Received March 9, 2020, accepted March 22, 2020, date of publication April 6, 2020, date of current version April 27, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2985762

AuthPrivacyChain: A Blockchain-Based Access Control Framework With Privacy Protection in Cloud

CAIXIA YANG[1], LIANG TAN[1,2], NA SHI[1], BOLEI XU[3], YANG CAO[4], AND KEPING YU[5,6], (Member, IEEE)

[1]College of Computer Science, Sichuan Normal University, Chengdu 610101, China
[2]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[3]College of Information Engineering, Shenzhen University, Shenzhen 518060, China
[4]Department of Social Informatics, Kyoto University, Kyoto 606-8501, Japan
[5]Global Information and Telecommunication Institute, Waseda University, Tokyo 169-8050, Japan
[6]Shenzhen Boyi Technology Company, Ltd., Shenzhen 518125, China

Corresponding author: Keping Yu (keping.yu@aoni.waseda.jp)

This work was supported in part by the National Natural Science Foundation of China under Grant 61373162, in part by the Sichuan Science and Technology Support Project under Grant 2019YFG0183, and in part by the Japan Society for the Promotion of Science (JSPS) Grants-in-Aid for Scientific Research (KAKENHI) under Grant JP18K18044.

ABSTRACT Cloud is a computing model that provides sharing and supports ubiquitous on-demand access computing, providing new data processing and services for many industries, significantly reducing user computing and storage costs, and improving ease of use. With the development of cloud-scale and intensification, cloud security has become an essential issue in the field of cloud computing. Access control is one of the critical security technologies for protecting sensitive data stored in the cloud by enterprises and individuals. Since the centralized access control mechanism is adopted in the cloud, the sensitive data in the cloud are easy to be tampered with or leaked by hackers or cloud internal managers. To address this issue, we propose a blockchain-based access control framework with privacy protection called AuthPrivacyChain. Firstly, we use the account address of the node in blockchain as the identity, and at the same time, redefine the access control permission of data for the cloud, which is encrypted and stored in blockchain. After that, we design processes of access control, authorization, and authorization revocation in AuthPrivacyChain. Finally, we implement AuthPrivacyChain based on enterprise operation system (EOS), and the results show that AuthPrivacyChain can not only prevent hackers and administrators from illegally accessing resources, but also protect authorized privacy.

INDEX TERMS Cloud computing, cloud security, access control, blockchain, privacy protection.

I. INTRODUCTION

Cloud computing [1]–[3], as a new computing model, can provide users with services of omnipresence, and reduce the cost of user storage and computing, and improve the convenience of use, so more and more businesses and individuals choose to store data in cloud. However, with the development of cloud computing scale and intensification, research on fog computing and edge computing has also gradually risen [4]–[6], cloud security issues have become an important factor restricting cloud computing development [7]–[9]. In July 2017, the cloud security alliance (CSA) published a ''Security Guidance for Critical Areas of Focus in Cloud Computing v4.0'' [10], which identified 14 cloud computing security focus areas, among which access control is one of the core technologies of cloud security. Meanwhile, access control is also the current research hotspot [11]–[13], the purpose is to use access control to prevent resources stored in cloud from being accessed or stolen by illegal users. The main three service systems of cloud computing, infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS), all need to protect relevant resources through access control [14], so access control plays an important role in cloud.

Compared with the traditional computing model, the computing and storage mode of cloud computing have undergone

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

many changes, which are mainly reflected in the following five aspects: 1) users cannot control the resources in cloud; 2) lack of trust between users and cloud; 3) migration technology may cause data to change the security domain; 4) multitenant technology makes the access subject to be redefined; 5) virtualization technology may lead resources to be stolen on the same physical device [15]. With these challenges, a lot of researches in cloud access control has appeared in academia [16]–[21] and the industry has also tried to implement existing access control technologies. However, both of them have centralized storage and management modes for identity, key, authority, authentication information, etc [22]. So the access control technology still has two aspects problem of security and privacy:

1) An external attacker attacks the trusted center, tamper with the authorized database stored on the central server, and illegally access or steal the resources stored by users in cloud.
2) The system administrator of cloud manages the authorization database and has the right to access and manage the resources, so a malicious system administrator of cloud may take advantage of the privilege to illegally access the resources or tampering the authorization database to illegally access.

In this paper, we propose AuthPrivacyChain - a blockchain-based access control framework with privacy protection in cloud to solve the above problems. Our contributions are as follows:

- Decentralized access control architecture. Authprivacychain uses the decentralized and tamper-proof blockchain [23]–[25] to store access control rights and uses blockchain account address as the identity, then design the access control, authorization and authorization revocation process.
- Authorized privacy protection. Due to the transparency of blockchain, it is easy to disclose users' privacy. Authprivacychain encrypts and stores access control rights in blockchain, effectively protecting the privacy of users.
- Security. Authprivacychain can not only guarantee the confidentiality, integrity, availability, authenticity and accountability of the resources, but also resist various external and internal attacks.

Paper organization. The section II introduces related work. Section III introduces the problem and attack model. Section IV Introduces AuthPrivacychain's access control process, authorization process and authorization revocation process in detail. Section V analyzes the security of the AuthPrivacyChain. Section VI is experiments and performance tests. Finally, summarizes in Section VII.

II. RELATED WORK

In this section, we discuss the current research status of traditional cloud computing access control and then discuss the research status of using blockchain combined with cloud, especially to solve cloud security issues, including cloud access control issues. Finally, makes a brief summary of the current status of cloud access control research.

In the era of big data, the security of information content and storage has received much attention [26], [27], as one of the important means to solve cloud security in the field of cloud computing, there are many research achievements in cloud access control, mainly includes three aspects. First, in terms of the access control model of cloud computing, including task-based, attribute-based, usage control (UCON) based and Bell-LaPadula (BLP) based access control models. For attribute-based access control, [28] combined role-based access control (RBAC) and attribute-based access control (ABAC), the upper layer was composed of RBAC to support model validation and review, while the lower layer used the characteristics of ABAC to automatically create RBAC models. For UCON-based access control, [29] proposed a new UCON model to solve subject attribute variability and obligation processing in cloud. For BLP-based access control, [30] proposed a virtual machine system based on the BLP model, which realized simple virtual machine isolation and efficient sharing. Second, in terms of access control based on the attribute-based encryption (ABE) cryptosystem, [31] proposed an access control framework that could authenticate users and protected the privacy of data in cloud, and ABE encryption storage of data could be carried out after the authentication. Reference [32] proposed a multi-authorization center access control model and certificate authority (CA) managed unique user identifier (UID) and the unique authorization identifier (AID) for each user. Finally, In terms of multi-tenancy and virtualized access control in cloud, [33] proposed a multi-tenancy based on access control model, tenants were only responsible for managing their own access control, while cloud service provider (CSP) was responsible for adding, deleting and managing cloud tenants, and was responsible for related security issues. Reference [34] proposed and designed role-based multi-tenant access control (RB-MTAC). This model can determine user identity and applicable roles through user identity management and achieves data and program isolation through efficient management of tenant access rights, aiming to improve multi-tenant security and privacy in cloud. Reference [20] proposed a hypervisor-based multi-tenant access control mechanism: CloudPolice, which used the hypervisor to dynamically coordinate the virtual machine access control policies.

With the rise of blockchain [35], [36], there are a lot of research achievements that combine blockchain technology with cloud technology. First, in the service combination, [37] proposed a service composition strategy based on the service overlay networks (SON) theory and designed an efficient service path generation algorithm across the service overlay layer. Second, in the service fees, [38] proposed a blockchain concept of market negotiation, aiming to solve the problem of multiple rounds of a bilateral negotiation between consumers and cloud service transactions in the future. Reference [39] introduced the fair payment framework BPay of cloud computing outsourcing services based on the blockchain, which was compatible with bitcoin blockchain and ethereum blockchain.

In addition, research on the use of blockchain to solve cloud security has become popular, but most of them are research on specific security aspects. First, in terms of data storage, [40] proposed a distributed cloud storage security architecture, and before uploading files, the files were divided into encrypted data blocks and then the design of a genetic algorithm designed to solve the problem of file copy placement. According to the European SUNFISH project data integrity problem, [41] proposed a used in cloud database design based on the blockchain, to ensure data integrity. Reference [42] proposed a safe system of distributed data storage and keyword search service BlockDS to solve the traditional reliance on a trusted third party in cloud storage as large storage providers. Second, in terms of data sources, [43] proposed a blockchain-based cloud data source framework to solve the problem of reliability data sources in cloud platforms by using blockchain consensus mechanism. Reference [44] proposed a distributed trusted cloud data source system to prevent tampering by collecting and verifying cloud data sources and embedding source data into blockchain transactions. Reference [45] designed a proof-of-stake (PoS) based coherence protocol CloudPoS to solve the problem of a consistency model based on an encrypted stream in the traditional data source system and improve security. Finally, in terms of deposit certificate, [46] proposed a cloud data deletion protocol to solve the behavior of tampering users by tampering with data deletion results when cloud server is not trusted. Reference [47] proposed a cloud computing electronic forensics model to improve evidence preservation based on merkle tree and formula algorithm, aiming at solving the centralized electronic forensics in cloud computing environment. Reference [48] combined blockchain and cryptographic signature techniques to propose a cloud forensics scheme, but the scheme relied too heavily on trusted center nodes CA and provenance auditor (PA).

