structure, where $Cn-FL$ means that client n is training on the federated mode. It can be observed that all clients show a similar rising trend, as well as higher convergence speed. At about epoch 35, all clients obtain the best performance and begin to be stable, which is faster than any client that in centralized training. Therefore, it can be concluded that the intrusion detection performance is affected by the data distribution of training data set held by different clients and the randomness of network parameters. Some edge servers may take longer to train to optimal performance due to poor data quality or inappropriate parameter initialization. In this scenario, FL can improve the training efficiency of all clients through parameter aggregation, enabling them to achieve optimal intrusion detection performance faster.

C. Security Evaluation

To evaluate the defense ability of our proposed architecture against MITM attack aiming at tampering the parameters, we design one experiment with one client attacked and another with two attacked. We manipulate the training data set in C1 to simulate the MITM attack that tries to disturb the system by introducing malicious data. The labels of the training data set in C1 are all set to the same to make it generate an invalid model. In a three-client FL system with one attacked, we compare the accuracy performance of centralized training and federated leaning, the result of which is shown in Fig. 4. Here, we just present the result of C1 and C2 to express the difference between normal client and attacked client. $Cn-S-Train$ or Test means the accuracy of client n on training or test data set. $Server-Test$ means that the accuracy on PS. From Fig. 4(a), the training and test accuracy of C1 is almost unchanged after starting. Because the data set is maliciously modified, the test accuracy of C1 is quite low while the training accuracy is close to 1. Client 2, which is not attacked, was able to train normally. Once the training data set is maliciously modified, the detection performance will be significantly decreased to weaken the intrusion detection of IIoT system. Fig. 4(b) illustrates the federated scenario, where even with a malicious data set, C1 can achieve regular detection accuracy. Because the parameter averaging strategy can mitigate the negative influence brought by a few nodes, valuable and correct information can also be spread to all clients. Therefore, it can be concluded that FL can reduce the negative impact of malicious nodes or attackers to a certain extent.

Further, we simulate the scenario where both C1 and C3 are subjected to MITM attacks, i.e., the local data sets of both are all assigned to 0. The intrusion detection performance of the FedAvg and the architecture proposed in this article are compared, and the results are shown in Fig. 5. Here, we only show the attacked C1 and the normal C2. Fig. 5(a) shows the FedAvg algorithm fails to train the model after two-thirds of the nodes are attacked. The model parameters of C1's training set have been compromised by tampering, resulting in the algorithm's detection performance being close to perfect. In contrast, C2 achieved an accuracy of approximately 0.8 on its training set. However, the FedAvg algorithm's detection ability is also compromised on the test set due to the distribution of erroneous parameters trained by C1 and C3 to C2. The proposed blockchain-reinforced FL architecture can address the problem of system failure caused by most nodes attacked,

Authorized licensed use limited to: Green University of Bangladesh. Downloaded on January 23,2025 at 14:02:44 UTC from IEEE Xplore. Restrictions apply.

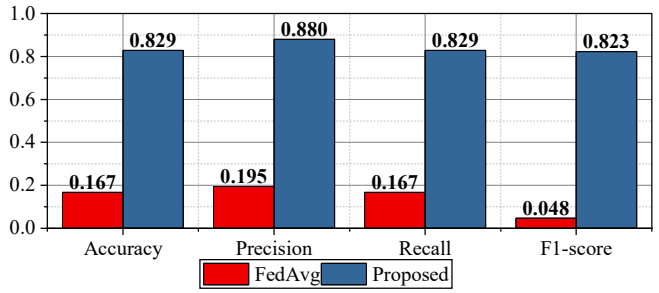

Fig. 6. Performance comparison as two clients attacked.

the results of which are shown in Fig. 5(c). Due to the security protection capability of the proposed method, only the parameters of C2 are used for aggregation, while the parameters propagated to the server by the attacked C1 and C3 are discarded. This ensures that the parameters distributed to all clients are normal. The accuracy of both clients and servers on the test set remains around 0.8. It is important to note that, despite the security protection, the final detection is still weaker than normal. Therefore, all clients can maintain relatively better detection performance than existing FedAvg.

