IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 17, 2021, accepted February 7, 2021, date of publication February 16, 2021, date of current version March 9, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3059863

Blockchain-Based IoT Access Control System: Towards Security, Lightweight, and Cross-Domain

SHUANG SUN, RONG DU, SHUDONG CHEN, AND WEIWEI LI
Institute of Microelectronics, Chinese Academy of Sciences, Beijing 100029, China
School of Microelectronics, University of Chinese Academy of Sciences, Beijing 100049, China

Corresponding author: Shudong Chen (chenshudong@ime.ac.cn)

This work was supported by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDC02070600.

ABSTRACT Most IoT devices cannot afford to be a blockchain node due to the high computation and storage loads. Thus, the blockchain is usually deployed on one delegate node, e.g., the edge device or cloud, which may encounters three drawbacks: (1) The delegate node becomes the single failure point when the number of delegate notes are limited. (2) The delegate node replicating the blockchain data can lead to privacy information leak. (3) The delegate node is vulnerable to the Distributed Denial of Service (DDoS) attack. To tackle these drawbacks, we consider to minimize the redundant of blockchain to make the IoT devices as the specialized blockchain nodes. In this paper, we integrate a permissioned blockchain (HLF), an attribute-based access control (ABAC) and an identity-based signature (IBS) to build a security, lightweight, and cross-domain blockchain-based IoT access control system. Specifically, we divided the IoT system into different function domains, named IoT domains. Then, we establish a local blockchain ledger for each IoT domain to enable more IoT devices as blockchain nodes. The local blockchain ledger records the IoT domain entities' attributes, policy files' digests, and access decisions. Meanwhile, we use the channel technology of HLF to realize cross-domain access and use the IBS to filter the legal access requests for each IoT domain to prevent DDoS attacks. We also design a policy decision point (PDP) selection algorithm that select multiple IoT devices (blockchain nodes) to achieve the real-time distributed policy decisions (off-chain). Finally, we implement and evaluate the proposed system to demonstrate its practicality.

INDEX TERMS IoT, blockchain, ABAC, HLF, IBS, PDP selection algorithm.

I. INTRODUCTION

With the rapid growth of smart devices and high speed networks, the Internet of Things (IoT) approached our lives gradually and silently. According to the forecast, more than 8.4 billion connected things joined this network worldwide in 2017, and will reach 20.4 billion by 2020 [1]. It is necessary to control the huge number of IoT devices remotely to perform the desired functionality and share information. However, the security and privacy issues bring economic loss to users and hampering the development of the IoT [2]. Therefore, access control is regarded as one of the most important technology to guarantee IoT security and privacy [8]–[11].

Many IoT access control schemes have been proposed [8]–[10], but almost all of them are based on a single-server architecture. A centralized scheme has limitations such as the single point of failure. Furthermore, a malicious or compromised server can easily modify the access policy and permit the illegal access requests. Therefore, it is necessary to design a secure decentralized access control scheme for the IoT system.

The decentralization, verifiability, and immutability properties enable blockchain to solve the above problems. A rising number of blockchain-based IoT access control systems have been proposed. These systems integrated the blockchain and IoT from various perspectives to implement efficient and scalable distributed IoT access control [8]–[11]. Specifically, they make distributed access decisions via smart contract or record access policies (authority credentials) on the blockchain to prove that users have granted access by querying the blockchain. However, the existing blockchain-based IoT access control systems generally have the following limitations:

1)Lack of security: Blockchain will suffer DDoS attacks when malicious users send access requests repeatedly.

2)Limited distribution: The distribution feature of the IoT system is determined by the proportion of entities who

The associate editor coordinating the review of this manuscript and approving it for publication was Wen Sun.

deployed the blockchain [32]. Due to resource-constrained and dynamic features (their state and assigned tasks are different at each stage), most IoT devices cannot act as blockchain nodes to deal with access control process logic in real-time and afford the storage pressure of blockchain ledger. Therefore, most schemes only use delegate servers (edge devices or cloud) as blockchain nodes. However, in an IoT system composed of lighted IoT devices, such as smart home, there is only one gateway device that cannot achieve Byzantine fault tolerance.

3)Lack of privacy: To deal with the access control process, delegate nodes need to record all access policies and process all access requests, which may cause leakage of policy information as well as the daily activities of users may be predicted.

Therefore, to benefit from the blockchain technology and avoid use delegate nodes to process the access control logic, we design a lightweight blockchain that enables most IoT domain to deploy its own blockchain network on their IoT devices.

Firstly, we select HLF as the blockchain platform. As a permissioned blockchain, HLF supports multiple blockchain ledgers and only the authorized users are allowed to store the data in the corresponding blockchain ledgers. Therefore, we establish a local blockchain ledger for each IoT domain. A local blockchain ledger only records access control data and processes access control logic of its own domain, which can reduce the storage and computation pressure of each blockchain node. In addition, the channel technology of HLF can realize cross-domain access of users between IoT domains.

Furthermore, to avoid the DDoS attack, we design the public key of IoT entities based on identity-based signatures (IBS) to enable edge servers to filter and forward access requests initiated by users who are inside the IoT domain or trusted domain to PDPs.

Finally, due to the dynamic characteristics, not all IoT devices who deployed blockchain can process the access requests in real-time. So we design a PDP algorithm, to select the blockchain nodes making real-time off-chain policy decisions. Moreover, we use the PBFT to make final access decisions consensus.

The rest of the article is organized as follows. Section II briefly introduces some preliminaries and related work. We describe the construction of the blockchain-based IoT attributed based access control system in section III. More specific scheme design details are described in section IV. In section V, the practicality of our proposed system is demonstrated by the performance evaluation. Finally, a short conclusion is given in section VI.

II. PRELIMINARIES AND RELATED WORKS

In this section, we review some relevant background knowledge and related works.

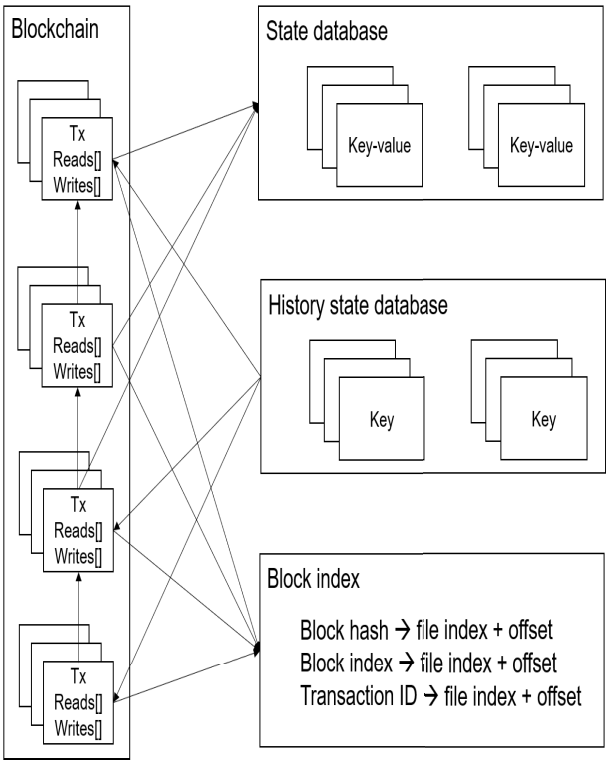

FIGURE 1. Ledger.