However, the current research using blockchain to solve cloud access control is still in its infancy. Reference [49] proposed a system to solve the medical data sharing problem of medical big data servers in the untrusted environment, the two-layer blockchain designed by the system was based on a completely consistent mechanism. Reference [50] proposed a multi-user system prototype to solve the problem of access control of data in the untrusted cloud and design a set of cryptographic protocols to ensure key privacy. Reference [51] proposed an ABE access control in cloud, but ABE was not extensible when the user revoked.

In conclusion, although the academic circle has made a lot of research achievements in the above, there are still many shortcomings in the related research on using blockchain to solve cloud access control, especially there are few research achievements in cloud access control with privacy protection.

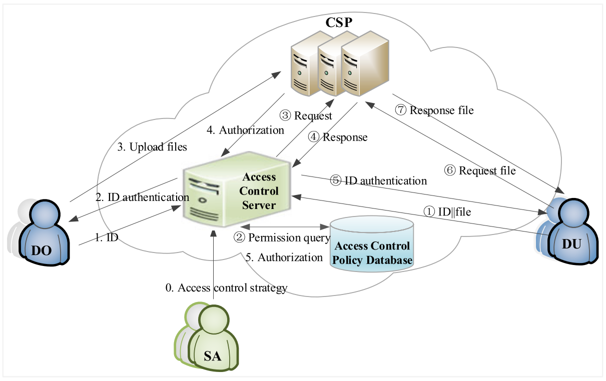

FIGURE 1. Cloud access control framework.

III. PROBLEM

Whether it is the three types of access control methods in cloud studied by academic or cloud access control actually used by industry, there are two common characteristics:

- **One or more trusted centers.** The three types of access control proposed by the academic, including traditional cloud access control models, encryption-based access control models, and virtualized access control, all require one or more trusted centers to store identities, keys, authorization rights, etc. First, in cloud access control model, the UCON-based model requires trusted centers to store access rights and permissions-related obligations and conditions. Second, in encryption-based access control, ABE-based access control requires one or more trusted centers to manage and distribute the keys. Finally, in virtualization access control, store the access control rights of virtual machines (VMs) in a trusted center. In addition, the cloud access control actually used in the industry requires a trusted center to store user identity information and access rights.
- **Internal trusted system administrator (SA).** Cloud provides three typical services for users: IaaS, PaaS and SaaS. Although they provide different services, all of them need internal trusted SA to manage access policies. For example, IaaS mainly provides users with computing power and storage space and requires SA to monitor and manage access to infrastructure environments. PaaS mainly provides users with a platform for developing and executing applications, which requires SA monitors and manages the access control of the platform; SaaS mainly provides users with the applications they need and also requires SA to manage and maintain the access control policies.

In conclusion, as shown in Figure 1, the advanced view of cloud access control can abstract four entities: CSP, data user (DU), data owner (DO), SA. SA managing the access control policy database (ACPD).

Due to the two characteristics mentioned above in cloud access control, there are two problems when attacked:

Problem 1 External Attackers Tamper With ACPD: Attackers attack trusted authorization centers, and tamper with

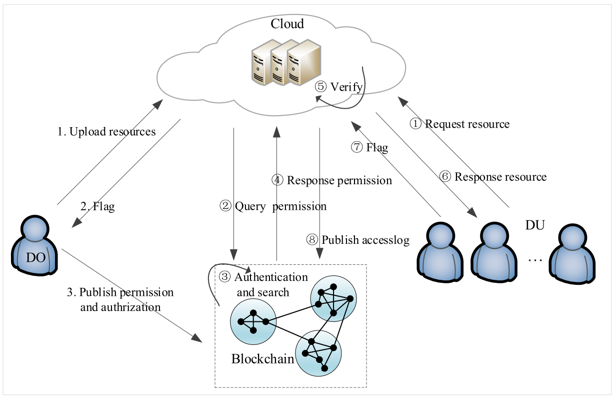

FIGURE 2. The system model of access control framework.

ACPD, causing data leakage or steal data. For instance, an attacker steals ACPD and impersonates an authorized user to access or steal resources, or tampers with ACPD, increasing permissions for illegal access, or deletes legitimate user permissions, destroying confidentiality, availability and integrity.

Problem 2 Malicious SA Privilegs Access or Tampers With ACPD For instance, malicious SA uses his/her privilege to bypass identity authentication and access resources illegally, causing privacy leaks and destroying confidentiality and integrity; malicious SA tampers with ACPD, causing privacy leaks and destroying confidentiality, integrity and availability.

IV. AN ACCESS CONTROL FRAMEWORK WITH PRIVACY PROTECTION

In order to solve the above problems, we propose AuthPrivacyChain - a blockchain-based access control framework with privacy protection in cloud. This Section contains the system model, initialization, access control, authorization, and revocation. The system model is shown in Figure 2, which is consisted of four entities:

- ***Cloud***. It provides authentication and data storage for users. *Cloud* determines access rights of *DU* or *DO* by *Blockchain*.
- ***Blockchain***. It is open, transparent, tamper-proof, and irreversible, and the same as the distributed database, we use it as an authorization policy database for access control.
- ***DO***. *DO* uploads the resources to *Cloud* and publishes the resource's access rights to *Blockchain*.
- ***DU***. *DU* can access the resources if he has permission from *Cloud*.

We assume *Cloud* is semi-trusted, that is, the software, hardware, asymmetric key and business processes of the *Cloud* are trusted, but the *Cloud* SA is not. *Blockchain* is assumed to be trustful. First, *DO* uploads the resources to the *Cloud* and then publishes authorization by registration transactions in blockchain. *DU* sends a resource request to *Cloud*, and *Cloud* queries blockchain, and judge whether the request has permission, finally reply to the request.

TABLE 1. Notation table.

Notations	Description
$E\,(key,M)$ $D\,(key,N)$	They are encrypt/decrypt function. E() is used to encrypt M and D() is used to decrypt N by *key* which can be symmetric or asymmetric.
$Hash\,(M)$	It represents hash function.
K	It represents asymmetric key, K_{pub} represents public key. K_{pri} represents private key.
ks	It represents symmetric key
K_C	It represents asymmetric key pair of *Cloud*. K_C includes K_{pubC} and K_{priC}.
K_{DO}	It represents asymmetric key pair of *DO*. K_{DO} includes K_{pubDO} and K_{priDO}.
K_{DU}	It represents asymmetric key pair of *DU*. K_{DU} includes K_{pubDU} and K_{priDU}.
K_W	It represents asymmetric key that generates wallet address. K_W includes K_{pubW} and K_{priW}.
$Addr$	It represents wallet address in blockchain, meanning the identity of the user

TABLE 2. The definitions of set and field.

Set/Field	Description
resInfo	It represents a set of resource information. **resInfo** ={*resOwnerID, resID, resName, reshash, resCAP, resURL*}. where *resOwnerID*: host of the resource. *resID*: the unique identifier of the resource. *resName*: resource name. *reshash*: resource hash. *resCAP*: resource access permission. *resURL*: resource access address.
resCAP	It represents access permission of resource. ***resCAP*** = {*accessPolicy*, *acessConstCond*, *accessMode*, *timespan*, *resCAPhash*}. where *accessPolicy*: access control policies, such as DAC, BLP, RBAC, etc. *acessConstCond*: access constraints and conditions. *accessMode*: resource access mode, *accessMode*= {up, down, migrate,delete}. *timespan*: access time interval, including start and end. *resCAPhash*: *resCAP* hash.
resCAP_S	It represents encrypted *resCAP*.
bresInfo	It represents resource information in blockchain. ***bresInfo***={ *actionInfo*, *resInfo* }, where *actionInfo*: the added information of the resource.
cloudInfo	It represents *Cloud* basic information set, ***cloudInfo*** ={ *CloudID*, *CloudName*, *CloudUrl*}, where *CloudID*: cloud record number). *CloudName*: cloud name. *CloudUrl*: cloud access address.
tran	It represents a transation of blockchain set, ***tran***={*tranID*, $Addr_{from}$, $Addr_{to}$, *value*, *bresInfo*}, where *tranID*: transaction number. $Addr_{from}$: transaction originator. $Addr_{to}$: transaction receiver. value: transaction value. *resInfo*: resource information.

Next, we will introduce some symbols used later. First, we listed some important functions and symbols, as shown in Table 1. Then, we listed some key fields and set, as shown in Table 2.

A. INITIALIZATION

Initialization consists of three entity registrations (*Cloud*, *DO* and *DU*) and resource publishing.

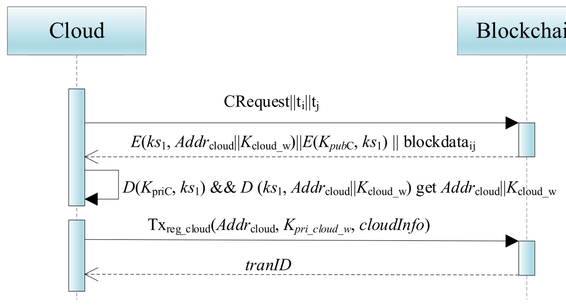
Cloud
Blockchain
CRequest||t_i||t_j
$E(ks_1, Addr_{cloud}||K_{cloud\_w})||E(K_{pubC}, ks_1)$ || $blockdata_{ij}$
$D(K_{priC}, ks_1)$ && $D(ks_1, Addr_{cloud}||K_{cloud\_w})$ get $Addr_{cloud}||K_{cloud\_w}$
$Tx_{reg\_cloud}(Addr_{cloud}, K_{pri\_cloud\_w}, cloudInfo)$
$tranID$

FIGURE 3. Registration workflow of cloud.

