Computer Science and Information Systems 20(4):1541–1568 https://doi.org/10.2298/CSIS230401063G

BI-FERH: Blockchain-IoT Based Framework for Securing Smart Hotel

Quanlong Guan[1,2,4], Jiawei Lei[1,3], Chaonan Wang[3], Guanggang Geng[3], Yuansheng Zhong[4], Liangda Fang[1,2,3], Xiujie Huang[1,2,⋆], and WeiQi Luo[1,2,3]

[1] Guangdong Institute of Smart Education,
Jinan University, Guangzhou 510632, China
[2] College of Information Science and Technology,
Jinan University, Guangzhou 510632, China
[3] College of Cyber Security, Jinan University,
Guangzhou 510632, China
[4] Key Laboratory of Safety of Intelligent Robots for
State Market Regulation, Guangdong Testing Institute of Product
Quality Supervision, Guangzhou 510670, People's Republic of China

Abstract. IoT devices and applications are growing rapidly as a result of the advancement of IoT technology. In the case of smart hotels with many IoT devices, the majority of the data generated by those devices contains the private information of users, which is susceptible to being changed and leaked during transmission and storage. To overcome it, this paper proposes a blockchain-IoT based Framework for securing smart hotels(BI-FERH) to enhance the security of hotel information systems. The high performance BI-FERH architecture takes advantage of real-time data transmission capabilities offered by IoT devices. Sensitive data generated by IoT devices is protected in BI-FERH, enhancing tamper-proof capabilities. The results of the experiment demonstrate that BI-FERH can increase the security of smart hotel systems while preserving operational efficacy. An innovative and safe solution for the information management system of smart hotels is offered by the BI-FERH framework.

Keywords: Blockchain, Smart Hotel, IoT, Privacy Protection, Hyperledger Fabric, Data Security.

1. Introduction

IoT devices and blockchain technology are increasingly mature, enhancing people's quality of life. Smart hotels leverage modern technologies like IoT, cloud computing, smart devices, and big data to enhance guest experiences [1]. Over time, IoT devices have elevated the hotel industry's service levels with innovations like Smart Speakers, Social Robots [2], and Hotel intelligent guidance [3]. However, the use of IoT devices in hotels introduces security risks, as they handle sensitive personal information. Protecting this data effectively is challenging, as it faces threats like data leakage and manipulation during service enhancement [4]. Unfortunately, scant literature exists on hotel information security despite its significance, as hotels harbor private data that, if misused, could

⋆ Corresponding author.

lead to cybercrimes. The term ”cybercrime” denotes illegal activities employing computer technology to compromise systems or data. Cybercrime has surged globally, resulting in substantial financial losses, with data breaches causing losses to rise from 3.8 million in 2021 to 4 million in 2022 [5]. Notably, small businesses, including hospitality, face cyberattacks that compromise customer data [6]. Such businesses often lack the means to safeguard data, which, if misused, can facilitate crimes like fraud. Enhancing information systems' security is imperative to protect user data, which entails improving the information management system of smart hotels to utilize guest data effectively and withstand cyberattacks.

Presently, hotel information management systems typically adopt a centralized data storage strategy, which poses challenges in ensuring data integrity due to potential tampering and lack of traceability. Furthermore, central databases become vulnerable to hacking attacks, compromising data availability. Data reliability issues are inherent in this model, with limited security improvement options such as key replacement and enhanced management. The proliferation of data in smart hotels amplifies data protection complexities, necessitating an effective and secure information management system to safeguard guest data. Unfortunately, comprehensive security solutions for hotel information systems against various attackers are lacking.

This paper introduces the Blockchain-IoT-based Framework for sEcuring smaRt Hotel (BI-FERH), aiming to provide a secure and reliable data management solution for the hospitality sector. This framework addresses data integrity compromise and enhances information systems' resilience against malicious attackers. Combining real-time data transmission of IoT devices with blockchain's tamper-evident and traceable characteristics, the BI-FERH framework promises efficient and secure information systems. By leveraging blockchain, the framework elevates data security and trustworthiness in hotels, contributing to enhanced smart hotel data security.

The paper's main contributions are as follows:

(1) Introducing the novel BI-FERH framework, combining blockchain and IoT to secure hotel data while enhancing customer service.

(2) Employing edge computing and edge servers to establish blockchain network nodes at the IoT device layer, enhancing data redundancy and decentralizing resources in the hotel industry.

(3) Designing and implementing the BI-FERH framework, coupled with an Autoencoder + TCN machine learning model to create an Intrusion Detection System (IDS). The IDS identifies 8 types of attack traffic with 97% accuracy, benefiting from the distributed structure's tamper-evident and traceable data attributes.

(4) Conducting experimental tests on the BI-FERH framework, revealing 30% greater efficiency compared to similar blockchain applications, with 10% lower memory consumption and 7% lower CPU consumption. The study delves into usability, performance, and security aspects of the experimental results.

The paper is organized as follows: Section II presents related work, Section III details the BI-FERH solution, Section IV discusses experimental tests, Section V analyzes experimental results, and Section VI concludes the paper. Some parts of this article were first presented in Guan et al. [7].

2. Related Work

The blockchain, introduced by Satoshi Nakamoto [8], presents a peer-to-peer payment and electronic cash system. As Bitcoin's underlying technology, it has immense potential for applications and development, impacting services reliant on trusted third parties. Blockchain finds utility in data structures, verification methods, communication protocols, and information storage [9]. Its hashing algorithm converts variable-length input data to fixed-length digests irreversibly [10]. Public key cryptography, an asymmetric encryption algorithm, verifies identities within blockchain networks [11], enhancing security for data transmission [12]. Consensus mechanisms foster trust among blockchain nodes and ensure transaction consistency [13], spawning diverse algorithms like Proof of Work (POW), Proof of Stake (POS), and Practical Byzantine Fault Tolerance (PBFT).

The Internet of Things (IoT) has seen widespread adoption, necessitating secure, accessible, and reliable infrastructure to process and store data. Blockchain addresses challenges in traditional IoT security protocols [14], especially as a centralized model for IoT data communications introduces privacy and security issues [15].

Studies like Donet, Pérez-Solà, and Herrera-Joancomartí's [16] analyze bitcoin network attributes, while the Bitcoin system prioritizes user privacy and anonymity through public key concealment [17]. Smart contracts elevate blockchain interactivity [18], with Ethereum showcasing blockchain and smart contract synergy, albeit hindered by proof-of-work's inefficiency [19]. Hyperledger Fabric, a Linux Foundation-backed project, employs the Practical Byzantine Fault Tolerance (PBFT) mechanism, enabling transaction validation [20]. Hao et al. [21] determined PBFT's superiority over proof-of-work (PoW) regarding latency and throughput. Fabric's chaincode enables application interaction [22], granting high throughput (>20,000 tps) without proof-of-work [23]. Comparative evaluations of Ether and Hyperledger Fabric address aspects such as throughput, latency, security, and scalability [24] [25].

Table 1 summarizes differences between these blockchain platforms.

Table 1. Platform comparison

Name	Type	Consensus	Smart contract language	Cost	Security
Bitcoin	Public	PoW	Stack based script	Extremely high	Extremely high
Ethereum	Public	PoW/PoS	Solidity	High	High
Hyperledger Fabric	Consortium	Solo/Kafka/PBFT	Go/Java	low	Higher

IoT devices have navigated various challenges, as highlighted by Da Xu, He, and Li in 2014 [26], who outlined the IoT industry's development and dilemmas at that time. Edge computing meets low-latency requirements for compute or data-intensive tasks [27], aligning well with efficient IoT operation. Combining edge architectures with blockchain is viable [28]. Expanding edge computing, Du et al. explored a blockchain-enhanced EC market, where data service operators rent edge nodes, leasing them to user terminals for computation offloading [29]. The advent of blockchain facilitated IoT and blockchain integration [30]. Kshetri [31] proposed blockchain's use in supply chains to mitigate cost

and risk. Blockchain elevates IoT interactivity, data security, reliability, and scalability [30]. Khan and Salah [32] surveyed IoT security issues, suggesting blockchain solutions. Blockchain's impact extends beyond finance to non-monetary domains like smart factories [33], traceable supply chains [34], fake news detection [35], smart homes [36], medical records [37], smart hospitals [38], secure transportation [39], decentralized voting [40], and more. The IoT-blockchain combination streamlines data transactions [41]. Evaluating merged systems challenges researchers, with Lao et al. using throughput and consensus security to assess IoT ecosystems and blockchain platforms [42]. Alshudukhi et al. introduced a blockchain-microservices security framework for IoT federated cloud systems [43].

Inspired by Daidone et al. [44] who used blockchain to protect IoT privacy, this paper addresses IoT devices' role in enhancing hotel service quality. Privacy concerns persist due to the sensitive data inherent in the hotel industry. Sahu and Gutub [5] highlighted personal information leakage risks in hotels and proposed grayscale steganography for protection. Presently, hotel data is centralized, risking tampering and leakage. A blockchain-based privacy scheme for IoT is feasible [45]. To elevate hotel services and data security, this paper proposes a blockchain-integrated smart hotel framework. Martinez-Rendon et al. [46] merged blockchain with edge IoT architecture for heightened security. Edge computing efficiently transfers IoT data, while blockchain safeguards guest data in hotels.