A. HYPERLEDGER FABRIC

In our system, we use the hyperledger fabric (HLF) to construct the system and focus on the key component of HLF as follows:

- **Peer**. Blockchain nodes maintain ledgers and read/write on the ledgers via chaincode.
- **Ledger**. The ledger is the sequenced, tamper-resistant record of all state transitions in the HLF. As shown in Fig.1, the ledger mainly consists of two parts: the blockchain file system and the status database [1]. The blockchain file system is a transaction log, which is composed of blocks. Each transaction is a result of chaincode invocation submitted by participating parties and results in a set of asset key-value pairs that can be committed to the ledger. The state database holds current value of a set of blockchain file system states, e.g. key-value pairs.
- **Chaincode**. Chaincode is software defining an asset or assets, and the transaction instructions for modifying the asset(s). Chaincode enforces the rules for reading or altering key-value pairs or other state database information.
- **Membership Services** (**MSP**). MSP authenticates, authorizes, and manages identities on a permissioned blockchain network. In addition, MSP creates channels between the organization.
- **Channel**. Channel is a private "subnet" of communication between two or more specific network blockchain nodes, to conduct private and confidential transactions. Each peer must be authenticated and authorized to join the channel [30].

B. IDENTITY-BASED SIGNATURE

Identity-based signature (IBS) allows users to use their identity (ID) as their public keys. In our system, IoT domains use IBS to filter access requests to prevent DDoS attacks.

We combine the role (user, IoT device, edge device) information and IoT domain information as the ID of an entity. Thus, the filter could distinguish whether the access request is initiated by its own domain or trusted domain users via verifying the IBS. Specifically, we use the device's ID number and its domain information as the IoT device's ID, that is, $devID = \{deviceID@domainA\}$. Similarly, we use the MAC address and its domain information as the edge device's ID, that is, $edevID = \{macAddress@domainName\}$. The user's ID consists of the user's name and domain information, such as $uID = \{userName@domainName\}$. Assuming that a user, Bob, and a camera are in the smart home named domain A. Bob's user ID is $uID = \{Bob@domainA\}$ and the camera's device ID is $devID = \{VID\_0AC8PID\_3340@domainA\}$.

Here, we suggest using Barreto-Libert-McCullagh-Quisquater (BLMQ) scheme (Li 2009) for our implementation. It comprises four algorithms.

- **BLMQ.Setup($\mathbf{k}, \mathbf{G_1}, \mathbf{G_2}, \mathbf{G_T}$) $\rightarrow$ params** : Given the security parameter k, and group $< G_1, G_2, G_T >$, which has the same prime order $q > 2^k$. Q is the generator of the group G_2. A mapping $\phi : G_2 \rightarrow G_1$, satisfies $P = \phi(Q)$, then defines $g = e(P, Q)$. Select $s \in z_q^*$ as the master key, calculate $Q_{pub} = sQ$. Select the hash function $H_1 : \{0, 1\}^* \rightarrow Z_q^*, H_2 : \{0, 1\}^* \times G_T \rightarrow Z_q^*$, then the system public the parameter $params =< G_1, G_2, G_T, e, P, Q, Q_{pub}, \tilde{O}, H_1, H_2 >$
- **BLMQ.SkGen(ID) $\rightarrow$ $\mathbf{sk_{ID}}$** : the KGC generate a private key for the $user_i's$ identity ID, $sk_{ID} = (H_1(ID) + s)^{-1}P$
- **BLMQ.Sign($\mathbf{sk_{ID}}, \mathbf{m}$) $\rightarrow <$ h, S $>$**: The signature of message m is calculated as follows

  - select random number $x \in Z_q^*$, calculate $r = g^x$
  - calculate $h = H_2(m, r)$
  - calculate $S = (x + h)sk_{ID}$

  The signature is $< h, S >$
- **BLMQ.Verify(ID, $<$ h, S $>$) $\rightarrow$ 0/1** : Verify whether $h = H_2(m, e(S, H_1(ID)Q + Q_{pub})g^{-h}$ holds. If it holds, the $< h, S >$ is a valid signature. Otherwise, it's invalid.

C. ATTRIBUTE-BASED ACCESS CONTROL

The attribute-based access control (ABAC) model generalizes DAC, MAC, and RBAC, and is more flexible, scalable, and secure in dynamic environments [25]. In addition, ABAC is particularly suitable for organizations where individuals rotate frequently [26]. Therefore, the ABAC model satisfies the lightweight, massive, dynamic, and real-time characteristics of IoT system.

The definition of attribute-based access control is as follows:

1) S, R, and E represent three entities respectively: subject, accessed resource, and environment.

2) $SA_k(1 \leq k \leq K)$, $RA_m(1 \leq m \leq M)$, $EA_n(1 \leq n \leq N)$ represents the attributes of the subject, accessed resource, and environment respectively.

3) ATTR(s), ATTR(r), ATTR(e) represent the attribute assignment relationship of subject, resource, and environment respectively, such as:

$$ATTR(s) \in SA_1 \times SA_2 \times \cdots \times SA_K$$
$$ATTR(r) \in RA_1 \times RA_2 \times \cdots \times RA_M$$
$$ATTR(e) \in EA_1 \times EA_2 \times \cdots \times EA_N$$

4) Generally, a policy role is used to determine whether a subject s can access resource r in a specific environment e. The policy can be expressed as a function that returns a boolean value with the attribute of s, r, and e as input parameters.

$$Rule : Can_{access}(s, r, e) \leftarrow f(ATTR(s), ATTR(r), ATTR(e))$$

If the return value of the function f is true, the subject is permitted to access resource r. Otherwise, the access request is denied.

D. POLICY EXECUTION FRAMEWORK

The policy execution framework describes the relationship between entities and the process of the access control. As shown in Fig.2, the framework consists of policy enforcement point (PEP), policy decision point (PDP), policy administration point (PAP), policy information point (PIP) [18].

PEP receives an access request and sends the request to the context handle, which transforms the request into the standard request format and sends the standard request to PDP. The PDP retrieves the policy related to the request in the PAP and then make a policy decsion for the request. During the evaluation process, the context handle sends the attributes query messages to PIP, which will search the attribute value from the corresponding attribute authority and then return the attributes to the context handle. After the evaluation, the PDP returns the policy decision to the context handle, which makes a corresponding response to the PEP. The PEP executes the decision result, decided whether the access request is permitted or denied.

