IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 9, 2020, accepted April 16, 2020, date of publication April 22, 2020, date of current version May 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2988951

A Blockchain-Based Access Control Framework for Cyber-Physical-Social System Big Data

LIANG TAN[1,2], NA SHI[1], CAIXIA YANG[1], AND KEPING YU[3,4], (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]Global Information and Telecommunication Institute, Waseda University, Tokyo 169-8050, Japan
[4]Shenzhen Boyi Technology Company Ltd., Shenzhen 518125, China

Corresponding author: Liang Tan (tanliang2008cn@126.com)

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 Cyber-Physical-Social System (CPSS) big data is specified as the global historical data which is usually stored in cloud, the local real-time data which is usually stored in the fog-edge server (*FeS*) of the mobile terminal devices or sensors, and the social data which is usually stored in the social data server (*SdS*), moreover adopts a centralized access control mechanism to offer users' access strategy which can easily cause CPSS big data to be tampered with and to be leaked. Therefore, a blockchain-based access control scheme called BacCPSS for CPSS big data is proposed. In BacCPSS, account address of the node in blockchain is used as the identity to access CPSS big data, the access control permission for CPSS big data is redefined and stored in blockchain, and processes of authorization, authorization revocation, access control and audit in BacCPSS are designed, and then a lightweight symmetric encryption algorithm is used to achieve privacy-preserving. Finally, a credible experimental model on EOS and Aliyun cloud is built. Results show that BacCPSS is feasible and effective, and can achieve secure access in CPSS while protecting privacy.

INDEX TERMS CPSS, CPS, access control, blockchain, transaction.

I. INTRODUCTION

Cyber-Physical-Social System (CPSS) [1]–[3] integrates the cyber, physical and social spaces together. One of the ultimate goals of CPSS is to make our lives more convenient and intelligent by providing prospective and personalized services for users [4]–[7]. CPSS big data is complex and heterogeneous, and records all aspects of users' lives in the forms of image, audio, video and text. Generally, the collected or generated data in CPSS satisfies 4Vs (volume, variety, velocity, and veracity) of big data. CPSS big data is specified as the global historical data, the local real-time data and the extensive social data. Firstly, cloud computing [8], [9] in processing global historical data, which acts as a powerful paradigm for implementing the data-intensive applications, has an irreplaceable role; secondly, with the increasing computing capacity and communication capabilities of mobile terminal devices and sensors, fog-edge computing [10]–[12], as an important and effective supplement of cloud computing, has been widely used to process the local real-time data; finally, for coordination between physical system, information system and social networks composed of human beings, social data server [13], [14] which integrates human knowledge, mental capabilities, and sociocultural elements has become a more and more essential part.

Whether cloud platform, fog-edge server or social data server, they adopt a centralized access control mechanism to offer users' access. However, the security and privacy issues of CPSS big data [15] have been widely concerned [16], the access right in authorization database of cloud platform, fog-edge server and social server is easily being tampered by administrator or attackers, this centralized management method is prone to lead to disclose CPSS big data.

In this paper, we propose an access control scheme called BacCPSS for CPSS that is based on the blockchain [17]–[19] which has the characteristics of decentralization, without tampering and trustworthiness. The main contributions are as follows:

1) Formally analyze the threats existing in the distributed architecture of CPSS big data and the traditional access control model.

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Arafatur Rahman.

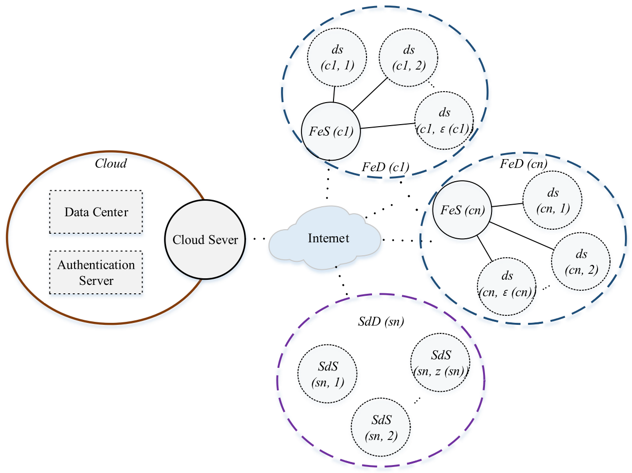


FIGURE 1. **Distributed architecture for CPSS big data.**

2) The BacCPSS access control architecture is proposed, using the account address of the blockchain node as the identity, redefining access permissions, designing the initialization, access control, authorization, authorization revocation and audit processes, and using lightweight symmetric encryption algorithm to achieve privacy protection.
3) Establish an experimental model on EOS and Alibaba Cloud, and prove the effectiveness of BacCPSS by evaluating the three indicators of defined computation overhead, storage overhead, and throughput.

This manuscript is organized as follows, background and threat model are summarized in section II, followed by related work in section III. We propose the detailed construction of our blockchain-based access control scheme for CPSS in section IV. Section V is the security and performance analysis respectively. Section VI is experiment and evaluation. Finally, we end up with a conclusion and future work in section VII.

II. BACKGROUND AND THREAT MODEL

A. DISTRIBUTED ARCHITECTURE FOR CPSS BIG DATA

The distributed architecture for CPSS big data is shown in Figure 1. Including cloud (*Cloud*), decentralized distributed fog-edge domains (*FeD*) and decentralized distributed social data domains (*SdD*).

Firstly, the decentralized cloud architecture is considered. Cloud (*Cloud*) connects a large number of fog-edge servers (*FeS*) and social data servers (*SdS*), and communicates these fog-edge servers by Internet and achieves real-time data, meanwhile communicates these social data servers by Internet too and achieves social data. Cloud can achieve a reliable and collaborative control and management by integrating, storing and coordinating social resources, computing resources and physical resources. Cloud domain contains cloud server, data center, and authentication server. As the entrance and exit of cloud, the cloud server is used to computing and responding to external request. The data center is used to store data that will be computing or consolidated from fog-edge servers and social data servers. The authentication server provides access control services and identity authentication services. In Figure 1, Cloud represents the cloud domain.

Secondly, decentralized distributed fog-edge architecture is considered. Fog-edge server integrates a large number of resource-constrained devices, such as mobile terminal devices, sensors, and so on, into a topological structure with decentralized features. Each device belongs to a unique fog-edge manager which called fog-edge server, and each fog-edge server has many different devices. The fog-Edge server and all its subordinate devices form a fog-edge domain, devices can cooperate within or between fog-edge domains to achieve specific needs, and fog-edge domains can cooperate with each other [20]. To this end, the fog-edge server of fog-edge domains needs to establish, evaluate, and update trust relationships with each other. In Figure 1, $FeS(x)$ represents the fog-edge server, $ds(x, y)$ represents devices managed by $FeS(x)$, $FeD(x)$ represents the fog-edge domain, and $\varepsilon(x)$ represents the maximum number of devices in $FeD(x)$. Formally, we have $x \in \{\mu \in N* | 1 \leq \mu \leq \alpha, \alpha \in N*\}$, $y \in \{\omega \in N* | 1 \leq \omega \leq \varepsilon(x), \varepsilon(x) \in N*\}$, where α is the number of fog-edge domain in the network. In Figure 1, $1 \leq c1 \leq \alpha$, $1 \leq cn \leq \alpha$, and other fog-edge domains except $FeD(c1)$ and $FeD(cn)$ are omitted. In this architecture, we assume that there is a secure and stable communication link between *FeS* and devices, which are implemented by their own protocols and communication mechanisms in each fog-edge domain.

Finally, the decentralized distributed social data system architecture is considered. Social data system includes many social data collection systems which can collect social data, many social data search systems which can search social data and many social intelligent systems which can mine, analyze, depth learn social data. Every system is able to upload social data to data center of cloud via internet by communicating with cloud server. In Figure 1, $SdS(x)$ represents the social data server, *SdD* represents the social data domain, $z(x)$ represents the maximum number of *SdS* in *SdD*. Formally, $x \in \{\lambda \in N* | 1 \leq \lambda \leq \beta, \beta \in N*\}$, $y \in \{\theta \in N* | 1 \leq \theta \leq z(x), z(x) \in N*\}$, where β is the number of social data system in the network. In Figure 1, $1 \leq sn \leq \beta$, and other social data system except $SdS(sn)$ are omitted.

