A Blockchain-based MQTT Protocol Optimization Algorithm

Wei Gao[1], Lixia Zhang[2] and Yun Ju[3,*]

[1]*State Grid Shanxi Electric Power Company, Taiyuan 030001, China*
[2]*State Grid Shanxi Province Electric Power Company Information and Communication Branch, Taiyuan 030001, China*
[3]*North China Electric Power University, Beijing, China*
E-mail: juyun1982@ncepu.edu.cn
[*]*Corresponding Author*

Received 27 October 2022; Accepted 22 November 2022;
Publication 08 May 2023

Abstract

The communication protocol is an important support to realize the communication between equipment and Internet. And it covers all aspects of the IoT (Internet of things) system. To address the security problem of forging or tampering of key data in traditional IoT protocols, this paper designs an improved MQTT (message queue telemetry transmission) protocol that uses blockchain technology to ensure the security of transmitted data in the process of data transmission. Because the information in the blockchain is not tamperable, which in turn ensures that data stored in brokers are not maliciously tampered with. Through simulation experiments, it is proved that this scheme is lightweight, efficient and easy to implement, which helps to protect the security of IoT data.

Keywords: MQTT protocol, blockchain, message proxy server, IoT, security.

Journal of ICT Standardization, Vol. 11_2, 135–156.
doi: 10.13052/jicts2245-800X.1122

1 Introduction

In recent years, IoT technology [1] has a wide application especially in the industrial field [2–4]. And a large number of sensors, controllers and other infrastructures are interconnected through the network to successfully connect the physical world with the information world. It makes the IoT technology develop deeply. The most widely used communication protocol in IoT technology is MQTT (Message Queue Telemetry Transport Protocol). It is a standard application layer protocol with low complexity, low power consumption, low occupancy implementation and low overhead [5]. And it is easy to implement. Currently, the MQTT protocol has a wide application in resource-constrained power IoT environments to enable heterogeneous communication of distributed grid devices such as sensors.

However, the MQTT protocol only provides a very simple security model. And the protocol itself does not have any built-in security scheme other than the username password authentication method. While the number of edge access users is increasing, it is facing serious problems in terms of security and privacy [6]. At present, there are some schemes to study the security of MQTT protocol, which mainly include the secure transmission of data [7], the access control authority of topic [8] and the security of data itself [9].

In terms of data secure transmission, the most commonly used solution is the scheme based on SSL/TLS (Secure Socket Layer/Transport Layer Security) protocol [10] in the industry. This scheme embeds SSL/TLS protocol between MQTT protocol and TCP protocol to encrypt the data transmission process to ensure the security of the data transmission process. Although this scheme brings high security to data transmission and communication, most of the terminal devices are resource-constrained in the communication environment of the Internet of Things. On this basis, embedding SSL/TLS protocol will bring extra overhead. It is not friendly to resource-constrained terminal devices. Subsequently, some scholars propose lightweight SSL/TLS schemes. The most representative one is AugPAKE (Aug-password-authenticated key agreement) protocol scheme [11]. It puts client and server authentication in SSL/TLS offline, followed by online implementation. In the literature [12], AugPAKE protocol and HOTP (An HMAC-Based One-Time Password Algorithm) policy are used in combination to ensure high security of data during transmission. Although this scheme ensures the security of data transmission, it increases the consumption of time and computational resources when more and more publishers and subscribers enter the system, which greatly reduces the efficiency of communication.

In terms of topic access control authority, literature [13] proposes a scheme based on static authorization codes, i.e., publishers generate static authorization codes for the corresponding topics, and subscribers request the authorization codes to ensure the subscription of the corresponding topics to achieve the access control of the topics and ensure the security of their privacy. The literature [12] introduces OAuth2.0 protocol to achieve topic access control, and the authorization mechanism of this scheme is more flexible, but the process is more complicated. During the authentication process, too many responses can consume a lot of resources on the server. The literature [14] proposes to secure customer access control to related topics by embedding authorization codes into the device hardware, which is similar to the literature [13] and is a static authorization with a very simple authorization mechanism process, but it may not be very flexible and has the drawback of poor real-time performance. The literature [15] introduces HTTP (HyperText Transfer Protocol) to achieve real-time authorization, which is a more flexible authorization mechanism, but this scheme also has certain problems due to the mixture of two protocols, MQTT and HTTP, and security issues regarding the HTTP protocol also need to be considered.

In terms of the security of the data itself, some lightweight encryption algorithms are mainly used to ensure the security of the data of resource-constrained end devices and the security of the data during transmission. The literature [16] proposes a symmetric encryption scheme based on hash and heterogeneous implementation, which is friendly to resource-constrained end devices with less overhead in communication, computation and storage, but the symmetric encryption scheme is not very secure. Therefore, the ECC-based elliptic curve encryption asymmetry algorithm [17] is proposed again, whose security is based on the computational difficulty of using the rational points on the elliptic curve to form the discrete logarithm of the ellipse on the Abel addition group, which has the advantages of low broadband requirement, small storage space occupation and providing equivalent or higher level of security with smaller keys. However, the algorithm suffers from the drawback that the implementation of encryption and decryption operations takes longer than other mechanisms and is difficult to implement in practical engineering applications.

