Scalable Blockchain-Based Access Control Algorithm for Large-Scale IoT Networks with Byzantine Nodes

Ningyu Yang(✉), Xizheng Yang, Rong Chai, and Chengchao Liang

School of Communications and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, People's Republic of China
2645636106@qq.com

Abstract. In this paper, we investigate the access control problem in large-scale Internet of things (IoT) scenarios with Byzantine nodes, and propose a scalable blockchain-based access control architecture and a reliable access control consensus algorithm. Firstly, a two-layer blockchain network architecture is constructed, which is composed of a higher-layer main cluster and lower-layer sub-clusters. A cost optimal-based clustering algorithm is proposed to construct the sub-clusters of the blockchain. To achieve efficient consensus among main cluster nodes, a modified Raft (MRaft) algorithm is proposed. Then, the practical Byzantine fault tolerance (PBFT) algorithm is presented for the sub-clusters. The performance of the proposed consensus algorithm is evaluated through simulations.

Keywords: Blockchain · Internet of things · Access control · Consensus algorithm · Byzantine fault tolerance

1 Introduction

The Internet of things (IoT) has continued gaining in popularity and importance in everyday life in recent years [1]. With the rapid development of communication and manufacturing technologies, The trillion of IoT devices will be connected to the Internet in the near future [2,3]. These devices are expected to perform required functions and share data in various types of applications. In practice, the data access of the IoT devices is in general highly limited and only the authorized users are allowed to access the data. To avoid data leakage caused by illegal access, efficient access control schemes should be designed. However, the ubiquitous and densely deployed IoT devices pose challenges and difficulties to the design of access control schemes [4].

In recent years, some studies have been conducted on the access control problem in IoT scenarios. In [5], the authors proposed an attribute-based secure access control mechanism utilizing federated deep learning, which allows users to access specific medical data only if their trust values exceed the designated threshold. The authors in [6] design a secure industrial data access control scheme

F. Gao et al. (Eds.): Chinacom 2023, LNICST 590, pp. 126–141, 2024.
https://doi.org/10.1007/978-3-031-67162-3_9

for cloud-assisted industrial IoT (IIoT), leveraging ciphertext policy attribute-based encryption to achieve fine-grained access control.

Some studies apply blockchain technology in IoT systems and design blockchain-based access control schemes [7–9]. In [7], the authors present a blockchain-based IoT access control system that leverages a permissioned blockchain, attribute-based access control (ABAC), and identity-based signature to create a lightweight and secure, low computational and cross-domain access control solution. In [8], the author propose a bidirectional-linked blockchain based on chameleon hash functions to resist attacks for lightweight systems. In [9], the authors design a hierarchical IoT architecture, and uses the practical Byzantine fault tolerance (PBFT) algorithm for reaching consensus among network nodes.

While blockchain-based access schemes have been proposed for IoT systems [7–9], the low scalability and high complexity of the proposed schemes may not be suitable for the practical applications of the IoT scenarios. To tackle this issue, we study the access control problem in large-scale IoT scenarios with Byzantine nodes in this paper. A scalable layered blockchain network architecture is proposed which consists of a higher-layer main cluster and a number of lower-layer sub-clusters. Then, the consensus algorithms for both the main cluster and the sub-clusters are designed. Specifically, a modified Raft (MRaft) algorithm and PBFT algorithm are proposed respectively for the main cluster and sub-clusters.

2 System Model

2.1 Proposed Model

In this paper, we consider an IoT scenario, which is composed of M IoT nodes and N edge devices.

IoT devices monitor environmental information, collect environmental data, and send to the associated edge devices directly or forward to edge device via IoT devices with high-performance computation capability.

After receiving data from IoT devices, edge devices store data in their local storage, and store the attributes of the data, such as digital summary, file name, file type, file address, file size, ownership, file sensitivity level, creation time, label, file description, and other information packaged into a transaction, which is then published to a blockchain network composed of edge devices.

The cost of storing this information of the data is much less than storing the data itself, so doing so can let other domain know the existence of the data and related information without storing the data itself, so as to achieve a special distributed storage and cope with possible access requests.

Edge devices can also forward the data to the core network to achieve remote environmental monitoring, which is not detailed here.

Suppose a device (referred to as access requester) tends to access data from another device (referred to as destination device), it first sends an access request message to the system (typically a centralized server, in this paper, it referred

to as the entire blockchain nodes). According to the attributes of the visitor (referred to as the subject attribute), the attributes of the target resource (referred to as the object attribute) and the environment attribute, the system judges whether the access request is legitimate, and determines whether the access requestor has the right to access the corresponding data, and records the access request in the system log for audit. In the system, the access policy is written by the system administrator, and the administrator writes the corresponding rules according to various attributes (Fig. 1).