1) REGISTRATION OF CLOUD

For becoming a legitimate user or node of blockchain, *Cloud*, *DO* and *DU* must registration in blockchain firstly. Before designing registration workflow, we introduce three function including *KGen*, *AGen* and *SynData*, and one interface *ISave*, which all have been realized by blockchain. *KGen()* aims to generate K_W which input null, and output K_W; *AGen()* aims to create wallet address, and input K_{pub}, output *Addr*; *SynData()* aims to synchronize data from time t_i to t_j. It input t_i and t_j, output $blockdata_{ij}$. *ISave()* is a blockchain storage interface that inputs address, signature private key, storage content and time, and outputs *tranID*.

Then we design the registration workflow of *Cloud* is shown in Figure 3, including three steps.

① *Cloud->Blockchain*: *CRequest*|| t_i|| t_j. *Cloud* sends a request for registration to *Blockchain*.

② *Blockchain->Cloud*: E(ks_1, $Addr_{cloud}$|| K_{cloud_w})|| E(K_{pubC}, ks_1)|| $blockdata_{ij}$. *Blockchain* calls KGen() to generates K_{cloud_w}($K_{pub\_cloud\_w}$, $K_{pri\_cloud\_w}$), and then calls AGen($K_{pub\_cloud\_w}$) to generate $Addr_{cloud}$. *Blockchain* uses ks_1 to encrypt $Addr_{cloud}$|| K_{cloud_w}), and uses K_{pubC} to encrypt ks_1, then sends E(ks_1, $Addr_{cloud}$|| K_{cloud_w})|| E(K_{pubC}, ks_1)|| $blockdata_{ij}$ to *Cloud*. *Cloud* calls D(K_{priC}, ks_1) to decrypt to get ks_1, and calls D(ks_1, $Addr_{cloud}$|| K_{cloud_w}) to decrypt to get $Addr_{cloud}$|| K_{cloud_w}, where $blockdata_{ij}$ represents the data generated in blockchain from t_i to t_j. Usually, cloud will automatically call SynData(t_i, t_j) to synchronize $blockdata_{ij}$ to the local database.

③*Cloud->Blockchain*: Tx_{reg_cloud}($Addr_{cloud}$, $K_{pri\_cloud\_w}$, *cloudInfo*). *Cloud* calls Tx_{reg_cloud} to save *cloudInfo* to *Blockchain*, where Tx_{reg_cloud} is a smart contract, as shown in Algorithm 1.

Algorithm 1 Smart Contract Tx_{reg_cloud}

```
Input: $Addr_{cloud}$, $K_{pri\_cloud\_w}$, cloudInfo
Output: tranID
1 Set chainID and select access nodeID;
2 Gets the current timestamp;
3 tranID = ISave($Addr_{cloud}$, $K_{pri\_cloud\_w}$, cloudInfo,
    timestamp);
4 return tranID;
```

In Smart Contract Tx_{reg_cloud}, line 1 configures the connection of blockchain, line 2 gets the current timestamp, line 3 calls blockchain storage interface to publish the registration transaction, finally, line 4 returns the *tranID*.

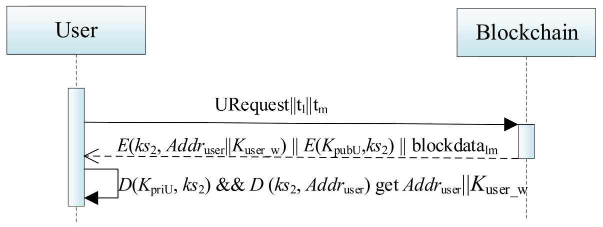
User
Blockchain
URequest||t_l||t_m
$E(ks_2, Addr_{user}||K_{user\_w})$ || $E(K_{pubU}, ks_2)$ || $blockdata_{lm}$
$D(K_{priU}, ks_2)$ && $D(ks_2, Addr_{user})$ get $Addr_{user}||K_{user\_w}$

FIGURE 4. Registration workflow of DO and DU.

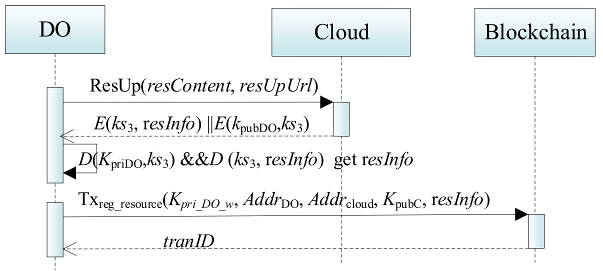
DO
Cloud
Blockchain
ResUp($resContent$, $resUpUrl$)
$E(ks_3, resInfo)$ ||$E(k_{pubDO}, ks_3)$
$D(K_{priDO}, ks_3)$ &&$D(ks_3, resInfo)$ get $resInfo$
$Tx_{reg\_resource}(K_{pri\_DO\_w}, Addr_{DO}, Addr_{cloud}, K_{pubC}, resInfo)$
$tranID$

FIGURE 5. Registration workflow of resource publishing.

2) REGISTRATION OF USER

We design the registration workflow of *DO* and *DU*, as shown in Figure 4, includes two steps. Because the registration process of *DO* is the same as *DU*, we collectively call it user registration.

① *User->Blockchain*: *URequest*|| t_l|| t_m. *User* sends a registration request to *Blockchain*.

② *Blockchain->User*: E(ks_2,$Addr_{user}$|| K_{user_w})|| E(K_{pubU}, ks_2)|| $blockdata_{lm}$. *Blockchain* generates K_{user_w}, and calls AGen($K_{pub\_user\_w}$) to generate $Addr_{user}$, and then sends E(ks_2, $Addr_{user}$|| K_{user_w})|| E(K_{pubU},ks_2)|| $blockdata_{lm}$ to *User*. *User* calls the D(K_{priU}, ks_2) and D(ks_2,$Addr_{user}$|| K_{user_w}) to decrypt to get $Addr_{user}$|| K_{user_w}, where $blockdata_{lm}$ represents the data generated in blockchain from t_l to t_m. Usually, *User* will automatically call SynData(t_l, t_m) to synchronize $blockdata_{lm}$ to the local database.

3) RESOURCE PUBLISHING

Resource publishing is that *DO* upload resources to cloud and publish the metadata on of resources to blockchain. Before designing the registration workflow, we introduce one function *ResUp()* and one interface *ISend()* of blockchain. *ResUp()* aims to upload resources to cloud by *DO*, which inputs *resContent* and *resUpURL*, outputs *resInfo*; *ISend ()* is a blockchain transaction interface, which inputs K_{pri_from}, $Addr_{from}$, $Addr_{to}$, *index*, *content* and *timestamp*, and outputs *tranID*.

We design the registration workflow of resource publishing, it is shown in Figure 5, include three steps.

① *DO->Cloud*: ResUp(*resContent*, *resUpUrl*). *DO* calls ResUp to upload resource to *Cloud*.

② *Cloud->DO*: E(ks_3, *resInfo*)|| E(k_{pubDO}, ks_3). *Cloud* returns E(ks_3, *resInfo*) to *DO*, and *DO* decrypts it with ks_3 to get *resInfo*.

③ *DO->Blockchain*: $Tx_{reg\_resource}(K_{pri\_DO\_w}, Addr_{DO}, Addr_{cloud}, K_{pubC}, resInfo)$. *DO* calls $Tx_{reg\_resource}$ to publish the information of resource registration to *Blockchain*, where $Tx_{reg\_resource}$ is a smart contract, as shown in Algorithm 2.

Algorithm 2 Smart Contract $Tx_{reg\_resource}$

```
    Input: $K_{pri\_DO\_w}, Addr_{DO}, Addr_{cloud}, K_{pubC}, resInfo$
    Output: $tranID$
1   Set $chainID$ and select access $nodeID$;
2   $hash\_resID$ = Hash($resInfo.resID$);
3   $resCAP\_S$ = E($K_{pubC}$, $resInfo.resCAP$);
4   if $resCAP\_S$ is null then
5       return false;
6   else
7       resInfo[resCAP] = $resCAP\_S$;
8   end
9   bresInfo = null + resInfo;
10  Gets the current $timestamp$;
11  tranID = ISend($K_{pri\_DO\_w}, Addr_{DO}, Addr_{cloud}$,
    $hash\_resID$, $bresInfo$, $timestamp$);
12  return $tranID$;
```

In smart contract $Tx_{reg\_resource}$, line 1 configures blockchain, line 2 computes resource unique identifier by hash, line 3-8 effectively encrypts access control permission, line 9 assigns *bresInfo*, line 10 gets a timestamp, line 11 calls ISend and line 12 finally returning *tranID*.

B. ACCESS CONTROL

The access control is that the user requests resource in cloud, and cloud determines whether users can access the resource according to the permissions stored in blockchain. If the user has rights, cloud will let users access the resource, and the access record will be stored in blockchain. Before designing workflow of access control, we introduce one function that is *VerifyCap* and one interface that is *IQuery*, *VerifyCap()* which realized by *Cloud* is aim to verify the inclusion relationship between two permission set, which input $resCAP_1$, K_{priC}, and $resCAP\_S_2$, and outputs *true* or *false*, if $resCAP_1$ contains $resCAP\_S_2$, returns *true*, otherwise returns *false*. *IQuery()* which realized by blockchain is a transaction query interface, which inputs *index*, $Addr_1$ and $Addr_2$ where index represents query index value, $Addr_1$ and $Addr_2$ represents trading parties, and outputs *tran*.