Finally, Fig. 6 compares the intrusion detection performance of the two algorithms over 100 epochs with two-thirds of the clients under attack. The intrusion detection performance of Fedavg is severely compromised, rendering the IIoT system unable to differentiate between normal and malicious traffic. This vulnerability can be exploited by attackers during the model's corruption to launch more impactful attacks. In this case, our proposed algorithm improves the accuracy, precision, recall, and F1-score of intrusion detection by more than three times. Therefore, our proposed architecture maintains the security of the FL process even when most nodes in the system are under attack. The intrusion detection model's security ensures the system can distinguish between normal and malicious traffic, preventing attackers from utilizing this method to create attack vectors.

D. Overhead Evaluation

We use Perfmon to monitor the system performance, to evaluate the overhead caused by blockchain, which measures the total resource utilization of all system components, including the PS, clients, and blockchain nodes. We both take 160 s for the system with blockchain and without blockchain, where the CPU and memory overhead is collected. The results are shown in Fig. 7. The red dash line represents the result of FL process without blockchain reinforcement. The black solid line represents the result of FL process with blockchain reinforcement. Different colored shadows correspond to the duration of two rounds in each scene. The network model, hyperparameter and running environment are identical. In Fig. 7(a), we present the CPU overhead through the percentage of Process Time, which represents the percentage of current CPU utilization. It can be seen that during the training process, due to the need for the system to verify the parameters written by the client to the blockchain, the CPU utilization in the blockchain environment is higher than that of the FL only system. Due to the need to write and verify blocks, the overall time cost

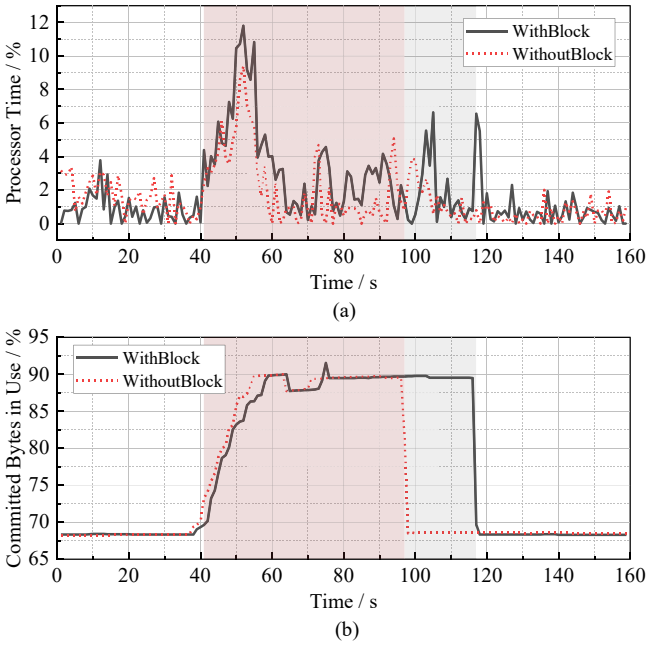

Fig. 7. Overhead evaluation of the proposed architecture. (a) Processor time (CPU). (b) Usage of committed bytes in use (memory).

is also higher, which is related to the difficulty of the PoW algorithm, the computational power of the system, and the set stopping conditions. During parameter aggregation, the PS only queries data from the blockchain, so the CPU overhead generated is not significant, as shown in the figure for 70–80 s. Throughout the entire training process, the introduction of blockchain resulted in an additional CPU computational overhead of 37.45%. The PoW algorithm mainly relies on CPU computation, so the memory overhead occupied by the two is relatively similar, as shown as Fig. 7(b). Therefore, the introduction of blockchain will inevitably bring additional system overhead, which increases with difficulty, number of nodes, etc.

To present the monetary costs the transactions introduce to the system, we evaluate the gas costs of the operations in our proposed architecture. To implement the workflow, we deploy a smart contract providing the interfaces, which manage the interaction between FL and blockchain. Deploying it on our private chain costs 68 8450 gas. Therefore, according to [44], the bootstrap cost of the proposed architecture is 0.0068845 ether, since we set the gas price as 1Gwei. To record a 128-bit parameter into the smart contract for verification, a FL client should pay 17 1049 gas. The ether cost of each recording operation is 0.000171049 ether, which is a fixed operation cost. Since we deploy the For a FL system with 3 clients, the gas cost of each training round is about 0.5M. If we record all of the parameters into the blockchain, the gas cost will be so huge that it exceeds the gas limit (50M). And the size of the parameters increases rapidly with the size of the network. Hence, storing the MD5 value rather than the original parameters saves a lot of computing resources. For the server that controls the FL training, verifying the confidentiality of a parameter costs 0 gas, because it does not change the state of the smart contract. Another operation that costs more gas

Authorized licensed use limited to: Green University of Bangladesh. Downloaded on January 23,2025 at 14:02:44 UTC from IEEE Xplore. Restrictions apply.

is flushing the value stores in the smart contract, the gas cost of which is 76 1012.