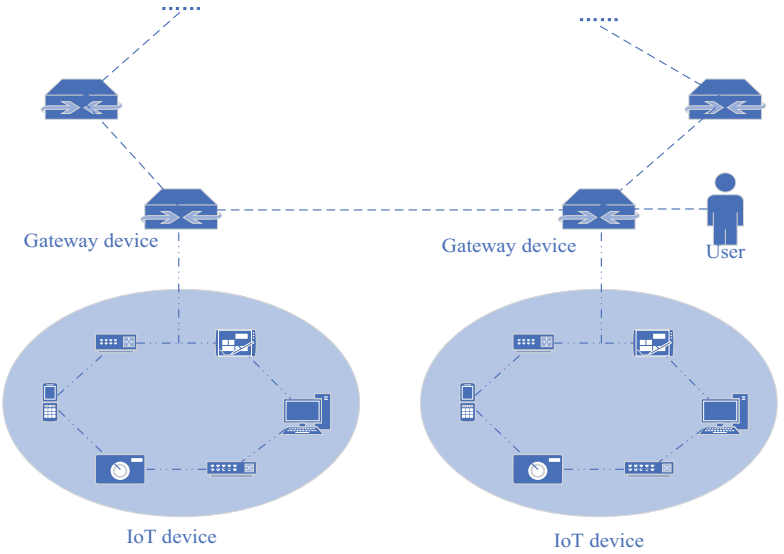

Fig. 1. System model

2.2 XACML

XACML is an OASIS standard specified in XML for access control policy and decision request description. XACML supports finer-grained access control by evaluating access requests based on conditions like attributes of the requesting subject (e.g. user identity and role) and attributes of the targeted resource (e.g. resource type and location). The XACML model is shown in the Fig. 2. Policy administrator point (PAP) is responsible for writing access control policies, policy decision point (PDP) is responsible for evaluating access requests, policy enforcement point (PEP) is responsible for receiving access requests and sending them to PDP for evaluation, and policy information point (PIP) is responsible for providing attribute information for PDP, and context handler is responsible for providing attribute information for PIP.

In this article, we consider a distributed permission verification system and use blockchain technology to ensure the scalability and the security of the system and reduce the possibility of nodes doing evil. We consider a consortium blockchain consisting of alliances, where each alliance is a node in the blockchain and a domain of IoT devices (sub-clusters).

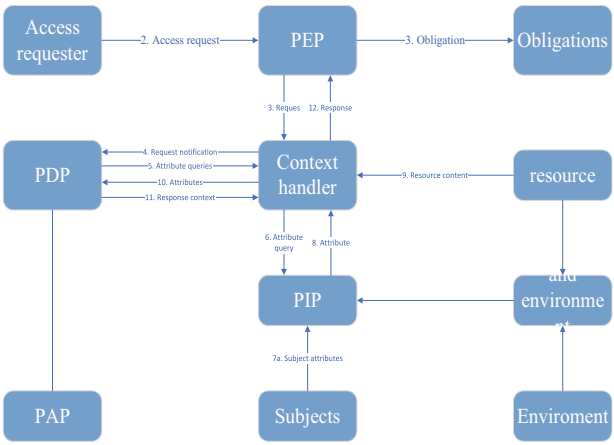

Fig. 2. XACML model

Although the distributed scheme reduces the possibility of nodes doing evil, in practical IoT scenarios, some devices may fail or be attacked, resulting in improper or malicious operations. For instance, a device may refuse to respond, postpone response, or maliciously response with false information. Such devices are referred to as Byzantine devices. In this paper, the access control scheme is designed for the IoT scenario with Byzantine devices.

2.3 Access Control

Architecture Overview. The proposed two-layer blockchain network architecture which consists of a higher-layer main cluster and multiple lower-layer sub-clusters. Specifically, the higher-layer main cluster is composed of edge devices and IoT devices with high-performance computation capability whereas the lower-layer sub-clusters are mainly composed of IoT devices, each lower-layer sub-cluster is a IoT domain. Within the main cluster and sub-clusters, the nodes are capable of conducting information interaction with each other. In order to achieve inter-cluster information interaction, each lower-layer sub-cluster is associated with one of higher-layer main cluster nodes, which acts as a destination device or relay node to forward data between various clusters. For convenience, let the higher-layer main cluster nodes be the main nodes. We assume that each sub-cluster can only access one main cluster node.