We design an access control workflow, as shown in Figure 6, include six steps.

① *User->Cloud*: E(ks_4, $Addr_{user}$|| *resInfo'*)|| E(K_{pubC},ks_4). *User* sends a request to *Cloud*, where *resInfo'* represents resource what user want to access.

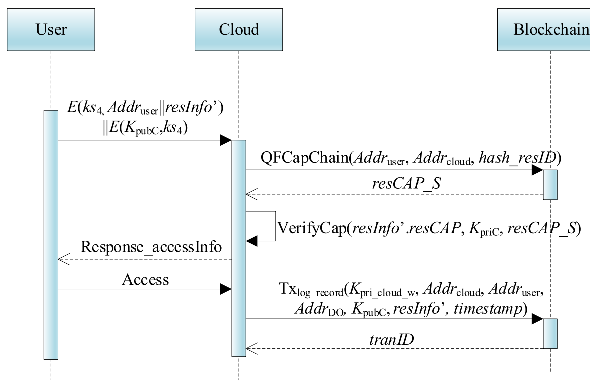
User
Cloud
Blockchain
$E(ks_4, Addr_{user}||resInfo')$
$||E(K_{pubC},ks_4)$
QFCapChain($Addr_{user}$, $Addr_{cloud}$, $hash\_resID$)
$resCAP\_S$
VerifyCap($resInfo'.resCAP$, K_{priC}, $resCAP\_S$)
Response_accessInfo
Access
$Tx_{log\_record}(K_{pri\_cloud\_w}, Addr_{cloud}, Addr_{user},$
$Addr_{DO}, K_{pubC}, resInfo', timestamp)$
$tranID$

FIGURE 6. Access control overview.

Algorithm 3 Smart Contract *QFCapChain*

```
    Input: $Addr_{user}, Addr_{cloud}, hash\_resID$
    Output: $bresInfo$, $resCAP\_S$
1   Set $chainID$ and select access $nodeID$;
2   if $Addr_{user}$ == $null$ | $Addr_{cloud}$ == $null$ | $hash\_resID$
    == $null$ then
3       $resCAP\_S$ = null;
4   else
5       tran = IQuery($hash\_resID, Addr_{user}, Addr_{cloud}$);
6       $resCAP\_S$ = $tran.bresInfo.resInfo.resCAP\_S$;
7   end
8   return $resCAP\_S$
```

② *Cloud->Blockchain*: *QFCapChain*($Addr_{user}$, $Addr_{cloud}$, *hash_resID*). *Cloud* firstly decrypts to get $Addr_{user}$, *resInfo'*, and computing Hash(*resInfo.resID*) to get *hash_resID*, then calls *QFCapChain* to get *resCAP_S*, where *QFCapChain* is a smart contract, as shown in Algorithm 3.

③ *Blockchain->Cloud*: *resCAP_S*. *Blockchain* returns *resCAP_S* to *Cloud*.

④ *Cloud*: VerifyCap(*resInfo'.resCAP*, K_{priC}, resCAP_S). *Cloud* decrypt *resCAP_S* to get *resCAP* of $Addr_{user}$, and compare with *resInfo'.resCAP*, if *resCAP* contains *resInfo'.resCAP*, the user is allowed to access the resource; otherwise, access is not allowed.

⑤ *Cloud->User*: *Response_accessInfo*. *Cloud* returns *Response_accessInfo* to *User*, if *Response_accessInfo* is *true*, then *User* can access the resource, otherwise, access is not allowed.

⑥ *Cloud->Blockchain*: $Tx_{log\_record}(K_{pri\_cloud\_w}, Addr_{cloud}, Addr_{user}, Addr_{DO}, K_{pubC}, resInfo', timestamp)$. *Cloud* calls Tx_{log_record} to publish logs of access resource in blockchain, where Tx_{log_record} is a smart contract, as shown in Algorithm 4.

C. AUTHORIZATION

Authorization is divided into direct authorization and indirect authorization. Direct authorization means that the *DO*

Algorithm 4 Smart Contract Tx_{log_record}

```
Input: $K_{pri\_cloud\_w}$, $Addr_{cloud}$, $Addr_{user}$, $Addr_{DO}$, $K_{pubC}$,
       $resInfo'$, $timestamp$
Output: $tranID$
1  Set $chainID$ and select access $nodeID$;
2  $hash\_resID$ = Hash($resInfo'.resID$);
3  $resCAP\_S$ = E($K_{pubC}$, $resInfo'.resCAP$);
4  if $resCAP\_S$ is null || $timestamp$ is illegal then
5      return false;
6  else
7      resInfo'[resCAP] = $resCAP\_S$;
8      $accessTime\_S$ = E($K_{pubC}$, $timestamp$);
9  end
10 $actionInfo = Addr_{user} + accessTime\_S$;
11 $bresInfo = actionInfo + resInfo'$;
12 $tranID$ = ISend($K_{pri\_cloud\_w}$, $Addr_{cloud}$, $Addr_{DO}$,
   $hash\_resID$, $bresInfo$, $timestamp$));
13 return $tranID$;
```

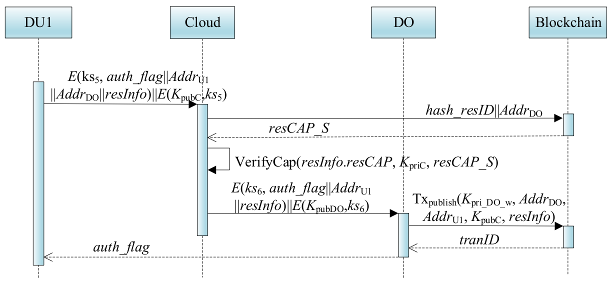
DU1
Cloud
DO
Blockchain
$E(ks_5, auth\_flag||Addr_{U1}$
$||Addr_{DO}||resInfo)||E(K_{pubC},ks_5)$
$hash\_resID||Addr_{DO}$
$resCAP\_S$
VerifyCap($resInfo.resCAP$, K_{priC}, $resCAP\_S$)
$E(ks_6, auth\_flag||Addr_{U1}$
$||resInfo)||E(K_{pubDO},ks_6)$
$Tx_{publish}(K_{pri\_DO\_w}, Addr_{DO},$
$Addr_{U1}, K_{pubC}, resInfo)$
$tranID$
$auth\_flag$

FIGURE 7. Direct authorization process.

authorizes the *DU*, and the indirect authorization is authorized by the granted *DU* to other users.

1) DIRECT AUTHORIZATION

Direct authorization is that the owner of the resource grants access to other users. We design direct authorization workflow of resource publishing, it is shown in Figure 7, include seven steps.

① *DU1->Cloud*: E(ks_5, *auth_flag*|| $Addr_{U1}$|| $Addr_{DO}$|| *resInfo*)|| E(K_{pubC}, ks_5). *DU1* sends an authorization request to *Cloud*, where *auth_flag* is authorization flag. *Cloud* decrypts to get $Addr_{U1}$, $Addr_{DO}$ and *resInfo*.

② *Cloud->Blockchain*: *hash_resID*|| $Addr_{DO}$. *Cloud* computes Hash(*resID*) to get *hash_resID*, and calls *QFCapChain* ($Addr_{DO}$, $Addr_{cloud}$, *hash_resID*).

③ *Blockchain->Cloud*: *resCAP_S*. *Blockchain* returns *resCAP_S* of resource to *Cloud*.

④ *Cloud*: VerifyCap(*resInfo.resCAP*, K_{priC}, *resCAP_S*). *Cloud* checks that $Addr_{DO}$ is indeed the host of resource, and calls *VerifyCap* to confirm that the permissions requested by *DU1* are within the scope of host's authorization. If the request is outside the scope of the authorization, the authorization process is terminated immediately. otherwise, go to ⑤.