3. Blockchain-IoT based Framework for securing smart Hotel(BI-FERH)

To improve the quality of hotel services and data security, Blockchain-IoT based Framework for securing smart Hotel(BI-FERH) is proposed and shown by Figure 1 with three parts. The first part is made up of IoT devices and artificial intelligence for quality service. The second part consists of blockchain for data reliability and security. And the last part is connected by the Internet between users, devices, and databases. Hotel guests and staff request different services from the server through their respective Internet API interfaces. For example, the booking of various hotel services, the setting of IoT devices, and the viewing of hotel guests' private data, etc. The privacy data generated by the IoT devices is stored in the Hyperledger Fabric blockchain network. The confidentiality, integrity, and availability of the system data are improved by using blockchain data security and tamper-evident features. Through the use of channels in Fabric, data interoperability of multiple hotels can be achieved, while hotels outside the channels can achieve data confidentiality.

3.1. System Framework

As shown in Figure 2, the BI-FERH framework consists of five independent modules, which are the application layer, edge IoT layer, IDS layer, edge server layer, and blockchain layer. The functions of each layer in the framework are described below.

- Application Layer: Hotel guests and staff access specific services through this layer, like check-in, information management, and remote control of smart devices. It serves as the interface for interacting with the hotel system. It links to the next layer through human-machine interaction and network connections.

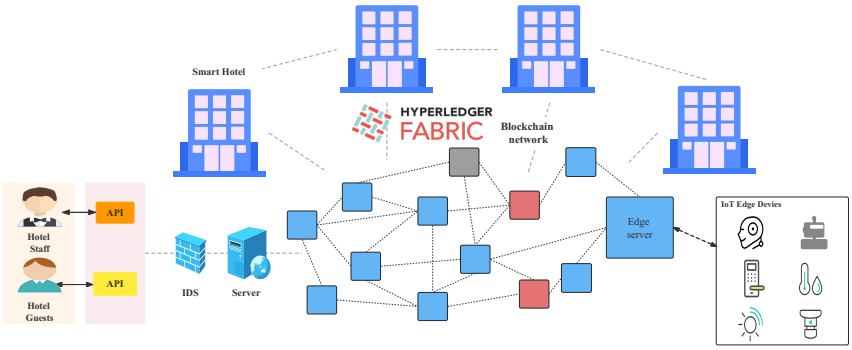


Fig. 1. BI-FERH architecture

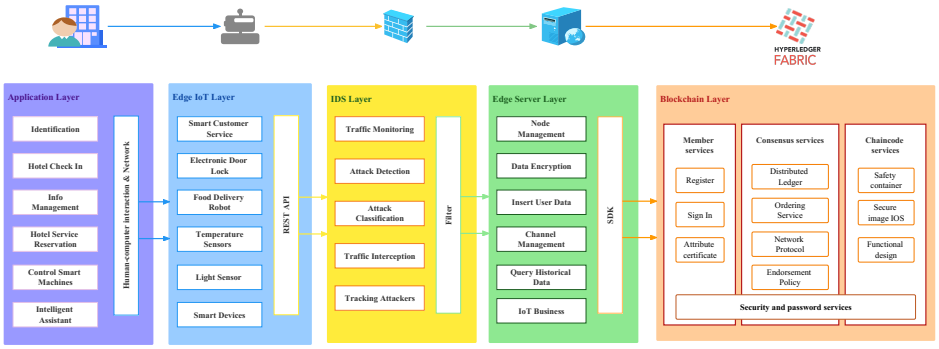


Fig. 2. System framework

- Edge IoT Layer: This layer provides hardware support for smart hotels. Given the limited computing power of IoT devices, edge computing architecture is adopted to efficiently process operations. Complex tasks are sent to edge servers for processing.
- IDS Layer: This layer aims to prevent malicious attacks on the information system. It collects network data via traffic monitoring for attack identification and classification. Malicious traffic is intercepted, and attacker information is traced.
- Edge Server Layer: This layer has two primary functions: processing edge device data and interacting with the blockchain layer. Edge devices send data for processing and receive processed data. It manages the blockchain network using the SDK, storing necessary data in the blockchain.
- Blockchain Layer: Hyperledger Fabric is chosen as the platform. It includes three key modules: the member service module, consensus service module, and chaincode module. Secure communication among these modules relies on security and cryptographic services, protected by public-private key pairs and TLS protocols. The member service module handles member vetting, registration, and authentication.

The Consensus Service module ensures secure ledger booking and data agreement among nodes. The chaincode service module automates business logic calculations.

3.2. Edge IoT Architecture

In the BI-FERH framework, IoT devices employ an edge IoT architecture. Edge computing is utilized to reduce latency and bandwidth by processing data closer to its source. This architecture comprises edge servers and edge devices. Edge devices collect data and send it to the edge server. The edge server processes data, provides computational resources, and returns results to edge devices. Figure 3 illustrates the fusion of blockchain and edge IoT architecture. Edge devices deliver intelligent services and data processing occurs on the edge server. The edge server, doubling as a blockchain network node, broadcasts and stores data. This harnesses blockchain features to enhance data reliability and security.

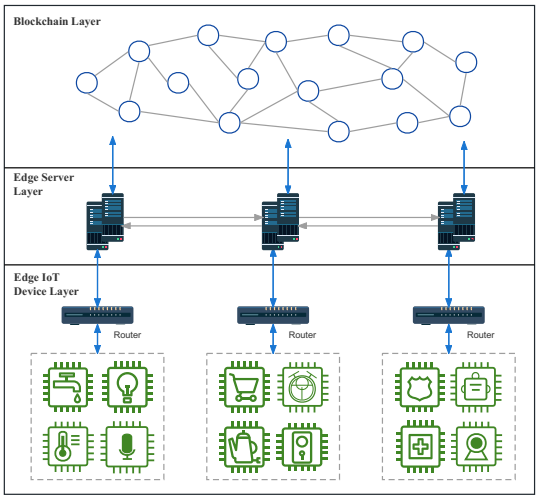


Fig. 3. Edge IoT Architecture

3.3. Intrusion Detection System

Traditional security tools are inadequate against sophisticated network attacks. Machine learning, especially in intrusion detection, is on the rise. In the BI-FERH framework (Figure 4), an autoencoder + TCN hybrid model is used for identifying and classifying attack traffic in network data.Network traffic's feature values enter the autoencoder (AE). Unlike manual feature engineering, AE automatically transforms input into suitable training data. Encoded data then goes into a temporal convolutional network (TCN), excelling in processing time-series data.After TCN's convolutional operation, data is flattened and sent to a fully connected layer for attack classification training. This hybrid approach strengthens BI-FERH's ability to detect and classify network attacks.

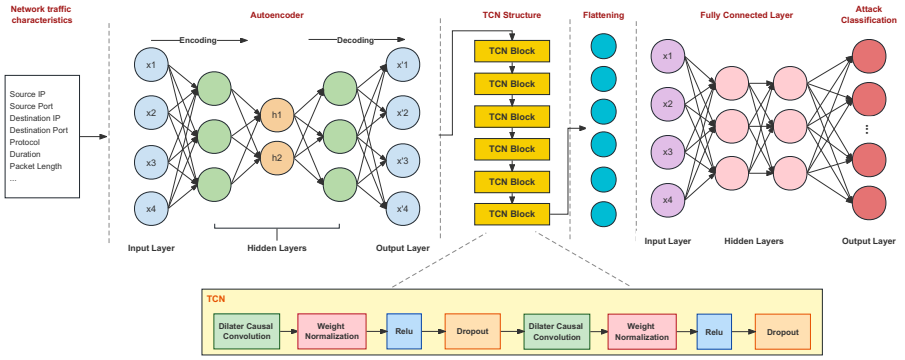


Fig. 4. Autoencoder + TCN Architecture

3.4. Workflow

As shown in Figure 5, the workflow of the BI-FERH solution consists of five main parts. The details of the steps in the workflow will be described in this section. The symbols used are explained in Table 2.

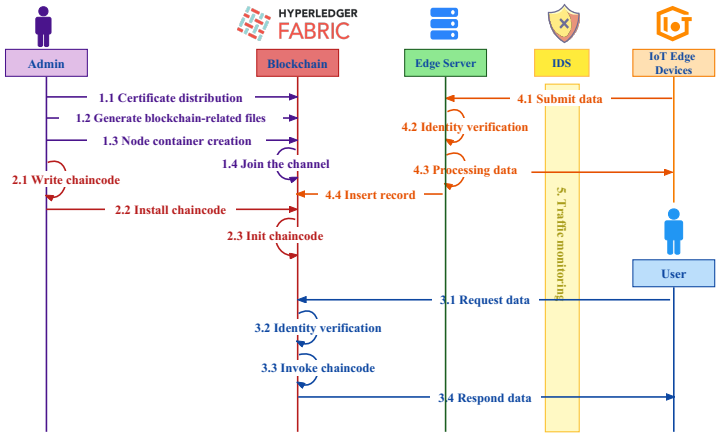


Fig. 5. Workflow of BI-FERH

Part 1 The main purpose of this part is the construction of the Hyperledger Fabric blockchain, which requires the admin to design and configure the network structure and finally build the basic network architecture. The first part mainly includes 4 steps.