Denote the number of sub-clusters as K, which also represents the number of main nodes. The types of the main nodes can be determined based on the number of main nodes and the number of edge devices, denoted by N. If $N \geq K$, all the main nodes are edge devices, i.e., K edge devices are chosen as main nodes and the remaining edge devices work as sub-cluster nodes. If $N < K$, all the edge devices are selected as main nodes and $K - N$ IoT devices with computing capabilities also act as main nodes. The proposed two-layer blockchain architecture is shown in Fig. 3.

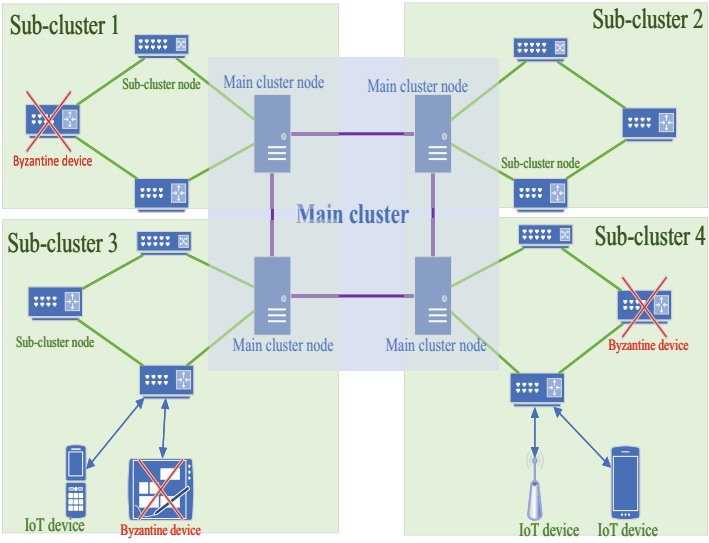

Fig. 3. Two-layer blockchain architecture

Access Procedure. In the proposed blockchain network, the IoT devices are required to upload their attributes, resources, and access control policies to the blockchain network.

Assume that system nodes has completed clustering, and PAP has written an access control policy and stored it at the PDP, a complete access request is as follows.

1. A registered users send access requests to the domain closest to them
2. Edge device in a domain perform PEP functions, but don't need PEP for anything at this time except receiving this request message.
3. The edge device starts to execute the work of the context handler, packaging the voting application into a standard format, which is equivalent to the node has a proposal whether to agree to the access request.

4. If follows XACML, context handler may need to query PIP for attribute information, in this paper, since the edge devices has all the attribute information of IoT devices and datas, we assume that the edge devices also perform the PIP function. In order to avoid PDP from requesting relevant information back and forth from the edge device, we let the edge device package the relevant subject attributes, resource attributes, and environment attributes together into a transaction and publish it to the blockchain network.
5. The blockchain system judges the access request of the transaction and uses the MRaft public consensus algorithm to reach a consensus on the result, and store the result in the blockchain.

To achieve efficient and secure access control, the blockchain nodes should reach consensus on the attributes, resources, access control permissions, and the access records of the IoT devices.

2.4 Cost Function Optimization-Based Sub-cluster Association Algorithm

Let Φ denote the set of blockchain nodes, $\Phi = \{\mathrm{I}_1, \ldots, \mathrm{I}_i, \ldots, \mathrm{I}_T\}$, where I_i denotes the i-th blockchain node, $1 \leq i \leq T$, and T is the number of blockchain nodes.

Cost Function of Formulation. Addressing the energy consumption sensitiveness of the devices and the link transmission performance, we define the cost function as the weighted sum of the energy consumption of the devices and the bit error rate (BER) of the transmission links. Let $\Psi_{i,k}$ denote the cost function when node I_i is associated with node I_k, which is modeled as

$$\Psi_{i,k} = \omega_1 E_{i,k} + \omega_2 \delta_{i,k}, \tag{1}$$

where $E_{i,k}$ represents the energy consumption of node I_i when sending messages to node I_k, and $\delta_{i,k}$ represents the link error rate of node I_i when sending messages to node I_k. ω_1, ω_2 is the weight of energy consumption and the link error rate.

The energy consumption of node I_i when sending messages to node I_k can be computed as

$$E_{i,k} = P_{i,k} t_{i,k}, \tag{2}$$

where $P_{i,k}$ represents the transmit power of node I_i when sending data to node I_k, and $t_{i,k}$ represents the transmission time of node I_i when sending data to node I_k. The transmission time is given by

$$t_{i,k} = \frac{S_{i,k}}{R_{i,k}}, \tag{3}$$

where $S_{i,k}$ represents the amount of data transmitted by node I_i when sending data to node I_k, and $R_{i,k}$ represents the transmission rate of link between node I_i and node I_k. The transmission rate is modeled as

$$R_{i,k} = B \log_2 (1 + \gamma_{i,k}), \tag{4}$$

where B represents the available bandwidth, $\gamma_{i,k} = \frac{P_{i,k} h_{i,k}}{N_0 B}$, $h_{i,k}$ represents the channel gain between node I_i and node I_k, and N_0 represents the noise power spectral density.