Algorithm 5 Smart Contract $Tx_{publish}$

```
Input: $K_{pri\_DO\_w}$, $Addr_{DO}$, $Addr_{U1}$, $K_{pubC}$, $resInfo$
Output: $tranID$
1  Set $chainID$ and select access $nodeID$;
2  $hash\_resID$ = Hash($resInfo.resID$);
3  $resCAP\_S$ = E($K_{pubC}$, $resInfo.resCAP$);
4  if $resCAP\_S$ is null then
5      return false;
6  else
7      resInfo[resCAP] = $resCAP\_S$;
8  end
9  $actionInfo = actionAuth + null$ ;
10 $bresInfo = actionInfo + resInfo$ ;
11 Gets the current $timestamp$;
12 $tranID$ = ISend($K_{pri\_DO\_w}$, $Addr_{DO}$, $Addr_{U1}$,
   $hash\_resID$, $bresInfo$, $timestamp$));
13 return $tranID$;
```

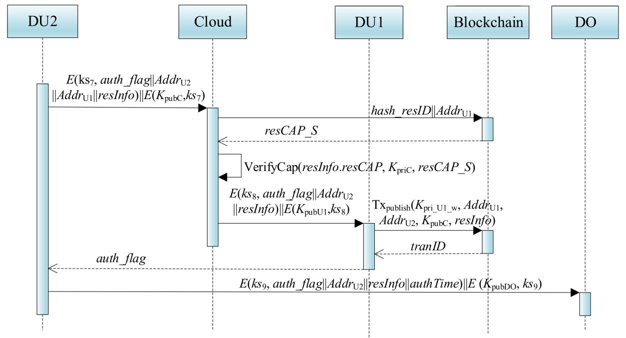
DU2
Cloud
DU1
Blockchain
DO
$E(ks_7, auth\_flag||Addr_{U2}$
$||Addr_{U1}||resInfo)||E(K_{pubC},ks_7)$
$hash\_resID||Addr_{U1}$
$resCAP\_S$
VerifyCap($resInfo.resCAP$, K_{priC}, $resCAP\_S$)
$E(ks_8, auth\_flag||Addr_{U2}$
$||resInfo)||E(K_{pubU1},ks_8)$
$Tx_{publish}(K_{pri\_U1\_w}, Addr_{U1},$
$Addr_{U2}, K_{pubC}, resInfo)$
$tranID$
$auth\_flag$
$E(ks_9, auth\_flag||Addr_{U2}||resInfo||authTime)||E(K_{pubDO}, ks_9)$

FIGURE 8. Indirect authorization process.

⑤ *Cloud->DO*: E(ks_6, *auth_flag*|| $Addr_{U1}$|| *resInfo*)|| E(K_{pubDO}, ks_6). *Cloud* sends authorization request of *DU1* to *DO*.

⑥ *DO->Blockchain*: $Tx_{publish}$($K_{pri\_DO\_w}$, $Addr_{DO}$, $Addr_{U1}$, K_{pubC}, *resInfo*). *DO* calls $Tx_{publish}$ to publish authorization to *Blockchain*. $Tx_{publish}$ is a smart contract, as shown in Algorithm 5.

⑦ *DO->DU1*: *auth_flag*. *DO* sends *auth_flag* to *DU1*.

2) INDIRECT AUTHORIZATION

Indirect authorization means that the authorized *DU* authorizes other users, including at least five entities: *DO* (resource owner), *DU1* (authorizer), *DU2* (requester), *Blockchain* and *Cloud*. the specific interaction process is shown in Figure 8.

From Figure 8, the steps of indirect authorization are basically the same as direct authorization, the difference between indirect authorization and direct authorization is that authorized users need to send indirect authorization notice (E(ks_9, *auth_flag*|| $Addr_{U2}$|| *resInfo*|| *authTime*)|| E(K_{pubDO}, ks_9)) to *DO*, so we will not be repeated here.

D. AUTHORIZATION REVOCATION

Authorization revocation refers to the revocation of authorization by authorized users. Authorization revocation is a

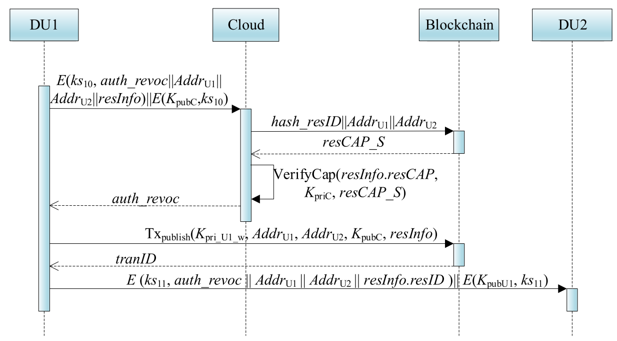
DU1
Cloud
Blockchain
DU2
E(ks10, auth_revoc||AddrU1||AddrU2||resInfo)||E(KpubC,ks10)
hash_resID||AddrU1||AddrU2
resCAP_S
VerifyCap(resInfo.resCAP, KpriC, resCAP_S)
auth_revoc
Txpublish(Kpri_U1_w, AddrU1, AddrU2, KpubC, resInfo)
tranID
E (ks11, auth_revoc || AddrU1 || AddrU2 || resInfo.resID)|| E(KpubU1, ks11)

FIGURE 9. Basic authorization revocation.

very complex process. In order to standardize the authorization revocation process, we define the following rules:

- Who authorizes, who revokes.
- The owner can revoke the direct authorization and indirect authorization.
- Only when the authorized user revokes all authorizations can the previous level authorized user to revoke the authorization of this authorized user.

According to the above three rules, authorization revocation includes basic authorization revocation and complex authorization revocation. The basic authorization revocation is the basic operating unit of authorization revocation, which means that the authority that no longer has indirect authorization is revoked. For example, we assumed that *DU1* authorizes to *DU2*, and *DU2* no longer has any indirect authorization. The specific interaction process is as follows. We design workflow of basic authorization revocation, it is shown in Figure 9, include six steps.

① *DU1->Cloud*: E(ks_{10}, *auth_revoc*|| $Addr_{U1}$|| $Addr_{U2}$|| *resInfo*)|| E(K_{pubC}, ks_{10}). *DU1* sends the information of authorization revocation to *Cloud*, where *auth_revoc* is flag of authorization revocation request.

② *Cloud->Blockchain*: *hash_resID*|| $Addr_{U1}$|| $Addr_{U2}$. *Cloud* calls *QFCapChain* to query whether or not $Addr_{U1}$ have authorized permission to $Addr_{U2}$.

③ *Blockchain->Cloud*: *resCAP_S*. *Blockchain* returns *resCAP_S* to *Cloud*.

④ *Cloud->DU1*: *auth_revoc*. *Cloud* decrypt *resCAP_S* to get *resCAP*, if *resCAP* is not null, then *Cloud* send *auth_revo* to *DU1*.

⑤ *DU1->Blockchain*: $Tx_{publish}$($K_{pri\_U1\_w}$, $Addr_{U1}$, $Addr_{U2}$, K_{pubC}, *resInfo*). *DU1* calls $Tx_{publish}$ to revoke authrozation, where $resInfo.resCAP = null$.

⑥ *DU1->DU2*: E(ks_{11}, *auth_revoc*|| $Addr_{U1}$|| $Addr_{U2}$|| *resInfo.resID*)|| E(K_{pubU1}, ks_{11}). *DU1* send *auth_revoc*|| $Addr_{U1}$|| $Addr_{U2}$|| *resInfo.resID* to *DU2*, notifying that the *DU2* permission has been revoked.

Complex authorization revocation usually refers to the authorization revocation with one or more indirect authorizations. However, all complex authorization revocations are completed by basic authorization revocations step by step. For example, we assume the following scenario: *DU1* has

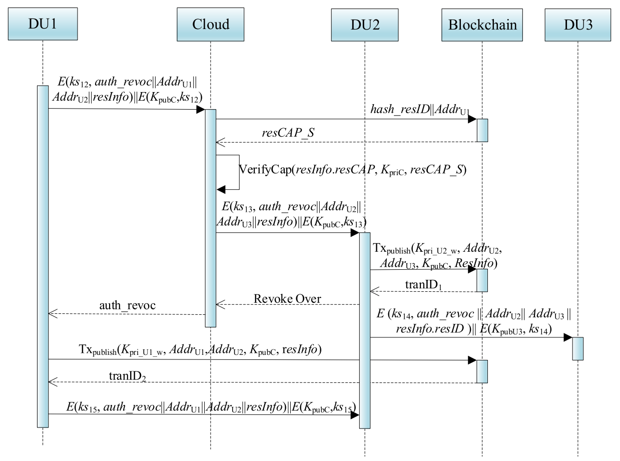
DU1
Cloud
DU2
Blockchain
DU3
E(ks12, auth_revoc||AddrU1||AddrU2||resInfo)||E(KpubC,ks12)
hash_resID||AddrU1
resCAP_S
VerifyCap(resInfo.resCAP, KpriC, resCAP_S)
E(ks13, auth_revoc||AddrU2||AddrU3||resInfo)||E(KpubC,ks13)
Txpublish(Kpri_U2_w, AddrU2, AddrU3, KpubC, ResInfo)
tranID1
auth_revoc
Revoke Over
E (ks14, auth_revoc || AddrU2|| AddrU3 || resInfo.resID)|| E(KpubU3, ks14)
Txpublish(Kpri_U1_w, AddrU1, AddrU2, KpubC, resInfo)
tranID2
E(ks15, auth_revoc||AddrU1||AddrU2||resInfo)||E(KpubC,ks15)

FIGURE 10. Complex authorization revocation.

the authorization, *DU1* authorizes to *DU2*, and *DU2* not only uses the permission to access the resource but also authorizes the authorization to *DU3*. As can be seen from Figure 10, the specific interaction process is as follows.

V. CHARACTERISTICS AND SECURITY ANALYSIS

A. CHARACTERISTIC ANALYSIS

For decentralization, traditional cloud access control includes identity authentication, authorization, access permissions, and auditing. Cloud as a trusted center for access control and all processes are centralized. User is usually authenticated by username and password which is stored by cloud authentication database. Authorization-related information is directly stored in cloud ACPD. The access record is usually stored in cloud log database. However, for AuthPrivacyChain, it does not have a trusted center, nor does it require users to treat cloud as a trusted center. User is authenticated by *Addr* which generates by K_W. Authorization-related information is directly stored in blockchain. Access record is stored in blockchain too. **For performance**, in terms of identity authentication, AuthPrivacyChain is better than traditional access control, because it uses *Addr* instead of username and password. In terms of identity authentication authorization and access control, AuthPrivacyChain is slightly lower than the traditional access control, because it needs to access blockchain, but it can reduce the gap by choosing the appropriate super node to access blockchain.