E. PRACTICAL BYZANTINE FAULT TOLERANCE

The PBFT can process as follows [36], [37]:

(1) Select a primary node from the whole network, which is responsible for determining whether to proccess a request.

(2) Pre-Prepare: Each node broadcasts request to the whole network. The primary node verifies the validity of the request and then broadcasts the valid request.

(3) Prepare: Once a node receives the request, it will process the request then broadcasts the response to the network.

(4) Commit: If a node receives 2f (f is the tolerable number of Byzantine nodes) responds from other nodes, and these responses are the same as its own respond, this node will broadcast a commit message to the network.

(5) Reply: Once a node receives $2f + 1$ pieces of commit messages, it will pack all transactions into a new block, and record it on the blockchain ledger.

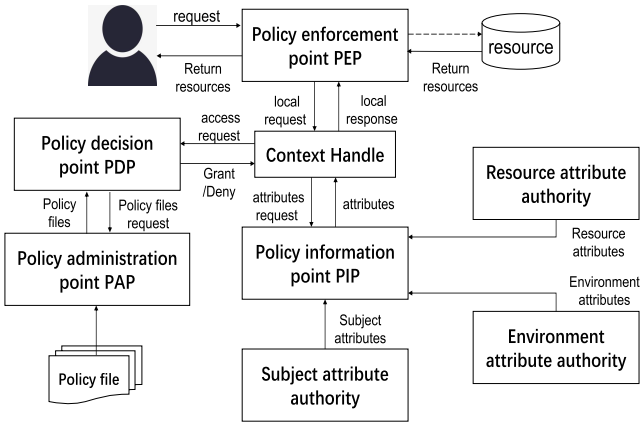

FIGURE 2. ABAC policy execution framework.

F. RELATED WORK

Authors in [21] proposed a smart contract-based access control for the IoT, wihch record access policy on the blockchain. However, it cannot take advantage of distributed damage-tolerant, decentralization features of blockchain. Authors in [22] use table structure to store access rights of users, with the increase of devices and users, the contents of the access rights table will increase rapidly. Framework FairAccess [23] introduces access token which contains the access right of users. It solves the high table query complexity problem and guarantees the privacy of the access policy. But this scheme requires each device to establish its access policy, resulting in contract redundancy and account management difficulties. Oscar Novo considered the redundancy of smart contract in [24], he proposed a scheme that only one access control smart contract is deployed on the blockchain, but this scheme cannot achieve fine-grained access for different devices. Wang *et al.* [27]. proposed a blockchain-based distributed access control system ADAC, using the attributed-based access control model. Islam and Madria [28], realizes IoT access control based on HLF and ABAC model and claims that it is the first time to implemented IoT access control based on permission blockchain. Putra proposed an architecture and mechanism of blockchain and smart-contract based access control for IoT [29]. The proposed blockchain architecture consists of the main blockchain network and several sidechain networks, but the author did not discuss the relationship between side chains and the interaction between the main chain and side chains in depth. To provide security and privacy for IoT, Dai *et al.* [33] proposed charge the buyer Ether to prevent denial of service (DoS) attacks. Groß *et al.* [34] introduced a novel method based on IPsec and TLS. Ukil *et al.* [35] proposed a privacy management method to evaluate the risk of sharing data with others. Liu *et al.* [38] proposed an access control system in IoT, which is based on Hyperledger Fabric blockchain framework. Tan *et al.* [39] proposed a blockchain-based access control scheme called BacCPSS for Cyber-Physical-Social System (CPSS) big data. Sun *et al.* [40] proposed a trusted and efficient cross-domain access control system based on blockchain and rbac model. As far as we know, the existing blockchain-based IoT access control system do not consider the security of the deployment of the blockchain. This article explores how to reduce the storage and computing burden of blockchain nodes from the perspective of resource constraints of IoT devices, so that more IoT devices can be used as blockchain nodes, that enhancing the invulnerability and attack cost of the blockchain system.

III. PROPOSED SYSTEM

This section presents the consists of the proposed system.

A. COMPONENTS

The IoT divide entities into a single authority domain according to their location and function, such as Alice's smart home. In the system, entities consist of IoT devices, edge devices, local blockchain ledgers, and users. With the local blockchain ledger, each IoT domain constructs its decentralized access control system. To support cross-domain access, we use MSP to build channels between IoT domains. Meanwhile, A KGC is used to generate the private key for the entities' ID. A database is used to decrease the storage overheads of the local blockchain ledger.

- **KGC** : KGC generates private keys for entities' ID (described in Section II.A). The ID of entities contains the IoT domain information, which is convenient for edge devices to determine whether the access request is legal.
- **Blockchain** : We use HLF to establish a local blockchain ledger for each IoT domain. Each local blockchain ledger records its IoT domain entities' attributes, policy files' digest, and access decisions. We deploy the local blockchain ledger on IoT devices.
- **IoT device** : IoT device owners define access control policies to decide who can access their resources. In addition, IoT devices are blockchain nodes. Meanwhile, IoT devices act as PDPs to make policy decisions.
- **Edge device** : The edge device is the boundary of an IoT domain and acts as an interface to the external world to access the resource within the IoT domain. In our system, we regard the edge devices as PEP. It responsible for filtering the access requests which are initiated by the IoT domain or trusted domain users and the accessed devices are in the IoT domain. Moreover, the edge device runs slectePDP (described in Section IV.B) algorithm to select PDPs to make the policy decisions, then collects the policy decisions returned by the PDPs, and coordinates PDPs use the PBFT algorithm to commit a final access decision, finally returns the final access decision to users.
- **MSP** : MSP establishes channels between IoT domains. Domains connected by a channel regard other domains as trusted domains.
- **Channel** : The channel is responsible for ledger sharing between IoT domains.
- **User** : Users are access requesters. They send access requests to corresponding PEP.
- **Database** : Database stores the policy files. IoT devices belongs to a local domain save their access policies in a file and then store the file in the database.