B. TRADITIONAL ACCESS CONTROL MODEL

Traditional access control framework is shown in Figure 2, it is divided into identity authentication, access control, access permission and audit [21], [22]. When a user sends a request, the system's access control module first verifies the user's identity, and once the verification is passed, then performs corresponding operations on the resource according to the access policy and the corresponding permissions in the authorization database. Finally, access is recorded in a log for auditing and tracking.

C. ACCESS CONTROL MODEL OF CPSS BIG DATA

Furthermore, in distributed CPSS, *Cloud* identifies, authenticates, and connects *FeS* and *SdS*, *FeS* identifies, authenticates,

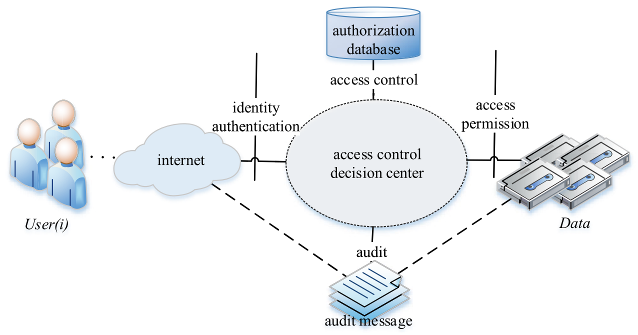


FIGURE 2. Traditional access control framework.

TABLE 1. Access matrix M of users.

Access Subject *User(i)*	Target Data (*Data*)			
	DataF(i)	***DataFC(i)***	***DataSD(i)***	***DataSDC(i)***
	$<dataf_1,$ T_1, $ds(c_1,1)$, *FeD(c1)*>	<*DataF(1)*, T_{upload}>	$<datasd_i,$ $T_{datasdi},$ *SdS(i), SdD*>	<*DataSD(i)*, $T_{DataSD(i)upload}$>
$<u1, FeD(c1)>$	r, p	x	x	x
$<u1, FeD(c2)>$	r, p	x	x	x
$<u1, FeD(x)>$	r, p	x	x	x
$<u1, SdD(1)>$	x	x	r, p	x
$<u1, SdD(2)>$	x	x	r, p	x
$<u1, SdD(x)>$	x	x	r, p	x
$<u(n), CloudD>$	x	r, p	x	r, p

and connects and manages all decentralized *ds*, moreover, *SdS* identifies, authenticates, and manage all social data. Therefore, it is important for users to obtain permission to access the target data on the *FeS*, *SdS* and data center of *Cloud* through access control framework. In order to more clearly describe the access control framework in distributed CPSS, some notations are defined next as follow.

- ***User(i)***. Represents user identity that generates access requests, includes the user's *ID* and the management domain which belongs to. Formally, $User(i) = \langle UID_i, Domain\rangle$, in which, $Domain = \{FeD, SdD, CloudD\}$, in which, $FeD = \{FeD(c1), FeD(c2), \cdots, FeD(cn)\}$, which represents the physical devices management domain in CPSS, $SdD = \{SdD(s1), SdD(s2), \cdots, SdD(sn)\}$, which represents the social data management domain in CPSS, and *CloudD* represents the cloud domain in CPSS.
- ***DataF(i)***. Represents the local real-time data set stored in a *FeS*, includes data block *i*, the time of data block *i* collection, devices which data belongs to and the management domain which belongs to. Formally, $DataF(i) = \langle dataf_i, T_{dataf_i}, ds(c_i, 1), FeD(c_i)\rangle$.
- ***DataFC(i)***. Represents the global historical data set stored in the data center of *Cloud*, includes $DataF(i)$ and the upload time of $DataF(i)$. Formally, $DataFC(i) = \langle DataF(i), T_{DataF(i)-upload}\rangle$, where $T_{DataF(i)-upload}$ represents the upload time of $DataF(i)$.
- ***DataSD(i)***. Represents the social data set stored in the social data sever, includes data block *i*, the production time of the social data block *i*, the social data sever and domain. Formally, $DataSD(i) = \langle datasd_i, T_{datasd_i}, SdS(i), SdD\rangle$.
- ***DataSDC(i)***. Represents the social data set stored in the data center of cloud sever, includes $DataSD(i)$ and the upload time of $DataSD(i)$. Formally, $DataSDC(i) = \langle DataSD(i), T_{DataSD(i)-upload}\rangle$, where $T_{DataSD(i)-upload}$ represents the upload time of $DataSD(i)$.
- ***Data***. Represents all data in CPSS, including all $DataF(i)$, all $DataFC(i)$, all $DataSD(i)$ and all $DataSDC(i)$.
- ***Right***. Represents the set of permission. Formally, $Right = \{own, execute, read, write, delete, download\}$.
- ***A***. Represents a set of access which describes a request subject, target data, access mode and access strategy. Here, $A = (u, d, r, p)$, where u represents the subject attribute, d represents the data attribute, r represents the permission attribute, p represents the policy attribute. And $u \subseteq User(i)$, $d \subseteq Data$, $r \subseteq Right$. A must contain u, d, and r, where p is optional. If p does not exist, it means using the default access policy called Discretionary Access Control (DAC).
- ***M***. Represents the user access matrix, ensuring that only operations authorized in the permission set matrix can be performed, as shown in Table 1.

D. THREAT MODELS

In CPSS, since the domain management server involves operations on the security attributes of data, users have security requirements for the data they want to access [23]. Generally, users' security requirements for information system are based on the following aspects:

1) Confidentiality: to prevent information from being leaked to unauthorized users.
2) Integrity: to prevent the unauthorized users from modifying the information.
3) Availability: to ensure the accessibility of authorized users to system information.

Due to the centralized nature of the traditional access control framework for CPSS, there are two kinds of security threats to the above security requirements and one problem as follows.

Attack 1. Stealing or Modifying ***Data****:* An attacker illegally steals or modifies the target data on *FeD*, *SdD* or *CloudD* management server leading to the leakage of the target data and destroying its confidentiality and integrity requirements. Specific operations including SQL injection, identity hijacking etc. can enable an attacker to bypass identity verification and directly steal or modify target data stored in *FeD*, *SdD* and *CloudD* management server.

Attack 2. Modifying ***M*** *in the Authorization Database:* An attacker illegally tampers or destroys the authorization database on *FeD*, *SdD* and *CloudD* management server, resulting in tampered data or unavailability of permissions in *M*. The specific operation is that after the attacker modifies the authorization database by adding himself as an authorized user, and then legally passing access control through this identity. This attack modifies the access control policy of the decision center, which will be more serious and worrying than the previous threat.

Problem 1. It Is Difficult for Users to Manage Their Identities: The number of domains has increased dramatically with the development of CPSS. Users accessing different domains need to register different accounts, which greatly increases difficulties of them in identity management.

III. RELATED WORK

Since 2007, American government has treated Cyber-Physical System (CPS) as a new development strategy. Some researchers from various countries discussed the related concepts, technologies, applications and challenges during CPS week and the international conference on CPS subject [24]. The results of these researches are mainly divided into energy control, security control, transmission and management, model-based software design, control technique, and system resource allocation [25]–[27], moreover, access control in CPS has caused widespread concern [28], [29].

With the development of artificial intelligence, deep learning and other technologies, on the basis of CPS, CPSS further incorporates social information, expands the scope of research to social network system, and has been applied in many fields such as smart enterprises, smart transportation, smart homes, and smart medical care [30]–[34]. In literature [35], many necessary constraints in CPSS are being considered together, e.g., the execution time, energy consumption, economic cost, security as well as reliability. Yang *et al.* proposed a general model for tensor computation that optimizes the execution time, energy consumption, and economic cost with acceptable security and reliability. To provide high-quality, proactive, and personalized services for humans, Wang *et al.* [36] proposed a tensor-based cloud-edge computing framework which includes the cloud and edge planes, in which the cloud plane is used to process large-scale, long-term, global data, which can be used to obtain decision making information. The edge plane is used to process small-scale, short-term, local data, which is used to present the real-time situation and provide personalized services for humans. To provide lower-latency, real-time, more effective, and proactive services for human, Wang *et al.* [37] proposed an edge cloud-assisted CPSS framework for smart cities which migrates some tasks from the cloud center to network edge devices and puts the services and resources closer to users. Sharma *et al.* [38] proposed a privacy aware access control model with k- anonymity for CPSS. While enabling functionality, it allows users access at different privacy levels by generating an anonymized data set in accordance with the privacy clearance of a certain request. To solve the big deal of computation and the complexity of networking contextual, Hussein *et al.* [3] proposed a dynamic social structure of things called DSSoT and proposed a novel smart services framework in CPSS. Moreover, they did a proof of concept for an application scenario called Airport Dynamic Social by using DSSoT. In order to protect the important data stored in CPS, Akhuseyinoglu *et al.* [39] proposed an access control framework composed of a cyber-physical access control model (CPAC) and a generalized action generation model (GAGM), and provided a CPS example scheme for medical treatment by using an algorithm which is enforcing authorization policies.

