BC-BLPM: A Multi-Level Security Access Control Model Based on Blockchain Technology

Xiang Yu[1, 2], Zhangxiang Shu[2], Qiang Li[2*], Jun Huang[2]

[1] College of Computer Science, National University of Defense Technology, Changsha 410005, China
[2] College of Electronic Engineering, National University of Defense Technology, Hefei 230037, China
[*] The corresponding author, email: lychfeei@163.com

Abstract: Traditional multi-level security (MLS) systems have the defect of centralizing authorized facilities, which is difficult to meet the security requirements of modern distributed peer-to-peer network architecture. Blockchain is widely used in the field of access control with its decentralization, traceability and non-defective modification. Combining the blockchain technology and the Bell-LaPadula model, we propose a new access control model, named BC-BLPM, for MLS environment. The "multi-chain" blockchain architecture is used for dividing resources into isolated access domains, providing a fine-grained data protection mechanism. The access control policies are implemented by smart contracts deployed in each access domain, so that the side chains of different access domains storage access records from outside and maintain the integrity of the records. Finally, we implement the BC-BLPM prototype system using the Hyperledger Fabric. The experimental and analytical results show that the model can adapt well to the needs of multi-level security environment, and it has the feasibility of application in actual scenarios.

Keywords: multi-level security (MLS); access control; blockchain; multi-chain; smart contract

I. INTRODUCTION

With the development of computer network technology, digital resources (e.g., electronic documents, videos, databases, or even services) have realized network-based interconnection on the whole. However, with the continuous expansion of network resources, a large number of high-value data assets stored in the network are likely to be the target of external or internal attackers to infiltrate and control. Protecting these resources from various security threats is a very important issue that has received widespread attention. Access control technology is one of the important technologies for network security. It is usually used to regulate access behavior to key resources or valuable resources of the network, such as data, services, computing systems, storage space, etc. It restricts access to users in the network by executing access rules, and prohibits unauthorized users from illegally accessing the network resources, thereby more effectively protecting the network infrastructure and data stored in the network.

The Multiple Level Security (MLS) [1], as a security policy based on mandatory access control, describes the ability of a computer or network system to handle different security level information and prevent users or resources from obtaining unauthorized information. It has been widely applied in government, military, banking, enterprise and other departments. In MLS, information is stored in objects, and each object is assigned a security level label. In addition, the subject usually represents the user or the currently active session (such as a process in the OS). Similarly, each subject is assigned A license that represents a security level label as well. Traditional MLS models in-clude Bell-LaPadula [2], Biba [3], Clark-Wilson [4], and C-

Received: Oct. 24, 2019
Revised: Apr. 28, 2020
Editor: Rongpeng Li

hinese Wall [5]. Among them, the Bell-LaPadula model provides confidentiality protection, the Biba model and the Clark-Wilson model mainly provide integrity protection, and the Chinese Wall model provides protection for the system environment based on conflicts of interest. At present, most multi-level security access control systems are based on centralized architecture, such as MUSHI [6], 3-D Correlation [7] and others. Centralized access control mechanism relies on trusted third-party provisioning permission management services. However, once the decision nodes in the system are maliciously attacked or controlled, it is easy to cause single-point failures, which makes the whole system face threats such as illegal, over-the-level access and malicious tampering of access logs, thus failing to provide credible and reliable access control service for honest visiting subjects. At the same time, the efficiency of centralized access control services will inevitably be affected by factors such as hardware infrastructure and network transmission bandwidth, there are problems of high construction and maintenance costs and low transmission efficiency, which cannot meet the needs of distributed network environment applications.

Blockchain technology can effectively solve the problem of insufficient centralized server for multi-level security system. The basic thought of blockchain technology is to use a distributed peer-to-peer network to ensure secure, transparent information transfer and storage of transactions. Information can be shared among multiple peer nodes without the intervention of the central manager, and data cannot be changed once it is linked into the public ledger of the blockchain. Through the consensus protocol, each transaction in the blockchain can have its own proof of validity, thus ensuring the Decentralization.

The blockchain features non-tamperable, decentralized, and time-series data, enabling multi-level secure access control based on blockchain technology to avoid single-point failures and provide non-repudiation access logs, thus effectively securing the information system. With the promotion of smart contract [8] in the field of blockchain technology, it is possible to realize a more lightweight multi-level security policy through the coding and instantiation of smart contracts, and improve the flexibility of multi-level security access control. Therefore, the combination of blockchain technology and multi-level security access control has a promising application prospect.

To construct a distributed multi-level security access control mechanism, this work proposes a new multi-level security access control model BC-BLPM (Blockchain-based Bell-LaPadula Model) based on blockchain technology and Bell-LaPadula(BLP) model. The model introduces the concept of "multi-blockchain", which divides the operational data of different departments or organizations in the network environment through multiple sidechains, so that the data of different business organizations can be logically isolated; by using the blockchain as a third-party management intermediary, the single-point failures and untrustworthy problems that may be caused by traditional centralized authorization settings are eliminated; provides access control services for user equipment and network resources by writing BLP-based security policies in smart contracts and deploying them in blockchain network. After the execution of the smart contract is completed, the execution process will be written in the form of a transaction and will be irreversibly linked into the blockchain.

Compared to traditional MLS systems (such as MUSHI [6]), BC-BLPM has the following advantages:

1) Confidentiality: The BLP-based access control policy primitives ensure the legitimacy of access authorization. At the same time, BC-BLPM enhances the security of the pol-icy itself through the form of smart contract deployment.

2) Integrity: The data integrity of the access process is maintained by the entire network nodes or the designated restricted nodes. Once the data record is linked into the blockchain, it cannot be revoked or changed.

3) Availability: Through the distributed verification mechanism of the blockchain and the mandatory and automated execution rules of the smart contract, various malicious attacks such as replay attacks, identity forgery attacks and denial of service attacks can be effectively prevented.