Step 1 In order to secure communication and identity, certificates need to be issued for all members such as peers, orderers, channels, users, and devices. The certificate is generated and issued by the CA authority.

$$CA \rightarrow \{Cert_{user}, Cert_{orderer}, Cert_{peer}, Cert_{channel}, Cert_{device}\} \quad (1)$$

Table 2. Symbol descriptions

Symbol	Meaning
CA	Certificate Authority
$Cert$	Certificates and digital signature files
$user$	Hotel guests and staff
$device$	IoT devices
$channel$	Hyperledger Fabric channel
$conf_{net}$	Blockchain network configuration file
$conf_{node}$	Blockchain node docker configuration file
$File_{channel}$	Channel files for blockchain networks
$File_{Anchor}$	Anchor node files for building blockchain networks
$File_{Genesis}$	The first block data of the blockchain
$Image$	Docker Image
$Container$	Docker Container
$Ledger$	A ledger for recording information in the Hyperledger Fabric
CC	Chaincode in Hyperledger Fabric
$Code(business)$	Code corresponding chaincode for business logic requirements
SDK_{Go}	SDK written in golang in Hyperledger Fabric
$TxId$	Transaction id in blockchain
IDS	Intrusion Detection System
$Traffic$	Network traffic of users and IoT
$Invoke(CC_{Init})$	Invoke chaincode to initialize node information
$CC_{Identify}(Request)$	Chaincode for identity authentication of HTTP request packets
$Response$	Response package processed by chaincode
$Install(CC)$	Install chaincode for blockchain
$Traffic_{attatck}$	Attack traffic in the network
$Request$	HTTP business request package
$Response$	Response information for data processing.

Step 2 The admin designs the network structure to generate the configuration file $conf_{net}$. According to the configuration file, the relevant files needed for the blockchain network are generated, including the channel, anchor node and creation block files.

$$Generate\,(conf_{net}) \rightarrow \{File_{channel}, File_{Anchor}, File_{Genesis}\} \tag{2}$$

Step 3 According to the node configuration file $conf_{node}$ written by the admin, build docker images for peer and orderer nodes, and then store them in the docker container to run. The respective certificates also need to be packed into the image.

$$Build\,(conf_{node}, Cert) \rightarrow Image \xrightarrow{run} Container \tag{3}$$

Step 4 After the container is started, the container, ledger, and related files are added to the channel together to build out a complete blockchain environment.

$$\{Container, File_{channel,Anchor,Genesis}, Ledger\} \xrightarrow{join} Channel \tag{4}$$

Part 2 After the blockchain base environment is constructed, the admin needs to design the corresponding chaincode for the actual application and deploy it to the blockchain. This part is completed in three steps.

Step 1 The admin writes the chaincode for the blockchain application according to different business logic.

$$Code(business) \rightarrow CC \tag{5}$$

Step 2 Install the chaincode into all peer nodes via the SDK_{Go}.

$$Install(CC) \xrightarrow{SDK_{Go}} Peer \tag{6}$$

Step 3 Initialize the chaincode of the peer node by invoking the initialization function of chaincode through SDK_{Go}.

$$Invoke(CC_{Init}) \xrightarrow{SDK_{Go}} Peer \tag{7}$$

Part 3 This section focuses on the process of hotel guests and staff interacting with the system as users. The interaction between the user and the system can be divided into three steps.

Step 1 The user generates a request package Request with userId, certificate, target url and data packaged together.

$$\{userId, data, Cert_{user}, url\} \rightarrow Request \tag{8}$$

Step 2 After receiving the user request, the server determines whether the request has permission based on the user's identity and the requested data. If the user has permission, the request is accepted, and if not, the request is rejected.

$$CC_{Identify}\,(Request) \rightarrow \begin{cases} 1 & Accept \\ 0 & Reject \end{cases} \tag{9}$$

Step 3 After accepting the request, the SDK_{Go} invokes the chaincode according to the url and data to complete the corresponding operations, and finally returns the response to the user.

$$CC(url, data) \xrightarrow{SDK_{Go}} Response \tag{10}$$

Part 4 The fourth part is the process of IoT devices interacting with the blockchain network. It is mainly for the data processing of IoT devices. This part can be divided into three steps.

Step 1 Send the deviceId, certificate, target url and data of the IoT device packaged as $Request$ to the blockchain.

$$\{deviceId, data, Cert_{device}, url\} \rightarrow Request \tag{11}$$

Step 2 Authenticate the IoT device, based on the deviceId and certificate of the device. If the authentication is passed, the access request is accepted, otherwise it is rejected.

$$CC_{Identify}\,(Request) \rightarrow \begin{cases} 1 & Accept \\ 0 & Reject \end{cases} \tag{12}$$

Step 3 After receiving the request from the device, it can be divided into two types depending on the url of the request. One is to store the device data, first encrypt the privacy data, then call the chaincode through the SDK_{Go} to store it in the blockchain, and finally get the new ledger data and transaction id. The other is to execute the request from the device and return the corresponding result.

$$Request \rightarrow \begin{cases} CC_{Insert}(Encrypt(data), deciveId, url) \xrightarrow{SDK_{Go}} \{Ledger, TxId\} \\ CC(url, data) \xrightarrow{SDK_{Go}} Response \end{cases} \tag{13}$$

Part 5 The last part is the IDS used to resist malicious attackers by monitoring the access traffic of users and IoT devices, identifying malicious access data and blocking it.

Step 1 All network traffic entering the edge server must be inspected by the IDS, and packets that are judged to be malicious traffic will be discarded, and only legitimate traffic can pass through.

$$IDS(Traffic) \rightarrow \begin{cases} Attack \rightarrow Reject \\ Legal \rightarrow Accpet \end{cases} \tag{14}$$

Step 2 When illegal traffic is denied, the type of attack is identified and the attack event is written to the security log.

$$IDS(Traffic_{Attack}) \rightarrow Log(IP_{src}, Port_{src}, IP_{dst}, Port_{dst}, Type_{Attack}) \tag{15}$$

3.5. Chaincode Design

The most important parts of BI-FERH are identity verification, data insertion and querying of data. The pseudo-code of the identity chaincode is verified by algorithm 1, and its time complexity is *O(n)*. The certificate(*cert*) and *Id* value of the user or IoT device are as input parameters. The output value *Pass* of the algorithm is a Boolean value, True means the authentication is passed and belongs to a legitimate account in the blockchain, and False means the verification fails. The function **CheckCA**(*Cert*) returns the CA of the user's certificate as *MyCA* variable. Then this *MyCA* would be checked whether it belongs to the trusted root CA(*RootCAlist*) or intermediate CA(*IntermediateCAlist*). After that, **CheckId**(*Cert*) function is used to check if the user is the MSP licensed user(*MSPlist*) and to determine if the *Id* value is the same as the one entered. Finally, the certificate is checked within the validity period by **TimeOut**(*Cert*) function. if all judgments have been passed, it means the authentication is passed.

Algorithm 2 pseudocode inserts data into the blockchain, with a time complexity of *O(n)*. Inputs include IoT device certificate (*Cert*), device ID (*deviceId*), and access data (*AccessData*). Output is transaction ID (*TxId*) or error message (*error*).

Verify device identity using **CC.Identify**(*Cert,deviceId*). If authentication fails, return -1. Check device permission with **CheckPermission**(*deviceId*). If not permitted, return -1. Parse incoming *AccessData* into *Day*, *Room*, *Time*, and *Operator* using **Split**(*AccessData*). Encrypt *Time* for user privacy using AES encryption and Base64 encoding. Package information and invoke **stub.PutState**(*Day,data*) to store data in blockchain. Returns transaction ID or error message. Output *error* if message not empty, else return *TxId*.

Storage key is *Day*, used as primary key. Using room number as key leads to inefficiencies during retrieval due to data volume growth over time.

Algorithm 1 CC.Identify(*Cert*, *Id*): Verify the legitimacy of identity

Input: *Cert*, *Id(userId or deviceId)*
Output: *Pass (bool)*

Pass ← *True*
MyCA ← **CheckCA**(*Cert*)
if *MyCA* ∉ *RootCAlist* and *MyCA* ∉ *IntermediateCAlist* **then**
    Pass ← *False*
end if
if CheckId(*Cert*) ∉ *MSPlist* **then**
    Pass ← *False*
end if
if CheckId(*Cert*)! = *userId* **then**
    Pass ← *False*
end if
if TimeOut(*Cert*) ← *True* **then**
    Pass ← *False*
end if

return *Pass*

Algorithm 2 CC.Insert(*Cert*, *deviceId*, *AccessData*): Insert data into the blockchain

Input: *Cert*, *deviceId*, *AccessData*
Output: *TxId or error*

TxId ← -1
Legal ← **CC.Identify**(*Cert*, *deviceId*)
if *Legal* == *False* **then**
    return *TxId*
end if
if CheckPermission(*deviceId*) == *False* **then**
    return *TxId*
end if
Day,Room,Time,Operator ← **Split**(*AccessData*)
encryptedT ← **AESEncrypt**(*Time*)
encodedT ← **Base64Encode**(*aes-encrypted*)
data ← {*Day, Room, encodedT, Operator*}
Txid,err ← **stub.PutState**(*Day, data*)
if *error* ! = *null* **then**
    return *error*
end if

return *Txid*

Algorithm 3 pseudocode queries blockchain data with time complexity *O(n)*. Inputs: user's credentials (*Cert*), *userId*, and *Request*. Output: hotel room access record (*Room-Record*) or error message (*error*).

Call **Identify**(*Cert,userId*) to verify identity, return *null* if successful. If verified, match *userId* to admin (*AdminList*), staff (*StaffList*), or guest (*GuestList*). Admins can view all data, staff can view staff data within chosen time range, guests can view own room data from check-in to present. Use **CheckInTime**(*roomNum*) to query target room's latest check-in time. Use **NowTime()** for current time. Call **GetHistoryForKey**(*day*) to query modification records (*raw*) for key. Extract required data by *roomNum*. Decode and decrypt time info for data restoration. If data extraction produces error, return *error*; else return *RoomRecord* based on user's identity.