The BER in (1) of link between node I_i and node I_k is modeled as

$$\delta_{i,k} = 0.2 \exp\left(-\frac{1.5\gamma_{i,k}}{\alpha - 1}\right), \tag{5}$$

where α represents the modulation order of the transmitted signal.

Sub-cluster Association Algorithm. Given the defined cost function, we propose a cost function optimization-based sub-cluster association algorithm. Let Φ^m denote the set of main nodes and K denote the number of main nodes, i.e., $|\Phi^m| = K$. Without loss of generality, we assume that each main node is associated with one sub-cluster, hence, the number of sub-clusters is also denoted by K. The detail of the proposed algorithm is described as follows.

1. Initial Selection of Main Nodes: Choose K nodes from the set Φ and label them from 1 to K sequentially and denote them as $\mathrm{I}_1^m, ..., \mathrm{I}_K^m$. In particular, if $N \geq K$, randomly select K edge devices as main nodes, otherwise, all the edge devices are selected as main nodes and $K - N$ IoT devices with computing capabilities are chosen as main nodes. Let Φ^m denote the set of selected main nodes, and Φ^s denote the set of sub-cluster nodes, we obtain $\Phi^s = \Phi / \Phi^m$.
2. Determine Association Strategy: For each sub-cluster node, the cost functions when associating various main nodes are evaluated and the main node offering the minimum cost function is selected for association. if $k^* = \text{argmin } \Psi_{i,k}$, where $\Psi_{i,k}$ represents cost function between sub-cluster node I_i^s and main node I_k^m, $\mathrm{I}_k^m \in \Phi^m$, then sub-cluster node I_i is associated with main node $\mathrm{I}_{k^*}^m$. Let Φ_k^s denote the set of nodes in the k-th sub-cluster,
3. Update of Main Nodes: Let $\Psi_{i,k}^0$ denote the total cost between node I_i in the k-th sub-cluster and other nodes in the k-th sub-cluster, modeled as

$$\Psi_{i,k}^0 = \sum_{\mathrm{I}i' \in \Phi_k, \mathrm{I}i' \neq \mathrm{I}_i} \Psi_{i,i'}, \tag{6}$$

   Let $k' = \text{argmin } \Psi_{i,k}^0$, then $\mathrm{I}_{k'}$ becomes the new main node of the k-th sub-cluster, the main node of the k-th sub-cluster is updated to $\mathrm{I}_{k'}^m$, $\mathrm{I}_k^m = \mathrm{I}_{k'}^m$.
4. Convergence Evaluation of The Algorithm: Repeat steps 2. and 3. until the main node set and the association between main nodes and sub-cluster nodes no longer update.

3 Proposed MRaft Algorithm Based Main Cluster Consensus Scheme

In this section, we propose an MRaft-based main cluster consensus algorithm.

3.1 Basic Idea of the Proposed MRaft Algorithm

To achieve high consensus efficiency, low latency, and low communication overhead among main nodes, the Raft algorithm can be applied. The main idea of Raft algorithm can be summarized as follows [10]. All the nodes in the main cluster can be classified into three roles: leader node, follower node and candidate node. The leader node is responsible for receiving and processing requests, and the follower node is responsible for receiving and executing the leader node's instructions. The candidate node may become new leader node in the case that previous leader node cannot be accessed. The Raft algorithm is composed of two stages: leader election and log replication. In leader election stage, a leader node is selected based on the votes from network nodes. In log replication stage, the log files are stored in the local caches of the follower nodes and then executed when the consensus among network nodes is reached.

While the Raft algorithm is capable of achieving consensus among network nodes, it cannot verify whether node messages have been tampered with, making it unable to support Byzantine fault tolerance. To address this issue, we propose an MRaft algorithm which introduces a supervisory node and message verification mechanism in the framework of the Raft algorithm. The main functions of the supervisory node include collecting pre-preparation messages from main nodes, verifying the legality of pre-preparation messages, generating and packaging reply certificates, and broadcasting reply certificates to the main cluster.

3.2 MRaft Algorithm Procedures

MRaft algorithm is used for consensus between the main clusters, in combination with the scenario presented in this paper, as described above, when a sub-cluster receives an access request from a user, the edge devices of this sub-cluster will package the request into a transaction, and publish it to the blockchain network, MRaft algorithm is used to reach consensus on the block containing the transaction.