4) Decentralization: The data from different organizations are maintained by different sidechains, and the operations between them are isolated from each other. The failure of a single service node in the network does not affect the user's access, avoiding the problem of single-point failures. This advantage is not available in traditional MLS systems.

5) Auditability: All access behaviors from different users are recorded in the public ledger of the blockchain. Since the data storage of the blockchain has the characteristics of openness and transparency, the data in the public ledger is visible to all subjects, which facilitates the auditing of the legitimacy of user's access within a period of time by authoritative institutions.

6) Scalability: The newly participating node can be added to the blockchain network by downloading the ledger copy of the original node.

7) Lightweight: User equipment and network resources can use BC-BLPM without any additional components. Meanwhile, the resource provider does not need to remain online in real time during the process of providing access control services.

The rest of the paper is structured as follows: Section II introduces the related work. Section III and Section IV describes the architecture and the implementation respectively. In Section V, we evaluate the performance and the security of the model through experimental and analytical methods. And finally, we have the conclusion of this paper in Section VI.

II. RELATED WORK

2.1 Bell-laPadula Model

The Bell-LaPadula (BLP) model[2, 9] is a state machine model for the MLS environment proposed by D. Elliott Bell and J. Leonard LaPadula. It is the earliest and most popular MLS model. Since the introduction of this model, many research works have been carried out in various ways driven by actual demand, and many important research results have been obtained. And we here select the research results in recent years for discussion. For example, Zeng et al. [10] proposed a dynamic sensitive data flow security model for federated cloud systems that assigns cloud system entities different security levels for a given security grid based on BLP security rules and cloud security rules, and formalizes them. The method assigns a security policy to the entity movement process between different clouds. Tan et al. [11] proposed a centralized pervasive computing environment/multi-level security (CPCE/MLS) system was designed. In Ref. [11], By introducing a server storage terminal, a multi-level security access control mechanism based on BLP model, process creation monitoring mechanism and auditing are implemented that provide security for pervasive computing environments. In order to solve the problem of cooperation between IoT terminals and virtual machines, Dong et al. [12] designed a task-oriented IoT multi-level collaborative access control model VR-BLP. The model divides tasks into multiples through improved BLP state transition rules. The level limits the access between the virtual machine and the IoT terminal, and implements network isolation, process isolation, and shared memory isolation. In the context of medical big data security, Freitas et al. [13] proposed a BLP-based access control model for federal cloud workflow deployment, which divides medical services into multiple security states and performs specific security. Policies implement state transitions between different services.

In summary, the Bell-LaPadula model, as the most classic multi-level security access control model, has been widely applied in many different research fields such as computer systems and network security. From the perspective of confidentiality, the model controls the flow direction of data through the security policy of "no read-up" and "no write-down" to prevent data flowing from high-confidentiality subject to low-confidentiality subject. Therefore, the MLS system based on the BLP security policy can effectively prevent illegal entities from stealing data, thereby maintaining the security state of confidential data, which is also the key to achieving multi-level security. Therefore, we choose the Bell-LaPadula model as the basic framework for the multi-level secure access control model. However, most BLP-based MLS systems still have authorization centralization, lack of auditing tools and data integ-rity protection mechanisms, etc., and need to improve the traditional Bell-LaPadula model through external channels, which is reflected in many existing research [14–16]. Compared with these improved methods, the biggest difference of this work lies in the adoption of blockchain technology, and the distributed execution of access control policy together with the immutable record of access behavior in the blockchain network, so as to overcome the inherent defects of Bell-LaPadula model.

2.2 Blockchain and Access Control

In recent years, the application and research of blockchain technology in cryptocurrency [17], identity management [18], reputation system [19], Internet of Things [20], cloud storage [21] and other fields have been deepening. A blockchain is defined as a decentralized, cryptographic-based arbitration network that does not require the participation of trusted third parties. In this network, nodes that are not trusted to each other can perform "end-to-end" interaction in an encrypted, verifiable form without a trusted intermediary. This interaction process is also called "Transaction" in the blockchain system. The transactions generated by the entire network in unit time are packaged into a "block" as a data record, the integrity of the data is protected by a hash function, and finally linked to the end of the chain-based ledger maintained by the entire network nodes. The verification of a transaction correctness relies on the distributed consensus protocol [22].

Traditional access control technologies such as role-based access control (RBAC) [23], attribute-based access control (ABAC) [24], usage control (UCON) [25], etc., and both of which need to introduce a trusted centralized authorization entity to provide access control services for users. The decentralization, non-tamperable, and time-series data of the blockchain determine its ability to play an independent trusted third-party role and provide access control services using its own data logging and consensus functions. Meanwhile, blockchains can also be used to transform existing access control models, decentralizing centralized authorization centers. In recent years, many research efforts have focused on the combination of blockchain technology and access control.

Some researchers have proposed storing and executing access control policies through blockchain transactions. Zyskind et al. [26] used blockchain to protect the user's private data in a mobile service scenario involving untrusted third parties. Among the article, there are two types of transactions accepted by blockchain: one for access control management transaction T_{access} and the other for data storage and retrieval transaction T_{data}. Through the creation of these two kinds of transactions, the access control policy is stored on the blockchain and the response to the legitimate access request. However, implementing such access control requires the resource requester and the owner to be online at the same time, which is not desirable in practical application scenarios. Ouaddah et al. [27] focused on the access control problems existing in IoT devices and proposed the FairAccess model. As a distributed access control model implemented by blockchain, this model stores the access control policy in the blockchain with the form of a 2-tuple (resource, request) through the blockchain transaction, and can also be applied as a log database to provide auditing capabilities. The disadvantages of FairAccess are the same as in Ref. [26], which requires both resource requesters and resource operators to stay online at the same time. Similar to the first two schemes, Ma et al. [28] proposed to store key data on the blockchain and manage it by a group of trustees who are responsible for controlling access to such data.