B. SECURITY ANALYSIS

1) FOR CONFIDENTIALITY

On the one hand, no matter the general user or the SA, AuthPrivacyChain can ensure that the data stored by the user in cloud will not be disclosed to or used by the unauthorized person, thus ensuring the confidentiality of the *DO*'s data. On the other hand, in AuthPrivacyChain, the information transmission among *DU*, *DO*, *Cloud* and *Blockchain* are encrypted, and the access control permission is also encrypted and stored in blockchain, so ensuring the confidentiality and privacy of AuthPrivacyChain. **For integrity**. On the one hand, for the data uploaded by users, AuthPrivacyChain provides eight protection mechanisms for data integrity and permission integrity for users. It sets the *reshash* member in the data structure *resInfo* and the *resCAPhash* in the data structure *resCAP*. On the other hand, AuthPrivacyChain can provide users with system integrity, which can ensure that the system can perform the predetermined functions in a normal way and avoid intentional or unintentional unauthorized manipulation. **For availability**. AuthPrivacyChain can work quickly and cannot refuse access to authorized users. **For authenticity**. On the one hand, *DU*, *DO*, *Cloud* and *Blockchain* in AuthPrivacyChain can be verified and trusted, because to join the system, they must first public their own certificate (public key). On the other hand, for the information transmission of AuthPrivacyChain, information and information sources are correct to be able to verify that the user is who he claims to be and that every input to the system comes from a trusted source. Because when AuthPrivacyChain delivers the information, it requires the private key of the receiver to be decrypted. **For accountability**, AuthPrivacyChain is not an absolutely safe system. Therefore, AuthPrivacyChain must be able to track down the party responsible for security leakage. We design AuthPrivacyChain to keep activity records of entities in blockchain, so as to allow post-audit analysis, and then track security events or resolve disputes.

From the above analysis, we can see that AuthPrivacyChain can not only prevent attacks from external users but also prevent internal management attacks when request access to resources, everyone must be authenticated, including SA.

VI. EXPERIMENT AND EVALUATION

A. EXPERIMENTAL ENVIRONMENT

We implemented a prototype to analyze performance of the framework. Our experimental environment is all based on Alibaba Cloud, configured as 2 core, 8G RAM, 100G storage, and the system is ubuntu 16.04. There are three test machines, and *Blockchain*, *Cloud* and users (*DO* or *DU*) are deployed. The experiment uses two typical test chains of EOS, namely *Kylin* and *Jungle*, and a local test chain based on EOS.

- *Kylin*: chainID is 5fff1dae8dc8e2fc4d5b23b2c7665c97f9e9d8edf2b6485a86ba311c25639191.
- *Jungle*: chainID is e70aaab8997e1dfce58fbfac80cbbb8fecec7b99cf982a9444273cbc64c41473.
- *Local*: chainID is cf057bbfb72640471fd910bcb67639c22df9f92470936cddc1ade0e2f2e7dc4f.

Cloud registration contract: 461c37c15d5dfbfdcf82e0fd758d0b841d0b2deb6ff0e29369ed5cbac9ccfdb9. Access control contract: 33b43efc0b5a62f81769cdad69dabddbfff328e9d6943a30a2cbba6e195d1af1. Record contract: 559f99332294d9cb3a44690fc473d0a96cfa0963fa2012060869dfab624488cc.

Then, in the access control process, we need to find *resCAP_S* in blockchain and decrypt it. We use the EOS-encrypt library, use public key encryption, then use private key decryption. Moreover, we use the advanced encryption standard (AES) algorithm to ensure the privacy of communication.

We try to do many access control experiments as shown in Table 3. "\" means unexecuted.

TABLE 3. Access control results.

User	Identity	Authorization query	Authorization verification	Access result	Access record
DO	yes	yes	yes	allow	yes
DO	no	\	\	refuse	\
DU	yes	yes	yes	allow	yes
DU	yes	yes	no	refuse	\
DU	yes	no	\	refuse	\
DU	no	\	\	refuse	\
SA	no	\	\	refuse	\

TABLE 4. The overhead of SHA256.

Data Size/B	Time Cost/ms
16	0.463
32	0.467
64	0.535

B. PERFORMANCE EVALUATION

In the AuthPrivacy evaluation, we will focus on time performance overhead. The complete access control process includes four parts: identity authentication, authorization, access permission, and audit. In AuthPrivacyChain, *Addr* is used as identity authentication, and access permissions need to query *resCAP_S* in blockchain and verify permissions. The hash value of *resID* is used as a unique identifier which may affect the query efficiency, so we first test the hash cost overhead, as shown in Table 4.

Secondly, the overhead of authorization needs to be analyzed. we tested it on two typical open test chains of EOS, namely *Jungle* and *Kylin*, and three different types of nodes are tested on each test chain. The overhead of authorization in three types of nodes for *Kylin* is shown in Figure 11(a), the best average performance is the node of *K-1 (api-kylin.eoslaomao.com)* for *Kylin*. The overhead of authorization in three types of nodes for *Jungle* is shown in Figure 11(b), the best average performance is the node of J-2 (*jungle.eosam.sterdam.net*) for *Jungle*.

The experimental results show that the authorization publish performance is related to the selected blockchain and the nodes connecting blockchain. Selecting the appropriate blockchain and configuring the nodes will greatly improve the performance. As shown in Figure 11(c), the optimal performance of the *Jungle* is about 4.4s, the optimal performance of *Kylin* is about 0.4s, and the optimal performance of the localhost less than 0.02s.

Then we tested the read and write throughput of AuthPrivacyChain and the traditional cloud (Alibaba Cloud's MySQL database). The comparison is shown in Figure 12.The test

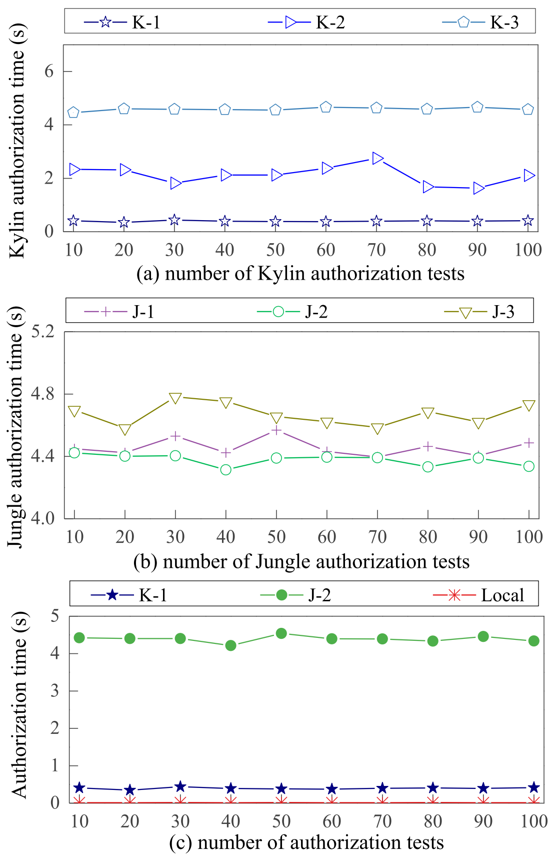

FIGURE 11. Comparison of authorization time, including ***Kylin***, ***Jungle***, **and *Local*. (a) *Kylin*: K-1 is *api-kylin.eoslaomao.com*, K-2 is *api-kylin.eosasia.one* and K-3 is *api.kylin.alohaeos.com*. (b) *Jungle*: J-1 is *jungle2.cryptolions.io*, J-2 is *jungle.eosam.sterdam.net*, J-3 is *api.jungle.alohaeos.com*. (c) *Jungle*, *Kylin* and *Local*'s authorized publish performance comparison.**

tool is JMeter, the Alibaba Cloud database is Mysql, and AuthPrivacyChain uses the *Kylin* test chain. Figure 12 (a) shows the read access throughput. AuthPrivacyChain has a small delay when processing read requests. Figure 12 (b) shows the throughput of permission to write. When the transaction is larger than 400, the throughput of AuthPrivacyChain processing request is slightly higher than that of Mysql.

Finally, the overhead of access control needs to be analysis too. We compare AuthPrivacyChain to the traditional cloud access control platform. AuthPrivacyChain uses *Kylin*'s node *api-kylin.eoslaomao.com*. The traditional cloud access control platform uses OneDrive and Alibaba Cloud. Assume that the user has logged in to the corresponding device, ignoring the interference of network factors and the time of user input, only considering the time of access control, as shown in Figure 13.

According to Figure 13, the overall performance of AuthPrivacyChain and traditional access control is very similar. Traditional access control is done in cloud, and AuthPrivacyChain's access control needs to interact with blockchain.