Algorithm 3 CC.Query(*Cert*, *userId*, *Request*)
Query the history of blocks in the blockchain

Input: *Cert*, *userId*, *Request*
Output: *RoomRecord or error*

```
1: Legal ← CC.Identify(Cert, userId)
2: tmpList ← []
3: if Legal == False then
4:     return null
5: end if
6: if userId ∈ AdminList then
7:     StartTime, EndTime, roomNum ← Request
8:     for day in range(StartTime, EndTime) do
9:         raw, error ← GetHistoryForKey(day)
10:        if error ! = null then
11:            return error
12:        end if
13:        for d in raw do
14:            if d[room] == roomNum then
15:                tempList.append(d)
16:            end if
17:        end for
18:    end for
19: else if userId ∈ StaffList then
20:    StartTime, EndTime,roomNum ← Request
21:    for day in range(StartTime, EndTime) do
22:        raw, error ← GetHistoryForKey(day)
23:        if error ! = null then
24:            return error
25:        end if
26:        for d in raw do
27:            if d[room] == roomNum and
               d[operator] == Staff then
28:                tempList.append(d)
29:            end if
30:        end for
31:    end for
32: else if userId ∈ GuestList then
33:    roomNum ← Request
34:    StartTime ← CheckInTime(roomNum)
35:    EndTime ← NowTime()
36:    for day in range(StartTime, EndTime) do
37:        raw, err ← GetHistoryForKey(day)
38:        if error ! = null then
39:            return null
40:        end if
41:        for d in raw do
42:            if d[room] == roomNum then
43:                tempList.append(d)
44:            end if
45:        end for
46:    end for
47: end if
48: for t in tempList do
49:    tmp ← AESDecrypt(Base64Decode(t[time]))
50:    t[time] ← tmp
51: end for
52: RoomRecord ← tempList
53: return RoomRecord
```

4. Experiment

The system integrates IoT electronic door locks with blockchain, as shown in Figure 6. When the electronic door lock in the smart hotel activates, it wirelessly transmits details like room number, operator, and timestamp to the server. Network traffic undergoes attack identification via IDS, eliminating malicious traffic. The server encrypts and stores

data in the blockchain using Hyperledger Fabric. The experimental part adopts the same configuration and experimental test method as the paper[7].

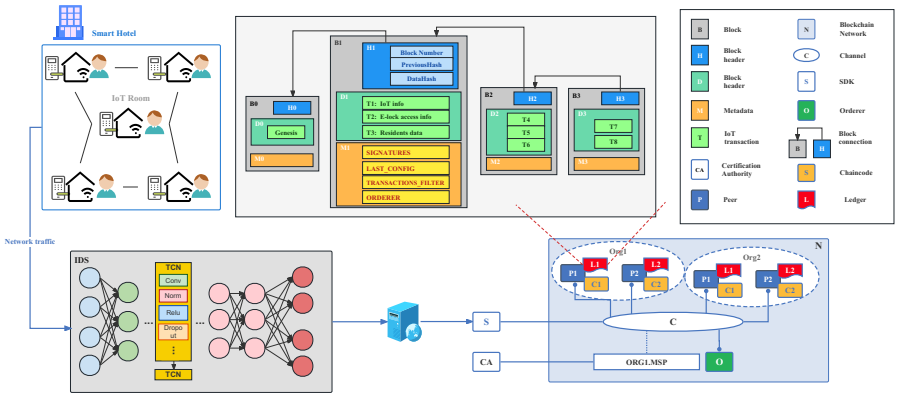


Fig. 6. Framework of this experimental system

4.1. Environment Configuration

The configuration information of the experimental environment is consistent with our previous work on the paper[7]. Refer to Table 3 for a detailed test environment overview.

4.2. System performance

The Fabric network is initialized using the SDK, involving steps such as channel creation, node addition, chaincode packaging, installation, approval, submission, and initialization. Table 4 presents time consumption for each phase in the BI-FERH system. Notably, the endorsement strategies of organizations contribute to the time-consuming approval, submission, and initialization phases. Despite this, blockchain network setup takes under 8 seconds on average, which is remarkably swift compared to manual network creation. SDK construction time is also rapid, and next, the performance of SDKs built with different programming languages will be compared.

The SDK can be developed in Go, Node.js, or Java. This paper opts for Go due to its lightweight, efficient, and parallel nature, as analyzed in [47]. The experiment compares data update and query operations times across different SDKs.In Figure 7, time spent by various SDKs for ledger update operations is depicted. Go (blue curve) stands out, offering optimal performance even with a slight increase in time as node count grows. For instance, with few nodes, Go is around 100 ms faster compared to other languages. This performance advantage amplifies with more nodes; at 16 nodes, Go is approximately 0.5s quicker than other languages.

Figure 8 illustrates query operation times across different SDKs. Queries involve only Fabric's consensus algorithm and a single-node ledger inquiry. Thus, the overall time is minimal. Notably, Java's query time increases with node count, unlike other SDKs. Go and Node.js lead with the best results, both achieving around 20ms. Java, comparatively,

Table 3. Development environment information

Hardware	
CPU	AMD R7 5800H
Hard Disk	516G
Memory	16G
Software	
OS	Windows 10 Unbutu 20.04
Docker	v20.10.7
Docker-compose	v1.29.2
Golang	v1.15.5
Platform	Hyperledger Fabric

Table 4. Time cost for initialization operations

Phase	Time(ms)
SDK generation	26
Create channel	48
Join Channel	320
Packing chaincode	670
Install chaincode	274
Organization approval	2617
Commit chaincode	2547
Initialize chaincode	2625

takes approximately 45ms. In data updates and query efficiency, the Go SDK outperforms other languages.

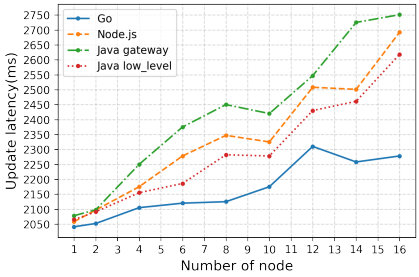


Fig. 7. Update speed of different SDK

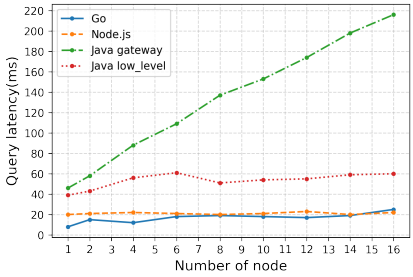


Fig. 8. Query speed of different SDK

Table 5 displays resource usage during data insertion for different nodes in the experiment. It includes node names, maximum/average CPU usage, memory consumption, and block input/output. CPU and memory consumption are highest for peer and CouchDB nodes, essential for data processing. Peers and CouchDB have average CPU usage of 13% and 20%, and memory usage around 194.5MB and 80.4MB respectively. The Orderer node, vital for consensus, consumes less CPU (7.74%) and memory (96.66MB). Chaincode and CA nodes show lower resource demands.

This system ensures efficient data processing with minimal resource utilization despite significant block input/output. Future experiments will further compare its superiority against other blockchain applications.

4.3. Comparison with traditional structure

In order to compare with the traditional database, a MySQL-IoT traditional database system with exactly the same functions as the BI-FERH system is constructed in this exper-

Table 5. Resource consumption

NAME	CPU (Max)	CPU (Avg)	MEM USAGE	BLOCK I/O
Chaincode-Peer0	1.93%	1.87%	12.05MB	9.58MB/0B
Chaincode-Peer1	2.07%	1.76%	14.18MB	9.29MB/0B
Peer0	22.76%	15.58%	171MB	16.6MB/8MB
Peer1	23.77%	10.69%	218.1MB	17.3MB/7MB
Orderer	9.87%	7.74%	94.66MB	5.8MB/8MB
Ca.Org1	0.36%	0.10%	16.56MB	20.1MB/311KB
CouchDB0	24.87%	20.01%	85.77MB	20.5MB/19.4MB
CouchDB1	25.88%	19.74%	75.14MB	5.88/20.3MB

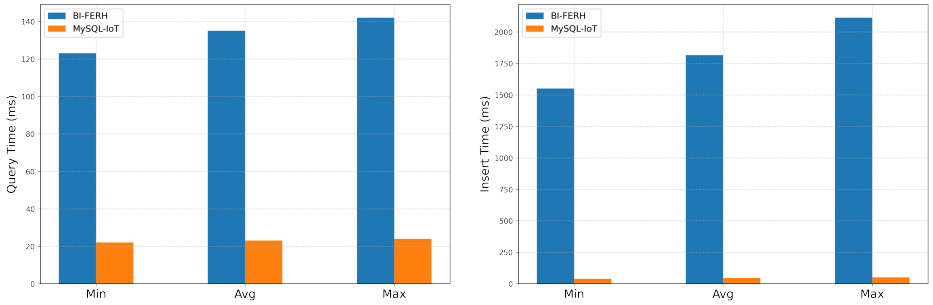
iment. Both of them have completed the same functions, such as data encryption, permission control, login management, etc., except for the different data storage structures. The experiment compares the concurrent performance and data processing efficiency of the two systems by sending requests for queries and inserting data in multiple threads. Figure 9 is the content of the inserted data, which contains the packet number, date, operator number, specific time, and room number. The experiment uses 100 threads to send packets continuously for 10 minutes to two servers, containing data queries, insertions, and updates. The total number of packets received by a single system was 289192, and the packet loss rate was 0% for both systems. By recording the data processing time of the two systems, the final test results were obtained as shown in Figure 10.

```
{
    "id": 1283,                          //Packet ID
    "Day": "2022-07-15",                 //Date
    "Operator": "Staff-cleaner-18",      //Operator number
    "Time": "1657864954",                //Specific time stamp
    "Username": 701,                     //Room number

}
```