In summary, there are few researches on the realization of data security access in the system by constructing an access control model in CPSS. We propose a blockchain-based secure access control framework called BacCPSS to achieve granted and security access in cloud domain, fog-edge domain and social domain. It will be described in detail later in section IV.

IV. OVERVIEW OF BacCPSS IN CPSS

To solve problems described in section II, we propose BacCPSS. BacCPSS is a novel access control scheme based on blockchain which can preserve privacy, as shown in Figure 3. It removes the central authorization database from the traditional architecture and adds blockchain. This scheme mainly includes *DO*, *DV*, *FeS*, *SdS*, *Cloud* and *Blockchain* six entities. We next describe some notations that will be later used.

- ***DO***. A data owner information set, $DO = \{DOAddr, PK_{DO}, SK_{DO}, Enc(), Dec()\}$, including DO's user address, public key PK_{DO} and private key SK_{DO}, the function *Enc*() which is to encrypt, and the function *Dec*() which is to decrypt, both *Enc*() and *Dec*() select a lightweight symmetric encryption and decryption algorithm, and the same as follow.
- ***DV***. A data visitor information set, $DV = \{DVAddr, PK_{DV}, SK_{DV}, Enc(), Dec()\}$, including DV's user address, public key PK_{DV} and private key SK_{DV}.
- ***FeS***. A fog-edge server information set, $FeS = \{FeS\text{-}Addr, PK_{FeS}, SK_{FeS}, Enc(), Dec()\}$, including user address, public key PK_{FeS}, and private key SK_{FeS}.
- ***SdS***. A social data server information set, $SdS = \{SdS\text{-}Addr, PK_{SdS}, SK_{SdS}, Enc(), Dec()\}$, including user address, public key PK_{SdS}, and private key SK_{SdS}.
- ***Cloud***. A cloud information set, $Cloud = \{CloudAddr, PK_{Cloud}, SK_{Cloud}, Enc(), Dec()\}$, including user address, public key PK_{Cloud}, and private key SK_{Cloud}.
- ***right***. A permission flag set. An 8-bit binary number is used to represent the above-formed permission set right. Without this permission, the bit is represented as 0. If the number of permissions is insufficient, it is reserved as an extension bit. Therefore, the permission set flag of *DO* is initialized to "11111100", and the permission set flag of the *DV* is initialized to "00000000".
- ***permiCap***. A capability set of users, $permiCap = \{DVAddr, <FeSAddr.data, CloudAddr.data, SdSAddr.data>, right\}$, which includes the visitor's address

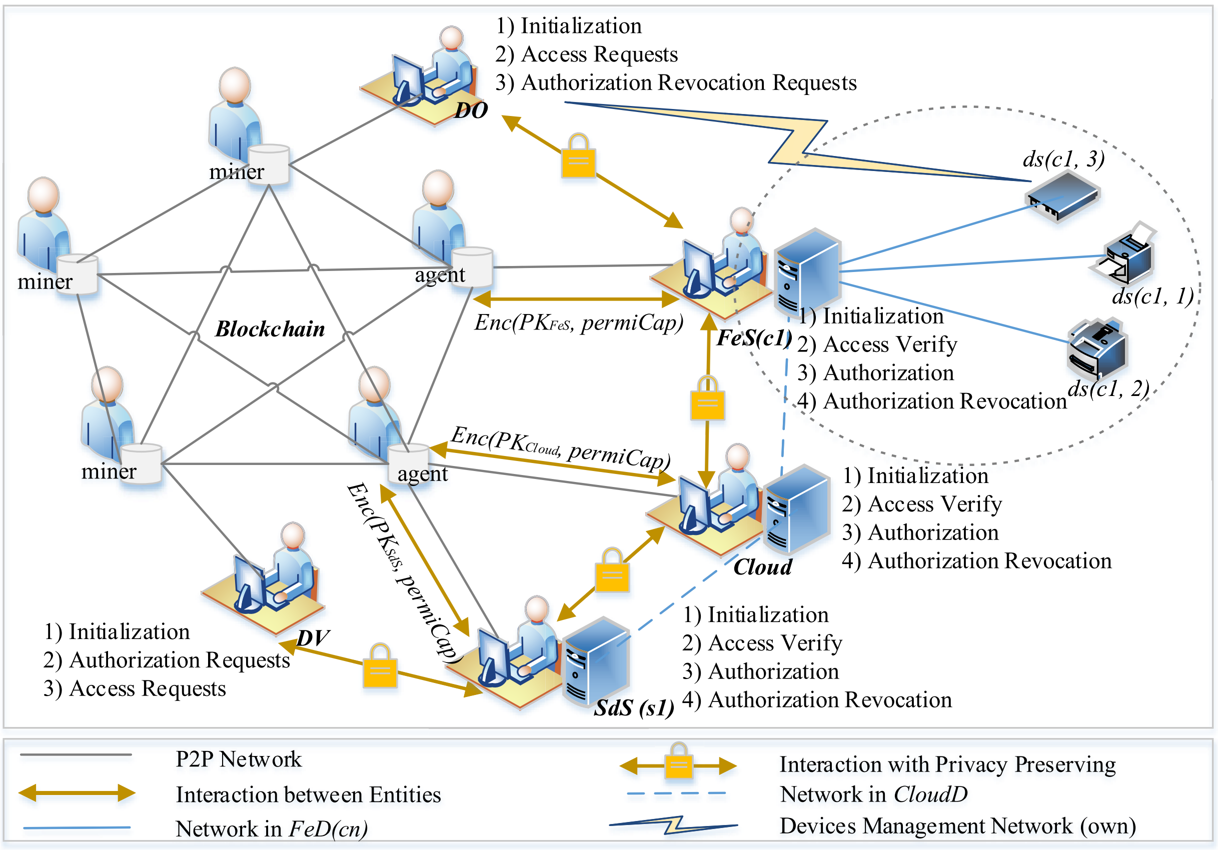


FIGURE 3. system framework of BacCPSS.

DVAddr, target data in this *FeS*, *SdS* or *Cloud*, and the specific permissions the visitor has. Where $data \subsetneq Data$.

- ***Blockchain***. A database of decentralization, without tampering and trustworthiness. *Blockchain* stores *permiCap* of user and *accesslog* which is user access log for *Data*.

Here we introduce the trading interface in blockchain:

- ***sendTransaction (from, to, value, data)***. Where *from* represents the address of sender, *to* represents the address of receiver, and *value* represents the amount of transaction, default is 0, and *data* indicates additional information. In the scenario of this article, we add the relevant access control information to *data*. This function returns hash value of this transaction.

In BacCPSS, access control mainly includes initialization, access, authorization and authorization revocation. We have listed the main functions of each entity, including but not the only. In order to protect the privacy of sensitive data, we have designed an interactive process between entities. Throughout BacCPSS, we use Diffie-Hellman (DH) method to consult and obtain the shared key (*Ks*) between entities. Given the efficiency of asymmetric encryption, we only use public keys to protect symmetric keys, as shown in Figure 4. Next, we give detailed function introductions of each entity in BacCPSS.

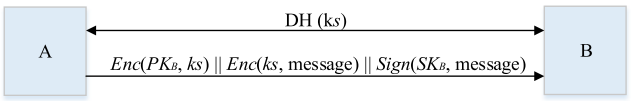


FIGURE 4. Interaction for shared key.