In conclusion, both the authorization and access control performance are related to the configuration of blockchain. The choice of blockchain has an impact on performance. For

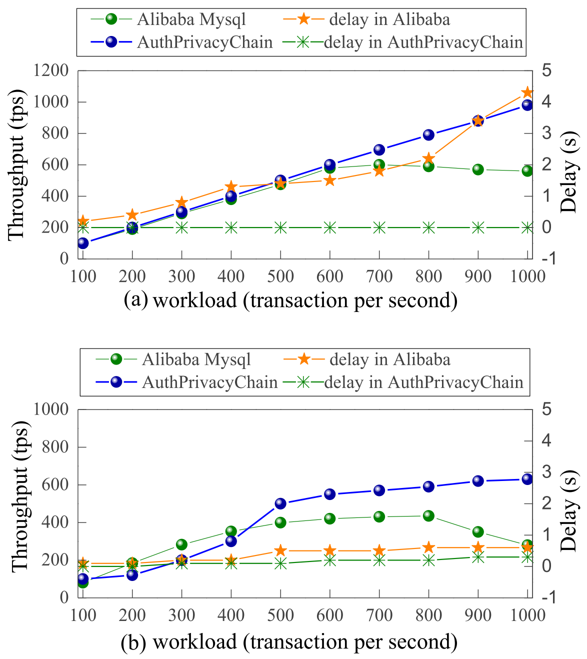

FIGURE 12. Comparison of throughput and delay.

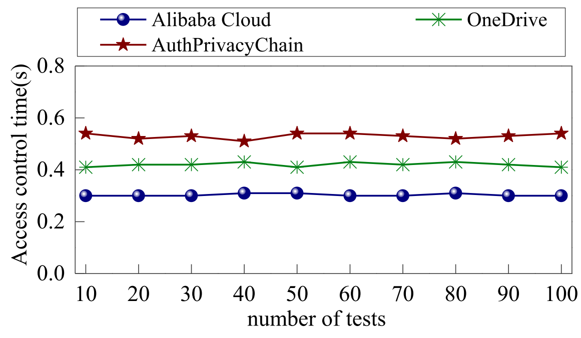

FIGURE 13. Traditional and AuthPrivacyChain access control performance comparison.

the same blockchain, you can configure nodes to achieve better performance.

VII. CONCLUSION

Among the security-related problems of the existing blockchain and cloud, the research results of using blockchain to solve the privacy protection access control in cloud are few. Most of the traditional cloud access control has one or more trusted centers and trusted internal administrators, so it is very likely to suffer internal and external attacks.

In order to solve the problem of illegal access to resources by attackers in cloud, this paper designs an access control framework AuthPrivacyChain with privacy protection in cloud environment. All authorization-related transactions are posted by the user to blockchain. This paper implements the framework model based on the EOS blockchain and regards access permission and other information as an additional description of blockchain transactions. The experimental results show that only users with access rights can access resources. so our solution can satisfy with confidentiality, integrity, availability, authenticity, and accountability, and can not only prevent attacks from external users but also prevent internal management attacks.

ACKNOWLEDGMENT

(Caixia Yang and Liang Tan are co-first authors.)

REFERENCES

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. Special Publication 800-145, 2011.

[2] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf, "NIST cloud computing reference architecture," *NIST Special Publication*, vol. 500, no. 211, pp. 1–28, 2011.

[3] M. Armbrust, I. Stoica, M. Zaharia, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, and A. Rabkin, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, 2010.

[4] J. Wu, M. Dong, K. Ota, J. Li, and Z. Guan, "FCSS: Fog-computing-based content-aware filtering for security services in information-centric social networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 7, no. 4, pp. 553–564, Oct. 2019.

[5] K. Yu, M. Arifuzzaman, Z. Wen, D. Zhang, and T. Sato, "A key management scheme for secure communications of information centric advanced metering infrastructure in smart grid," *IEEE Trans. Instrum. Meas.*, vol. 64, no. 8, pp. 2072–2085, Aug. 2015, doi: 10.1109/TIM.2015.2444238.

[6] X. Lin, J. Li, J. Wu, H. Liang, and W. Yang, "Making knowledge tradable in edge-AI enabled IoT: A consortium blockchain-based efficient and incentive approach," *IEEE Trans. Ind. Informat.*, vol. 15, no. 12, pp. 6367–6378, Dec. 2019.

[7] Y. Q. Zhang, X. F. Wang, X. F. Liu, and L. Liu, "Survey on cloud computing security," *J. Softw.*, vol. 27, no. 6, pp. 1328–1348, 2016.

[8] Z. Tari, X. Yi, U. S. Premarathne, P. Bertok, and I. Khalil, "Security and privacy in cloud computing: Vision, trends, and challenges," *IEEE Cloud Comput.*, vol. 2, no. 2, pp. 30–38, Mar. 2015, doi: 10.1109/MCC.2015.45.

[9] M. Almorsy, J. Grundy, and I. Müller, "An analysis of the cloud computing security problem," 2016, *arXiv:1609.01107*. [Online]. Available: http://arxiv.org/abs/1609.01107

[10] *Cloud Security Alliance, Security Guidance V4.0*. Accessed: Apr. 16, 2020. [Online]. Available: https://c-csa.cn/i/file/20171225/2017122523220533533.pdf

[11] C. Lee, P. Chung, and M. Hwang, "A survey on attribute-based encryption schemes of access control in cloud environments," *Int. J. Netw. Secur.*, vol. 15, no. 4, pp. 231–240, 2013.

[12] R. Charanya and M. Aramudhan, "Survey on access control issues in cloud computing," in *Proc. Int. Conf. Emerg. Trends Eng., Technol. Sci. (ICETETS)*, Feb. 2016, pp. 1–4, doi: 10.1109/ICETETS.2016.7603014.

[13] J. M. Ferris, "Providing access control to user-controlled resources in a cloud computing environment," U.S. Patent 8 984 505, Mar. 17, 2015.

[14] S. Namasudra and P. Roy, "Secure and efficient data access control in cloud computing environment: A survey," *Multiagent Grid Syst.*, vol. 12, no. 2, pp. 69–90, May 2016, doi: 10.3233/MGS-160244.

[15] Y. Wang, J. Yang, C. Xu, X. Ling, and Y. Yang, "Survey on Access Control Technologies for Cloud Computing," *J. Softw.*, vol. 26, no. 5, pp. 1129–1150, 2015.

[16] J. Zhou, Y. Zhang, and Y. Gao, "Research of ABAC model based on usage control under cloud environment," *J. Comput. Appl.*, vol. 31, no. 12, pp. 3692–3694, 2014.

[17] J. Zhu and Q. Wen, "SaaS access control research based on UCON," in *Proc. 4th Int. Conf. Digit. Home*, Nov. 2012, pp. 331–334, doi: 10.1109/ICDH.2012.50.

[18] Y. Zhu, D. Ma, C.-J. Hu, and D. Huang, "How to use attribute-based encryption to implement role-based access control in the cloud," in *Proc. Int. Workshop Secur. Cloud Comput.-Cloud Comput.* New York, NY, USA: ACM, 2013, pp. 33–40.

[19] Y. Wang, D. Zhang, and H. Zhong, "Multi-authority based weighted attribute encryption scheme in cloud computing," in *Proc. 10th Int. Conf. Natural Comput. (ICNC)*, Aug. 2014, pp. 1033–1038, doi: 10.1109/ICNC.2014.6975982.

[20] L. Popa, M. Yu, S. Y. Ko, S. Ratnasamy, and I. Stoica, "CloudPolice: Taking access control out of the network," in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw. (Hotnets)*. New York, NY, USA: ACM, 2010, pp. 1–6.

[21] P. He, R. Huang, N. Chen, and Z. Li, "Research progress on side-channel attacks in cloud environment," *Appl. Res. Comput.*, vol. 35, no. 4, pp. 969–973, 2018.

[22] J. Guo, W. Yang, K. Lam, and X. Yi, "Using blockchain to control access to cloud data," in *Proc. Int. Conf. Inf. Secur. Cryptol.* Cham, Switzerland: Springer, 2018, Art. no. 274C288, doi: 10.1007/978-3-030-14234-6_15

[23] Y. Yuan and F.-Y. Wang, "Blockchain: The state of the art and future trends," *Acta Autom. Sinica*, vol. 42, no. 4, pp. 481–494, 2016.

[24] J. Chen, J. Wu, H. Liang, S. Mumtaz, J. Li, K. Konstantin, A. K. Bashir, and R. Nawaz, "Collaborative trust blockchain based unbiased control transfer mechanism for industrial automation," *IEEE Trans. Ind. Appl.*, early access, Dec. 13, 2019, doi: 10.1109/TIA.2019.2959550.

[25] X. Shen, Q. Pei, and X. Liu, "Survey of blockchain," *J. Netw. Inf. Secur.*, vol. 2, no. 11, pp. 11–20, 2016.

[26] K. Yu, S. Eum, T. Kurita, Q. Hua, T. Sato, H. Nakazato, T. Asami, and V. P. Kafle, "Information-centric networking: Research and standardization status," *IEEE Access*, vol. 7, pp. 126164–126176, 2019, doi: 10.1109/ACCESS.2019.2938586.

[27] X. Qi, Y. Su, K. Yu, J. Li, Q. Hua, Z. Wen, J. Lopez, and T. Sato, "Design and performance evaluation of content-oriented communication system for IoT network: A case study of named node networking for real-time video streaming system," *IEEE Access*, vol. 7, pp. 88138–88149, 2019.

[28] J. Huang, D. M. Nicol, R. Bobba, and J. H. Huh, "A framework integrating attribute-based policies into role-based access control," in *Proc. 17th ACM Symp. Access Control Models Technol. (SACMAT)*. New York, NY, USA: ACM, 2012, pp. 187–196.