E. Complexity Analysis

Assuming there are $|C|$ clients, each holds local data set with size of $|D|$; model parameter size is $|W|$; p clients participating in the model aggregation in each federated round; each client conducts l local epochs in every round.

Computational Complexity: Each client conducts SGD locally to update its local model. The computation complexity of each client performing local training is $\mathcal{O}(|D| \cdot |W|)$. After training, each client computes the hash value of its parameters, which costs $\mathcal{O}(1)$. Since it is hard to evaluate the computational complexity of PoW, we assume that it is an operation determined by the mining difficulty d, represented as $\mathcal{O}(d)$. After conducting PoW, the nodes that have not obtained accounting rights, i.e., nodes that have not packaged blocks, will verify the legality of the blocks upon receiving the broadcast, introducing complexity $\mathcal{O}(1)$, since all nodes conducts in parallel. After local training, the clients participated in the FL reports their models to central server, so the computational complexity of aggregation is $\mathcal{O}(p \cdot |W|)$. Before the aggregation, each incoming parameter is verified, which leads to complexity of $\mathcal{O}(p)$. Compared to the existing FedAvg, the proposed architecture introduces additional $\mathcal{O}(d) + \mathcal{O}(p)$ computational complexity for each round.

Communication Complexity: There are five main phrases that introduces communication complexity, namely, parameter recording, block broadcasting, parameter uploading, parameter verification, and global model distribution. The parameter recording operation is represented as a transaction broadcasted to the blockchain network, costing $\mathcal{O}(|C|)$. The block broadcasting also introduces $\mathcal{O}(|C|)$ communication complexity because of the block broadcasting. For the parameter uploading, the communication time is represented as $\mathcal{O}(p \cdot |W|)$. Since there are p clients participating in the model aggregation and only the fixed-length hash values are verified, the communication time of parameter verification is $\mathcal{O}(p)$. After global aggregation, the updated model distribution costs $\mathcal{O}(|C| \cdot |W|)$. Therefore, compared to the existing FedAvg, the proposed architecture introduces additional $\mathcal{O}(|C|) + \mathcal{O}(p)$ communication complexity.

F. Security Analysis

In this section, we present the security analysis of the proposed blockchain-reinforced FL architecture, focusing on the ability of data privacy protection, MITM attack prevention, and data traceability.

Data Privacy Protection: For intrusion detection scenarios, attackers usually illegally access the system by targeting the attack surface and certain fragile attributes of the system. Therefore, the intrusion data collected in IIoT edge servers may contain the attacker's attack behavior, revealing the system's vulnerabilities. The adoption of FL technology enables the proposed architecture to have the ability to protect data privacy.

MITM Attack Prevention: In each federated round, the updated parameters are transmitted through the network between clients and sever, where attackers could launch MITM attacks, passing tampered parameters to the server to disrupt the normal training process. In the IIoT intrusion detection scenario, if the performance of the AI model is affected, the system will be unable to distinguish between attack traffic and normal traffic, and may even lead to more serious system level security issues. In this architecture, we integrate the blockchain into FL process to prevent tampered parameters from being added to the aggregation process. Therefore, the proposed architecture achieves the MITM attack prevention.

Data Traceability: The proposed architecture provides data traceability through the integration of blockchain, where the FL parameters in every round can be extracted anytime to verify the source client and recording time. The data traceability is useful for troubleshooting and security auditing of IIoT system.

VI. Conclusion

As an emerging technology, the security of IIoT is different from a traditional network security problem. Massive devices and distributed structure obstruct the application of many security measures. In this article, combining the blockchain and FL, a cooperative intrusion detection architecture is proposed in edge-cloud IIoT structure. The edge servers collect network traffic, running federated client applications to extract features. Through FL, attack characteristics from different clients are aggregated and spread to all clients, which significantly enhances the detection capability for unknown attacks. Inspired by the tamper-proof mechanism of blockchain, we establish a P2P network between cloud and edge servers to build a private chain. A smart contract to record the information abstract of the transmitted parameter is deployed in blockchain. Based on a Web 3 technique, the client stores record into blockchain and the server verifies parameter through querying. In this way, the MITM attack can be effectively prevented. The simulation results show that the proposed architecture can realize effective cooperative intrusion detection. For future work, more refined and efficient parameter aggregation approaches will be studied and deployed to improve detection accuracy. Besides, the parameter communication between federated server and clients are still plaintext, which may lead to information disclosure. More secure parameter sharing approaches will be studied.