A. INITIALIZATION

The main function of initialization is registering *DO*, *DV*, *FeS*, *SdS* and *Cloud* to *Blockchain*, let them become legal light nodes of Blockchain. Firstly, *DO*, *DV*, *FeS*, *SdS* and *Cloud* all need to download and install "Geth" and connect to *Blockchain* service, and then generate respective key pairs and send the public key to *Blockchain* to generate account addresses. Specific steps are shown in Figure 5. It is worth mentioning that if you need to manage your account systematically, you can use a wallet program, such as "MetaMask", which can synchronize all block data. Finally, *FeS* publishes permiCap of *DO* in its *FeD* to *Blockchain*, *SdS* publishes

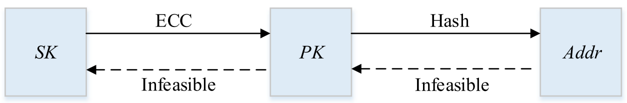


FIGURE 5. Generation of address.

permiCap of *DO* in its *SdD* to *Blockchain*, and *Cloud* publishes permiCap of *DO* in this *CloudD* to *Blockchain*. These three processes are basically same, and each process is implemented by function *initPublish*.

- ***initPublish (FeSAddr (or SdSAddr, or CloudAddr), DOAddr, permiCap)***. The function is to encrypt and publish the permiCap of *DO* who owns the data and stores in *FeS*, *SdS* or *Cloud* to *Blockchain*. Parameters include the address of *FeS*, *SdS* or *Cloud*, the address of *DO* and specific access permission capability *permiCap*, which is equal to {*DOAddr*, *FeSAddr.data* (or *SdSAddr.data*, or *CloudAddr.data*), "11111100"}, the executor of this function is *FeS*, *SdS* or *Cloud*, which returns "True" for success and "False" for failure, as shown in Algorithm 1.

Algorithm 1 ***initPublish***

```
Input   FeSAddr/SdSAddr/CloudAddr, DOAddr, permiCap
Output  True or False
Begin
  01. if(permiCap. right! = "11111100") then
  02.     return False;
  03. end if;
  04. if(permiCap. DVAddr! = DOAddr) then
  05.     return False;
  06. end if;
  07. data = Enc ($PK_{FeS}$, permiCap ||initTime);
      or data = Enc ($PK_{SdS}$, permiCap ||initTime);
      or data = Enc ($PK_{Cloud}$, permiCap ||initTime);
  08. tx = sendTransaction (FeSAddr, DOAddr, 0, data);
      or tx = sendTransaction (SdSAddr, DOAddr, 0, data);
      or tx = sendTransaction (CloudAddr, DOAddr, 0, data);
  09. return True;
End
```

In Algorithm 1, it needs to first check whether the published *permiCap* is owned by *DO* and determine the beneficiary of this *permiCap* as the *DO*. Then *FeS* (or *SdS*, or *Cloud*) uses its public key PK_{FeS} (or PK_{SdS}, or PK_{Cloud}) to encrypt *permiCap*||*initTime*, and finally call *sendTransaction* through *FeS* (or *SdS*, or *Cloud*) to publish the *permiCap* of the *DO* to *Blockchain*.

B. ACCESS PROCESS WITH PRIVACY PRESERVING

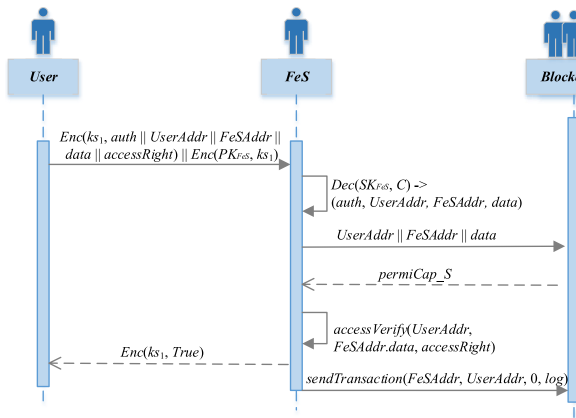
After initializing successfully, the *DV* may initiate a visit request to *FeS* or *Cloud* with "*accessRight*". We set the flag "auth", when its value equal 0, it means a visit request, that a value equal 1 indicates authorization request, and that a value equal 2 indicates authorization revocation request. "True" and "False" in the message indicate whether the operation was successful, and # indicates the transaction *ID*. To clearly describe this process, we next definite a function called *accessVerify*.

- ***accessVerify (DVAddr, FeSAddr.data (or SdSAddr.data, or CloudAddr.data), accessRight)***. The function is to verify whether this visitor has the access right, parameters include address of *DV*, target data and access right. When this access request verification is successful, it returns "True", otherwise, it returns "False", as shown in Algorithm 2.

Algorithm 2 ***accessVerify***

```
Input   DVAddr, FeSAddr.data/SdSAddr.data
        /CloudAddr.data, accessRight
Output  True or False
Begin
  01. search from Blockchain by DVAddr and FeSAddr.data
      (or SdSAddr.data, or CloudAddr.data) to get permiCap;
  02. Dec ($SK_{FeS}$, result);
      or Dec ($SK_{SdS}$, result);
      or Dec ($SK_{Cloud}$, result);
  03. if(permiCap.right & accessRight == 0) then
  04.     return False;
  05. end if;
  06. return True;
End
```

In Algorithm 2, *FeS* (or *SdS*, or *Cloud*) needs to search the latest *permiCap* from *Blockchain* by *DVAddr* and *FeSAddr.data* (or *SdSAddr.data*, or *CloudAddr.data*), and decrypt it with SK_{FeS} (or SK_{SdS}, or SK_{Cloud}), specific search methods are not introduced here. By doing bitwise AND ("&") operations between *permiCap.right* and *accessRight* to get a result. If the result equals 0, it means the *DV* has no access, otherwise, *FeS*, *SdS*, or *Cloud* can let this *DV* visit. After using PK_{FeS} (or PK_{SdS}, or PK_{Cloud}) to encrypt this message "*DVAddr*||*FeSAddr* (or *SdSAddr*, or *CloudAddr*) ||*accessRight*||*accessTime*", *FeS* (or *SdS*, or *Cloud*) publishes the access record with privacy preserving to *Blockchain*. Performer of this function is *FeS*, *SdS* or *Cloud*. The specific access process is shown in Figure 6. And detailed steps are as follows:



User
FeS
Blockchain
Enc(ks1, auth || UserAddr || FeSAddr || data || accessRight) || Enc(PKFeS, ks1)
Dec(SKFeS, C) -> (auth, UserAddr, FeSAddr, data)
UserAddr || FeSAddr || data
permiCap_S
accessVerify(UserAddr, FeSAddr.data, accessRight)
Enc(ks1, True)
sendTransaction(FeSAddr, UserAddr, 0, log)

FIGURE 6. **Access process with privacy protection.**

1) *User -> FeS*: $Enc(ks1, auth\|UserAddr\|FeSAddr\|data\| accessRight)\|Enc(PK_{FeS}, ks1)$. User sends a request for data to *FeS*, where *auth* is a flag indicating the type of request.
2) *FeS -> Blockchain*: *UserAddr*‖*FeSAddr*‖*data*. *FeS* decrypts $Enc(PK_{FeS}, ks1)$ by SK_{FeS} to get $ks1$, and decrypts $Enc(ks1, auth\|UserAddr\|FeSAddr\|data \|acc\text{-}essRight)$ by $ks1$ to get *auth*, *UserAddr*, *FeSAddr*, *data* and *accessRight*, and then sends a request to *Blockchain* with *UserAddr*‖*FeSAddr*‖*data*.
3) *Blockchain -> FeS*: *permiCap_S*. *Blockchain* searches and returns *permiCap_S* to *FeS*.
4) *FeS*: *accessVerify* (*UserAddr*, *FeSAddr. data*, *access-Right*), *FeS* decrypts *permiCap_S* by SK_{FeS} to get *per-miCap*, and calls *accessVerify*() to verify whether the *User* can access the data by comparing *accessRight* with *permiCap*, if no, it is over, else, got to next step.
5) *FeS -> User*: $Enc(ks1, True)$. *FeS* returns $Enc(ks1, True)$ to *User*. *User* decrypts it, and accesses the data by *accessRight*.
6) *FeS -> Blockchain*: *sendTransaction*(*FeSAddr*, *User-Addr*, 0, *log*). *FeS* sends access log which is $Enc(PK_{FeS}, permiCap\|accessTime)$ to *Blockchain* by calling *send-Transaction*(), where *log* is a flag indicating the type of log.