Other researchers have adopted smart contracts as a storage vehicle for access control policies. Among them, Cruz et al. [29] proposed a blockchain-based RBAC model, which used Ethereum platform and smart contracts as the infrastructure to achieve cross-organization access control, and also proposed the concept of cross-organization role availability. Wang et al. [30] designed a data access control model for cloud storage architecture, providing data access and sharing services by calling smart contract interfaces. Azaria et al. [31] focused on the data privacy protection and access permission management of medical systems, proposed a model called MedRec suitable for electronic medical records. The model is implemented by Ethereum platform. By introducing the hierarchical relationship of smart contracts, the smart contract calling path of the tree structure is designed, enabling the patient or doctor to view/update/revoke the electronic medical record only by calling the top-level contract. But on the other hand, the hierarchical relationship of the multi-contracts also increases the storage overhead of the network nodes. Different from the Refs. [29–31], Maesa and Mori [32] combined the attribute-based access control model and the blockchain technology to design a universal access control system for managing the calling permissions of smart contracts. Through the external ABAC infrastructure (such as PAP, PDP, PEP, etc.), the system embeds the policy into the smart contract through the XAMCL language, and implements a lightweight access control mechanism for the calling relationship of smart contract.

Until now, the combination of blockchain technology and access control technology is mainly concentrated in three application areas, namely, Internet of

Things, cloud storage and medical systems. Most of these fields have open scenarios, that is, the scene where operating devices are not restricted. These scenarios are mainly concerned with the protection of privacy, and the flexibility of data sharing. The issues of confidentiality and availability in multi-level security are usually placed in the third place or even lower. Therefore, until now, neither the access control mechanism based on blockchain implementation, nor the existing access control model based on the blockchain improvement and the access control method with additional encryption means, have fully realized the requirement of multi-level security access control mechanism. In comparison, the access control model proposed in this work focuses on the MLS environment with higher requirements for data confidentiality and user operation legitimacy, through the "multi-chain" architecture and access control policy based on smart contracts, without infringing on MLS system under the premise of security properties, effectively limits the visitors access to network resources, so as to realize the decentralized multi-level security access control mechanism. According to the previous research, the access control model proposed in this work is the first one to consider the application of blockchain technology to multi-level security access control mechanism.

III. ARCHITECTURE

3.1 Overview

BC-BLPM is a decentralized multi-level security access control model that combines blockchain technology and BLP security policy primitives. It consists of the resource layer, access control layer, and transaction-forwarding layer as is shown in Figure 1. In the resource layer, different types of data resources, e.g., databases, electronic files, pictures and so on, stored in a medium such as a computer or a smart mobile device are accessed into the model via an encrypted data transmission channel. In the access control layer, A number of data processing nodes are responsible for forming blockchain-based access domains, which can be used as an abstraction of different business departments in a large enterprise or business organization, maintain different data resources respectively(i.e., maintain the data resources stored carrier), and record access behavior requested from the outside by updating the distributed Ledger of the sidechains. In the transaction-forwarding layer, several high-performance data processing nodes establish communication relationships with gateway nodes of each access domain, and are responsible for packing, sorting, and forwarding blockchain transactions generated in the access control layer.

In BC-BLPM, the access control layer is composed of multiple access domains, which realizes the isolation of business logic. That is, updating the data of the sidechain in one domain does not affect the others, which ensures the operational isolation of inter-domain data. This is the key to achieving a multi-level security environment. For example, for a large enterprise, the isolation of operational data from various business units can minimize the risk of core data flows from high security units to low ones. At the same time, each access domain deploys a smart contract that includes access control policies. Through the execution of the smart contract, the calling process of the access control policy is completed. And the model creatively uses the combined public key algorithm system (CPK) as the verification component of the identity validity for the external access subject. The subject can verify the validity of its own identity information based on the cryptosystem. The model can be expressed in the form of a 6-tuple (AD, PL, ET, R, D, MR), where:

1)$AD = \{ad_1, ad_2, \ldots, ad_n\}$, represents all access domain collections in the network that are added to the BC-BLPM access control layer.

2)$PL = \{pl_1, pl_2, \ldots, pl_n\}$, represents all the set of access control policies that are deployed in the network.

3)$ET = \{et_1, et_2, \ldots, et_n\}$, represents the set of entities in the network, which can be divided into user entities and resource entities. Each type of entity can map clearance level or classification through function F. By combining public key algorithms to sign identity data, signature data $Sig_{CPK}\{et_i\}(1 \leqslant i \leqslant n)$ based on entity identity can be obtained.

4)$R = \{r_1, r_2, \ldots, r_n\}$, represents the set of access requests that all access-request entities are generating. Similarly, the different decision results of these request counterparts can be represented by the set $D = \{d_1, d_2, \ldots, d_n\}$ as well.

5)$MR : F(ET) \times R \times PL \rightarrow D$, represents the mapping relationship between the access request and the decision result. The establishment of this relationship also requires the participation of the entity security level set $F(ET)$ and the access control policy set PL.