Algorithm 1 Smart Contract

```
function uploadAttr(attributes, ID, sig)
% Invoked by the entities to upload their attributes
% sig = BLMQ.sign(sk_ID, attributes)
    require(ID's domain ∈ IoT domain);
    if BLMQ.Verify(ID, sig) == true then
        stub.PutState(ID, attributes);
    else
        return 0;
function uploadPolicy(digest, devID, sig)
% Invoked by the IoT devices to upload their policy files' digest
% digest = hash || signature
% sig = BLMQ.sign(sk_devID, digest)
    require(devID's domain ∈ IoT domain)
    if BLMQ.verify(devID, sig) == true then
        stub.PutState(devID, digest);
    else
        return 0;
function getAttr(ID, PDP_ID,sig)
% Invoked by the PDPs to get the entity's attributes
% sig=BLMQ.sign(sk_PDP_ID,ID)
    require(PDP_ID's domain ∈ IoT domain)
    if BLMQ.Verify(PDP_ID, sig) == true then
        return attributes;
    else
        return 0;
function getPolicy(devID, PDP_ID, sig)
% Invoked by the PDPs to get the device's policy file's digest
% sig=BLMQ.sig(sk_PDP_ID, devID)
    require(PDP_ID's domain ∈ IoT domain)
    if BLMQ.Verify(PDP_ID, sig) == true then
        return digest;
    else
        return 0;
```

B. SMART CONTRACT

Smart contract (chaincode), together with the ledger, is the key component of blockchain system. In our system, a smart contract provides an interface for entities to obtain and upload attributes and policy files' digest onto the blockchain.

As shown in algorithm 1, before the system process access requests. IoT domain's entities upload their attributes onto their local blockchain ledger via invoking uploadAttr, and the IoT devices upload their access policy files' digest onto their local blockchain ledger via invoking uploadPolicy.

During the access control process, PDPs get attributes via invoking getAttr and get policy files' digest via invoking getPolicy, then PDPs make an off-chain policy decision respectively.

IV. SYSTEM DESIGN

In this section, we give the design details of the system and describe the policy decision point (PDP) selection algorithm.

A. SYSTEM ARCHITECTURE

Fig. 3 shows the architecture of our proposed system. We will now describe the System Setup, Request Control, PDPs execution, and PEP execution in our system.

•**System Setup** : HLF establishes a local blockchain ledger for each IoT domain. The IoT domain's entities upload their attributes and policy files' digest onto the local blockchain ledger and then store the policy files in the database. KGC generates the corresponding private key sk_{ID} for the IoT domain entities' ID. To realize cross-domain access control, MSP builds channels between IoT domains. IoT devices join the channel to share their local blockchain ledger.

•**Request Control** : Users send requests to the edge device. A request consists of the request number, the user's ID, the accessed device's ID, and the identity-based signature (IBS) signed by user's private key sk_{uID}, that is, request = {requestNum, uID, devID, BLMQ.Sign(sk_{uID},m)}, where m={requestNum, uID, devID}. The edge device distinguish whether the requester belongs to its IoT domain by the uID, and distinguish whether the resource belongs to its IoT domain or trusted domain by the devID. If so, the edge device verifies the signature BLMQ.verify(uID, BLMQ.Sign(sk_{uID}, m)). If the signature is valid, the edge device runs the policy decision point selection algorithm (described in IV.B). Finally, The edge device forwards the access request to the selected PDPs.

•**PDPs Execution** : If the request is a cross-domain access request, the selected PDP who joined the channel obtains corresponding attributes and policy file's digest via invoking getAttr and getPolicy, and retrieves the policy file from the database according to the digest, while the PDPs who are not in the channel get the attributes and policy from the channel PDP member. If the access request is an intra-domain access request, each selected PDP obtains corresponding attributes and policy file's digest via invoking getAttr and getPolicy, and retrieves the policy file from the database according to the digest. Then each PDP makes an off-chain policy decision. Finally, each PDP returns the result to the edge device. A result consists of the PDP's ID (IoT device's ID), the request number, the policy decision (permit/deny), and the IBS signed by PDP's private key sk_{devID}, that is, result={devID, requestNum, decision, BLMQ.sign(sk_{devID}, m)}, where m={devID, requestNum, decision}.

•**PEP Execution** : PDPs use PBFT consensus algorithm to commit a final access decision.

(1)Pre-Prepare: the edge device (PEP) collects the results result1 = {$dev1\_ID$, requestNum, decision, BLMQ.sign(sk_{dev1_ID}, m)}, result2 = {$dev2\_ID$, requestNum, decision, BLMQ.sign(sk_{dev2_ID}, m)}, $\cdots$ resultn = {$devn\_ID$, requestNum, decision, BLMQ.sign(sk_{devn_ID}, m)}, then verify their signatures and determine whether the results are returned by the selected PDPs via their ID. Finally, the edge device broadcasts the valid results of PDPs to the network.

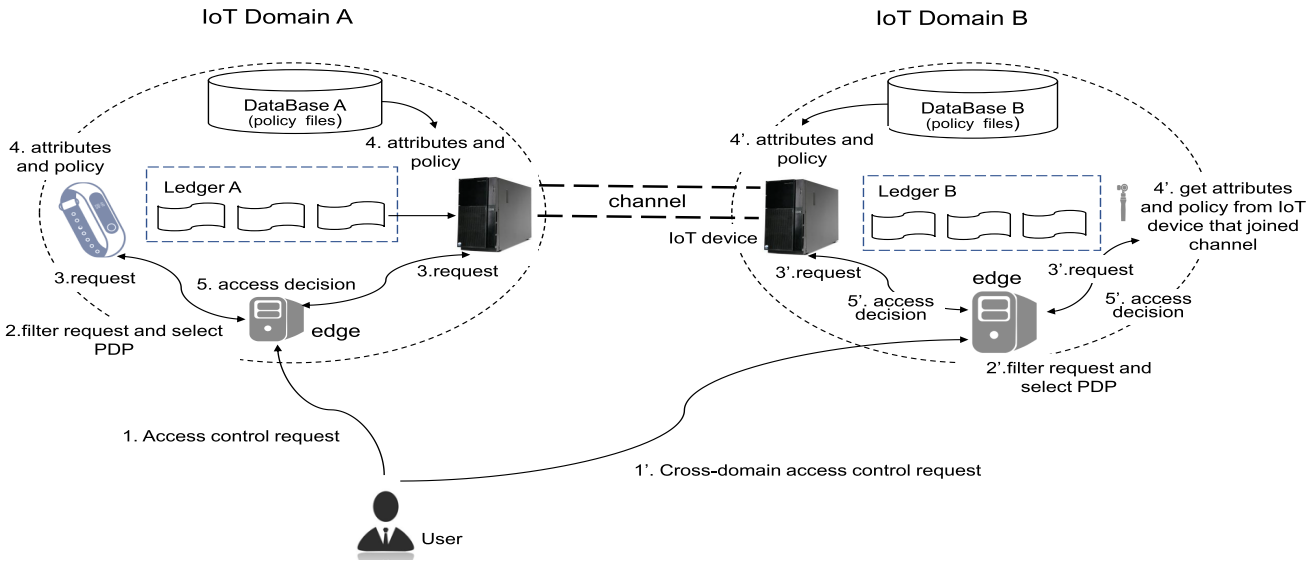

FIGURE 3. **Blockchain-based IoT access control system architecture. After the system setup, a user sends request to the edge device. Edge device filters the illegal request and forwards the request to selected PDPs. Each PDP makes a policy decision and commits a final access decision. Edge devices return the access decision to users.**

Algorithm 2 accessDecision

```
Input: PDPs' policy decision results
Output: bool % 0: deny; 1: permit
function accessDecsion(result[1],. . ., result[n])
    int count = 0;
    for j = 1; j <= n; j ++ {
        if result[j].decision == permit then
            count++;
    }
    if count > ⌊n/2⌋ then
        return 1;
    else
        return 0;
```

(2)Prepare: Once a PDP receives the results from the edge device, it makes a final access decision via invoking accessDecision, described in algorithm 2. Then the PDP broadcasts its final access decision to the network.