C. AUTHORIZATION PROCESS WITH PRIVACY PRESERVING

In this section, we will present a privacy-protected authorization process for how to authorize *DV* to access to the data in *FeS* (or *SdS*, or *Cloud*). In this case, the value of "auth" is equal to 1. In fact, authorization can be divided into direct authorization and indirect authorization. Due to the real needs of CPSS, only direct authorization is considered here. To clearly describe this process, we next definite a function called *authGrant*.

- ***authGrant (DVAddr, FeSAddr.data (or SdSAddr.data, or CloudAddr.data), addRight)***. The function is to grant new rights to a visitor, parameters include address of requester, address of *FeS*, *SdS* or *Cloud* where the target data is requested, and the new rights will be granted. Success or failure will return the latest *permiCap* of this user, as shown in Algorithm 3.

Algorithm 3 ***authGrant***

```
Input   DVAddr, FeSAddr.data/SdSAddr.data
        /CloudAddr.data, addRight
Output  permiCap
Begin
  01.  search from Blockchain by DVAddr and FeSAddr.data
       (or SdSAddr.data, or CloudAddr.data) to get permiCap;
  02.  Dec ($SK_{FeS}$, result);
       or Dec ($SK_{SdS}$, result);
       or Dec ($SK_{Cloud}$, result);
  03.  permiCap.right = permiCap.right | addRight;
  04.  return permiCap;
End
```

In Algorithm 3, *FeS* (or *SdS*, or *Cloud*) also needs to search the latest *permiCap* from *Blockchain* by *DVAddr* and *FeSAddr.data* (or *SdSAddr.data*, or *CloudAddr.data*), and decrypt it with SK_{FeS} (or SK_{SdS}, or SK_{Cloud}). By doing bitwise OR ("|") operations between *permiCap.right* and *addRight* to get the latest *permiCap*. Since the right initialization value of each *DV* is not null, if the *FeS* (or *SdS*, or *Cloud*) grants successfully, this operation finally returns the latest *permiCap* and *FeS* (or *SdS*, or *Cloud*) will using PK_{FeS} (or PK_{SdS}, or PK_{Cloud}) to encrypt "*DVAddr*‖*FeSAddr* (or *SdSAddr*, or *CloudAddr*)‖*permiCap*‖*grantTime*" and publish the authorization record with privacy preserving to *Blockchain*. Performer of this function is *FeS*, *SdS* or *Cloud*.

D. AUTHORIZATION REVOCATION PROCESS WITH PRIVACY PRESERVING

In this section, we will present a privacy-preserving authorization revocation process detailing how to revoke *DV*'s right. In this case, the value of "auth" is equal to 2. To clearly describe this process, we next definite a function called *authRevoke*.

- ***authRevoke (DVAddr, FeSAddr.data (or SdSAddr.data, or CloudAddr.data), deleteRight)***. The function is to revoke the right of *DV* to the target data in *FeS* (or *SdS*, or *Cloud*). Parameters include the address of *DV* who needs to be revoked, target data, and revoked rights. Success or failure will return the latest *permiCap* of this user, as shown in Algorithm 4.

In Algorithm 4, *FeS* (or *SdS*, or *Cloud*) also needs to search the latest *permiCap* from *Blockchain* by *DVAddr* and *FeSAddr.data* (or *SdSAddr.data*, or *CloudAddr.data*), and decrypt it with SK_{FeS} (or SK_{SdS}, or SK_{Cloud}). By doing bitwise AND ("&") operations between *permiCap.right* and

Algorithm 4 ***authRevoke***

```
Input   DVAddr, FeSAddr.data/SdSAddr.data
        /CloudAddr.data, deleteRight
Output  True or False
Begin
  01.  search from Blockchain by DVAddr and FeSAddr.data
       (or SdSAddr.data, or CloudAddr.data) to get permiCap;
  02.  $Dec\ (SK_{FeS}, \text{result})$;
       or $Dec\ (SK_{SdS}, \text{result})$;
       or $Dec\ (SK_{Cloud}, \text{result})$;
  03.  if(permiCap. right & deleteRight == 0) then
  04.      return permiCap;
  05.  end if;
  06.  $permiCap.right = permiCap.right \wedge deleteRight$;
  07.  return permiCap;
End
```

deleteRight to get a result. If the result equals 0, it means the *DV* does not have this permission which not need to revoke. If the result equals 1, it means *deleteRight* $\subseteq$ *permiCap.right*, then *FeS* (or *SdS*, or *Cloud*) needs do XOR ("$\wedge$") operation to get the latest *permiCap*. If the *FeS* (or *SdS*, or *Cloud*) revokes successfully, then using PK_{FeS} (or PK_{SdS}, or PK_{Cloud}) to encrypt "*DVAddr*‖*FeSAddr* (or *SdSAddr*, or *CloudAddr*)‖*permiCap*‖*revokeTime*" and publishing the authorization revocation record with privacy preserving to *Blockchain*. Performer of this function is *FeS*, *SdS* or *Cloud*.

V. SECURITY AND FEATURE ANALYSIS

Blockchain is a kind of distributed ledger technology based on peer-to-peer (P2P) network that provides services by using consensus mechanism and encryption algorithm. It links a large number of data blocks and has tamper-proof, completely transparent and block verifiability characteristics. In BacCPSS, after user, *FeS*, *SdS* or *Cloud* becomes normal non-malicious node in the blockchain through the initialization phase, the entire traditional centralized access control becomes decentralized. Moreover, each behavior triggers a transaction. Next, we will conduct a detailed security analysis of BacCPSS to address the threat models proposed above.

A. SECURITY ANALYSIS

In CPSS, our solution is able to resist Attack 1 and Attack 2 existing in section II by taking advantage of the features of decentralization and transaction mechanism. Firstly, once the attacker uses the identity *DVAddr* to initiate access to the target data in a *FeS*, or a *SdS*, or *Cloud* without authorization, it will trigger the *FeS* (or *SdS*, or Cloud) to query {*DVAddr, FeSAddr.data, right*} (or {*DVAddr, SdSAddr.data, right*}, or {*DVAddr, CloudAddr.data, right*}) from *Blockchain* to obtain the *permiCap* which is the latest access capability of this *DV* for target Data, so as to prevent Data tampering and disclosure. Secondly, since *permiCap* is stored in *Blockchain* and shared by all participating nodes, an attacker cannot agree to pass even if he tampers with the local *permiCap*, which results in the invalid operation and protects the authorized data *M* in the *FeS* (or *SdS*, or *Cloud*) from being tampered with and destroyed. Then, because of *DVAddr*, *DOAddr* and *FeSAddr* (or *SdSAddr*, or *CloudAddr*) authenticating during the whole access control process, SK_{DO}, SK_{DV} and SK_{FeS} (or SK_{SdS}, or SK_{Cloud}) are much more secure and infeasible than the current normal identity (username/password), so BacCPSS greatly reduces the possibility that an attacker impersonates an authorized user to log in and improves the security of access control of target data. Finally, in BacCPSS, since every valid behavior of a node in *Blockchain* includes access records "$DVAddr \| FeSAddr \| Enc(PK_{FeS}, permiCap \| accessTime)$(or $Enc(PK_{SdS}, permiCap \| accessTime)$, or $Enc(PK_{Cloud}, permiCap \| accessTime)$)", authorization records "$DVAddr \| F\text{-}eSAddr \| Enc(PK_{FeS}, permiCap \| grantTime)$ (or $Enc(PK_{SdS}, permiCap \| grantTime)$, or $Enc(PK_{Cloud}, permiCap \| grantTime)$)" and authorization revocation records "$DVAddr \| FeS\text{-}Addr \| Enc(PK_{FeS}, permiCap \| revokeTime)$ (or $Enc(PK_{SdS}, permiCap \| revokeTime)$, or $Enc(PK_{Cloud}, permiCap \| revokeTime)$)", they will be generated as a transaction and recorded to *Blockchain*. Therefore, it can guarantee the security requirements of the recorded information during the process of audit.