Fig. 9. Experimental sample data

Regarding query times (Figure 10(a)), the distributed database exhibits an average query time of 135 ms (ranging from 123 to 142 ms). In comparison, the MySQL database has an average query time of 23 ms (ranging from 22 to 25 ms). Despite the disparity, this difference in response speed is minor and unlikely to significantly impact user experience, as users can obtain desired history in a single query. In terms of data insertion operations (Figure 10(b)), inserting data into the distributed database takes an average of 1815 ms (ranging from 1550 to 2113 ms), whereas the MySQL database averages 45 ms (ranging from 39 to 51 ms). The noticeable gap in data insertion time is expected, considering the inherent complexity of blockchain data insertion steps. However, according to research from the Encyclopedia of Software Testing Technology, most users accept response times under 4 seconds. Thus, the 2-second insertion time remains well within acceptable limits. Moreover, data insertion doesn't involve active user interaction; it's simply the IoT device submitting data to the server for processing. Overall, the system's feasibility is promising.



(a) Query time comparison (b) Insert time comparsion

Fig. 10. Comparison with MySQL-IoT

Hyperledger Fabric offers quicker consensus compared to traditional blockchain systems using proof-of-work. This experiment contrasts Fabric's speed with proof-of-work difficulties of 15 and 20 [48]. In Figure 11, Fabric's consensus outperforms proof-of-work by around 20 ms at low difficulties. As nodes increase, Fabric maintains efficiency. However, with proof-of-work's difficulty at 20, consensus time escalates significantly as nodes grow. At 100 nodes, Fabric takes about 0.2 s while proof-of-work (difficulty 20) takes 6.3 s. Fabric's consensus efficiency is thus prominent.

Table 6 provides a visual comparison between the two frameworks. MySQL-IoT boasts efficient data processing, especially in updates. Yet, its security is compromised. In contrast, BI-FERH sacrifices some efficiency for vastly improved data security. Its distributed storage structure resists DDoS attacks, while blockchain storage ensures traceability and tamper resistance. Hyperledger Fabric's chaincode enhances blockchain scalability. TLS secures data transmission. BI-FERH strengthens data security without harming business efficiency, offering a secure IoT storage solution.

Table 6. BI-FERH vs. MySQL-IoT

	BI-FERH	MySQL-IoT
Storage System	Hyperledger Fabric v2.2	MySQL v5.7.26
Storage Structure	Distributed	Centralized
DDoS Defense Capability	Strong(IDS 99.9% recognition rate)	Weak
Traceability	Strong(TxId)	Weak
Data Redundancy	Strong(Distributed Storage)	Weak(Centering)
Untamperability	Strong(Hash Chain)	Weak
Scalability	Chaincode	None
Data Packet Loss Rate	0%	0%
Data Transmission Protocol	TLS	TCP
Data Query (average time)	135ms	23ms
Data Update (average time)	1815ms	45ms

4.4. Comparison with blockchain applications

Comparing the BI-FERH system's performance with similar blockchain applications is essential. To validate its advantages over other blockchain solutions, the proposed approach is evaluated against systems from four papers: [34], [37], [38], and [39], all previously discussed in the related work section.

The experiment involved sending data insertion requests to different blockchain systems at varying rates: 50, 100, 150, 200, and 250 transactions per second. Each phase had 1000 transaction requests over five rounds, totaling 5000 transactions. This assessed concurrency performance and transaction processing speed. Resource consumption, including CPU, memory, and block I/O, was recorded for system evaluation.

Table 7 displays specific data for the five systems across phases, including rates, throughput, and processing times. Initially ranking 3rd in the first phase, the BI-FERH approach gains ground with higher sending speeds. It secures 2nd place at 100/s. In later phases, it consistently leads. At 250/s, BI-FERH achieves 125 tps throughput and 8 ms processing time. This is 28.33 tps higher and 2.79 ms faster than the 2nd place, and 77.66 tps higher with 13.12 ms saved compared to the 5th place.

Table 7. Detailed experimental data with other blockchain applications

Phrase	Transactions No.1 to 1000			Transactions No.1001 to 2000			Transactions No.2001 to 3000			Transactions No.3001 to 4000			Transactions No.4001 to 5000		
Indicators / Reference	Sending Rate	Throughput (TPS)	Average Time(ms)	Sending Rate	Throughput (TPS)	Average Time(ms)	Sending Rate	Throughput (TPS)	Average Time(ms)	Sending Rate	Throughput (TPS)	Average Time(ms)	Sending Rate	Throughput (TPS)	Average Time(ms)
BI-FERH	50/s	25.0	40	100/s	50.0	20	150/s	75.24	13.29	200/s	100.0	10.0	250/s	125.0	8.0
[34]	50/s	48.07	20.8	100/s	50.76	19.7	150/s	34.21	29.23	200/s	37.89	26.39	250/s	47.34	21.12
[37]	50/s	23.25	43	100/s	40.0	25	150/s	41.2	25	200/s	37.04	27	250/s	50.0	20
[38]	50/s	19.08	52.4	100/s	37.87	26.4	150/s	56.49	17.7	200/s	74.62	13.4	250/s	92.67	10.79
[39]	50/s	25.25	39.6	100/s	42.21	23.69	150/s	54.67	18.29	200/s	63.49	15.75	250/s	70.82	14.12

Figure 12 displays a line graph illustrating the time required by the five systems to process 5000 data points. The x-axis represents phases with increasing data sending speeds. The blue curve, representing the BI-FERH method, completes the task in the shortest time. Observing the graph, the blue curve initially takes the second longest time, then reverses this trend after 3000 transactions, outperforming other methods. Changes in slope reflect concurrency and data processing capabilities. BI-FERH excels in handling large data volumes and maintaining good concurrency, avoiding transaction blockages and ensuring efficient data processing.

Figure 13 presents a bar chart depicting resource usage and block data volume for the five blockchain systems during transaction processing. The chart illustrates CPU and memory consumption for peer and orderer nodes. Figure 13(a) and (b) reveal CPU and memory consumption across the five systems. The BI-FERH system (blue) exhibits lower resource consumption than the other applications, with slightly more memory usage in peer nodes compared to the application represented by purple. Notably, BI-FERH consumes minimal CPU resources across all nodes. Efficient resource usage is crucial for maintaining overall computer performance. Although blockchain systems inherently demand additional computational resources due to distributed data storage and security-enhancing operations, a good system should minimize resource usage while ensuring efficient performance. In this context, the BI-FERH system excels by consuming fewer resources than other applications.

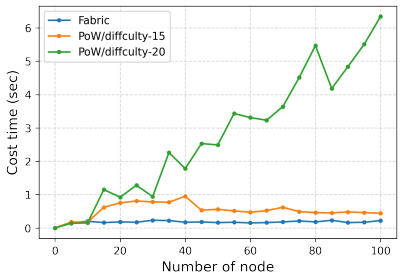


Fig. 11. Consensus consumption for Fabric and PoW

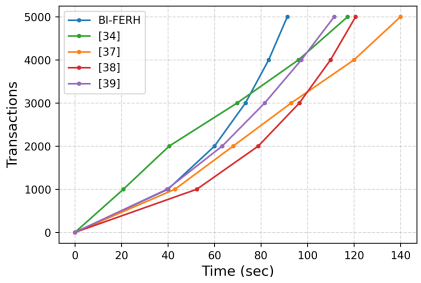


Fig. 12. Transaction speed comparison

Figure 13(c) and (d) display block data volume in the five blockchain networks. Larger volumes signify more content and efficient data transmission. BI-FERH (blue) consistently shows higher data volume than other apps, except in orderer node's block output. Overall, BI-FERH excels in block data volume, implying its network handles data efficiently.

The experiment compares BI-FERH across various dimensions, emphasizing its transaction speed, resource usage, and block data volume advantages.

4.5. Attack traffic identification

The experiment employs the CICIDS2017 dataset [49], encompassing diverse network attacks like DDoS, DoS, brute force, and web attacks. This dataset comprises 14 attack types and normal access traffic, as outlined in Table 8. The data is divided into 70% training and 30% testing sets. The BI-FERH framework utilizes an Autoencoder + TCN model. For comparison with other contemporary studies, experiments are also performed on the CICIDS2017 dataset using Autoencoder + LSTM and Autoencoder + BRNN models.

To measure the accuracy of different models, the following four common machine learning evaluation metrics are used for the experiments.

Accuracy: Accuracy is the ratio of correctly predicted traffic to all traffic.