This paper focuses on the communication between edge devices and brokers in the IoT scenario. Based on the existing work, it designs an application layer protocol of Internet of things based on MQTT message protocol to complete process and standardized safety certification to ensure the

authenticity and verifiability of the transmitted data. The main contributions are as follows:

(1) It can realize the functions of data upload and control delivery between the side device and the broker, and realize the connection between the IoT sensing control device and the Internet;
(2) Propose a scheme to establish a blockchain network with each side device and broker as nodes, to ensure data traceability and non-tampering.

2 MQTT Communication

In the current use scenarios of MQTT protocol, there are two ways to connect to broker: direct access and gateway access, according to whether the terminal devices of the IoT have the ability to access the network.

The first is direct access. The Internet of Things terminal device itself has a communication module, with the ability to connect to the network directly. The second is gateway access. IoT terminals do not have the network access capability themselves and need to access the network through a unified gateway.

The application layer protocol of IoT studied in this paper runs between gateway and broker to realize message transmission between gateway and broker. In the case of direct access, the terminal device acts as the gateway. So the research method in this paper is applicable to the above two connection modes.

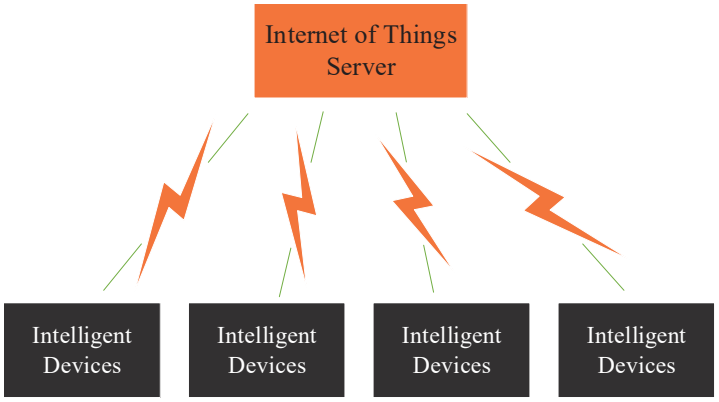

Figure 1 Direct access.

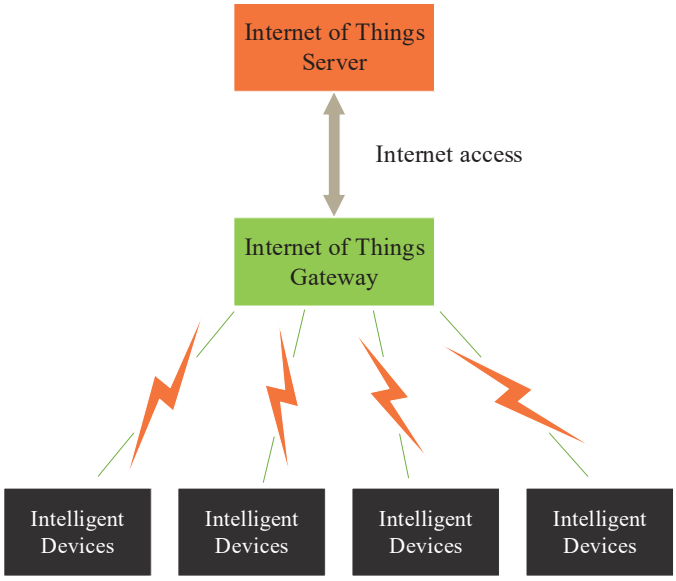

Figure 2 Gateway access.

2.1 Analysis of MQTT Communication Model

There are three roles in the MQTT protocol communication model, which are publisher, broker and subscriber. Both message publishers and message subscribers belong to the client side. Figure 3 shows the principle of its communication model.

Broker: a central server that manages MQTT messages;

Publisher: an entity that publishes messages to subscribers through a broker;

Subscriber: an entity that receives messages through an broker and can subscribe to topic messages.

Topic: published topics that follow a hierarchical structure with different levels of topics separated by slashes (e.g. region/house/consumption).

Payload: message payload, referring to the specific message content.

The construction of MQTT protocol based on TCP/IP protocol. It establishes a directed connection between the client and the broker, providing an ordered, lossless, bidirectional byte stream based transmission between the two.