B. FEATURE ANALYSIS

In BacCPSS, Due to the whole access control is done with *Blockchain*, after initialization, *DO*, *DV*, *FeS*, *SdS* and *Cloud* have their own *DOAddr*, *DVAddr*, *FeSAddr*, *SdSAddr* and *CloudAddr*, no additional identity registration is required. They can use this global account address when accessing target data in different domains. Therefore, BacCPSS can solve the problem of difficult identity management for users.

VI. EXPERIMENT AND EVALUATION

A. EXPERIMENTAL ENVIRONMENT

In this section, we build a prototype of the BacCPSS framework. First, all experimental environments were configured based on Alibaba cloud Linux ubuntu 16.04 (2 core 8G, 100G storage), we use 4 machines to play *FeS*, *SdS*, *Cloud* and *DO* (or *DV*). Second, the Kylin test chain on EOS was used as *Blockchain* of BacCPSS, version is v1.8.4, and chainNode of Kylin test chain is "5fff1dae8dc8e2fc4d5b23b2c7665c97f9e9d8edf2b6485a86b-a311c25639191". Then, before processes of publishing, access, authorization and authorization revocation, *permiCap* needs to be encrypted. After being searched from *Blockchain*, *permiCap* needs to be decrypted. Therefore, we use a lightweight algorithm eosjs-ecc for encryption

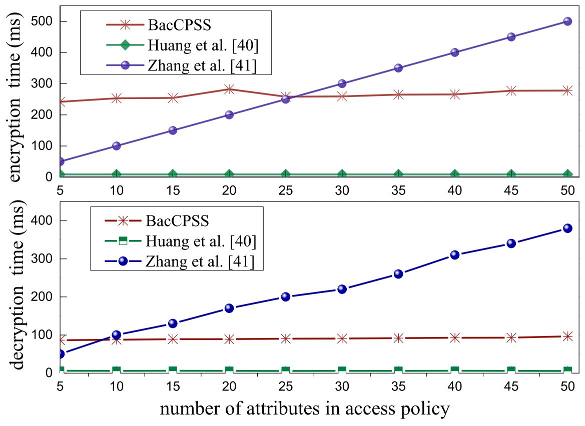


FIGURE 7. comparison of computational overhead for encryption and decryption.

and decryption. Moreover, we use the SEA algorithm to implement information interaction between various entities with privacy preserving. Finally, we implemented a prototype of the BacCPSS framework and performed performance analysis.

B. PERFORMANCE EVALUATION

In order to evaluate the scheme, we define three evaluation indicators as follows:

- **Computation overhead**. In BacCPSS, we need to sacrifice a certain amount of computing cost to take security measures for *permiCap* and off chain transaction information, mainly refers to time overhead.
- **Storage overhead**. That is, the storage space required for access control rights storage. As the number of rights (attributes) increases, so does the space for storing attributes. In BacCPSS, we need to functionalize the user's rights and publish them to the blockchain for storage through transactions, so we focus on the storage consumption of *permiCap*.
- **Throughput**. That is, the number of transactions (accesses) that can be processed per second. The throughput is always an important index to evaluate online system of network. In BacCPSS, because we store *permiCap* by publishing transactions, the network latency and throughput are the same as normal publishing transactions, and are closely related to how to select chains. Choosing different chains will lead to different results.

For BacCPSS, we focused on time performance overhead. The complete access control process includes four parts: identity authentication, access control, access permission, and audit. In BacCPSS, the access control part includes encrypting and decrypting *permiCap*, and publishing the encrypted *permiCap*. Therefore, we first tested the encryption and decryption time in BacCPSS. Since BacCPSS contains *FeS*, *SdS* and *Cloud*, we multiplied the encryption and decryption time by 3 types, as shown in Figure 7. The number of attributes used in the experiment ranged from 5 to 50, with

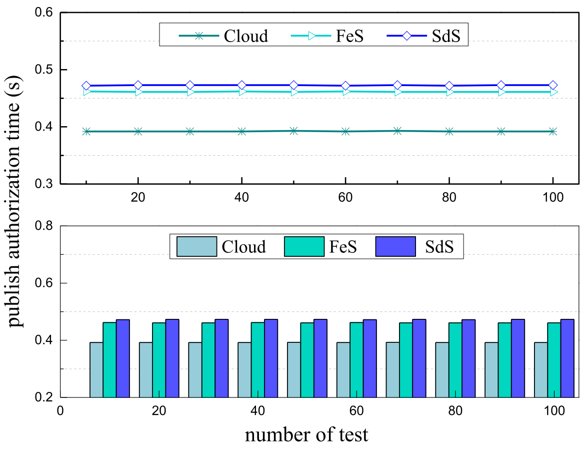


FIGURE 8. comparison of time overhead for *FeS*, *SdS*, and *Cloud* in BacCPSS.

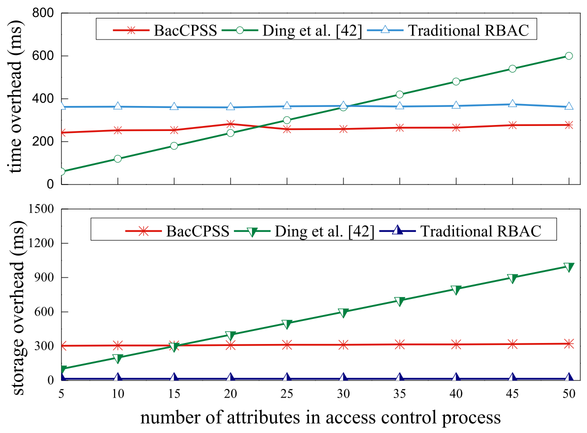


FIGURE 9. comparison of time and storage overhead.

each group averaging 10 measurements. Although there are differences in specific experimental environments, we use similar data for experiments, and we can see that BacCPSS has a low computing cost and a good operation.

Then, we tested the time of the authorization publish, including the authorization costs of *FeS*, *SdS*, and *Cloud*. As shown in Figure 8, the three are very close in terms of authorization time.

As access control divided into 4 steps, we compared time of every part between traditional scheme and our work. The result shows in Figure 9. Since traditional access control only requires storage permissions, the time and storage overhead remain the same. Experiments show that although BacCPSS uses *Blockchain*, it costs less storage than Ding *et al.* [42].

Since there are two cases for *DV* to access data of *FeS* or *SdS* in BacCPSS. Case 1, when *DV* directly accesses data of *FeS* or *SdS* or accesses data they owned in *Cloud*, it only needs one access control process (*DV* accesses to *FeS*, or *DV* accesses to *SdS*, or *DV* accesses to *Cloud*). Case 2, When *DV* applies to *Cloud* for access to data which are not owned in *FeS* or *SdS*, *DV* needs to go through two access control processes (*DV* first access *FeS* or *SdD*, and then *DV* access *Cloud*).

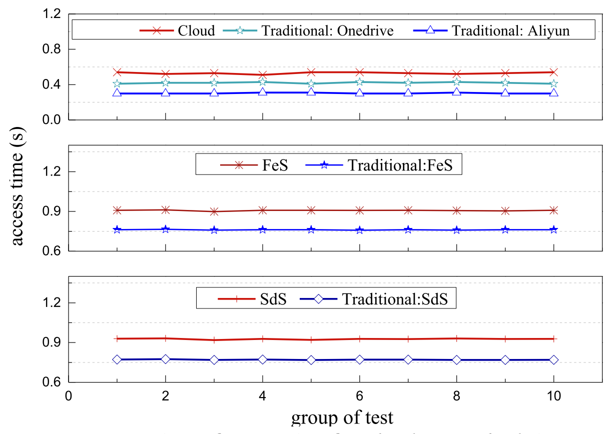


FIGURE 10. comparison of access time for *Cloud*, *FeS* and *SdS* in BacCPSS with traditional access control.

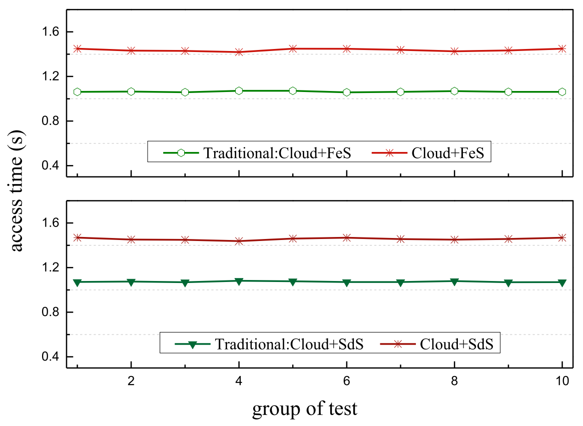


FIGURE 11. comparison of access time for *Cloud* + *FeS*, *Cloud* + *SdS* in BacCPSS with traditional access control.