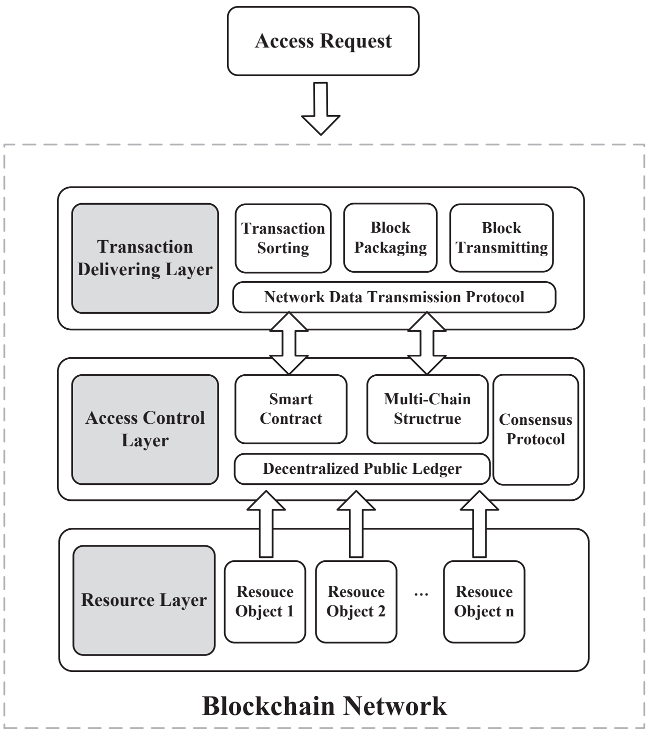


Figure 1. *BC-BLPM framework.*

3.2 Access Control Policy

We reuse the security theorem of the traditional Bell-LaPadula model and establish the access control policy for multi-level security environments. Below, we describe the policy from three aspects: basic elements, security attributes, and state transition rules.

3.2.1 Elements

We first define the elements of the Policy: 1) Subject and Object. One user is a subject, a specific data resource that exists in the network represents an object. Users can obtain permission to access a certain type of data resource by submitting an access request.

2) Access attribute set. The access attribute set is A contains two sub-elements, which is $A = A_o \cup A_\varphi$. Regarding A_o as the access attribute set in the traditional Bell-LaPadula model, where $A_o = \{r, a, w\}$ is composed of 3 kinds of sub-attributes, r stands for read-only, a stands for Write-only, and w stands for read-write; A_φ represents the resource-shared attribute set between two different objects, where $A_\varphi = \{sd\}$, and sd denotes a data-sending operation.

3) Security function set. F is a set of security functions, and $F = \{f_s(s), f_c(s), f(o)\}$ contains 3 sub-elements, where $f_s(s)$ represents the highest clearance level of the subject s, $f_c(s)$ represents the current clearance level of the subject s, and $f(o)$ represents the security level of the object o.

4) Current access set. $B = S \times O \times A$ indicates the current access set. The arbitrary element of B is b, and $b = (s_i, o_j, x)$ indicates that one subject s_i performs an x access behavior on another object o_j.

5) Access matrix. The access matrix M contains a sequence of access attributes that record the subject's access behavior to the object, i.e. $M_{ij} \subseteq A$.

6) Current system state. $v = (b, M, f) \in V$ represents the state value of the system, where $b \in B$, $f =< f_s, f_o >\in F$.

7) State transition rule. The rule function $\rho : R \times V \to D \times V^*$, where $R = RA \times S \times O \times X$ represents the request set, $X = A \cup F$; $D = \{yes, no, error, ?\}$ represents the system's response set ; V and V^* represent the system state before and after the state transition, respectively.

3.2.2 Security Properties

The security properties of the policy are implemented based on the original security theorem of the Bell-LaPadula model, and are extended according to the actual elements of the policy and the access requirements of the subject. We give the following theorems:

Theorem 1. *Simple-Security Property*

State $v = (b, M, f) \in V$, $x \in A_o$ satisfies the simple-security property if and only if $\forall(s, o, x) \in b$, there are: ①$x = a$; ②$x = r$ or $x = w \Rightarrow f_s(s) \rhd f(o)$.

Here, $\rhd$ means "dominate", refers to the partial order between security levels or clearance levels. It means that, for $\forall(s, o, x) \in b$, when the subject's tries to access the object o with the access attribute r or w, the highest clearance level of the subject's must be higher than or equal to the security level of the object o, then the state $v = (b, M, f)$ satisfies simple-security property.

Theorem 2. **-Property (Star-Property)*

State $v = (b, M, f) \in V$, $x \in A_o$ satisfies *-Property if and only $\forall(s, o, x) \in b$, $s \in S/S^*$ where

S^* is a trusted subject, there are: ①$x = r \Rightarrow f_c(s) \rhd f(o)$; ②$x = a \Rightarrow f_c(s) \lhd f(o)$; ③$x = w \Rightarrow f_c(s) = f(o)$.

It indicates that, for $\forall(s,o,x) \in b$, $s \in S/S^*$, when the subject's tries to access the object o with the access attribute r, the state $v = (b, M, f)$ can satisfies the *-Property only if the current clearance level of the subject s is higher than or equal to the security level of the object o; when the subject s tries to access the object o with the access attribute a, $v = (b, M, f)$ can satisfies the *-Property only if the current clearance level of the subject s is less than or equal to the security level of the object o; similarly, when the subject s tries to access the object o with the access attribute w, the state $v = (b, M, f)$ can satisfies the *-Property only if the current clearance level of the subject s is equal to the security level of the object o.

Theorem 3. *Data-Flow Property*

State $v = (b, M, f) \in V$, $x \in A_\varphi$ satisfies the Data-flow property if and only if $\forall(s_1, o_1, o_2, x) \in b$, $s_1 \in S/S^*$ where S^* is a trusted subject, there are: $\neg(f_c(s_1) \lhd f(o_1))$ and $x = s \Rightarrow f(o_2) \rhd f(o_1)$.

It specifies the direction of data transmission between objects. When the data transfer action between two objects is involved, only meet the current clearance level of the operation subject s_1 is not dominated by the security level from the data-sending object o_1, and the security level of the data-receiving object o_2 dominates the security level of o_1, the state $v = (b, M, f)$ satisfies the data-flow property.

Theorem 4. *Discretionary-Security Property*

For the state $v = \{v_1 \cup v_2 \cup \ldots \cup v_n\}$ and $v_i = (b, M, f) \in V$, $x \in A_o$ satisfies the discretionary-security property if and only if $\forall(s, o, x) \in b$, there is: $x \in M_{ij}$.