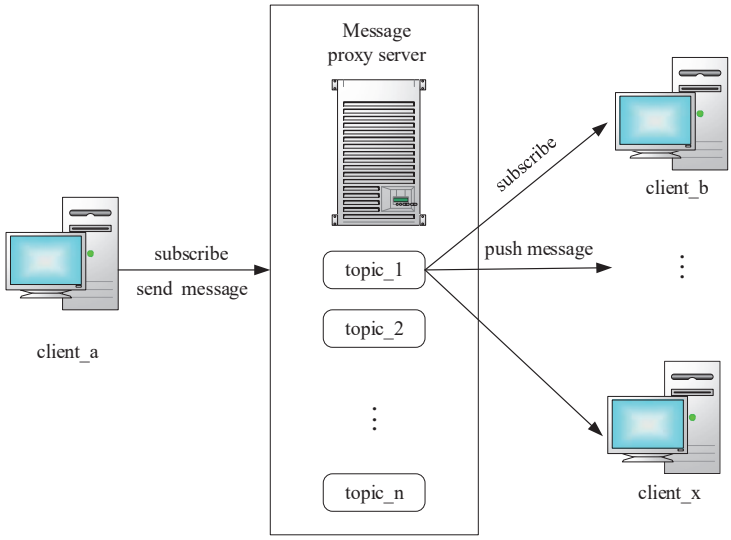

Figure 3 Communication model.

2.2 MQTT Communication Principle

MQTT protocol adopts publish/subscribe mode based on client server. Both gateways and edge devices can connect to an MQTT message broker server as clients and deliver data by subscribing to topics and publishing messages to them. Clients are connected to a broker. Broker manages topics of interest to individual clients and is responsible for forwarding messages.

2.2.1 MQTT data packet

In MQTT protocol, an MQTT packet is composed of three parts: Fixed Header, Variable Header, and Payload.

(1) Fixed Header: MQTT fixed header has two bytes. The first byte contains the MessageType, the DUP message retransmission flag bit, the RETAIN flag bit, and the QoS level flag bit. The second byte starts with the remaining length field, which is the total length of the variable packet header and the message load.
(2) Variable Header: Different types of messages have different variable headers, which mainly include the protocol name, version number, connection flag, user authorization, heartbeat time and other information.
(3) Message Body: Indicating the specific content received by the client. the message body according to the different message types encapsulate the

Table 1 MQTT message format

Bit	7	6	5	4	3	2	1	0
Byte1	Message Type				DUP	QoS		RETAIN
Byte2	Remaining Length							
……	Variable Header							
ByteM								
……	Payload							
ByteN								

information they need, and MQTT load supports a maximum of 256MB size of data.

2.2.2 MQTT protocol encryption

At present, there are some proxy server based on the MQTT protocol, such as EMQX (Erlang/Enterprise/Elastic MQTT Broker) and Mosquitto. They adopt the transmission way of SSL/TLS encryption authentication.

(1) One-way SSL/TLS authentication:The client authenticates the server, and the server does not authenticate the client

(2) Two-way SSL/TLS authentication: The client and server authenticate each other. That is, between the client and server need certificate exchange. Regardless of the encryption and authentication scheme, they all need to provide a CA-signed certificate. Installation certificates require a special administrative body to generate certificates, manage certificates, and issue them to clients. Certificates also need to be configured and their life cycle managed.

The MQTT protocol itself recommends using a combination with TLS (Transport Layer Security) encryption as an IoT communication scheme, but using TLS encryption has the following drawbacks:

(1) TLS not only occupies hardware storage space but also increases CPU usage, which is not suitable for IoT terminal devices with limited resources;

(2) The addition of TLS makes communication expensive because many bytes are spent for authentication and encryption. This is contrary to the design of MQTT itself. And making the MQTT protocol less advantageous on resource-constrained devices.

The whole process is cumbersome, which is also the reason why most clients use plaintext for data transmission in the actual application of IoT devices.

2.3 Data Management

Clients receive messages by subscribing to a specific topic, and the topic name can identify the source of the message. Gateway ID and client ID can uniquely identify an edge device, and the MQTT account can be used to identify the operational user.

A gateway serves as a medium of communication between a device and a broker, and uses data items to hold all the real-time data values of the devices it connects to. DeviceId, serviceId and num can uniquely identify a piece of data on a device. After the gateway uploads the data to the broker through the MQTT protocol, the broker can keep all the continuous historical data in the database for data analysis and other work.

To more clearly represent the meaning and function of each data item at the broker, Figure 2 shows an example of the message implementation format. In the same message can transfer multiple data to reduce the frequency of message delivery in the network.

To show the universality of the IoT system, it is necessary to abstract a universal object to represent different information uploaded by different IoT terminal devices. The different types and functions of IoT terminal devices make it very difficult for IoT devices to connect to a common system. Therefore, it is necessary to design a unified description format to facilitate the docking with the background data of the general IoT system.