Therefore, we first discuss case 1. For comparing access control of *Cloud*, *FeS*, *SdS* in BacCPSS with that in traditional access control, we tested 10 times respectively, as shown in Figure 10.

Then, we discussed case 2. For comparing *FeS* + *Cloud* and *SdS* + *Cloud* in BacCPSS with that in traditional access control, we tested 10 times respectively, as shown in Figure 11. It can be seen from Figure 10 and Figure 11 that no matter in case 1 or case 2, the time overhead of BacCPSS is slightly higher than that of traditional access control. Analyzing the reasons, we believe that there are two main aspects. One is that when accessing, BacCPSS needs to query the permission on *Blockchain*; the other is that when making access control judgments, BacCPSS need to decrypt *permiCap*, because *permiCap* are cipher texts stored in *Blockchain*.

In process of access, we need to search the latest *permiCap* from *Blockchain*, it is a great impact on the efficiency of BacCPSS. Therefore, we test throughput of *FeS*, *SdS* and *Cloud*. Because the storage overhead is at least 300 bytes but does not increase indefinitely, the size of the *permiCap* used in the experiment ranged from 300 to 1200 bytes is sufficient, with an average of 10 measurements each time, as shown in Figure 12. The results show that in (a) and (b), because *FeS* and *SdS* need to read *permiCap* from the *Cloud*,

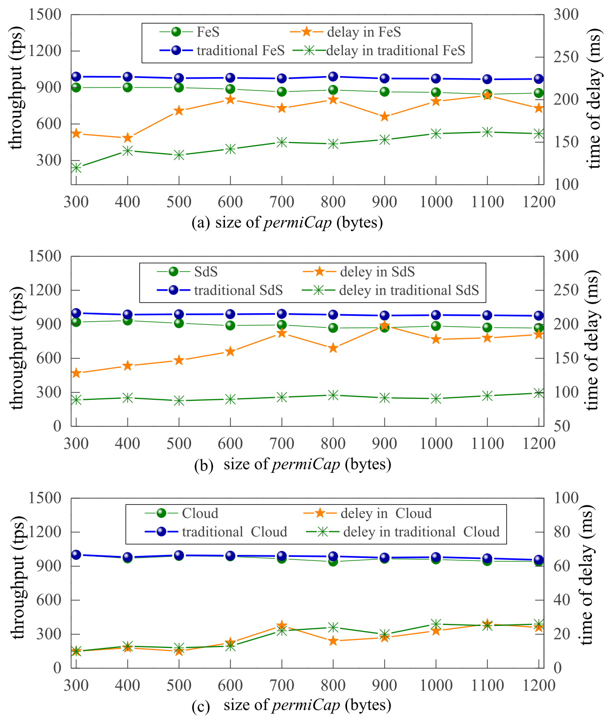


FIGURE 12. separate comparison of throughput and delay for *FeS* and traditional *FeS*, *SdS* and traditional *SdS*, *Cloud* and traditional *Cloud*.

the throughput of *FeS* and *SdS* is about 900tps, which is slightly lower than the traditional method, and because of the network, the delay will be higher than the traditional method. In (c), because *Cloud* is a full node, and it can read directly from the local. So, the throughput of *Cloud* is between 950tps and 1000tps, which is not much different from the traditional method, and the delay time is less than *FeS* and *SdS*.

In short, although *Blockchain* is used in BacCPSS, the experimental results show that the performance is relatively good. Furthermore, BacCPSS can effectively prevent attacks caused by the two threat models mentioned in section II. At the same time, no matter whether it is *FeS*, *SdS*, or *Cloud*, *permiCap* is encrypted and stored in *Blockchain*, which facilitates the management of access control.

VII. CONCLUSION AND FUTURE WORK

It is an important research direction to solve the security of access control in CPSS big data by utilizing the features of blockchain. This paper proposes a blockchain-based access control scheme called BacCPSS for privacy preserve in CPSS big data. Given the nature of CPSS big data, we redefined the rights in access and used lighter weight encryption algorithms to ensure privacy. In BacCPSS, all access control transactions are encrypted and issued by the domain management server, such as *FeS*, *SdS*, *Cloud*, and so on. Experiments have proved that this scheme is feasible and effective, and secure to implement access control for CPSS big data.

Since the whole authorization access process is recorded through the blockchain, there are two problems in this scheme. One is that it will take time to protect privacy, so we can find more lightweight and suitable methods to protect privacy for mobile terminal devices. Secondly, it is that for the operation of permission function and the chain of access records, we need to find an effective retrieval method to match encryption parameters in blockchain. In the future, we will work to address these shortcomings and focus on achieving more efficient and secure access control in CPSS big data.

REFERENCES

[1] Z. Liu, D.-S. Yang, D. Wen, W.-M. Zhang, and W. Mao, "Cyber-physical-social systems for command and control," *IEEE Intell. Syst.*, vol. 26, no. 4, pp. 92–96, Jul. 2011.

[2] F.-Y. Wang, "The emergence of intelligent enterprises: From CPS to CPSS," *IEEE Intell. Syst.*, vol. 25, no. 4, pp. 85–88, Jul. 2010.

[3] D. Hussein, S. Park, S. N. Han, and N. Crespi, "Dynamic social structure of things: A contextual approach in CPSS," *IEEE Internet Comput.*, vol. 19, no. 3, pp. 12–20, May 2015.

[4] X. Wang, L. T. Yang, Y. Wang, L. Ren, and M. J. Deen, "ADTT: A highly-efficient distributed tensor-train decomposition method for IIoT big data," *IEEE Trans. Ind. Informat.*, early access, Jan. 2020, doi: 10.1109/TII.2020.2967768.

[5] X. Zhou, W. Liang, S. Huang, and M. Fu, "Social recommendation with large-scale group decision-making for cyber-enabled online service," *IEEE Trans. Comput. Social Syst.*, vol. 6, no. 5, pp. 1073–1082, Oct. 2019, doi: 10.1109/TCSS.2019.2932288.

[6] X. Wang, L. T. Yang, Y. Wang, X. Liu, Q. Zhang, and M. J. Deen, "A distributed tensor-train decomposition method for cyber-physical-social services," *ACM Trans. Cyber-Phys. Syst.*, vol. 3, no. 4, pp. 1–15, Oct. 2019.

[7] X. Zhou, W. Liang, K. I.-K. Wang, and S. Shimizu, "Multi-modality behavioral influence analysis for personalized recommendations in health social media environment," *IEEE Trans. Comput. Social Syst.*, vol. 6, no. 5, pp. 888–897, Oct. 2019, doi: 10.1109/TCSS.2019.2918285.

[8] H. Li, K. Ota, M. Dong, and M. Guo, "Mobile crowdsensing in software defined opportunistic networks," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 140–145, Jun. 2017.

[9] X. Wang, L. T. Yang, J. Feng, X. Chen, and M. J. Deen, "A tensor-based big service framework for enhanced living environments," *IEEE Cloud Comput.*, vol. 3, no. 6, pp. 36–43, Nov. 2016.

[10] K. C. Okafor, "Dynamic reliability modeling of cyber-physical edge computing network," *Int. J. Comput. Appl.*, early access, pp. 1–11, Apr. 2019, doi: 10.1080/1206212X.2019.1600830.

[11] M. S. de Brito, S. Hoque, R. Steinke, A. Willner, and T. Magedanz, "Application of the fog computing paradigm to smart factories and cyber-physical systems," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 4, p. e3184, Apr. 2018.

[12] Y. Xu, G. Wang, J. Yang, J. Ren, Y. Zhang, and C. Zhang, "Towards secure network computing services for lightweight clients using blockchain," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–12, Nov. 2018, Art. no. 2051693.