For $v_j = (b, M, f) \in V$, $x \in A_\varphi$ satisfies the autonomous security property if and only if $\forall(s_i, o_j, o_k, x) \in b$, there is: $\exists o_j, o_k \in O$, $M_{ij} \subseteq M_{ik}$.

Theorem 5. *Fundamental Security Property*

The system can remain secure if and only if the initial state of the system is secure, and each state transition process satisfies simple-security property, *- property, data-flow property, and discretionary-security property.

3.2.3 State Transition Rules

The execution of the policy will cause the transition of the system state. The state transition function is: $\rho : R \times V \rightarrow D \times V^*$, where R is a set of requests and D is a set of decisions. Take R_k be arbitrary element of R, D_m be an arbitrary element of D, then the output of ρ remains secure if and only if $\forall(R_k, v) \in (D_m, v^*)$ and the Initial state v is secure, such that v^* is secure. According to this derivation principle, we present 4 types of state transition rules for the access control policy.

Rule 1. *Read-only*

Given the initial state $v = (b, M, f) \in V$, $s \in S/S^*$, the process of state transition request $R_k = (s_i, o_j, r)$ is:

$$\rho_r(R_k, v) = \begin{cases} (?, v), iff R_k \notin Dom(\rho_r) \\ (yes, (b^* = b \cup (s_i, o_j, r), M, f^* = \\ \qquad f \cup max\{f_c(s_i), f(o_j)\})) \\ iff(R_k \in Dom(\rho_r)) \\ \&(f_s(s_i) \rhd f(o_j)) \\ \&(r \in M_{ij}) \\ (no, v), else \end{cases} \quad . \tag{1}$$

When the subject s_i requests to access the object o_j with the access attribute r, the system first checks whether the request is legal, that is, whether it is in the scope $Dom(\rho_r)$ of the rule, and if not, the "unknown" state $(?, v)$ is returned. If $R_k \in Dom(\rho_r)$ is true, then the dominance relationship between the highest clearance level $f_s(s_i)$ and the security level $f(o_j)$ is judged. If $f_s(s_i)$ can dominate $f(o_j)$, then the system continues to check if the access attribute r satisfies the requirements of $r \in M_{ij}$. Only if these three requirements are met, the system can execute the access request and return to the "access permitted" state $(yes, v^* = (b^*, M, f^*))$, otherwise it returns the "access denied" state (no, v).

Rule 2. *Write-only*

Given the initial state $v = (b, M, f) \in V$, $s \in S/S^*$, the process of state transition request $R_k = (s_i, o_j, a)$ is:

$$\rho_a(R_k,v)=\begin{cases}(?,v), iff R_k \notin Dom(\rho_a(R_k,v))\\(yes,(b^*=b\cup(s_i,o_j,a),M,f))\\iff(R_k\in Dom(\rho_a(R_k,v)))\\\&(f_s(s_i)\rhd f(o_j)\wedge f_c(s_i)\lhd f(o_j))\\\&(a\in M_{ij})\\(no,v), else\end{cases}. \tag{2}$$

When the subject s_i requests to access the object o_j with the access attribute a, the system first checks whether the request is in the scope $Dom(\rho_a)$, and if not, the "unknown" state $(?,v)$ is returned. If $R_k \in Dom(\rho_a)$ is true, it continues to judge the clearance level of s_i and the security level of o_j. If both of it meet the requirement of "writing up", then the system checks whether the request attribute a satisfies the requirement of $a \in M_{ij}$. Only if these three requirements are met can the access request be executed and the "access permitted" state $(yes, v^* = (b^*, M, f))$ be returned, otherwise the "access denied" state (no, v) is returned.

Rule 3. *Read-write*

Given the initial state $v = (b, M, f) \in V$, $s \in S/S^*$, the process of state transition request $R_k = (s_i, o_j, w)$ is:

$$\rho_w(R_k,v)=\begin{cases}(?,v), iff R_k \notin Dom(\rho_w)\\(yes,(b^*=b\cup(s_i,o_j,a),\\M,f^*=f\cup f_c(s_i)))\\iff(R_k\in Dom(\rho_w))\\\&(f_s(s_i)\rhd f(o_j)\wedge f_c(s_i)\lhd f(o_j))\\\&(w\in M_{ij})\\(no,v), else\end{cases}. \tag{3}$$

When the subject s_i requests to access the object o_j with the access attribute w, the system first checks whether the request is in the scope $Dom(\rho_w)$, and if not, returns the "unknown" state $(?,v)$. If $R_k \in Dom(\rho_w)$ is true, then the dominance relationship between the highest clearance level $f_s(s_i)$, the current clearance level $f_c(s_i)$, and the security level $f(o_j)$ is judged, if $f_s(s_i)$ is for the dominant relationship of $f(o_i)$ while satisfying $f(o_i)$ dominates $f_c(s_i)$, it continues to check whether the request attribute w satisfies the requirement of $w \in M_{ij}$. Only if these three requirements are met can the access request be executed and the "access permitted" state $(yes, v^* = (b^*, M, f^*))$ be returned, otherwise the "access denied" status (no, v) is returned.