Figure 4 shows the MQTT message transfer format:

The number of information attributes count is 2; Information description msg is the temperature and humidity sensor work information report: Eventtime is the time stamp. There are two elements in the data array. The first element represents the temperature information temperature $= 36°C$, and the second element represents the humidity information humidity $= 6\%$.

2.3.1 Client ID format

The client ID consists of the English string + ”/” + English string, e.g.: devices/iot.

When a client connects to the MQTT proxy server, only one connection can be online with the same client ID, so the client ID is unique.

2.3.2 Topic format for publishing subscriptions

According to the customer ID, taking devices/iot as an example, the web server side will receive the topic related to devices/iot/. So topic of MQTT message topic prefix sent by IoT device terminal should be devices/iot/.

```
{
        "devices":
        [
                {
                        "deviceId":"D3343134IE3B",
                        #"deviceId" of gateway sub device
                        "services":
                        [
                                {
                                        "code":200,
                                        "count":2,
                                        "msg":"information reported successfully",
                                        "topic":"",
                                         "serviceId":"CS",
                                        "data":
                                        [{
                                        "name":"device_temperature",
                                        "datatype":"double",
                                        "val":"36",
                                        "unit":"℃",
                                        "comment":"temperature"
                                        },{
                                        "name":"device_humidity",
                                        "datatype":"double",
                                        "val":"6",
                                        "unit":"%",
                                        "comment":"humidity"
                                        }]
                                        "eventtime":"20200706T144831Z"
                                }
                        ]
                }
        ]
}
```

Figure 4 MQTT message delivery format.

2.3.2.1 *Format of published topics*

There are several MQTT topics that IoT device endpoints should subscribe to, using the MQTT client ID of devices/iot as an example.

(1) devices/iot/key
The topic indicates that the IoT terminal device transmits the encryption key of MQTT load information transmission to the server.

(2) devices/iot/xxx
The topic indicates that the MQTT client sends the encrypted MQTT load information to the server. That is, the data information that the IoT terminal device really wants to upload.

2.3.2.2 *Topic format for subscriptions*

There are several MQTT topics that IoT device endpoints should subscribe to, using the MQTT client ID of devices/IoT as an example.

(1) devices/iot/unauth
The received message indicates that the IoT end device is not registered in the web platform of the system framework.
(2) devices/iot/forbidden
The received message indicates that the device information has been registered, but the encryption key for message transmission has not been generated by the web server or the key has expired.
(3) devices/iot/key/success
The received message indicates that the encryption key for the server-side MQTT messaging was successfully generated. And the encrypted MQTT load message can be sent to the server.
(4) devices/iot/key/fail
The received message indicaties that the encryption key generation for the MQTT messaging on the Web server side failed.
(5) devices/iot/update/client
The received message indicates that the user has modified the MQTT information of the IoT end device and the delivered load message contains the MQTT client information to be updated by the IoT end device.
(6) devices/iot/update/code
The received message indicates that the server sends the url address information of the relevant file to realize the function of remote update of the terminal device.
(7) devices/iot/xxx/datas
The received message indicates that the data uploaded by the IoT end device to the topic devices/ iot /xxx for the first time is successful.
(8) devices/iot/xxx/success
The message received indicates the second and subsequent successful upload of data from the IoT end device to the subject devices/iot/xxx.
(9) devices/iot/xxx/error
The received message indicates an error in the devices/iot/xxx topic information sent by the IoT end device.

3 Blockchain

The blockchain architecture is mainly divided into four layers. The bottom data layer adopts Merkel tree blocks composed of asymmetric encryption,

digital signatures, time stamps, hash functions, and chains to connect to ensure that the overall data of the block chain can not be tampered with. The network layer, which releases data through propagation mechanism and verifies the authenticity of propagated data by verification mechanism. The consensus layer mainly uses algorithms to realize node trust and data consistency. The application layer includes the actual scenarios where blockchain is applied.

This protocol uses MQTT as the application layer messaging protocol to establish a blockchain network between all MQTT proxy servers and clients. When the client transfers data to the proxy server, it stores its data value in the block of the blockchain. The broker can verify the historical data stored in the blockchain to determine whether the data has been tampered with.

3.1 Key Management

In the blockchain network, each node has a unique node ID in the network. For the gateway, the unique identification gateway ID of the gateway can be used as the node ID.

In a scenario with multiple cloud servers, different cloud servers should use different node IDs. In addition, each node also has a key pair used for identity authentication, and each gateway and edge device entering the network must generate its own key pair (public key and private key).

Each node in the network keeps a list containing the node ID and public key of all other nodes. The node newly joining the network can announce its public key or synchronize the key information of the nodes in the network through message broadcasting. During data transmission, the gateway uses the private key to digitally sign the data. After receiving the data, other nodes use the public key of the gateway to verify the data, ensuring that the data is not tampered with or contaminated.