References

[1] G. Rathee, M. Balasaraswathi, K. P. Chandran, S. D. Gupta, and C. S. Boopathi, "A secure IoT sensors communication in industry 4.0 using blockchain technology," *J. Ambient. Intell. Humaniz. Comput.,* vol. 12, no. 1, pp. 533–545, Jan. 2021, doi: 10.1007/s12652-020-02017-8.

[2] X. Wang, K. Zhang, J. Wang, and Y. Jin, "An enhanced competitive swarm optimizer with strongly convex sparse operator for large-scale multiobjective optimization," *IEEE Trans. Evol. Comput.,* vol. 26, no. 5, pp. 859–871, Oct. 2022, doi: 10.1109/TEVC.2021.3111209.

[3] M. Al-Hawawreh and E. Sitnikova, "Developing a security testbed for Industrial Internet of Things," *IEEE Internet Things J.,* vol. 8, no. 7, pp. 5558–5573, Apr. 2021, doi: 10.1109/JIOT.2020.3032093.

[4] Z. Wei, F. Liu, C. Masouros, N. Su, and A. P. Petropulu, "Toward multi-functional 6G wireless networks: Integrating sensing, communication, and security," *IEEE Commun. Mag.,* vol. 60, no. 4, pp. 65–71, Apr. 2022, doi: 10.1109/MCOM.002.2100972.

Authorized licensed use limited to: Green University of Bangladesh. Downloaded on January 23,2025 at 14:02:44 UTC from IEEE Xplore. Restrictions apply.

[5] S. Messaoud, A. Bradai, O. B. Ahmed, P. T. A. Quang, M. Atri, and M. S. Hossain, "Deep federated Q-learning-based network slicing for industrial IoT," *IEEE Trans. Ind. Informat.,* vol. 17, no. 8, pp. 5572–5582, Oct. 2021, doi: 10.1109/TII.2020.3032165.

[6] P. Zhang, C. Wang, C. Jiang, and Z. Han, "Deep reinforcement learning assisted federated learning algorithm for data management of IIoT," *IEEE Trans. Ind. Informat.,* vol. 17, no. 12, pp. 8475–8484, Mar. 2021, doi: 10.1109/TII.2021.3064351.

[7] J. Li, L. Lyu, X. Liu, X. Zhang, and X. Lyu, "FLEAM: A federated learning empowered architecture to mitigate DDoS in industrial IoT," *IEEE Trans. Ind. Informat.,* vol. 18, no. 6, pp. 4059–4068, Jun. 2022, doi: 10.1109/TII.2021.3088938.

[8] Y. Song, T. Liu, T. Wei, X. Wang, Z. Tao, and M. Chen, "FDA^3: Federated defense against adversarial attacks for cloud-based IIoT applications," *IEEE Trans. Ind. Informat.,* vol. 17, no. 11, pp. 7830–7838, Jun. 2021, doi: 10.1109/TII.2020.3005969.

[9] J. Wang, Q. Chang, T. Gao, K. Zhang, and N. R. Pal, "Sensitivity analysis of Takagi–Sugeno fuzzy neural network," *Inf. Sci.,* vol. 582, pp. 725–749, Jan. 2022, doi: 10.1016/j.ins.2021.10.037.

[10] X. Hou, J. Wang, C. Jiang, X. Zhang, Y. Ren, and M. Debbah, "UAV-enabled covert federated learning," *IEEE Trans. Wireless Commun.,* vol. 22, no. 10, pp. 6793–6809, Oct. 2023, doi: 10.1109/TWC.2023.3245621.

[11] D. Li et al., "Blockchain for federated learning toward secure distributed machine learning systems: A systemic survey," *Soft Comput.,* vol. 26, no. 9, pp. 4423–4440, May 2022, doi: 10.1007/s00500-021-06496-5.

[12] L. Liu, W. Zhang, and C. Han, "A survey for the application of blockchain technology in the media," *Peer-to-Peer Netw. Appl.,* vol. 14, no. 5, pp. 3143–3165, Sep. 2021, doi: 10.1007/s12083-021-01168-5.