The MRaft algorithm is composed of four stages: request stage, pre-preparation stage, preparation stage, and response stage. The algorithm procedures in each stage are described below.

1. Request Stage: The edge device of each domain packages the access requests it receives into transactions, signs it with its own private key and sends it to the leader of the current view. In addition, some other information should be included in the transaction message, some of the important information are message type, transaction digest, view number.

2. Pre-Preparation Stage: Leader verifies the signatures of these transactions, assigns a sequence number α, to these transactions and arranges them by sequence number after a time, that is, $\{\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_k\}$, and k represents the number of transactions received by leader during this term. Then leader uses its own private key to sign and package these transactions into a block b, and then sends b to the followers and the supervisor node in the main cluster. Some other information should be included in the block message, which is similar to the transaction message, there is not much detail here.
3. Preparation Stage: Upon receiving block b from the leader node, the supervisory node verifies that the block b has been tampered with and that each transaction is legal. The follower nodes judge whether the access requests in the block b are passed according to the access control policy stored internally and set by the administrator. Because the system proposed in this paper uses consortium blockchain, each sub-cluster nodes also need to reach a consensus on the decision of the corresponding main cluster nodes, so the PBFT algorithm is used to reach a consensus on the decision of the corresponding main cluster nodes, and the decision made by main cluster node are legal only if sub-cluster nodes recognize it.
   When each sub-cluster makes a decision, the main nodes of each sub-clusters send a message containing the decision and some informations to the supervisory node and the leader node, some of these informations are the same as before, in particular, the main node needs to package the signature of the sub-cluster nodes into the message.
4. Response Stage: After the supervisor node and leader receive the decision returned by each main cluster node, the leader broadcasts the final decision to the system according to the decision of each master node, the supervisor node verifies the legitimacy of each message, and broadcasts the verification result. Each sub-cluster in the system judges the final decision of the system according to the message broadcast by the leader and the supervisor node. And the results of this round of consensus can be known. At this point, the round of consensus is completed.

Not only the consensus process of access control requests, but also some other operations in the system are completed through the above process.

For data resources, it is necessary to upload their attributes to achieve to some degree of distributed data storage. After receiving the data, the main node of each domain packages the data's digital summary, file name, file type, file address, file size, ownership, file sensitivity level, creation time, label, file description and other information into a transaction, and then sends it to the corresponding leader of the system. After the leader is packaged into blocks, it is broadcast to all the sub-clusters in the system.

The administrator uses the same scheme for uploading the access control policy. The administrator signs and packages the policy written by himself and then gives it to the main node of a domain in the system. The master node packages it into a transaction and then sends it to the current leader of the

system, and then deploys the access control policy in each domain of the system by consensus.

In the system designed in this paper, all IoT devices and edge devices need to upload its own attributes to the block chain, which can prevent people with evil thoughts from using a large number of low-cost devices to destroy the system. Similarly, users in the system also need to register with the system, so they also need to upload their own attributes. These things can also be packaged into transactions through the edge devices of a domain and broadcast to all domains in the block chain, which does not bring too much overhead or complexity to the system while improving the security of the system.

4 PBFT-Based Sub-cluster Consensus Algorithm and Main Cluster View Switching

In the proposed system model, there may exist Byzantine nodes in the sub-cluster, we propose a PBFT-based sub-cluster consensus algorithm so as to achieve the consensus within individual sub-clusters.

PBFT (Practical Byzantine Fault Tolerance) is a consensus algorithm used to deal with Byzantine failures in distributed systems, it is suitable for environments with limited capacity and number of nodes, it assumes the system can tolerate up to f faulty nodes. The number of nodes in the network must be at least $3f + 1$.

4.1 The PBFT-Based Sub-cluster Consensus Phase

The sub-cluster consensus phase is composed of four stages: pre-preparation stage, preparation stage, submission stage, and response stage. The sub-cluster consensus phase is described in detail below. We assume that the main node of the sub-cluster has always been the primary node.

1. Pre-Preparation Stage: The primary node receives the block b, and packages the block b into a pre-preparation message, it contains the block b, sequence number, digest of the operation etc. to all the replica nodes in the sub-cluster.
2. Preparation Stage: replica nodes validate and broadcast preparation messages to all nodes in the sub-cluster if receive $2f$ legal Pre-preparation messages from different replica. The preparation message contains sequence number, pre-preparation message digest, and signature of the replica node.
3. Submission Stage: In the Submission phase, each replica node sends Submission messages to the other replica nodes, including the sequence number, view number, and digest of the operation. When a replica node receives Submission messages from $2f + 1$ different replica nodes, it can apply the request to the local state machine and send a response to the client.
4. Response Stage: Once a replica node applies the request to the local state machine, it sends a response message to the primary node indicating that the request has been processed.