[13] X. Zhou, W. Liang, K. Wang, R. Huang, and Q. Jin, "Academic influence aware and multidimensional network analysis for research collaboration navigation based on scholarly big data," *IEEE Trans. Emerg. Topics Comput.*, early access, Jul. 2018, doi: 10.1109/TETC.2018.2860051.

[14] B. Omoniwa, R. Hussain, M. A. Javed, S. H. Bouk, and S. A. Malik, "Fog/Edge computing-based IoT (FECIoT): Architecture, applications, and research issues," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4118–4149, Jun. 2019.

[15] W. Guo, Y. Zhang, and L. Li, "The integration of CPS, CPSS, and ITS: A focus on data," *Tsinghua Sci. Technol.*, vol. 20, no. 4, pp. 327–335, Aug. 2015.

[16] X. Zhou, J. Chen, B. Wu, and Q. Jin, "Discovery of action patterns and user correlations in task-oriented processes for goal-driven learning recommendation," *IEEE Trans. Learn. Technol.*, vol. 7, no. 3, pp. 231–245, Jul. 2014, doi: 10.1109/TLT.2013.2297701.

[17] Y. Yuan and F. Wang, "Development status and prospect of blockchain technology," *J. Automat.*, vol. 42, no. 4, pp. 481–494, 2016.

[18] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F.-Y. Wang, "Blockchain-enabled smart contracts: Architecture, applications, and future trends," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 11, pp. 2266–2277, Nov. 2019.

[19] Y. Xu, J. Ren, G. Wang, C. Zhang, J. Yang, and Y. Zhang, "A blockchain-based nonrepudiation network computing service scheme for industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3632–3641, Jun. 2019.

[20] X. Wang, L. T. Yang, L. Kuang, X. Liu, Q. Zhang, and M. J. Deen, "A tensor-based big-data-driven routing recommendation approach for heterogeneous networks," *IEEE Netw.*, vol. 33, no. 1, pp. 64–69, Jan. 2019.

[21] R. S. Sandhu and P. Samarati, "Access control: Principle and practice," *IEEE Commun. Mag.*, vol. 32, no. 9, pp. 40–48, Sep. 1994.

[22] Y. Xu, J. Ren, Y. Zhang, C. Zhang, B. Shen, and Y. Zhang, "Blockchain empowered arbitrable data auditing scheme for network storage as a service," *IEEE Trans. Services Comput.*, vol. 13, no. 2, pp. 289–300, Mar. 2019, doi: 10.1109/TSC.2019.2953033.

[23] K. L. Liu, "Formal modeling and implementation of secure linux operating system and secure Web system," Inst. Softw., Chinese Acad. Sci., Beijing, China, 2001.

[24] (2019). *Cyber-Physical Systems and Internet-of-Things Week*. [Online]. Available: http://cpslab.cs.mcgill.ca/cpsiotweek2019/

[25] J. Shi, J. Wan, and H. Yan, "A survey of cyber-physical systems," in *Proc. Int. Conf. Wireless Commun. Signal Process. (WCSP)*, 2011, pp. 1–6.

[26] J. Z. Li, H. Gao, and B. Yu, "Concepts, features, challenges, and research progresses of CPSs," China Comput. Sci., CCF, Beijing, China, Tech. Rep. CCFP 0012, 2009, pp. 1–17. [Online]. Available: https://max.book118.com/html/2017/0803/125796656.shtm

[27] D. Ding, Q.-L. Han, Y. Xiang, X. Ge, and X.-M. Zhang, "A survey on security control and attack detection for industrial cyber-physical systems," *Neurocomputing*, vol. 275, pp. 1674–1683, Jan. 2018.

[28] J. Lopez and J. E. Rubio, "Access control for cyber-physical systems interconnected to the cloud," *Comput. Netw.*, vol. 134, pp. 46–54, Apr. 2018.

[29] Y. Xu, Q. Zeng, G. Wang, C. Zhang, J. Ren, and Y. Zhang, "An efficient privacy–enhanced attribute based access control mechanism," *Concurrency Comput., Pract. Exper.*, vol. 32, no. 5, p. e5556, Mar. 2020, doi: 10.1002/cpe.5556.

[30] X. S. Dong, Z. Shen, and G. Xiong, "Research on construction of CPSS platform for urban rail transit," *Acta Automatica Sinica*, vol. 45, no. 4, pp. 48–58, 2019.

[31] G. Xiong, F. Zhu, X. Liu, X. Dong, W. Huang, S. Chen, and K. Zhao, "Cyber-physical-social system in intelligent transportation," *IEEE/CAA J. Automatica Sinica*, vol. 2, no. 3, pp. 320–333, Jul. 2015.

[32] X. Zheng, Z. Cai, J. Yu, C. Wang, and Y. Li, "Follow but no track: Privacy preserved profile publishing in cyber-physical social systems," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1868–1878, Dec. 2017.

[33] J. Zeng, L. T. Yang, M. Lin, Z. Shao, and D. Zhu, "System-level design optimization for security-critical cyber-physical-social systems," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 2, pp. 1–21, Apr. 2017.

[34] S. M. M. Rahman, "Cyber-physical-social system between a humanoid robot and a virtual human through a shared platform for adaptive agent ecology," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 1, pp. 190–203, Jan. 2018.

[35] X. Wang, L. T. Yang, X. Chen, J.-J. Han, and J. Feng, "A tensor computation and optimization model for cyber-physical-social big data," *IEEE Trans. Sustain. Comput.*, vol. 4, no. 4, pp. 326–339, Oct. 2019.

[36] X. Wang, L. T. Yang, X. Xie, J. Jin, and M. J. Deen, "A cloud-edge computing framework for cyber-physical-social services," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 80–85, Nov. 2017.

[37] P. Wang, L. T. Yang, and J. Li, "An edge cloud-assisted CPSS framework for smart city," *IEEE Cloud Comput.*, vol. 5, no. 5, pp. 37–46, Sep. 2018.

[38] T. Sharma, J. C. Bambenek, and M. Bashir. (2020). *Preserving Privacy in Cyber-Physical-Social Systems: An Anonymity and Access Control Approach*. [Online]. Available: https://www.ideals.illinois.edu/handle/2142/106049

[39] N. B. Akhuseyinoglu and J. Joshi, "A risk-aware access control framework for cyber-physical systems," in *Proc. IEEE 3rd Int. Conf. Collaboration Internet Comput. (CIC)*, Oct. 2017, pp. 349–358.

[40] Q. Huang, Y. Yang, and L. Wang, "Secure data access control with ciphertext update and computation outsourcing in fog computing for Internet of Things," *IEEE Access*, vol. 5, pp. 12941–12950, 2017.

[41] P. Zhang, Z. Chen, J. K. Liu, K. Liang, and H. Liu, "An efficient access control scheme with outsourcing capability and attribute update for fog computing," *Future Gener. Comput. Syst.*, vol. 78, pp. 753–762, Jan. 2018.

[42] S. Ding, J. Cao, C. Li, K. Fan, and H. Li, "A novel attribute-based access control scheme using blockchain for IoT," *IEEE Access*, vol. 7, pp. 38431–38441, 2019.

LIANG TAN received the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, in 2007. He is 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 researches focus on information security and the IoT access control.

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.

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. He is currently a Junior Researcher with the Global Information and Telecommunication Institute, Waseda University. 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, the Keihin Electric Railway Corporation of Japan, and the 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.

Dr. Yu has had experience with editorial and conference organizations. He has served as a Leading Guest Editor of *Sensors* special issue Emerging Blockchain Technology Solutions for Real-world Applications (EBTSRA), the General Co-Chair and the Publicity Co-Chair of the IEEE VTC2020-Spring EBTSRA Workshop, the TPC Co-Chair of SCML2020, the Local Chair of MONAMI 2020, and the Session Chair of ITU Kaleidoscope 2016. Moreover, he has served as a TPC Member of the ITU Kaleidoscope 2020, the IEEE VTC2020-Spring, the IEEE CCNC 2020, the IEEE WCNC 2020, the IEEE VTC2019-Spring, the ITU Kaleidoscope 2019, the IEEE HotICN 2019, the IEEE ICCC 2019, the IEEE WPMC 2019, EEI 2019, and ICITVE 2019. He is also an Editor of the IEEE OPEN JOURNAL OF VEHICULAR TECHNOLOGY (OJVT).