(3)Commit: If a PDP receives 2f final access decisions from other PDPs (we assumed that the number of PDP n $\geq$ 3f+1), and these final access decisions are same as its own final access decision, the PDP will broadcast a commit message to the network.

(4)Reply: Once the PEP receives 2f+1 pieces of commit message, the PEP returns a final access decision to the requester and requested deivce.

B. SELECT POLICY DECISION POINT

To select IoT devices who can make real-time policy decisions, we deployed the resource management on edge devices to detect the resource status of each IoT device. As shown in Fig.4, each IoT device runs a node management service

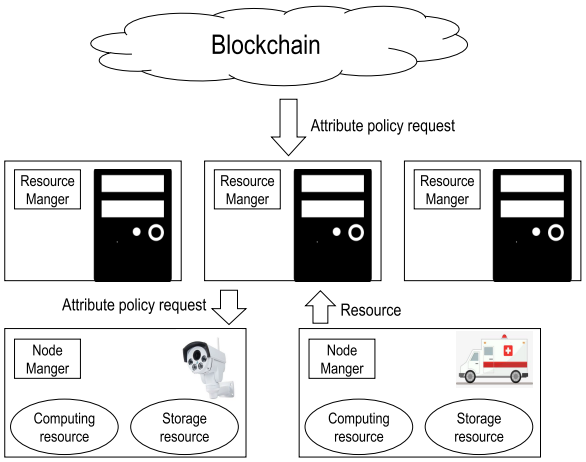

FIGURE 4. **Resource management.**

to monitor the resource usage and task execution states, and synchronize these states to the edge device.

As algorithm 3 shows, when the edge device receives an access request, it evaluates the computing and storage resources required to make the access decision. Then, the edge device selects IoT devices who have sufficient computing and storage resource as PDP, to make a real time, off-chain policy decision.

C. FEATURES OF OUR SYSTEM

The features of the proposed system are summarized as follows:

• **Decentralization**. There is no third-party authorities to store the attributes and policies. Moreover, most IoT device can act as a blockchain node. Considering IoT devices also have tasks in the IoT system, we only select a part of IoT

Algorithm 3 SelectPDP

```
Input: request, DevSet
Output: PDPs' ID set
% request={requestNum, uID, devID, sig}
% DevSet contains all IoT devices' ID belongs to the IoT domain
function SelectPDP
    attributes ← getAttr(uID)
    digest ← getPolicy(devID)
    policies ← ipfs get digest.hash
    res ← EvalResCons(request,attributes,policies)
    while (DevSet != null){
        if (ID.resource ≥ res) then
            PDPset.add(ID)
            DevSet.remove(ID)
    }
    return PDPset
```

devices with enough storage and computation resources to make the policy decision during one access control process epoch. In addition, the final policy decision is committed by PDPs via PBFT consensus algorithm.

• **Security**. The access request is signed by users, and the policy decision is signed by PDPs. Therefore, requests and policy decisions cannot be tampered or forged during transmission. In addition, the edge device filters the legal request via IBS that ensures edge devices filter illegal access requests. Thus, local blockchain ledger can resist DDoS attacks in some extent.

• **Lightweight and dynamic**. Each local blockchain ledger only records the attributes and access policy files' digest of entities who belogs to its IoT domain, which reduces the storage pressure of the blockchain nodes. Meanwhile, blockchain nodes only process their own IoT domain users' access requests, that decrease their computational overhead. At the same time, considering the dynamic feature of IoT devices, we only select blockchain nodes with enough residual resources as PDP to make an off-chain access decision.

V. PERFORMANCE EVALUATION

In this section, we give a performance evaluation of the proposed system, which may be broken up into two parts. Firstly, we studied the processing time of the system initial phase, which contains the System Setup. The Request Control, PDPs execution, and PEP execution constitute the execution phase are studied in the second part.

We have developed an IoT network using Raspberry Pi 3B+ devices. Raspberry Pi 3B+ devices can act as IoT devices and bundle many essential elements to perform as a humidity sensor, light, and camera. We divided the Raspberry Pi 3B+ devices into three IoT domains: IoT domain A (DO A), IoT domain B (DO B), and IoT domain C (DO C). DO A and DO B has four Raspberry Pi 3B+ devices, while DO C has 20 Raspberry Pi 3B+ devices, and all of them

TABLE 1. Devices' configuration.

Device	Operating System	CPU	Memory
RASPBERRY PI 3B+	Micro-control Linux	ARM Cortex-A53	LPDDR2 SDRAM 1GB
Lenovo Tianyi 510 pro	Ubuntu 18.04	LST Inter core i7@3.20GHZ	8G

are connected to a PC through a USB port. In each IoT domain, the PC act as an edge device. In addition, we use another PC to simulate a user device to send an access request. The configurations of devices are described in Table 1. Our blockchain network has been implemented in Hyperledger fabric (hlf) v1.2. We establish blockchain ledgers for three IoT domains respectively and create a channel between DO A and DO B. In each IoT domain, we assume that all IoT devices are peer nodes who maintain the blockchain ledger and installed a software, named Casbin [31], to make an off-chain ABAC policy decision using PML language. User devices using fabric client SDK to interact with the blockchain. They can upload thier attributes onto the local blockchain ledger. To evaluate the time cost of store policy files in IPFS, we build an IPFS network composed of three PC nodes.

A. INITIAL PHASE

During the initial phase, the KGC generates private keys for entities' ID. The MSP builds a channel between DO A and DO B. Entities of each IoT domain uploads their attributes and policy files' digest onto their local blockchain ledger via invoking uploadAttr and uploadPolicy, and then store policy files in IPFS. As shown in Table 2, we create 3 attributes for each entity and create 2 different ABAC policies for each IoT device.

•**Attribute storage efficiency**

After deploy the smart contract, we used fabric client SDK and shell script to test the attribute storage efficiency via invoking uploadAttr. As shown in Fig.5, we obtained the time cost of running 1000, 2000, $\cdots$, 9000 times uploadAttr. The time for uploading attributes on the local blockchain ledger increases almost linearly with the number of attributes (i.e. number of IoT domain entities).