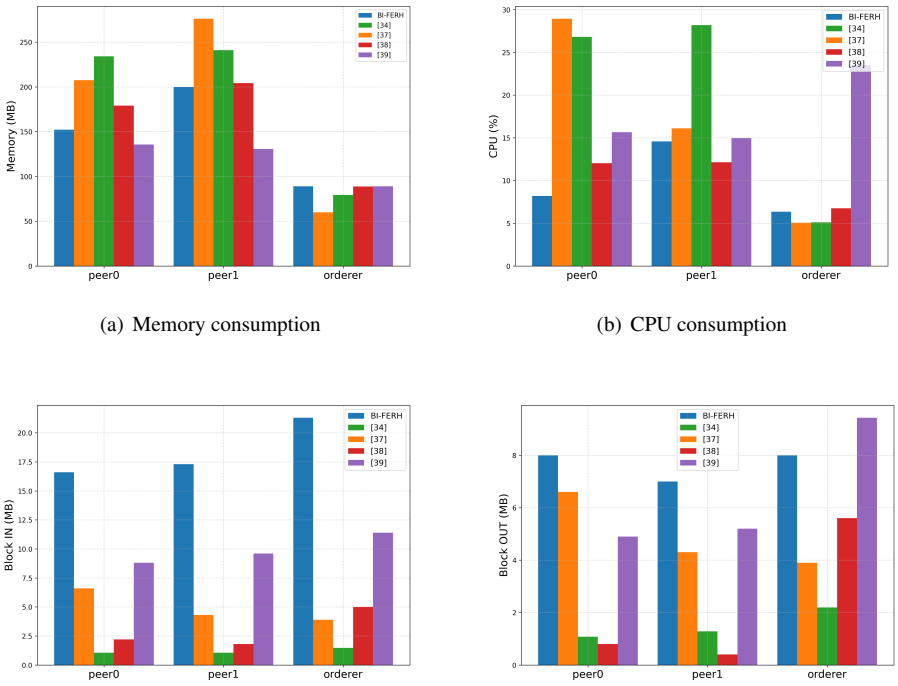
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{16}$$

Precision (PR): the ratio of the number of correctly classified positive samples to the number of samples determined to be positive by the classifier.

$$Precision = \frac{TP}{TP + FP} \tag{17}$$

Recall (RE): The ratio of the number of correctly classified positive samples to the number of true positive samples.

$$Recall = \frac{TP}{TP + FN} \tag{18}$$



(a) Memory consumption

(b) CPU consumption

(c) Block Input

(d) Block Output

Fig. 13. Resource consumption and performance

Table 8. Composition of CICIDS2017 dataset

Type	Total	Percent	Training Set	Percent	Test Set	Percent
DoS Hulk	230124	8.14%	161087	5.70%	69037	2.44%
DDoS	128025	4.53%	89618	3.17%	38408	1.36%
DoS GoldenEye	10293	0.36%	7205	0.26%	3088	0.11%
DoS slowloris	5796	0.21%	4057	0.14%	1739	0.06%
DoS Slowhttptest	5499	0.19%	3849	0.14%	1650	0.06%
PortScan	158804	5.62%	111163	3.93%	47641	1.69%
FTP-Patator	7935	0.28%	5555	0.20%	2381	0.08%
SSH-Patator	5897	0.21%	4128	0.15%	1769	0.06%
Bot	1956	0.07%	1369	0.05%	587	0.02%
Web Attack Brute Force	1507	0.05%	1055	0.04%	452	0.02%
Web Attack XSS	652	0.02%	456	0.02%	196	0.01%
Infiltration	36	0.00%	25	0.00%	11	0.00%
Web Attack Sql Injection	21	0.00%	15	0.00%	6	0.00%
Heartbleed	11	0.00%	8	0.00%	3	0.00%
BENIGN	2271320	80.32%	1589923	56.22%	681395	24.10%
Total	2827874	100%	1979512	70%	848362	30%

F1 Score: F1 Score is the balanced average of recall and precision.

$$F1 = \frac{2 * (Recall * Precision)}{Recall + Precision} \tag{19}$$

Table 9 shows the training and the final accuracy of the three models. From the table, it can be analyzed that the training time spent by the Autoencoder +TCN model is shorter, but there is not much difference between the accuracy rates of the three models. The overall results are all relatively good.

Table 9. Training situation

	Train Parameters	epoch	Batch Size	Training Time	Accuracy
Autoencoder + TCN	140320	10	512	25min	99.90%
Autoencoder + LSTM	3204688	5	512	42min	99.40%
Autoencoder + BRNN	2517822	5	512	183min	99.20%

Table 10 displays attack identification results using Precision, Recall, and F1 scores across the three models. Figure 14 visualizes F1 scores for 15 classification cases. The Autoencoder + TCN model achieves F1 scores of 0.97 or higher in 9 cases, showing its superior performance. All models accurately identify normal network traffic. In scenarios with limited data like Bot and web attacks, F1 values, while not reaching 0.97, still surpass those of other models. Increasing training data could enhance recognition capability in these scenarios.

Table 10. Attack scenario identification

CICIDS2017	**Autoencoder + TCN**			**Autoencoder + LSTM**			**Autoencoder + BRNN**		
	PR	**RE**	**F1**	**PR**	**RE**	**F1**	**PR**	**RE**	**F1**
DoS Hulk	0.999	1.0	0.998	0.991	0.996	0.993	0.972	0.998	0.986
DDoS	1.0	0.998	0.999	0.995	0.987	0.991	0.997	0.991	0.993
DoS GoldenEye	0.985	0.991	0.988	0.965	0.736	0.835	0.804	0.867	0.834
DoS slowloris	0.971	0.971	0.971	0.772	0.347	0.479	0.816	0.656	0.727
DoS Slowhttptest	0.965	0.982	0.974	0.507	0.572	0.537	0.766	0.632	0.691
PortScan	0.994	0.998	0.996	0.992	0.997	0.994	0.992	0.995	0.994
FTP-Patator	0.998	0.993	0.996	0.937	0.995	0.965	0.876	0.958	0.915
SSH-Patator	0.975	0.988	0.981	0.665	0.988	0.795	0.912	0.474	0.624
Bot	0.979	0.726	0.834	0	0	0	0	0	0
Web Attack Brute Force	0.669	0.858	0.752	0	0	0	0	0	0
Web Attack XSS	0.500	0.015	0.30	0	0	0	0	0	0
Infiltration	0.308	0.727	0.432	0	0	0	0	0	0
Web Attack Sql Injection	0.332	0.500	0.400	0	0	0	0	0	0
Heartbleed	0	0	0	0	0	0	0	0	0
Benigh	1.0	1.0	1.0	0.996	0.998	0.997	0.997	0.998	0.998

5. Results and discussions

Table 11 offers a comparison between BI-FERH and other blockchain applications across five categories: blockchain type, platform, IoT device, consensus algorithm, and data algo-

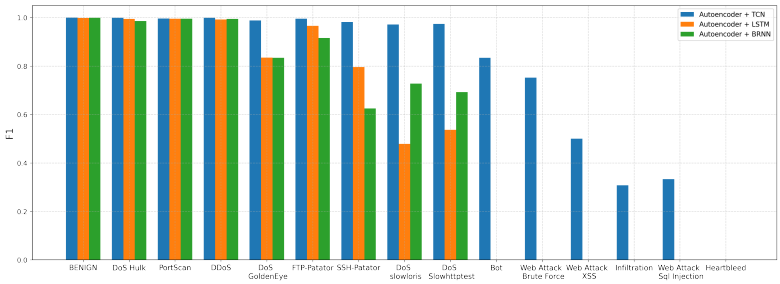


Fig. 14. F1 values for different models of attack identification

rithm. BI-FERH's choice of consortium and private blockchains aligns with non-currency applications. Hyperledger Fabric as the platform provides scalability and performance advantages over other platforms like Indy and Composer. Kafka consensus suits the hotel application's cluster environment, enhancing efficiency. PBFT's node trust issues don't apply here. BI-FERH enhances data protection via AES encryption and Base64 encoding, contrasting with other applications that lack data processing, except for watermarking in one case.

Table 11. Comparison of related applications

Reference	Description	Blockchain type	Platform	IoT	Consensus	Data algorithms
BI-FERH	Smart hotel info management system	Consortium blockchains	Hyperledger Fabric	√	Kafka	AES128, Base64
[11]	Mobility data transactions	Consortium blockchains	Hyperledger Indy	×	PBFT	×
[34]	Transparent supply chain for coffee	Consortium blockchains	Hyperledger Fabric	√	RAFT	×
[35]	Tracking sources of fake news	Private blockchain	BloXroute	×	PBFT	ARX Digital Watermark
[36]	Security system for smart home	Private blockchains	Hyperledger Composer	√	PoW	×
[37]	Electronic healthcare record system	Consortium blockchains	Hyperledger Composer	×	PBFT	×
[38]	Smart Hospital patient data detection	Consortium blockchains	Hyperledger Composer	√	Solo	×
[39]	Smart city safety transportation System	Consortium blockchains	Hyperledger Fabric	√	Solo	×
[40]	IoT decentralized voting system	Public blockchains	Solidity	√	PoW	×

5.1. Availability

The experiments, focusing on system performance and comparison with traditional storage structures, highlight BI-FERH's usability. The Fabric SDK plays a key role in node operation and data processing. Network initialization and construction took just 9,127ms. Comparison of SDKs in different languages favored Go for data updates and queries, affirming its suitability. Assessing traditional centralized storage versus BI-FERH's blockchain

approach aimed to gauge efficiency trade-offs. Results indicate that the efficiency sacrificed for decentralized storage through blockchain remains acceptable for user functionality.