[13] D. Wang, M. Wu, Z. Wei, K. Yu, L. Min, and S. Mumtaz, "Uplink secrecy performance of RIS-based RF/FSO three-dimension heterogeneous networks," *IEEE Trans. Wireless Commun*., vol. 23, no. 3, pp. 1798–1809, Mar. 2024, doi: 10.1109/TWC.2023.3292073.

[14] S. N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhelifa, and A. Bani-Hani, "Blockchain smart contracts: Applications, challenges, and future trends," *Peer-to-Peer Netw. Appl.,* vol. 14, no. 5, pp. 2901–2925, Sep. 2021, doi: 10.1007/s12083-021-01127-0.

[15] R. Taheri, M. Shojafar, M. Alazab, and R. Tafazolli, "Fed-IIoT: A robust federated malware detection architecture in industrial IoT," *IEEE Trans. Ind. Informat.,* vol. 17, no. 12, pp. 8442–8452, Dec. 2021, doi: 10.1109/TII.2020.3043458.

[16] Y. Wu, H. N. Dai, and H. Wang, "Convergence of blockchain and edge computing for secure and scalable IIoT critical infrastructures in industry 4.0," *IEEE Internet Things J.,* vol. 8, no. 4, pp. 2300–2317, Feb. 2021, doi: 10.1109/JIOT.2020.3025916.

[17] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of blockchains in the Internet of Things: A comprehensive survey," *IEEE Commun. Surveys Tuts.,* vol. 21, no. 2, pp. 1676–1717, 2019, doi: 10.1109/COMST.2018.2886932.

[18] A. H. Sodhro, S. Pirbhulal, M. Muzammal, and L. Zongwei, "Towards blockchain-enabled security technique for Industrial Internet of Things based decentralized applications," *J. Grid Comput.,* vol. 18, no. 4, pp. 615–628, Dec. 2020, doi: 10.1007/s10723-020-09527-x .

[19] G. Puthilibai, T. Benil, S. Chitradevi, V. Devatarika, D. R. Ashwin Kumar, and U. Padma, "Securing IIoT sensors communication using blockchain technology," in *Proc. ICPECTS*, Chennai, India, 2022, pp. 1–4.

[20] J. Wang, C. Jiang, Z. Han, Y. Ren, R. G. Maunder, and L. Hanzo, "Taking drones to the next level: Cooperative distributed unmanned-aerial-vehicular networks for small and mini drones," *IEEE Veh. Technol. Mag.,* vol. 12, no. 3, pp. 73–82, Sep. 2017, doi: 10.1109/MVT.2016.2645481.

[21] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, "Anonymous reputation system for IIoT-enabled retail marketing atop PoS blockchain," *IEEE Trans. Ind. Informat.,* vol. 15, no. 6, pp. 3527–3537, Feb. 2019, doi: 10.1109/TII.2019.2898900.

[22] Y. Tian, T. Li, J. Xiong, M. Z. A. Bhuiyan, J. Ma, and C. Peng, "A blockchain-based machine learning framework for edge services in IIoT," *IEEE Trans. Ind. Informat.,* vol. 18, no. 3, pp. 1918–1929, Mar. 2022, doi: 10.1109/TII.2021.3097131.

[23] Z. Wei, C. Masouros, X. Zhu, P. Wang, and A. P. Petropulu, "PHY layer anonymous precoding: Sender detection performance and diversity-multiplexing tradeoff," *IEEE Trans. Wireless Commun*., vol. 23, no. 5, pp. 4531–4545, May 2024, doi: 10.1109/TWC.2023.3319532.

[24] K. Demertzis, L. Iliadis, N. Tziritas, and P. Kikiras, "Anomaly detection via blockchained deep learning smart contracts in industry 4.0," *Neural Comput. Appl.,* vol. 32, no. 23, pp. 17361–17378, Dec. 2020, doi: 10.1007/s00521-020-05189-8.

[25] R. Saha et al., "DHACS: Smart contract-based decentralized hybrid access control for Industrial Internet-of-Things," *IEEE Trans. Ind. Informat.,* vol. 18, no. 5, pp. 3452–3461, May 2022, doi: 10.1109/TII.2021.3108676.

[26] U. Narayanan, V. Paul, and S. Joseph, "Decentralized blockchain based authentication for secure data sharing in cloud-IoT," *J. Ambient. Intell. Humaniz. Comput.,* vol. 13, no. 2, pp. 769–787, Feb. 2022, doi: 10.1007/s12652-021-02929-z.