•**Policy storage efficiency**

The policy storage experiment can be divided into three parts: (1) IoT devices calculate the hash value and signature of the policy files; (2) IoT devices store the policy files in the off-chain database; (3) IoT devices store the policy files' digest (hash value, signature) onto the local blockchain ledger via invoking uploadPolicy. We use SHA256 as the hash function, BLMQ as the digital signature, and IPFS as the off-chain database to test the policy storage efficiency. The time cost for storing 5k, 11.4M, 136.7M, and 1.1G access policy files in the blockchain ledger are 64.497ms, 311.875ms, 26142.493ms, and 220309.111ms respectively.

TABLE 2. Example of experimental parameters.

Entity	ID	Attributes	Polices
user	Bob@domainA	Type: student; Group: A; Credit: 50	Wirte: sub.Group==A||B & sub.Credit $\geq$ 70 Read: sub.Group==A
IoT device	a020d3_bdf09c5@domainB	Type: camera; Group: B; Credit: 70	
Edge device	a020d3_ehg08be@domainA	Type: gateway; Group: A	

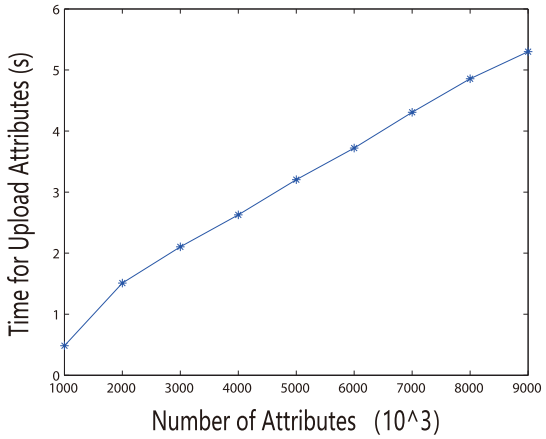

FIGURE 5. Time cost for entity uploads different numbers of attributes.

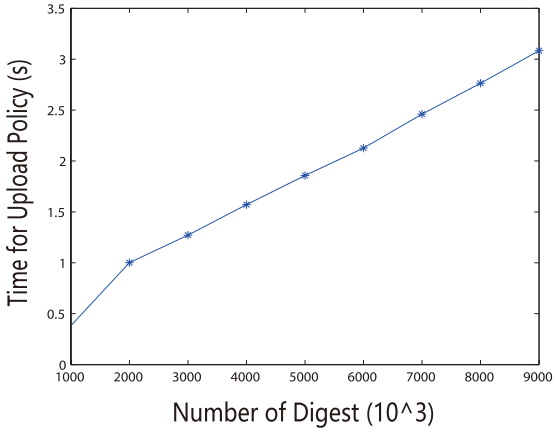

FIGURE 6. Time cost for entity uploads different numbers of policy files' digest.

Finally, The same as the attribute storage experiment, we test the user's policy files' diest storage efficiency via invoking uploadPolicy. We obtain the time cost of uploadPolicy with 1,000, 2,000, 3,000, $\cdots$, 9,000 running times, as shown in Fig.6. The time for uploading policy files' digest onto the local blockchain ledger increases almost linearly with the number of policy files' digests (IoT devices).

B. ACCESS CONTROL PHASE

During the access control phase, the edge device filters the illegal access request and selects the PDPs. PDPs make off-chain policy decisions and run the PBFT consensus algorithm to record the request and final access decision on the local blockchain ledger.

TABLE 3. Request filter performance and efficiency.

User,Edge,IoT	Filter Result	Filter Cost (s)	Communication Cost (s)
A,A,A	forward	0.0418	0.0032
A,A,B	forward	0.0442	0.0029
A,A,C	deny	0.0544	0.0031
$*$,A,+	deny	0.0618	0.0038

$*$ represents B and C; + represents A, B, and C

TABLE 4. Time cost for database retrieve and cipher algorithm execution (s).

	5K	11.4M	136.7M	1.1G
IPFS	0.005	0.028	0.607	18.333
SHA256	0.054	0.421	3.275	22.802
BLMQ.Verify	0.161	0.248	0.744	57.728

TABLE 5. Comparison between our system with other blockchain-based IoT access control system.

Feature	Our System	Fan's System [37]	Dorri's System [20]	Zhang's System [21]
Anti-DDoS	✓	×	×	×
Cross-Domain	✓	×	✓	✓
Lightweight	✓	✓	×	×

•Access request filter

We evaluated the access request filter performance and efficiency by sending different kinds of request to DO A's edge device. Concretely, we simulate requesters (i.e. belong to DO A, DO B, and DO C) send requests to access different IoT devices (i.e. belong to DO A, DO B, and DO C), and assume that there is no channel connect between DO A and DO C. The filter result, time consumption of filter, and communication are listed in Table 3.

•Attribute query efficiency

We use the fabric shell script to test the attribute query efficiency of PDPs via invoking getAttr. Specifically, we consider two different scenarios (i.e. intra-domain, cross-domain). As shown in Fig.7, we obtained the time cost of getAttr with 1000, 2000, $\cdots$, 9000 running times. It takes about 0.4ms for PDPs to obtain an attribute from the local IoT domain's blockchain ledger. While, it cost about 0.6ms for PDPs to obtain an attribute from one PDP who has joined the channel (actually, caused by the transmission delay between PDPs).

•Policy query efficiency

We use the fabric shell script to obtain both the intra-domain and cross-domain policy files' digest via invoking getPolicy to evaluate the policy query efficiency of PDPs.

TABLE 6. Communication and computation costs of IoT access control.

	Communication Cost	Computation Cost
System Setup	$n\|attr\| + m\|hash\| + nq\|G_1\|$	$mT_{BL} + lT_{ch} + nT_{BLMQ.skGen} + n.uploadAttr + mT_{IPFS.add} + m.uploadPolicy$
Request	$\|num\| + 2\|ID\| + q + q\|G_1\|$	$T_{BLMQ.Sign} + T_{BLMQ.Verify} + T_{selectPDP}$
PDPs Execution	$t\|attr\| + \|hash\| + \|File\| + \|ID\| + \|num\| + \|decision\| + q + q\|G_1\|$	$T_{policy\_decision} + T_{BLMQ.Sign} + t.getAttr + t.getPolicy + +T_{IPFS.get}$
PEP Execution	$s(\|ID\| + \|num\| + \|decision\| + q + q\|G_1\|) + s\|commit\| + \|decision\|$	$sT_{BLMQ.Verify} + sT_{accessDecision} + T_{PBFT}$

$*$ n is the number of entities and m is the number of domain, $|attr|$ is the length of a attribute of an entity, $|hash|$ is the length of a hash digest, q and $|G_1|$ is the parameter of BLMQ, $|num|$ is the length of the number of request, $|File|$ is the size of a policy file, $|commit|$ is the length of the PBFT commit message, $|decision|$ is the length of an access decision, s is the number of selected PDP, l is the channel number build on the system, uploadAttr, uploadPolicy, getAttr and getPolicy are smart contract algorithms in our system.