Rule 4. *Data-Transfer*

Given the initial state $v = (b, M, f) \in V$, $s \in S/S^*$, the process of state transition request $R_k = (s_i, o_j, o_k, sd)$ is:

$$\rho_s d(R_k,v)=\begin{cases}(?,v), iff R_k \notin Dom(\rho_s d)\\(yes,(b^*=b\cup(s_i,o_j,o_k,sd),\\M,f^*=f\cup f(o_k)))\\iff(R_k\in Dom(\rho_s))\\\&\begin{bmatrix}(f_s(s_i)\rhd f(o_j)\wedge\\\neg f_c(s_i)\lhd f(o_j))\wedge\\f(o_j)\lhd f(o_k)\end{bmatrix}\\\&(sd\in(M_{ij},M_{ik}))\\(no,v), else\end{cases}. \tag{4}$$

When the subject s_i requests to access the object o_j and o_k with the access attribute sd, the system first checks if the request is in the scope $Dom(\rho_s d)$, and if not, returns the "unknown" state $(?,v)$. If $R_k \in Dom(\rho_s d)$ is true, then the dominance relationship between $f_s(s_i)$ and $f(o_j)$, $f_c(s_i)$ and $f(o_j)$, $f(o_i)$ and $f(o_k)$, and Whether the request attribute sd exists in the access control matrix M of two objects are judged respectively. Only when $f_s(s_i)$ dominates $f(o_j)$, $f_c(s_i)$ is not dominated by $f(o_j)$ and $f(o_k)$ dominates $f(o_j)$, and the request attribute sd satisfies $sd \in (M_{ij}, M_{ik})$, the request can be executed to return the "access permitted" state $(yes, v^* = (b^*, M, f^*))$, otherwise it returns the "access denied" state (no, v).

3.2.4 Combined Public Key Cryptosystem

The concept of combined public key cryptosystem (referred to as CPK system) was first proposed in 1999 [33]. The CPK system is based on identification and does not require third-party certification. It can manage large-scale keys with fewer parameters. Its security is based on the intractability of discrete logarithms. One of the basic ideas of CPK is to use a small matrix to combine a large number of public and private key pairs to solve large-scale key management problems.

The CPK key management system can be constructed based on either a general finite-field discrete logarithm problem or an elliptic curve discrete logarithm problem. Under the same security requirements, the elliptic curve discrete logarithm problem requires a smaller key size, which has the advantages of fast running speed and small storage space. Therefore, this paper builds on the CPK system based on the discrete logarithm problem of elliptic curves over a finite field F_p (p is a prime number not equal to 2 and 3).

Given the elliptic curve cryptographic parameters $T = (a, b, G, n, p)$, a random seed public key sequence matrix PSK and seed private key sequence matrix SSK can be constructed:

$$PSK = \begin{bmatrix} R_{1,1} & R_{1,2} & \dots & R_{1,h} \\ R_{2,1} & R_{2,2} & \dots & R_{2,h} \\ \dots & \dots & \dots & \dots \\ R_{m,1} & R_{m,2} & \dots & R_{m,h} \end{bmatrix}$$

$$= \begin{bmatrix} (x_{1,1}, y_{1,1}) & (x_{1,2}, y_{1,2}) & \dots & (x_{1,h}, y_{1,h}) \\ (x_{2,1}, y_{2,1}) & (x_{2,2}, y_{2,2}) & \dots & (x_{2,h}, y_{2,h}) \\ \dots & \dots & \dots & \dots \\ (x_{m,1}, y_{m,1}) & (x_{m,2}, y_{m,2}) & \dots & (x_{m,h}, y_{m,h}) \end{bmatrix}$$

$$SSK = \begin{bmatrix} r_{1,1} & r_{1,2} & \dots & r_{1,h} \\ r_{2,1} & r_{2,2} & \dots & r_{2,h} \\ \dots & \dots & \dots & \dots \\ r_{m,1} & r_{m,2} & \dots & r_{m,h} \end{bmatrix}$$

Where $r_{(i,j)}$ is $R_{(i,j)} = (x_{(i,j)}, y_{(i,j)})$ for the discrete logarithm of the base point G, i.e. $R_{(i,j)} = (x_{(i,j)}, y_{(i,j)}) = [r_{(i,j)}] \cdot G$. Obviously, the elements $R_{(i,j)} = (x_{(i,j)}, y_{(i,j)})$ and $r_{(i,j)}$ at any corresponding position in the two matrices of PSK and SSK form a public and private key Yes, the combination of multiple points in PSK and discrete logarithms in SSK can also form a public-private key pair. Before the access control policy is implemented, the CPK system can be used to protect the identity validity of the subject $s \in S/S^*$. The subject first uses the random CPK seed private key sequence $SSK = \{r_1, r_2, \dots, r_n\}$ to generate its own signature ID private key, the generation steps are as follows:

(a) Calculate the hash value of ID using Hash algorithm to get $DigestID = hash(ID)$.

(b) If $MAP_0 = DigestID$, then calculate again $MAP_{i+1} = E_K(MAP_i)(i \geq 0)$, where $E_K(\cdot)$ is a block cipher operation with K as the encryption key. Let i not exceed the number of SSK (marked as len), get the sequence value $Seq_{IDSK} = (MAP_0 \parallel MAP_1 \parallel .. \parallel MAP_{n-1})$, $len \times k$ before interception bits (k is the number of unit bits) are used as the selection sequence $\{a_v(1 \leq v \leq n)\}$.

(c) We can use the selection sequence $\{a_v(1 \leq v \leq n)\}$ to combine SSK to obtain the identification private key $IDSK = \sum_{v=1}^{n} a_v \cdot r_v$.

The subject can use the generated signature private key to resign the blockchain transaction information to obtain the signature value CPK Sig. Then the verification of signature also be implemented using the CPK sequence $PSK = \{R_1, R_2, \dots, R_n\} = \{(x_1, y_1), (x_2, y_2), \dots, (x_3, y_3)\}$. Assuming that the ID of the subject has been maliciously tampered after signature, based on the anti-collision property of the Hash algorithm, the verifier cannot use the ID' (that is, the subject ID that has been tampered with) to restore the original identification verification public key $IDPK = \sum_{v=1}^{n} a_v \cdot R_v$, so the validity of the signature cannot be verified.