[27] W. Zhang et al., "Blockchain-based federated learning for device failure detection in industrial IoT," *IEEE Internet Things J.,* vol. 8, no. 7, pp. 5926–5937, Apr. 2021, doi: 10.1109/JIOT.2020.3032544.

[28] M. H. U. Rehman, A. M. Dirir, K. Salah, E. Damiani, and D. Svetinovic, "TrustFed: A framework for fair and trustworthy cross-device federated learning in IIoT," *IEEE Trans. Ind. Informat.,* vol. 17, no. 12, pp. 8485–8494, Dec. 2021, doi: 10.1109/TII.2021.3075706.

[29] P. Zhang, Y. Hong, N. Kumar, M. Alazab, M. D. Alshehri, and C. Jiang, "BC-EdgeFL: A defensive transmission model based on blockchain-assisted reinforced federated learning in IIoT environment," *IEEE Trans. Ind. Informat.,* vol. 18, no. 5, pp. 3551–3561, May 2022, doi: 10.1109/TII.2021.3116037.

[30] X. Hou, J. Wang, Z. Fang, Y. Ren, K.-C. Chen, and L. Hanzo, "Edge intelligence for mission-critical 6G services in space—air--ground integrated networks," *IEEE Netw.,* vol. 36, no. 2, pp. 181–189, Mar./Apr. 2022, doi: 10.1109/MNET.121.2100324.

[31] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Informat.,* vol. 16, no. 6, pp. 4177–4186, Jun. 2020, doi: 10.1109/TII.2019.2942190.

[32] B. Jia, X. Zhang, J. Liu, Y. Zhang, K. Huang, and Y. Liang, "Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in IIoT," *IEEE Trans. Ind. Informat.,* vol. 18, no. 6, pp. 4049–4058, Jun. 2022, doi: 10.1109/TII.2021.3085960.

[33] J. Xu, J. Lin, W. Liang, and K.-C. Li, "Privacy preserving personalized blockchain reliability prediction via federated learning in IoT environments," *Clust. Comput.,* vol. 25, no. 4, pp. 2515–2526, Aug. 2022, doi: 10.1007/s10586-021-03399-w.

[34] O. Malomo, D. Rawat, and M. Garuba, "Security through block vault in a blockchain enabled federated cloud framework," *Appl. Netw. Sci.,* vol. 5, no. 1, p. 16, Feb. 2020, doi: 10.1007/s41109-020-00256-4.

[35] S. Y. J. Ho Park, S. K. Singh, and J. H. Park, "PoAh-enabled federated learning architecture for DDoS attack detection in IoT networks," *Human-Centric Comput. Inf. Sci.,* vol. 14, no. 03, pp. 1–24, Jan. 2024, doi: 10.22967/HCIS.2024.14.003.

[36] 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.

[37] R. Yang et al., "Efficient intrusion detection toward IoT networks using cloud–edge collaboration," *Comput. Netw.,* vol. 228, Jun. 2023, Art. no. 109724, doi: 10.1016/j.comnet.2023.109724.

[38] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.,* vol. 10, no. 2, pp. 1–19, Jan. 2019, doi: 10.1145/3298981.

[39] C. Cachin and M. Vukolić, "Blockchain consensus protocols in the wild," 2017, *arXiv:1707.01873*.

[40] T. Qiu et al., "Edge computing in Industrial Internet of Things: Architecture, advances and challenges," *IEEE Commun. Surveys Tuts.,* vol. 22, no. 4, pp. 2462–2488, 4th Quart., 2020, doi: 10.1109/COMST.2020.3009103.

[41] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning," *IEEE Access,* vol. 10, pp. 40281–40306, 2022, doi: 10.1109/ACCESS.2022.3165809.

[42] D. J. Beutel et al., "Flower: A friendly federated learning research framework," 2020, *arXiv:2007.14390*.

[43] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger." 2014. [Online]. Available: http://gavwood.com/paper.pdf

[44] M. Pincheira, E. Donini, M. Vecchio, and R. Giaffreda, "An infrastructure cost and benefits evaluation framework for blockchain-based applications," *Systems,* vol. 11, no. 4, p. 184, Apr. 2023, doi: 10.3390/systems11040184.

Authorized licensed use limited to: Green University of Bangladesh. Downloaded on January 23,2025 at 14:02:44 UTC from IEEE Xplore. Restrictions apply.