The MRaft-PBFT consensus process is shown in Fig. 4.

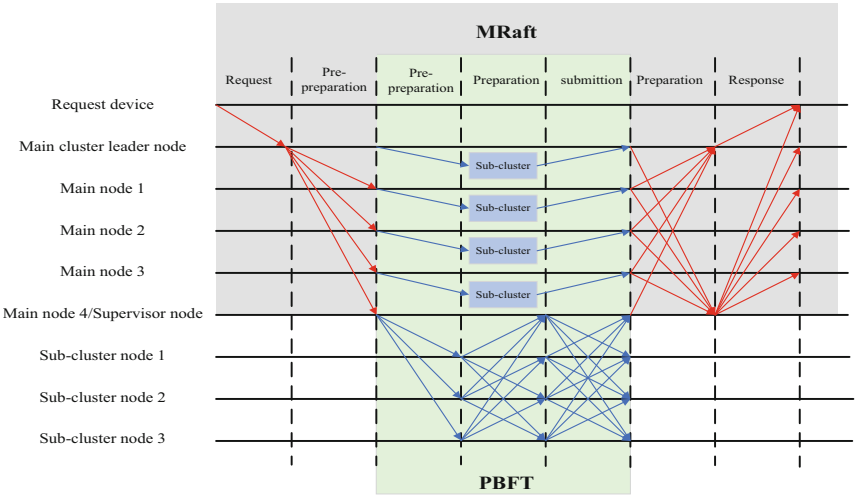

Fig. 4. MRaft-PBFT consensus process

4.2 The Procedure of View Switching

In the Raft algorithm, when the leader or is not offline, the original leader will always perform the functions of leader, and the view switch will be triggered only when it is offline, which is reasonable, but in the system designed in this paper, the burden of the node may be too heavy, and the system's ability to tolerate Byzantine nodes will decline to some extent. So in the MRaft algorithm, the system will automatically trigger the view switch after every v round consensus, Select the main node and supervisory node of the new view according to a certain scheme.

Let g and h denote the index of the leader node and supervisory node in main cluster, v denote the current view number, and l denote the number of the sub-clusters, we set $g = (v + 1) \bmod l$. That is, the node with index being g is selected as the leader node in the $v + 1$-th view, and the node with index being h is selected as the supervisory node in the $v + 1$-th view.

The consensus procedure of view switching is summarized below.

1. Message Transmission Stage Each main cluster node triggers a view switch, updates the view number to $v = v + 1$, and broadcasts the view switching message. The view switching message includes the message type, view number, the latest request message sequence number, the signature of the main cluster node, etc.
2. Reply Stage Each main cluster node receives the view switching message from other nodes, verifies the signature. If the verification is successful, the main cluster node sends a reply message to the selected main node and supervisory node.

3. New View Stage If the selected main node and supervisory node receives $2f$ view switching messages from different main cluster nodes with a view number of $v+1$, they broadcasts a new view message to the main cluster nodes. The new view message includes the new view message type, view number $v+1$, the new main node's signature on the view message, etc. After receiving the new view message if the verification is successful, the main node is updated successfully.

5 Simulations

The simulation scenario considered in this paper is an IoT system composed of multiple edge devices and IoT devices. The simulation area is 2000m $\times$ 2000 m, and the edge devices and IoT devices are randomly distributed in the simulation area. The simulation parameters are shown in Table 1, and all simulation results are the average of 500 independent experiments.

Table 1. Simulation parameters setting

Parameters	Value
Transmit power of IoT devices ($P_{i,k}$)	0.1 W
Amount of data of IoT devices ($S_{i,k}$)	[1,5] Mbits
Link bandwidth (B_k)	1 Mhz
Modulation order (α)	16
Noise power spectral density (N_0)	-174 dBm

Figure 5 shows the total cost versus the number of sub-clusters. It can be seen from the figure that as the number of sub-clusters increases, the total cost decreases. This is because as the number of sub-clusters increases, the number of nodes in individual sub-clusters decreases, resulting in smaller cost within sub-clusters and smaller total cost in turn. It can also be observed from the figure that as the number of network nodes increases, the total cost increases accordingly.