3.2 Block Generation

Figure 5 shows the block chain structure, and each block contains two parts: the block header and the block body. The block header stores Merkle root, previous block hash, timestamp, difficulty target value, Nonce and other data; the block body stores all node data. The Merkle root generated based on the hash of the data of each node within the block achieves the tamper-evidence of the data of each node within the block; each block is connected together by the hash of the previous block to form a block chain; the timestamp indicates the generation time of the block.

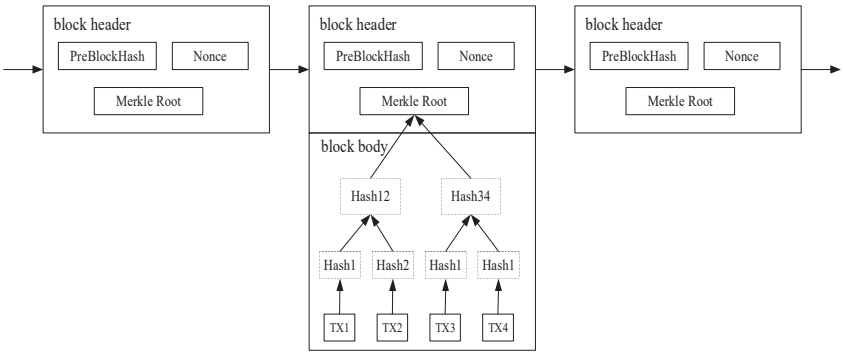

Figure 5 Block chain structure.

In common scenarios of the Internet of Things, client keeps sending data to borker. And this leads to many duplicate data being written to the blockchain, putting a large computational and bandwidth burden on the blockchain network. Therefore, in order to improve the performance and stability of the blockchain network, this paper will reduce the frequency of data writing to the blockchain.

This paper proposes two methods to write the data sent to the broker to the blockchain: periodic writing and change writing.

Periodic writing. It sets a time interval for each data item that is written to the blockchain. This time interval is usually greater than the data sent to the broker. Assuming that the time taken by the edge device to send to the cloud is 10 s, the time interval to write to the block is greater than 10 s. This method avoids real-time transmission of data and reduces the amount of data written to the blockchain.

Change writing. The gateway records the last data value written to the blockchain for each data item. And the gateway sets a threshold value for each data item. When the difference between the data value to be sent by the gateway and the last data value written to the blockchain is greater than or equal to the threshold, the new data value is written to the blockchain. Otherwise, it is only sent to the broker and not written to the blockchain. This approach greatly reduces the amount of duplicate data written to the blockchain.

The formats used to write data to the block chain are as follows:

<Gateway ID/Device ID/Data Item Sequence Number/Timestamp/Data Item Value>

The use of the above two methods can not only greatly reduce the amount of data written to the blockchain, but also ensure the correctness of broker data storage. The scheme use the Merkel tree data structure to form the data information from multiple nodes into a block content in a block.

3.3 Consensus Mechanism

The consensus mechanism, as the core technology of blockchain technology, is the support and guarantee to achieve important functions such as decentralization, anonymity, publicness, and information immutability. The purpose of this mechanism is to efficiently achieve strong and final consistency in distributed systems. A good consensus algorithm can greatly save the time required for blockchain network nodes to reach synchronized consistency, thus improving the operational efficiency of the whole blockchain system. The existing blockchain consensus algorithm, the distributed consistency algorithm represented by Practical Byzantine Fault Tolerance (PBFT), has the advantages of high fault tolerance, high efficiency and scalability for nodes. And this algorithm can be used in blockchain networks that need to handle a large amount of data writes.

This paper will define the node set as R and number the nodes sequentially, $\{0, 1, 2, \ldots, N-2, N-1\}$, N is the total number of nodes in the system. Assuming f Byzantine nodes (failing or malicious nodes) exist in the system, ensure that

$$N \geq 3f + 1 \tag{1}$$

For PBFT algorithm, the algorithm not only tolerate invalid nodes, but also consider malicious nodes. There are two extreme cases for the malicious and invalid nodes in set R:

(1) F invalid sections contain all the evil nodes, so as long as there is $f + 1$ node as a normal node, the client can receive $f + 1$ correct reply. And then the block chain system can achieve a normal consensus.
(2) In addition to f invalid nodes that can not send messages normally, there are also malicious nodes in the nodes that send messages normally. When the algorithm excludes these invalid nodes, in the $N - f$ messages received, assuming that there are false messages from F malicious nodes, at least $f + 1$ correct messages are required to reach a consensus on this message, that is,$f + 1$ normal nodes. And the consensus can be completed. The total number of nodes in the system is $3f + 1$.