[29] T. Tavizi, M. Shajari, and P. Dodangeh, "A usage control based architecture for cloud environments," in *Proc. IEEE 26th Int. Parallel Distrib. Process. Symp. Workshops PhD Forum*, May 2012, pp. 1534–1539, doi: 10.1109/IPDPSW.2012.193.

[30] G. Lin, S. He, H. Huang, J. Wu, and W. Chen, "Access control security model based on behavior in cloud computing environment," *Acta Automatica Sinica*, vol. 33, no. 3, pp. 59–66, 2012.

[31] S. Ruj, M. Stojmenovic, and A. Nayak, "Privacy preserving access control with authentication for securing data in clouds," in *Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2012, pp. 556–563, doi: 10.1109/CCGrid.2012.92.

[32] K. Yang and X. Jia, "Attributed-based access control for multi-authority systems in cloud storage," in *Proc. IEEE 32nd Int. Conf. Distrib. Comput. Syst.*, Jun. 2012, pp. 536–545, doi: 10.1109/ICDCS.2012.42.

[33] X.-Y. Li, Y. Shi, Y. Guo, and W. Ma, "Multi-tenancy based access control in cloud," in *Proc. Int. Conf. Comput. Intell. Softw. Eng.*, Dec. 2010, pp. 1–4.

[34] S.-J. Yang, P.-C. Lai, and J. Lin, "Design role-based multi-tenancy access control scheme for cloud services," in *Proc. Int. Symp. Biometrics Secur. Technol.*, Jul. 2013, pp. 273–279.

[35] H. Liang, J. Wu, S. Mumtaz, J. Li, X. Lin, and M. Wen, "MBID: Micro-blockchain-based geographical dynamic intrusion detection for V2X," *IEEE Commun. Mag.*, vol. 57, no. 10, pp. 77–83, Oct. 2019.

[36] J. Wu, M. Dong, K. Ota, J. Li, W. Yang, and M. Wang, "Fog-Computing-Enabled cognitive network function virtualization for an information-centric future Internet," *IEEE Commun. Mag.*, vol. 57, no. 7, pp. 48–54, Jul. 2019.

[37] L. Wang and X. Zhao, "Research on service composition strategy based on blockchain mechanism in cloud computing environment," *Appl. Res. Comput. (Chin.)*, vol. 26, no. 91, pp. 81–86, 2019.

[38] B. Pittl, W. Mach, and E. Schikuta, "Bazaar-blockchain: A blockchain for bazaar-based cloud markets," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Jul. 2018, pp. 89–96, doi: 10.1109/SCC.2018.00019.

[39] Y. Zhang, R. Deng, X. Liu, and D. Zheng, "Outsourcing service fair payment based on blockchain and its applications in cloud computing," *IEEE Trans. Services Comput.*, early access, Aug. 7, 2018, doi: 10.1109/TSC.2018.2864191.

[40] J. Li, J. Wu, and L. Chen, "Block-secure: Blockchain based scheme for secure P2P cloud storage," *Inf. Sci.*, vol. 465, pp. 219–231, Oct. 2018, doi: 10.1016/j.ins.2018.06.071.

[41] G. Edoardo, A. Leonardo, B. Roberto, L. Federico, M. Andrea, and V. Sassone, "Blockchain-based database to ensure data integrity in cloud computing environments,"ăin *Proc. Italian Conf. Cybersecur.*, Venice, Italy, Jan. 2017.

[42] H. G. Do and W. K. Ng, "Blockchain-based system for secure data storage with private keyword search," in *Proc. IEEE World Congr. Services (SERVICES)*, Jun. 2017, pp. 90–93, doi: 10.1109/SERVICES.2017.23.

[43] D. K. Tosh, S. Shetty, X. Liang, C. Kamhoua, and L. Njilla, "Consensus protocols for blockchain-based data provenance: Challenges and opportunities," in *Proc. IEEE 8th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, Oct. 2017, pp. 469–474, doi: 10.1109/UEMCON.2017.8249088.

[44] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "ProvChain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *Proc. 17th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*. Piscataway, NJ, USA: IEEE Press, May 2017, pp. 468–477.

[45] D. Tosh, S. Shetty, P. Foytik, C. Kamhoua, and L. Njilla, "CloudPoS: A proof-of-stake consensus design for blockchain integrated cloud," in *Proc. IEEE 11th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2018, pp. 302–309, doi: 10.1109/CLOUD.2018.00045.

[46] Y. Liu, Y. Zhou, R. Lan, and C. Tang, "Blockchain-based verification scheme for deletion operation in cloud," in *J. Comput. Res. Develop.*, vol. 66, no. 10, pp. 2199–2207, 2018.

[47] F. Huang, L. Xu, and X. Yang, "Blockchain Model of Cloud Forensics," *J. Beijing Univ. Posts Telecommun.*, vol. 40, no. 6, pp. 120–124, 2017.

[48] Y. Zhang, S. Wu, B. Jin, and J. Du, "A blockchain-based process provenance for cloud forensics," in *Proc. 3rd IEEE Int. Conf. Comput. Commun. (ICCC)*, Dec. 2017, pp. 2470–2473.

[49] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, "MeDShare: Trust-less medical data sharing among cloud service providers via blockchain," *IEEE Access*, vol. 5, pp. 14757–14767, 2017, doi: 10.1109/ACCESS.2017.2730843.

[50] I. Sukhodolskiy and S. Zapechnikov, "A blockchain-based access control system for cloud storage," in *Proc. IEEE Conf. Russian Young Researchers Electr. Electron. Eng. (EIConRus)*, Jan. 2018, pp. 1575–1578, doi: 10.1109/EIConRus.2018.8317400.

[51] S. Cui, M. R. Asghar, and G. Russello, "Towards blockchain-based scalable and trustworthy file sharing," in *Proc. 27th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2018, pp. 1–2.

CAIXIA YANG was born in Chengdu, Sichuan, China, in 1995. She received the bachelor's degree in engineering from Sichuan Normal University, in 2018, where she is currently pursuing the master's degree in computer science and technology. Her main research direction is information security.

LIANG TAN received the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, in 2007. He is currently a Professor with the School of Computer Science, Sichuan Normal University. His research interests include cloud computing, big data, trusted computing, and network security.

NA SHI was born in Meishan, Sichuan, China, in 1996. She received the B.S. degree in engineering from Sichuan Normal University, in 2018, where she is currently pursuing the master's degree in software engineering. Her research focuses on information security and the Internet of Things (IoT) access control.

BOLEI XU received the Ph.D. degree from the University of Nottingham, U.K. is currently a Postdoctoral Researcher with the College of Information Engineering, Shenzhen University. His main research interests include deep image processing, computer-aided diagnosis, and computer vision related areas.

YANG CAO received the B.S. degree from the School of Software Engineering, Northwestern Polytechnical University, China, in 2008, and the M.S. and Ph.D. degrees from the Graduate School of Informatics, Kyoto University, Japan, in 2014 and 2017, respectively. He was a Postdoctoral Fellow with the Department of Math and Computer Science, Emory University, USA. He is currently an Assistant Professor with the Department of Social Informatics, Kyoto University. His research interests are privacy preserving data analysis, data market, and blockchain.

KEPING YU (Member, IEEE) received the M.E. and Ph.D. degrees from the Graduate School of Global Information and Telecommunication Studies, Waseda University, Tokyo, Japan, in 2012 and 2016, respectively.

He was a Research Associate with the Global Information and Telecommunication Institute, Waseda University, from 2015 to 2019, where he is currently a Junior Researcher. He has hosted and participated in a lot of research projects, including the Ministry of Internal Affairs and Communication (MIC) of Japan, the Ministry of Economy, Trade and Industry (METI) of Japan, the Japan Society for the Promotion of Science (JSPS), the Advanced Telecommunications Research Institute International (ATR) of Japan, Keihin Electric Railway Corporation of Japan, and Maspro Denkoh Corporation of Japan. He is also the Leader and a Coauthor of the comprehensive book design and implementation of *Information-Centric Networking* (Cambridge University Press, 2020). He was involved in many standardization activities organized by ITU-T and ICNRG of IRTF, and contributed to the ITU-T Standards ITU-T Y.3071: Data Aware Networking (Information-Centric Networking) Requirements and Capabilities and Y.3033-Data Aware Networking-Scenarios and Use Cases. His research interests include smart grids, information-centric networking, the Internet of Things, blockchain, and information security. He had experience with editorial and conference organizations. Moreover, he has served as a TPC Member of the ITU Kaleidoscope 2020, IEEE VTC2020-Spring, IEEE CCNC 2020, IEEE WCNC 2020, IEEE VTC2019-Spring, ITU Kaleidoscope 2019, IEEE HotICN 2019, IEEE ICCC 2019, IEEE WPMC 2019, EEI 2019, and ICITVE 2019. He is also an Editorial Board Member of the IEEE Open Journal of Vehicular Technology (OJVT). He has served as a Leading Guest Editor of *Sensors* Special Issue *Emerging Blockchain Technology Solutions for Real-World Applications* (EBTSRA), *Peer-to-Peer Networking and Applications* Special Issue Blockchain for Peer-to-Peer Computing, a General Co-Chair and a Publicity Co-Chair of IEEE VTC2020-Spring EBTSRA Workshop, the TPC Co-Chair of SCML2020, the Local Chair of MONAMI 2020, and the Session Co-Chair of CcS2020 and ITU Kaleidoscope 2016.

• • •