5.2. Performance

The experiment's third segment compares the consensus algorithms' pivotal role in achieving data consensus among nodes, contrasting Fabric and PoW. In the fourth part, BIFERH's performance is gauged against other blockchain systems across five dimensions: transaction speed, memory consumption, CPU usage, and block input/output data volume. Results indicate this system outperforms others with 1.3x faster transaction speed, 10% lower memory consumption, 7% lower CPU usage, and increased block data input/output by 14MB and 3.8MB respectively. Overall, the system exhibits heightened transaction speed, improved concurrency, reduced memory and CPU consumption, and enhanced data transfer efficiency.

5.3. Security

Data Integrity Arthur Gervais et al.'s research on blockchain security [50] examines the potential for data tampering. If an attacker aims to alter a block before the latest one (h-th block), they must modify the prior block's hash and recalculate subsequent hashes to create a new longest chain. Assuming honest node hash computation speed as p times/s and the attacker's speed as q times/s, the hash's difficulty requires the first g binary digits to be 0. When no new node joins, an honest node's probability of acquiring a block is $p/2^g$, and the attacker's is $q/2^g$. Initially, differing nodes between the honest node and attacker are $z_0 = h$. Subsequently, z_{i+1} possibilities are: z_i+1, z_i-1, z_i with probabilities P_1, P_2, P_3, corresponding to events X_1, X_2, X_3.

$$z_{i+1} = \begin{cases} z_i + 1, & P_1 = \frac{p}{2^g}(1 - \frac{q}{2^g}) \\ z_i - 1, & P_2 = \frac{q}{2^g}(1 - \frac{p}{2^g}) \\ z_i, & P_3 = 1 - P_1 - P_2 \end{cases} \tag{20}$$

When $z_i+1 = -1$, it means that the attacker successfully tampered with the blockchain data. Within t seconds, there will be t times of changes in the number of nodes apart. Let n be the number of occurrences of X_1; when tampering is successful, X_2 will occur at least $(n + h + 1)$ times, Let j be the difference between the actual number of occurrences of event X_2 and the minimum number of occurrences, then the actual number of occurrences of X_2 is $(n+h+1+j)$, and the number of occurrences of event X_3 is $(t-2n-h-1-j)$, where $n \in [0, (t-1-h)/2], j \in [0, t-2n-h-1]$. Within t seconds, the probability of an attacker successfully tampering with blockchain data is

$$P_t(h) = \sum_0^{n_{max}} \sum_0^{j_{max}} \left(\frac{2!}{n!(h+n+1+j)!(t-2n-h-1-j)!} \cdot P_1^n P_2^{h+n+1+j} P_3^{t-2n-h-1-j}\right) \tag{21}$$

The final analysis shows that the success probability of the attacker decreases as the depth h of the tampered block increases and the probability of successful tampering increases as the attacker increases. If an attacker wants to tamper with the data of a specific

ledger on a local node, it requires the attacker to have the ability to forge signatures and modify the hash values of blocks on all nodes. So blockchain can prevent data tampering and makes data integrity guaranteed.

Signature forgery The external attacker's attempt to forge a trader's signature in the block involves brute force cracking to guess the private key, excluding key theft. The ECDSA elliptic curve digital signature algorithm's private key lengths of 160, 224, and 256 bits result in cracking times of 10^{12}, 10^{24}, and 10^{28} seconds, respectively, using a computer with one million instructions per second. The private key utilized in this system is 256 bits, requiring significant time and computational resources for cracking. Even with a high-cost professional machine, it takes a month to calculate the discrete logarithm of an elliptic curve with a prime order of 2^{120}. Consequently, elliptic curve algorithms with large prime orders substantially reduce the feasibility of brute force attacks, rendering signature forgery infeasible for attackers.

DDOS Defense Distributed Denial of Service (DDOS) attacks involve multiple attackers targeting multiple targets simultaneously, overloading servers to induce network paralysis and disrupt regular services. Traditionally, hotel industry information systems relied on centralized servers for data computation, storage, and business access. Any server attack would lead to widespread operational halts. In contrast, the BI-FERH framework leverages a distributed blockchain network, decentralizing computing and data resources. This decentralized structure poses challenges for DDOS attacks, and the framework employs an Intrusion Detection System (IDS) layer to pre-filter malicious access before processing. Experimental findings indicate the classification model accurately detects DDOS attack scenarios with a 99.9% accuracy rate. The IDS successfully identifies 8 distinct attack types with over 97% probability. This enhances the resilience of hotel information systems against external attacks.

6. Conclusion

This paper introduces the BI-FERH security framework for smart hotels, leveraging blockchain, IoT, and machine learning to establish a robust data management system. The framework's composition and technologies are explained, showcasing its potential to enhance security in the hotel industry. Through experiments, an IoT-based information management system for hotel door locks is developed, demonstrating its performance, efficiency, and security. The machine learning-driven IDS accurately identifies various attack scenarios in the CICIDS2017 dataset. In conclusion, this work provides a pioneering solution to address security challenges in the hotel industry's information systems.

Acknowledgments. This work is supported by the Science and Technology Planning Project of Guangdong (2021B0101420003, 2020B0909030005, 2020B1212030003, 2023ZZ03, 2020ZDZX3013), the Science and Technology Planning Project of Guangzhou(202206030007), Key Laboratory of Smart Education of Guangdong Higher Education Institutes, Jinan University(2022LSYS003), Guangdong Key Laboratory of Data Securit and Privacy Preserving (2017B030301004), the Opening Project of Key Laboratory of Safety of Intelligent Robots for State Market Regulation (GQI-

KFKT202205) and the High-Performance Public Computing Service Platform of Jinan University, NSF of Guangdong Province (No. 2021A1515011906).

References

1. Yang, H., Song, H., Cheung, C., Guan, J.: How to enhance hotel guests' acceptance and experience of smart hotel technology: An examination of visiting intentions. International journal of hospitality management 97, 103000 (2021)
2. Nakanishi, J., Baba, J., Kuramoto, I., Ogawa, K., Yoshikawa, Y., Ishiguro, H.: Smart speaker vs. social robot in a case of hotel room. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 11391–11396 (2020)
3. Shen, X., Shao, W., Zhang, Z., Xu, P.: Hotel intelligent guidance system based on zigbee technology. Microprocessors and Microsystems 77, 103160 (2020)
4. Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P., Sikdar, B.: A survey on iot security: application areas, security threats, and solution architectures. IEEE Access 7, 82721–82743 (2019)
5. Sahu, A.K., Gutub, A.: Improving grayscale steganography to protect personal information disclosure within hotel services. Multimedia Tools and Applications pp. 1–21 (2022)
6. Seamans, E.: The unique challenges of data security in the hospitality industry (2018), https://hospitalitylawyer.com/the-unique-challenges-of-data-security-in-the-hospitality-industry
7. Guan, Q., Lei, J., Wang, C., Geng, G., Zhong, Y., Fang, L., Huang, X., Luo, W.: A practical framework of blockchain in iot information management. In: 2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (2023)
8. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review p. 21260 (2008)
9. Wang, Y., Singgih, M., Wang, J., Rit, M.: Making sense of blockchain technology: How will it transform supply chains? International Journal of Production Economics 211, 221–236 (2019)
10. Maesa, D.D.F., Mori, P.: Blockchain 3.0 applications survey. Journal of Parallel and Distributed Computing 138, 99–114 (2020)
11. Lopez, D., Farooq, B.: A multi-layered blockchain framework for smart mobility data-markets. Transportation Research Part C: Emerging Technologies 111, 588–615 (2020)
12. Ali, J., Ali, T., Alsaawy, Y., Khalid, A.S., Musa, S.: Blockchain-based smart-iot trust zone measurement architecture. In: Proceedings of the International Conference on Omni-Layer Intelligent Systems. pp. 152–157 (2019)
13. Feng, J., Zhao, X., Chen, K., Zhao, F., Zhang, G.: Towards random-honest miners selection and multi-blocks creation: Proof-of-negotiation consensus mechanism in blockchain networks. Future Generation Computer Systems 105, 248–258 (2020)
14. Kamalov, F., Gheisari, M., Liu, Y., Feylizadeh, M.R., Moussa, S.: Critical controlling for the network security and privacy based on blockchain technology: A fuzzy dematel approach. Sustainability 15(13), 10068 (2023)
15. Albulayhi, A.S., Alsukayti, I.S.: A blockchain-centric iot architecture for effective smart contract-based management of iot data communications. Electronics 12(12), 2564 (2023)
16. Donet Donet, J.A., Pérez-Solà, C., Herrera-Joancomartí, J.: The bitcoin p2p network. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) Financial Cryptography and Data Security. pp. 87–102 (2014)
17. Biryukov, A., Khovratovich, D., Pustogarov, I.: Deanonymisation of clients in bitcoin p2p network. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. p. 15–29. CCS '14 (2014)