Based on the above two extreme cases, Maximum number of Byzantine nodes in the system supported by the PBFT algorithm is $N - 1/3$. All nodes work in the same view. The number v of the view is a consecutive integer starting from 0. When a view change occurs and needs to replace the primary node, the number P of the new primary node is obtained by formula (2):

$$p = v \bmod |R| \tag{2}$$

The basic flow of the PBFT algorithm in one view is as follows:

(1) After the client digitally signs the content (m) written to the block using its private key, it sends the request to the primary node selected in the block chain network;
(2) The primary node checks the validity of client requests and packages them into message broadcasts to slave nodes after passing the check;
(3) All nodes execute the consistency protocol in response to the client;
(4) When the client receives the same message from $f + 1$ different nodes, the consensus has correctly completed and the data has written into the block chain.

PBFT algorithm guarantees consistency protocol through three core stages: pre-prepare, prepare and commit. Figure 6 shows the execution process. Node 0 is the primary node, Node 1, Node 2, Node 3 is the slave node, Node 3 is the Byzantine node, and cannot participate in the consensus normally.

3.4 Data Validation Method Determination

In a cryptocurrency, a node is any computer connected to the network. In blockchain, there is a kind of redundant backup, and if all nodes need

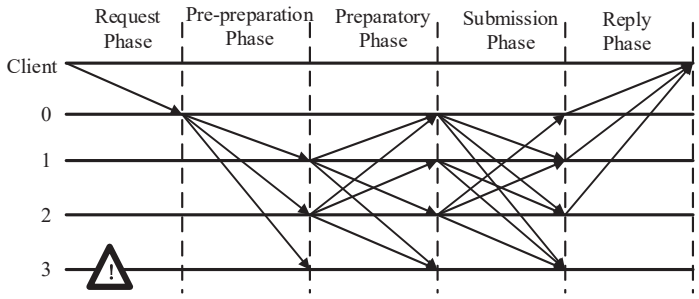

Figure 6 PBFT algorithm flow.

to keep all transactions and other data information of the whole network, there will inevitably be some drawbacks. For example, if a user wants to create a blockchain node of his own for project development without participating in the consensus process, then performing the synchronization of data will be a particularly huge task, which is both time-consuming and resource-intensive.

The blockchain classifies nodes into full nodes and light nodes.

Full nodes, that is, they keep a full network of transaction data, and can complete relevant verification transactions, and independently complete the connection with peer nodes. In other words, this kind of node keeps a complete blockchain network locally, on which any query, transaction verification and broadcast can be performed. Because of the existence of such nodes, it makes decentralization more possible and makes the blockchain network more secure at the same time.

Full nodes, which save all the transmitted data, and can complete the privacy verification of the relevant data and independently complete the connection with the peer nodes. In other words, this type of node preserves a complete blockchain network locally, on which any query can be verified and broadcasted, and it is because of the existence of such nodes that makes the blockchain network more secure.

Lightweight nodes, in the blockchain network, merkle-tree keeps the root hash composed of all the data in the current block, which is saved in the block header, and any change in the content of any piece of data will make the root hash change, thus changing the blockchain structure and not being recognized by the node. The lightweight nodes don't need to save all data contents, using the characteristics of merkletree, they only need to contain the blockheader and the data details related to themselves, and determine whether the data is in the current blockchain data list by Merkle proof.

In the blockchain network, due to the limited storage resources at the edge and usually no data verification request is initiated, it is not necessary to keep the information of all blocks in the broker for a long time. And when it consume the storage resources of the gateway, it can delete the old blocks and realize the recycling of storage resources. Brokers can choose to be full nodes or light nodes according to the performance and resources of their servers, but in the blockchain network, there must be at least one full node that works stably in the long term. The network should set more than two full nodes. When a full node restarts due to failure, it can synchronize all the blockchain information from other full nodes to improve the reliability and stability of the blockchain network.

3.5 Block Chain Function Implementation

According to the design of the blockchain network above, the implementation of this blockchain requires the definition of specific MQTT topics. Table 2 gives an example of the design of an MQTT topic that implements the blockchain functionality. Figure 7 gives an example of a blockchain write request message from the gateway to the master node, with the request source topic in the message header identifying the source of the request and used to receive confirmation of the result after consensus completion.

Table 2 Test results comparison

		Response Time/s	
Frequency	File Size/KB	Before Optimization	After Optimization
1	125	2	1.5
2	345	5	3
3	665	9	7
4	1000	14	13

```
{
        "devices":
        [
                {
                        "deviceId":"D3343134IE3B",
                        #"deviceId" of gateway sub device "services":
                        [
                          {
                                "code":200,
                                "count":2,
                                "msg":"information reported successfully",
                                "topic":"",
                                "serviceId":"CS",
                                "sign":"",   # digital signature
                                "data":[ ]   # data written to the block
                                "eventtime":"20200706T144831Z"# time stamp
                    }
                        ]
                }
        ]
}
```

Figure 7 Block chain write request.