3.3 Blockchain Component

The blockchain can be divided into two structures according to the type of participants: permissioned blockchain (including private/consortium blockchain) and permission-less blockchain. In a permission-less blockchain such as the Bitcoin system, each participating node can become a "miner", each can generate transactions and maintain the blockchain by verifying the validity of the transaction. A transaction can be recorded in the blockchain only if the transaction is verified and committed by most of the "miners" in the network (in principle, more than 50%). In the permissioned blockchain, only authorized participants can become "miners". Because of the authorization mechanism added in the permissioned blockchain, the unauthorized nodes cannot join to the network freely, which limits the scale and attributes of the application scenario, and is more suitable for some application scenarios that have high demand for system performance load and security confidentiality, such as government, banking, medical institutions, etc. As a multi-level secure access control model based on blockchain, BC-BLPM adopts the permissioned blockchain as the underlying architecture. This section describes the blockchain components of BC-BLPM, including transactions, blocks and smart contracts.

3.3.1 Transaction and Block Format

BC-BLPM uses two types of transactions, namely:

①Access Record Transaction(ART); ②Data Transfer Transaction(DTT).

The composition of the two transactions is shown in Figure 2 and Figure 3. The transaction headers of the two transactions contain common fields for blockchain trans-actions, such as transaction hash value TX Hash, transaction public key address TX Addr, subject's private key signature TX Sig and subject-based CPK signature CPK Sig, and the transaction body contains payloads that characterize the transaction structure. As a write set of transactions, the payload fields of the transactions need to be populated by the transaction creators.

Transaction Header			
TX Hash	TX Addr	TX Sig	CPK Sig
Transaction Payload			
Sbj_ID	Sbj_Level	Obj_ID	Obj_Level
AccReq_Attr		Judge_Result	

Figure 2. *Format of ART transaction.*

Transaction Header			
TX Hash	TX Addr	TX Sig	CPK Sig
Transaction Payload			
Sbj_ID	Sbj_Level	Data_Transfer	
SObj_ID	SObj_Level	RObj_ID	RObj_Level
AccReq_Attr		Judge_Result	

Figure 3. *Format of DTT transaction.*

The ART transaction is responsible for recording the decision result of the access operation. The transaction body contains the following payload fields:

- Sbj_ID: The unique identifier of the subject, including the subject's name, and IP address, MAC address, etc.
- Sbj_Level: The clearance level of the subject, consisting of the highest clearance level and the current clearance level.
- Obj_ID: The unique identifier of the object being accessed, including the name, IP address, MAC address, and the access domain number to which it belongs.
- Obj_Level: The security level of the object being accessed.
- AccReq_Attr: Access attributes requested by the subject, including read-only, write-only and read-write.
- Judge_Result: The judgment result of the access request from a subject, including "?", "0" and "1", where "?" indicates an illegal access warning, "0" indicates access is denied, and "1" indicates access is permitted.

Unlike the ART, the DTT transaction is responsible for recording the data transfer process between two different objects. In the DTT transaction body, the same fields as the ART transaction body are Sbj_ID, Sbj_Level, AccReq_Attr, and Judge_Result, and Its unique fields are:

- SObj_ID: The unique identifier of the data-sending object.
- SObj_Level: The security level of the data-sending object.
- RObj_ID: The unique identifier of the data-receiving object.
- RObj_Level: The security level of the data-receiving object.
- Data_Tansfer: Data transferred from the data-sending object to the data receiving object.

Data transfer operations are divided into intra-domain and inter-domain transfer modes. The inter-domain data transfer operation needs to record this data operation behavior in two different access domains, that is, to update the structure of the "independent" side chain, so according to The ID of the access control domain where the data receiving object is located divides DTT into two types of transactions: I_DTT (Inter-domain Data Transfer DTT) and C_DTT (Cross-Domain Data Transfer DTT).

As shown in Figure 4, the block is normally divided into block header and block body [28, 34]. The block header is used to record the basic parameters of the block and is linked with the previous block in the blockchain by recording the "PreHash" value. The block body is used to record the transaction data and the metadata of the block mining and verification process. All transaction data is computed by the parallel hash of the Merkle tree and recorded as a unique index value in the Merkle root to the block header. It should be noted that, unlike the permission-less blockchain based on the workload-based block mining process (i.e., the POW Protocol), the BC-BLPM model constructed by permissioned blockchain adopts a consensus mechanism for obtaining system consistency permissions to generate the block (we use the PBFT protocol in this work), therefore the block does not generate the target difficulty, random number and other fields required to execute the POW.

The transactions generated by the access behavior of different subjects form a chained data structure, as shown in Figure 5. This structure generates a new block by packaging a certain number of transactions per unit time, and is linked by the hash of the block. Among them, the C_DTT transaction must be packaged by an independent block, which is determined by the update requirements of cross-domain transactions, and also facilitates the development of later audit work.

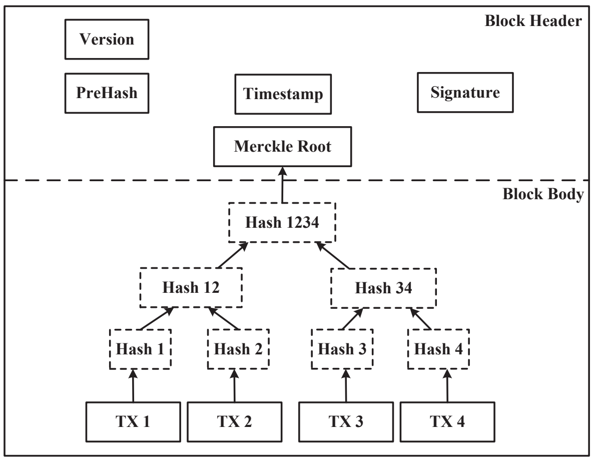


Figure 4. *Format of the block in BC-BLPM.*

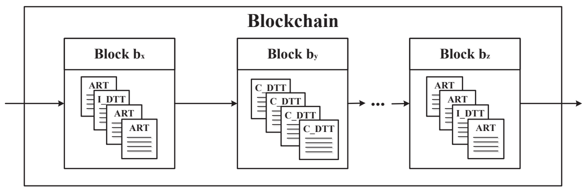


Figure 5. *Architecture of the blockchain ledger in BC-BLPM.*

3.3.2 Transaction Pool Structure

In order to ensure the continuity of the transaction packaging and on-chain process, so that each block can pack a fixed number of transactions as much as possible, and effectively reduce the delay generated by the block, this article uses the transaction pool to implement the sequential packaging process of transactions to blocks. The concept of transaction pool originally appeared in the Ethereum project, which means "a pool that temporarily stores transactions to be executed." The working mechanism of the transaction pool is shown in Figure 6: As long as a new transaction is generated, it will be first added to the transaction pool, whether it is generated by a local node or broadcast by other peer nodes. When the block needs to be packaged, the transaction will be extracted from this pool.