Figure 6 shows the consensus success rate versus the ratio of Byzantine nodes. It can be observed from the figure that as the ratio of Byzantine nodes increases, the consensus success rate decreases. This is because a larger number of Byzantine nodes results in difficulty in achieving consensus and leads to lower consensus success rate in turn. In the figure, we also examine the consensus success rate obtained for different number of sub-clusters. It can be seen from the figure that when the ratio of Byzantine nodes is relatively low, a larger number of sub-clusters yields slightly higher consensus success rate while when the ratio of Byzantine nodes increases, a larger number of sub-clusters results in a lower consensus success rate. The reason is that given the number of system nodes, when the number of sub-clusters is large, smaller nodes are contained in each

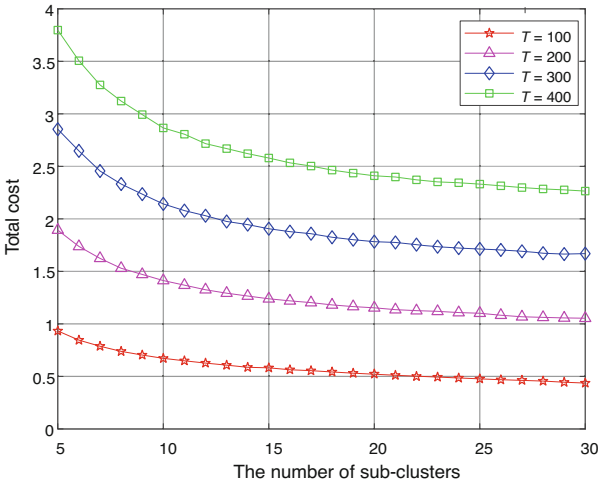

Fig. 5. The total cost vs number of sub-clusters

sub-cluster in general. In the case of low ratio of Byzantine nodes, smaller numbers of Byzantine nodes occur in individual sub-clusters, making it easier to reach consensus within the sub-clusters, resulting in a higher overall consensus success rate. On the other hand, when the ratio of Byzantine nodes increases, larger numbers of Byzantine nodes occur in each sub-cluster, thus leading to the difficulty in reaching consensus within sub-clusters. As a result, the consensus success rate decreases accordingly.

Figure 7 evaluates the relationship between consensus delay and the number of network nodes. For comparison, we plot the consensus delay obtained from our proposed MRaft-PBFT algorithm and the consensus algorithm proposed in [9]. In addition, we also apply Raft algorithm and PBFT algorithm to the considered network scenario, and examine the corresponding consensus delay. It can be seen from the figure that as the number of nodes increases, the consensus delay gradually increases. The reason is that as the number of network nodes increases, longer time is required for conducting information interaction among nodes, leading to higher consensus delay. It can also be observed from the figure that comparing the consensus delay of different algorithms, the consensus delay of our proposed PBFT-MRAFT algorithm is smaller than that obtained from the PBFT algorithm and the one proposed in [9]. This is because the PBFT algorithm and the consensus algorithm proposed in [9] require a larger number of information interaction among nodes, thus resulting in longer consensus delay.

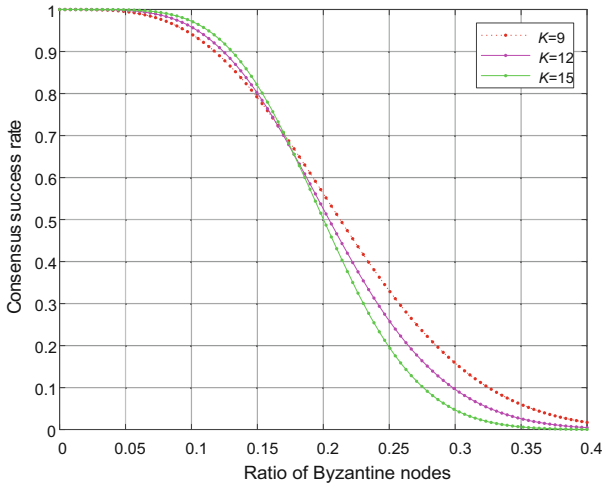

Fig. 6. Consensus success rate vs Ratio of Byzantine nodes

In Fig. 8, we examine the communication overhead of the proposed consensus algorithm, the algorithm proposed in [9]. Given the number of network nodes, the proposed MRaft-PBFT-based consensus requires smaller communication overhead than that of PBFT-based algorithm and the one proposed in [9]. This is because our proposed layered PBFT-MRaft algorithm leverages the advantages of PBFT and Raft algorithms and is capable of achieving efficient consensus with a relatively small communication overhead.