4 Analysis of Experimental Results

In order to verify the correctness of the improved MQTT protocol data transmission based on blockchain technology, the server set up the test environment, and build MQTT Broker. Two test IoT gateways push data to the Broker, and the pushed data follow the transmission data format. The test MQTT Broker address: mqtt.IoTmqtt.com.cn, port: 1883, standard test tool MQTT.fx, Figure 8 shows the configuration of the Broker in MQTT.fx. The JSON data received after decompression of the push data, Figure 9 shows the running result, which meets the expected transmission requirements.

This paper publishes experimental data designed according to quality of service QoS = 1 for MQTT protocol message publishing. After the server is running normally, test the MQTT protocol before and after optimization. First test the response time of the client receiving the message and test it several times under different message volumes to see the performance change. As shown in Table 2, after four tests, the amount of data sent to the JSON file each time increases. Comparing the response time before and after optimization, it can be seen that the modified structure can effectively reduce the response time, with an average decrease of about 22%.

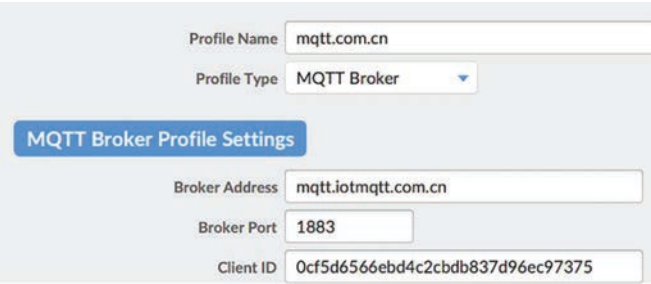

Figure 8 Broker configuration in MQTT.FX.

Figure 9 Result of operation.

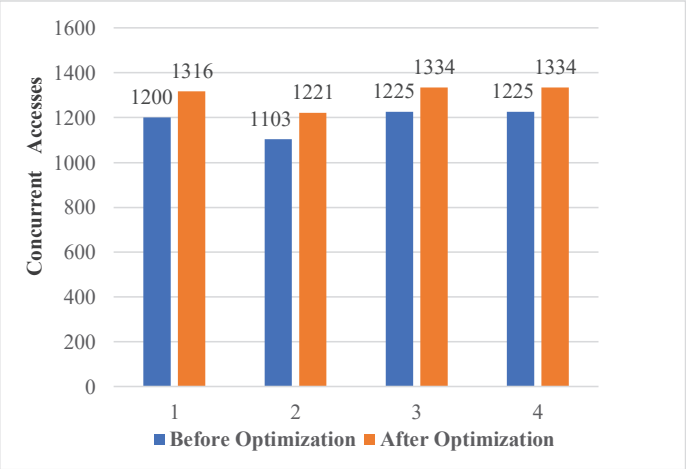

Figure 10 Concurrent access performance diagram for many clients.

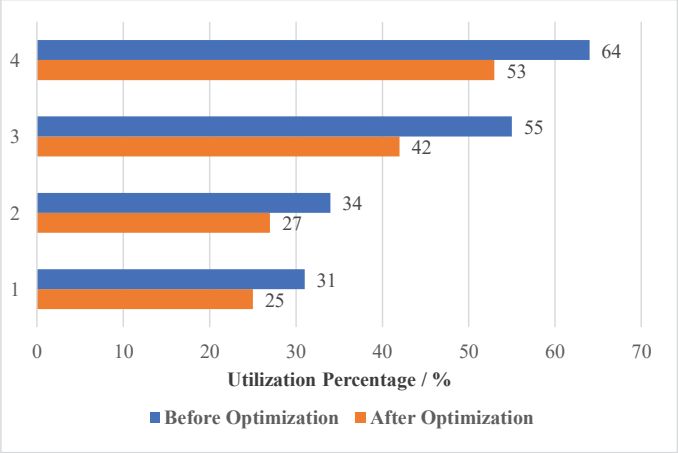

Figure 11 Proxy server CPU utilization.

Conduct concurrency tests on the server and manometer tests four times, sending 10,000 messages continuously in 5 minutes. As shown in Figure 10, the number of concurrent access is significantly better after optimization than before, with an average increase of about 9%.

In this paper, stress tests are conducted for the proposed method. The results of CPU occupancy during the stress test are shown in Figure 11. Although the complexity of the optimized subscription mechanism has

increased compared with that before the optimization, the increased complexity can be controlled at a reasonable level and does not affect the stable operation of the service.