18. Zou, W., Lo, D., Kochhar, P.S., Le, X.B.D., Xia, X., Feng, Y., Chen, Z., Xu, B.: Smart contract development: Challenges and opportunities. IEEE Transactions on Software Engineering 47(10), 2084–2106 (2021)
19. Wohrer, M., Zdun, U.: Smart contracts: security patterns in the ethereum ecosystem and solidity. In: 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE). pp. 2–8 (2018)
20. Sousa, J., Bessani, A., Vukolic, M.: A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In: 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). pp. 51–58 (2018)
21. Hao, Y., Li, Y., Dong, X., Fang, L., Chen, P.: Performance analysis of consensus algorithm in private blockchain. In: 2018 IEEE Intelligent Vehicles Symposium (IV). pp. 280–285 (2018)
22. Lee, J.W., Park, S.: A study on performance improvement of hyperledger fabric through batched chaincode message. In: 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS). pp. 259–262 (2020)
23. Javaid, H., Yang, J., Santoso, N., Upadhyay, M., Mohan, S., Hu, C., Brebner, G.: Blockchain machine: A network-attached hardware accelerator for hyperledger fabric. In: 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS). pp. 258–268 (2022)
24. Dinh, T.T.A., Wang, J., Chen, G., Liu, R., Ooi, B.C., Tan, K.L.: Blockbench: A framework for analyzing private blockchains. In: Proceedings of the 2017 ACM international conference on management of data. pp. 1085–1100 (2017)
25. Pongnumkul, S., Siripanpornchana, C., Thajchayapong, S.: Performance analysis of private blockchain platforms in varying workloads. In: 2017 26th International Conference on Computer Communication and Networks (ICCCN). pp. 1–6 (2017)
26. Da Xu, L., He, W., Li, S.: Internet of things in industries: A survey. IEEE Transactions on industrial informatics 10(4), 2233–2243 (2014)
27. Liu, B., Jiang, X., He, X., Qi, L., Xu, X., Wang, X., Dou, W.: A deep learning-based edge caching optimization method for cost-driven planning process over iiot. Journal of Parallel and Distributed Computing 168, 80–89 (2022)
28. Pavithran, D., Al-Karaki, J.N., Shaalan, K.: Edge-based blockchain architecture for event-driven iot using hierarchical identity based encryption. Information Processing & Management 58(3), 102528 (2021)
29. Du, Y., Wang, Z., Li, J., Shi, L., Jayakody, D.N.K., Chen, Q., Chen, W., Han, Z.: Blockchain-aided edge computing market: Smart contract and consensus mechanisms. IEEE Transactions on Mobile Computing (2022)
30. Reyna, A., Martín, C., Chen, J., Soler, E., Díaz, M.: On blockchain and its integration with iot. challenges and opportunities. Future generation computer systems 88, 173–190 (2018)
31. Kshetri, N.: 1 blockchain's roles in meeting key supply chain management objectives. International Journal of information management 39, 80–89 (2018)
32. Khan, M.A., Salah, K.: Iot security: Review, blockchain solutions, and open challenges. Future generation computer systems 82, 395–411 (2018)
33. Wan, J., Li, J., Imran, M., Li, D., et al.: A blockchain-based solution for enhancing security and privacy in smart factory. IEEE Transactions on Industrial Informatics 15(6), 3652–3660 (2019)
34. Ravi, D., Ramachandran, S., Vignesh, R., Falmari, V.R., Brindha, M.: Privacy preserving transparent supply chain management through hyperledger fabric. Blockchain: Research and Applications 3(2), 100072 (2022)
35. Dwivedi, A.D., Singh, R., Dhall, S., Srivastava, G., Pal, S.K.: Tracing the source of fake news using a scalable blockchain distributed network. In: 2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS). pp. 38–43 (2020)
36. Ammi, M., Alarabi, S., Benkhelifa, E.: Customized blockchain-based architecture for secure smart home for lightweight iot. Information Processing & Management 58(3), 102482 (2021)

37. Tanwar, S., Parekh, K., Evans, R.: Blockchain-based electronic healthcare record system for healthcare 4.0 applications. Journal of Information Security and Applications 50, 102407 (2020)
38. Jamil, F., Ahmad, S., Iqbal, N., Kim, D.H.: Towards a remote monitoring of patient vital signs based on iot-based blockchain integrity management platforms in smart hospitals. Sensors 20(8), 2195 (2020)
39. Abbas, K., Tawalbeh, L.A., Rafiq, A., Muthanna, A., Elgendy, I.A., El-Latif, A., Ahmed, A.: Convergence of blockchain and iot for secure transportation systems in smart cities. Security and Communication Networks 2021 (2021)
40. Li, Y., Susilo, W., Yang, G., Yu, Y., Liu, D., Du, X., Guizani, M.: A blockchain-based self-tallying voting protocol in decentralized iot. IEEE Transactions on Dependable and Secure Computing 19(1), 119–130 (2022)
41. Chen, F., Wang, J., Jiang, C., Xiang, T., Yang, Y.: Blockchain based non-repudiable iot data trading: Simpler, faster, and cheaper. In: IEEE INFOCOM 2022 - IEEE Conference on Computer Communications. pp. 1958–1967 (2022)
42. Lao, L., Li, Z., Hou, S., Xiao, B., Guo, S., Yang, Y.: A survey of iot applications in blockchain systems: Architecture, consensus, and traffic modeling. ACM Computing Surveys 53(1), 1–32 (2020)
43. Alshudukhi, K.S., Khemakhem, M.A., Eassa, F.E., Jambi, K.M.: An interoperable blockchain security frameworks based on microservices and smart contract in iot environment. Electronics 12(3), 776 (2023)
44. Daidone, F., Carminati, B., Ferrari, E.: Blockchain-based privacy enforcement in the iot domain. IEEE Transactions on Dependable and Secure Computing 19(6), 3887–3898 (2022)
45. Zhao, Q., Chen, S., Liu, Z., Baker, T., Zhang, Y.: Blockchain-based privacy-preserving remote data integrity checking scheme for iot information systems. Information Processing & Management 57(6), 102355 (2020)
46. Martinez-Rendon, C., González-Compeán, J., Sánchez-Gallegos, D.D., Carretero, J.: Cd/cv: Blockchain-based schemes for continuous verifiability and traceability of iot data for edge–fog–cloud. Information Processing & Management 60(1), 103155 (2023)
47. Foschini, L., Gavagna, A., Martuscelli, G., Montanari, R.: Hyperledger fabric blockchain: Chaincode performance analysis. In: ICC 2020 - 2020 IEEE International Conference on Communications (ICC). pp. 1–6 (2020)
48. Liu, H., Han, D., Li, D.: Fabric-iot: A blockchain-based access control system in iot. IEEE Access 8, 18207–18218 (2020)
49. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. Proceedings of the 4th International Conference on Information Systems Security and Privacy pp. 108–116 (2018)
50. Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. pp. 3–16 (2016)

Quanlong Guan received the M.S. and Ph.D. degrees from Jinan University, China, in 2006 and 2014, respectively, where he is currently a Professor of engineering. He is directing the Guangdong Research and Development Institute for the big data of service and application on education. His research has been funded by the National Natural Science Foundation of China and the Guangdong Key Technologies Research and Development Program of China. His research interests include network security, big data protection and processing, information system, big data application, and mobile security.

Jiawei lei received the B.S. degree in information security from Guangzhou University, Guangzhou, China in 2020. He is currently a graduate student with the College of Cyber Security, Jinan University, Guangzhou, China. His research interests include information system, data security, and blockchain.

Chaonan Wang is currently a full professor in the College of Cyber Security of Jinan University in China. She has been working in the area of system reliability engineering since she joined the graduate program in 2010 at the University of Massachusetts (UMass) Dartmouth, where she received her PhD degree in 2014. During 2014 to 2016, she continued her research as a Postdoctoral Research Associate at UMass Dartmouth. Her research focuses on reliability analysis, information system and performance evaluation of complex systems and networks. She is the recipient of "Shanghai Eastern Scholar" reward (2015). She has served as committee member for ORSC Reliability Society and Youth Academic Committee of Guangdong Computer Society.

Guanggang Geng received the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2008. He is currently a Professor with the College of Cyber Security or College of Information Science and Technology, Jinan University, Guangzhou, China. His current research interests include machine learning, information system, computer networking, anti-fraud, and web search.

Yuansheng Zhong is currently the deputy director of Electronics and Electromagnetic Compatibility Testing Lab of Guangdong Testing Institute of Product Quality Supervision. He is a senior engineer and a member of the C branch of the National Technical Committee for Electromagnetic Compatibility Standardization. He is engaged in the inspection and testing, certification, standard preparation and revision and scientific research in the fields of electronic products, intelligent robots, electromagnetic compatibility, information system, etc.

Liangda Fang received the B.Sc. degree from Guangzhou University, Guangzhou, China, the M.Sc. degree from the Guangdong University of Technology, Guangzhou, and the Ph.D. degree from Sun Yat-sen University, Guangzhou, all in computer science, in 2007, 2010, and 2015, respectively, He is currently an Assistant Professor with the Department of Computer Science, Jinan University, Guangzhou. His current research interests include artificial intelligence, information system and algorithmic learning theory.

Xiujie Huang received the M.Sc. degree in applied mathematics and the Ph.D. degree in communication and information system from Sun Yat-sen University, Guangzhou, China, in 2006 and 2012, respectively. She was a Post-Doctoral Fellow with the Department of Electrical Engineering, University of Hawaii at Manoa, Honolulu, HI, USA, from July 2012 to October 2013. Since November 2013, she has been with Jinan University, Guangzhou. Her research interests include information theory, information security and their applications in digital communications, IoT, IoV and storage systems.

Weiqi Luo received the B.S. and M.S. degrees from Jinan University, Guangzhou, China, in 1982 and 1985, respectively, and the Ph.D. degree from South China University of

Technology, Guangzhou, in 1999. He is currently a Professor with the School of Information Science and Technology, Jinan Univer- sity. He has published more than 100 high-quality papers in international journals and conferences. His research interests include network security, big data, information system and artificial intelligence.

Received: April 01, 2023; Accepted: August 20, 2023.