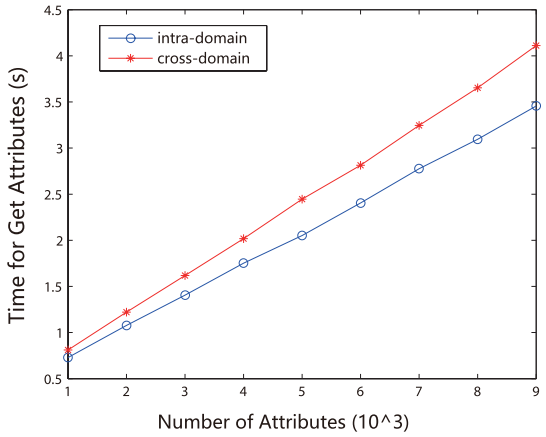

FIGURE 7. Time cost for PDP obtains different numbers of attributes.

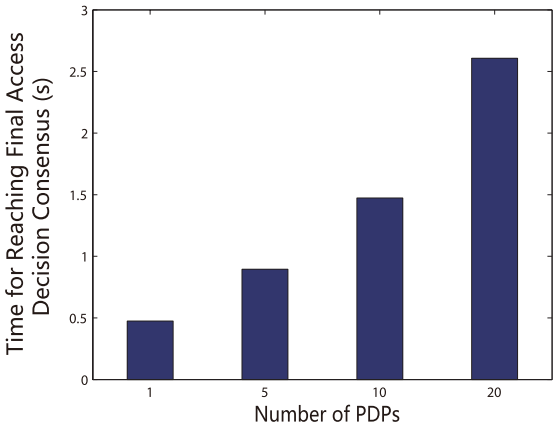

FIGURE 9. Time cost for reaching final access decision consensus.

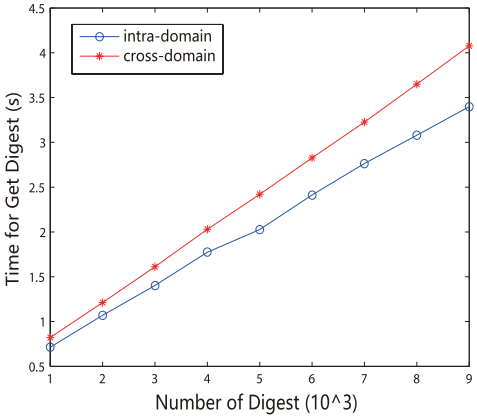

FIGURE 8. Time cost for PDP obtains different numbers of policy files' digest.

Fig.8 shows the time cost of getPolicy with 1000, 2000, $\cdots$, 9000 running times.

PDPs use the obtained hash of the digest to retrieve policy files stored in IPFS. Then, PDPs calculate the retrieved policy files' hash and compare it with the hash obtained from the blockchain ledger. Finally, PDPs verify the policy files' signature. We list the executing time in Table 4.

•PDP efficiency

To test the average execution time and memory cost for the access policy decision, we execute the same access policy 1000 times on one PDP. Experiments show that it takes a PDP 0.009831ms and 2.678KB to execute ABAC decisions for per-access policy.

•Access decision efficiency

We also use hlf to test the time cost for reaching a final access decision consensus. The experiment is run on different numbers of PDPs. As shown in Fig.9, the time for reaching consensus increases almost exponentially with the number of PDPs.

C. SYSTEM COMPARISON

To show the advantage of our system, we compared the proposed system with the existing blockchain-based IoT accesss control system in terms of the following features: "Anti-DDoS" means that the illegal access request will not affact the TPS of the blockchain, "Cross-Domain" means the system supports users to access resources accross domains, and we

evaluate the "Lightweight" feature by determining whether these systems have delegate nodes. Table 5 illustrates our system is more secure than other systems in terms of Anti-DDoS attacks. In addition, our system supports cross-domain access control. Moreover, our system has no delegate nodes.

D. SYSTEM PERFORMANCE

To show the utility of the system, we evaluate its communication and computation costs of each phase. As shown in table 6, the communication and computation cost is affected by the number of entities loacated in each domain.

VI. CONCLUSION

In this article, we propose a blockchain-based IoT access control system. We use ABAC model and HLF to establish a lightweight local blockchain ledger for each IoT domain. In addition, we build channels to realize cross-domain access control. To prevent the DDoS attacks, each edge device uses the ID of requestor to filter the illegal request. We also design a policy decision point selection algorithm to make off-chain and real-time policy decisions. Finally, we implement and evaluate the proposed system to demonstrate its practicality.