5 Conclusion

This paper proposes a blockchain-based MQTT protocol optimization strategy for the communication between edge devices and brokers in the Internet of Things scenario. Due to the non-tampering nature of block chains, this paper also proposes a scheme of building a block chain network with edge devices and gateways as nodes in the same network. This scheme writes data into the block chains at the same time of data transmission to ensure the integrity and authenticity of the data stored in the cloud. The simulation results show that this scheme is lightweight, efficient and easy to implement, and ensures the reliability of data transmission. It provides reliable technical support and guarantee for stable transmission of real-time data in the future under the conditions of remote devices with low hardware performance and unstable network conditions.

References

[1] Zhong C. L., Zhu Z. and Huang R. G. Study on the IOT Architecture and Access Technology; proceedings of the 2017 16th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES), F 13–16 Oct. 2017, 2017 [C].

[2] Xu L. D., He W. and Li S. Internet of Things in Industries: A Survey [J]. IEEE Transactions on Industrial Informatics, 2014, 10(4): 2233–43.

[3] Wollschlaeger M., Sauter T. and Jasperneite J. The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0 [J]. IEEE Industrial Electronics Magazine, 2017, 11(1): 17–27.

[4] Tao F., Cheng J. and Qi Q. IIHub: An Industrial Internet-of-Things Hub Toward Smart Manufacturing Based on Cyber-Physical System [J]. IEEE Transactions on Industrial Informatics, 2018, 14(5): 2271–80.

[5] Sadio O., Ngom I. and Lishou C. Lightweight Security Scheme for MQTT/MQTT-SN Protocol; proceedings of the 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS), F 22–25 Oct. 2019, 2019 [C].

[6] Andy S., Rahardjo B. and Hanindhito B. Attack scenarios and security analysis of MQTT communication protocol in IoT system; proceedings of the 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), F 19–21 Sept. 2017, 2017 [C].
[7] Amanlou S., Hasan M. K. and Bakar K. A. A. Lightweight and secure authentication scheme for IoT network based on publish–subscribe fog computing model [J]. Computer Networks, 2021, 199, 108465.
[8] V S., A V. and Pattar S. MQTT based Secure Transport Layer Communication for Mutual Authentication in IoT Network [J]. Global Transitions Proceedings, 2022, 3(1): 60–6.
[9] Yihan W. and Yongzhen L. Improved Design of DES Algorithm Based on Symmetric Encryption Algorithm; proceedings of the 2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA), F 22–24 Jan. 2021, 2021 [C].
[10] Kim S., Goo Y., Kim M., et al. A method for service identification of SSL/TLS encrypted traffic with the relation of session ID and Server IP; proceedings of the 2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS), F 19–21 Aug. 2015, 2015 [C].
[11] S. Shin K. K. Efficient Augmented Password-Only Authentication and Key Exchange for IKEv2 [J]. IETF, 2012.
[12] Ö. Y. and Dalkılıç G. Authentication and Authorization Mechanism on Message Queue Telemetry Transport Protocol; proceedings of the 2018 3rd International Conference on Computer Science and Engineering (UBMK), F 20–23 Sept. 2018, 2018 [C].
[13] Calabretta M., Pecori R. and Veltri L. A Token-based Protocol for Securing MQTT Communications; proceedings of the 2018 26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), F 13–15 Sept. 2018, 2018 [C].
[14] Fremantle P., Aziz B., Kopecký J., et al. Federated Identity and Access Management for the Internet of Things; proceedings of the 2014 International Workshop on Secure Internet of Things, F 10-10 Sept. 2014, 2014 [C].
[15] Niruntasukrat A., Issariyapat C., Pongpaibool P., et al. Authorization mechanism for MQTT-based Internet of Things; proceedings of the 2016 IEEE International Conference on Communications Workshops (ICC), F 23–27 May 2016, 2016 [C].

[16] Esfahani A., Mantas G., Matischek R., et al. A Lightweight Authentication Mechanism for M2M Communications in Industrial IoT Environment [J]. IEEE Internet of Things Journal, 2019, 6(1): 288–96.
[17] Sarwar K., Yongchareon S. and Yu J. Lightweight ECC with Fragile Zero-Watermarking for Internet of Things Security; proceedings of the 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), F 1–3 Aug. 2018, 2018 [C].

Biographies

Wei Gao received Bachelor degree from the School of Electronic and Information Engineering, Taiyuan University of Technology, Shanxi in 2003. And he received Master degree from the School of Electronic and Communication Engineering, Taiyuan University of Technology, Shanxi, in 2010.

Lixia Zhang received Bachelor degree from the School of Information Management and Information System, Tianjin university of Commerce, Tianjin,

in 2003. And she received Master degree from the School of Computer Application Technology, University of Science and Technology Beijing, Beijing, in 2010.

Yun Ju received the Master and Ph.D. degrees from the School of Control and Computer Engineering, North China Electric Power University, Beijing, in 2006 and 2014, respectively. He is currently an lecturer in School of Control and Computer Engineering, North China Electric Power University, China. His current research interests include computer science and control engineering. He has authored or co-authored more than 50 publications.