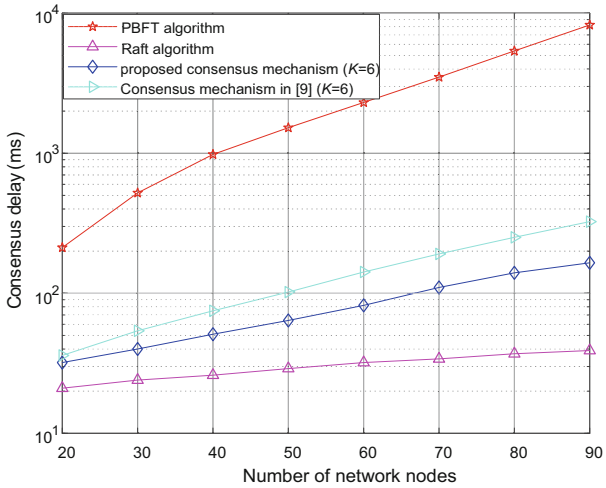

Fig. 7. Consensus delay vs number of nodes

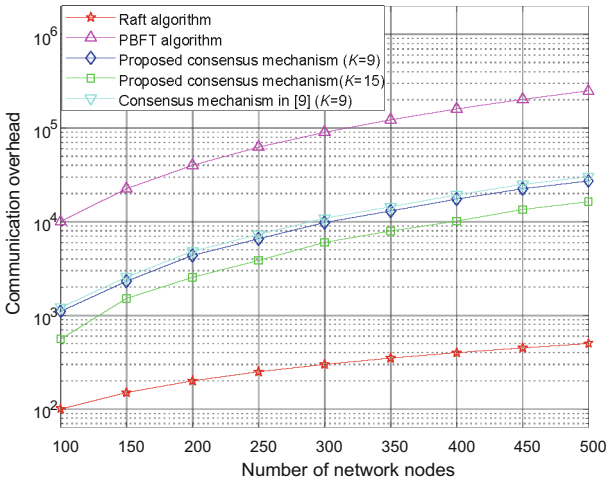

Fig. 8. Communication overhead vs number of nodes

6 Conclusions

In this work, the access control problem is studied for large-scale IoT scenarios with Byzantine nodes. Firstly, a two-layer blockchain network architecture is designed, which consists of a high-layer main cluster and multiple low-layer sub-clusters. A cost optimization-based clustering algorithm is proposed to construct the sub-clusters of the blockchain. To achieve efficient consensus among nodes, an MRaft-PBFT algorithm is proposed. Finally, simulations are conducted and the consensus success rate, consensus delay and communication overhead of the proposed algorithm are evaluated.

References

1. Bout, E., Loscri, V., Gallais, A.: How machine learning changes the nature of cyberattacks on IoT networks: a survey. IEEE Commun. Surv. Tutorials **24**(1), 248–279 (2022)
2. Li, Y., et al.: Toward location-enabled IoT (LE-IoT): IoT positioning techniques, error sources, and error mitigation. IEEE Internet Things J. **8**(6), 4035–4062 (2021)
3. Yaqoob, I., et al.: Internet of Things architecture: recent advances, taxonomy, requirements, and open challenges. IEEE Wirel. Commun. **24**(3), 10–16 (2017)
4. Mohd Aman, A.H., Yadegaridehkordi, E., Attarbashi, Z.S., Hassan, R., Park, Y.-J.: A survey on trend and classification of Internet of Things reviews. IEEE Access **8**, 111763–111782 (2020)
5. Lin, H., Kaur, K., Wang, X., Kaddoum, G., Hu, J., Hassan, M.M.: Privacy-aware access control in IoT-enabled healthcare: a federated deep learning approach. IEEE Internet Things J. **10**(4), 2893–2902 (2023)
6. Qi, S., Lu, Y., Wei, W., Chen, X.: Efficient data access control with fine-grained data protection in cloud-assisted IIoT. IEEE Internet Things J. **8**(4), 2886–2899 (2021)

7. Sun, S., Du, R., Chen, S., Li, W.: Blockchain-based IoT access control system: towards security, lightweight, and cross-domain. IEEE Access **9**, 36868–36878 (2021)
8. Xu, C., Qu, Y., Luan, T.H., Eklund, P.W., Xiang, Y., Gao, L.: A lightweight and attack-proof bidirectional blockchain paradigm for Internet of Things. IEEE Internet Things J. **9**(6), 4371–4384 (2022)
9. Li, W., Feng, C., Zhang, L., Xu, H., Cao, B., Imran, M.A.: A scalable multi-layer PBFT consensus for blockchain. IEEE Trans. Parallel Distrib. Syst. **32**(5), 1146–1160 (2021)
10. Ongaro, D., Ousterhout, J.: In search of an understandable consensus algorithm. In: Proceedings of the USENIX Annual Technical Conference (USENIX ATC), pp. 305–320 (2014)
11. Castro, M., Liskov, B.: Practical Byzantine fault tolerance. In: Proceedings of the 3rd Symposium on Operating Systems Design and Implementation, vol. 99, pp. 173–186 (1999)