REFERENCES

[1] *IoT Security White Paper*, 2018.
[2] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of Things security: A survey," *J. Netw. Comput. Appl.*, vol. 88, pp. 10–28, Jun. 2017.
[3] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, May 2018.
[4] A. Gabillon, R. Gallier, and E. Bruno, "Access controls for IoT networks," *Social Netw. Comput. Sci.*, vol. 1, no. 1, pp. 1–13, Jan. 2020.
[5] H. Shen and S. Liu, "A context-aware capability-based access control framework for the Internet of Things," *J. Wuhan Univ.*, vol. 60, pp. 424–428, 2014.
[6] L. J. Wu, K. Meng, S. Xu, S. Li, M. Ding, and Y. Suo, "Democratic centralism: A hybrid blockchain architecture and its applications in energy Internet," in *Proc. IEEE Int. Conf. Energy Internet*, Apr. 2017, pp. 176–181.
[7] L. Yong, "Identity-based non-delegatable universal designated verifier signature," *J. Tsinghua Univ.*, vol. 49, pp. 2133–2137, 2009.
[8] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://bitcoin.org/bitcoin.pdf
[9] *Ethereum Smart Contract Platform*. [Online]. Available: https://www.ethereum.org/
[10] *EOS.IO Technical White Paper V2*. [Online]. Available: https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md
[11] C. Cachin, "Architecture of the hyperledger blockchain fabric," in *Proc. Workshop Distributed Cryptocurrencies Consensus Ledgers*, 2016, pp. 1–4.
[12] K. X. Wu and H. Y. Gao, "Attribute-based access control for Web service with requester's attribute privacy protected," in *Proc. Int. Conf. Informational Technol. Environ.*, 2008, pp. 932–936.
[13] M. Hemdi and R. Deters, "Using REST based protocol to enable ABAC within IoT systems," in *Proc. IEEE 7th Annu. Inf. Technol., Electron. Mobile Commun.*, Oct. 2016, pp. 1–7, doi: 10.1109/IEMCON.2016.7746297.
[14] Q. Han and J. Li, "An authorization management approach in the Internet of Things," *J. Inf. Comput. Sci.*, vol. 9, no. 2, pp. 1705–1713, 2012.
[15] J. Wu, M. Dong, K. Ota, J. Li, and B. Pei, "A fine-grained cross-domain access control mechanism for social Internet of Things," in *Proc. IEEE 11th Int. Conf Ubiquitous Intell. Comput. IEEE 11th Int. Conf Autonomic Trusted Comput. IEEE 14th Int. Conf Scalable Comput. Commun. Associated Workshops*, Dec. 2014, pp. 666–671, doi: 10.1109/UIC-ATC-ScalCom.2014.140.
[16] H. Ouechtati, N. B. Azzouna, and L. B. Said, "Towards a self-adaptive access control middleware for the Internet of Things," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, vol. 1, Jan. 2018, pp. 545–550, doi: 10.1109/ICOIN.2018.8343178.
[17] K. San and L. Yin, "Attribute-role-based hybrid access control in the Internet of Things," in *Proc. Asia–Pacific Web Conf.*, 2014, pp. 333–343.
[18] F. Liang, L. H. Yin, and Y. C. Guo, "A survey of key technologies in attribute-based access control scheme," *Chin. J. Comput.*, vol. 40, no. 7, pp. 1680–1698, 2017.
[19] Y. Luo, Q. Shen, and Z. H. Wu, "PML: An interpreter-based access control policy language for Web services," *CoRR*, vol. abs/1903.09756, 2019.
[20] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2017, pp. 618–623.
[21] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, "Smart contract-based access control for the Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1594–1605, Apr. 2019.
[22] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "FairAccess: A new blockchain-based access control framework for the Internet of Things," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 5943–5964, Dec. 2016.
[23] O. Novo, "Blockchain meets IoT: An architecture for scalable access management in IoT," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1184–1195, Apr. 2018.
[24] *A Blockchain Platform for the Enterprise*. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/index.html
[25] B. Dundua and M. Rukhaia, "Towards integrating attribute-based access control into ontologies," in *Proc. IEEE 2nd Ukraine Conf. Electr. Comput. Eng. (UKRCON)*, Lviv, Ukraine, Jul. 2019, pp. 1052–1056.
[26] V. C. Hu, D. F. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, "Guide to attribute based access control (ABAC) definition and considerations," NIST Special Publication, Tech. Rep. 800-162, Jan. 2014.
[27] P. Wang, Y. Yue, W. Sun, and J. Liu, "An attribute-based distributed access control for blockchain-enabled IoT," in *Proc. Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Barcelona, Spain, Oct. 2019, pp. 1–6.
[28] M. A. Islam and S. Madria, "A permissioned blockchain based access control system for IOT," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Atlanta, GA, USA, Jul. 2019, pp. 469–476.
[29] D. R. Putra, B. Anggorojati, and A. P. P. Hartono, "Blockchain and smart-contract for scalable access control in Internet of Things," in *Proc. Int. Conf. ICT Smart Soc. (ICISS)*, Bandung, Indonesia, Nov. 2019, pp. 1–5.
[30] *A Blockchain Platform for the Enterprise/Architecture Reference/Channels*. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/channels.html
[31] *Casbin/Casbin*. [Online]. Available: https://github.com/casbin/casbin
[32] C. F. Liao, C. C. Hung, and K. Chen, "Blockchain and the Internet of Things: A software architecture perspective," in *Business Transformation through Blockchain*, vol. 2, H. Treiblmaier and R. Beck, Eds. London, U.K.: Springer Nature-Palgrave Macmillan, ch. 3, Jan. 2019.
[33] W. Dai, C. Dai, K.-K.-R. Choo, C. Cui, D. Zou, and H. Jin, "SDTE: A secure blockchain-based data trading ecosystem," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 725–737, 2020.
[34] H. Groß, M. Hölbl, D. Slamanig, and R. Spreitzer, "Privacy-aware authentication in the Internet of Things," in *Proc. 14th Int. Conf. Cryptol. Netw. Secur. (CANS)*, Marrakesh, Morocco, Dec. 2015, pp. 32–39.
[35] A. Ukil, S. Bandyopadhyay, and A. Pal, "IoT-privacy: To be private or not to be private," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada, Apr. 2014, pp. 123–124.
[36] K. Zheng, Y. Liu, C. Dai, Y. Duan, and X. Huang, "Model checking PBFT consensus mechanism in healthcare blockchain network," in *Proc. 9th Int. Conf. Inf. Technol. Med. Educ. (ITME)*, Hangzhou, China, Oct. 2018, pp. 877–881.
[37] K. Fan, Q. Pan, K. Zhang, Y. Bai, S. Sun, H. Li, and Y. Yang, "A secure and verifiable data sharing scheme based on blockchain in vehicular social networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 5826–5835, Jun. 2020.
[38] H. Liu, D. Han, and D. Li, "Fabric-IoT: A blockchain-based access control system in IoT," *IEEE Access*, vol. 8, pp. 18207–18218, 2020, doi: 10.1109/ACCESS.2020.2968492.

[39] L. Tan, N. Shi, C. Yang, and K. Yu, "A blockchain-based access control framework for Cyber-Physical-Social system big data," *IEEE Access*, vol. 8, pp. 77215–77226, 2020, doi: 10.1109/ACCESS.2020.2988951.
[40] S. Sun, S. Chen, and R. Du, "Trusted and efficient cross-domain access control system based on blockchain," *Sci. Program.*, vol. 2020, pp. 1–13, Aug. 2020.

SHUANG SUN is currently pursuing the Ph.D. degree with the School of Microelectronics, University of Chinese Academy of Sciences. Her research interests include cryptography, blockchain, the IoT, and distributed computing.

RONG DU received the B.Sc. and M.Eng. degrees from the Beijing University of Technology in 2013 and 2016, respectively. She is currently a Researcher with the Institute of Microelectronics, Chinese Academy of Sciences. Her current research interests include computer vision and pattern recognition.

SHUDONG CHEN received the B.Sc. and M.Sc. degrees from Southwest Jiao Tong University in 1999 and 2002, respectively, and the Ph.D. degree in computer science from Shanghai Jiaotong University in 2006. She was an Assistant Professor with the Department of Mathematics and Computer Science, Eindhoven University of Technology, from 2009 to 2011. She is currently a Professor with the Institute of Microelectronics, Chinese Academy of Sciences. Her main research interests include big data, cloud computing, and artificial intelligence.

WEIWEI LI is currently pursuing the Ph.D. degree with the School of Microelectronics, University of Chinese Academy of Sciences. His research interests include the IoT, distributed computing, and edge computing.