The transaction pool consists of transaction queues, including "queued" and "pending" sub-queues, where "queued" temporarily stores transactions that cannot currently be executed immediately, and "pending" stores currently executable transactions. When a large amount of transaction data is generated in a unit time, transactions exceeding the storage capacity of a single block are first stored in "queued", and a certain number of transactions are selected to form a new block according to the order of the queue. The data structure of the BC-BLPM blockchain described in Figure 5 shows that the record of the C_DTT transaction involves two different access control domains. And the transaction processing node needs to cross two different sidechains to execute the inter-domain of the block Update operation. Therefore, in order to ensure the synchronous update of transactions, this article deploys a transaction buffer pool composed of double transaction queues in the transaction transfer layer to separately process ART, I_DTT transactions within the domain and C_DTT transactions between domains. This transaction buffer pool working mode ensures that inter-domain transactions and intra-domain transactions are alternately chained. Due to the inevitable delay in the inter-domain transaction transfer process, the transaction buffer pool schedules the transaction queue based on the time stamp of inter-domain transactions.

3.3.3 Smart Contract Format

The concept of a smart contract was first proposed by Nick Szabo [8] and was defined as a "transaction agreement for executing contract terms through computers", that is, the contract was automatically executed by a code program. The essence of the smart contract is a computer algorithm that binds participan-

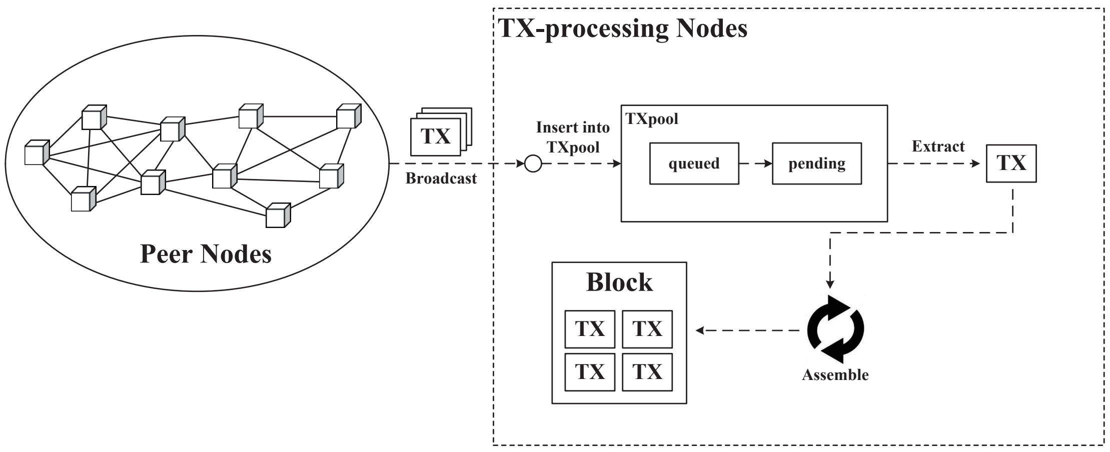


Figure 6. *BC-BLPM transaction pool structure.*

ts in the transaction processing platform with related transactions involved. In the blockchain, the smart contract refers to a program that can be automatically execute on the blockchain. It has the characteristics of programmability, atomicity, consistency, etc. The blockchain system can automatically execute the business logic of the blockchain by means of smart contracts packaged and deployed in its distributed nodes without the supervision of a third party.

In this work, smart contract is used as the policy execution carrier of the access control system, that is, the execution logic of BC-BLPM's access control policy is written in the smart contract, and different operation interfaces are provided, so that the smart contract can perform corresponding policy according to different access requests. One thing to be clear is that the interfaces encapsulate the execution logic of state transition rules and are embedded in smart contracts to provide external calls to smart contracts. Any entity can perform access control policies in BC-BLPM by deploying and executing the smart contracts in the peer nodes of the blockchain system. The definition of a smart contract interface is:

- $Initiate()$: This function is used to perform state initialization operations on the access domain and is executed by the system manager.
- $Read\_only()$, $Write\_only()$, $Write\_Read()$: These 3 types of interfaces represent the execution phases of the read-only, write-only, and read-write rules, respectively. The corresponding operations can be performed by inputting parameters such as subject and object information, and access requests.
- $Data\_Sending()$: This function represents the execution phase of the data-transfer rule, and is also called by BC-BLPM according to the content of the access request. A data transfer operation for different objects is performed by inputting the access request of the subject, the identification information of the data-sending object and the data-receiving object.

IV. IMPLEMENTATION

The actual deployment architecture of BC-BLPM is shown in Figure 7. This section describes the different interaction processes that BC-BLPM needs to complete during the deployment process. The interaction process can be divided into four phases: the initialization phase initializes the access domains by deploying smart contracts and CPK combination key suite; the access permission management phase completes access rights management for access-requesting users by executing the relevant policy interface of the smart contract; the data transfer phase implements the secure circulation of confidential data between the same domains or different domains; and the resource management phase provide convenience for the maintenance of resources.

4.1 Initialization

In the initialization phase, the system manager $m \in S^*$ as a trusted subject, needs to perform the operation of adding resource objects and deploying smart

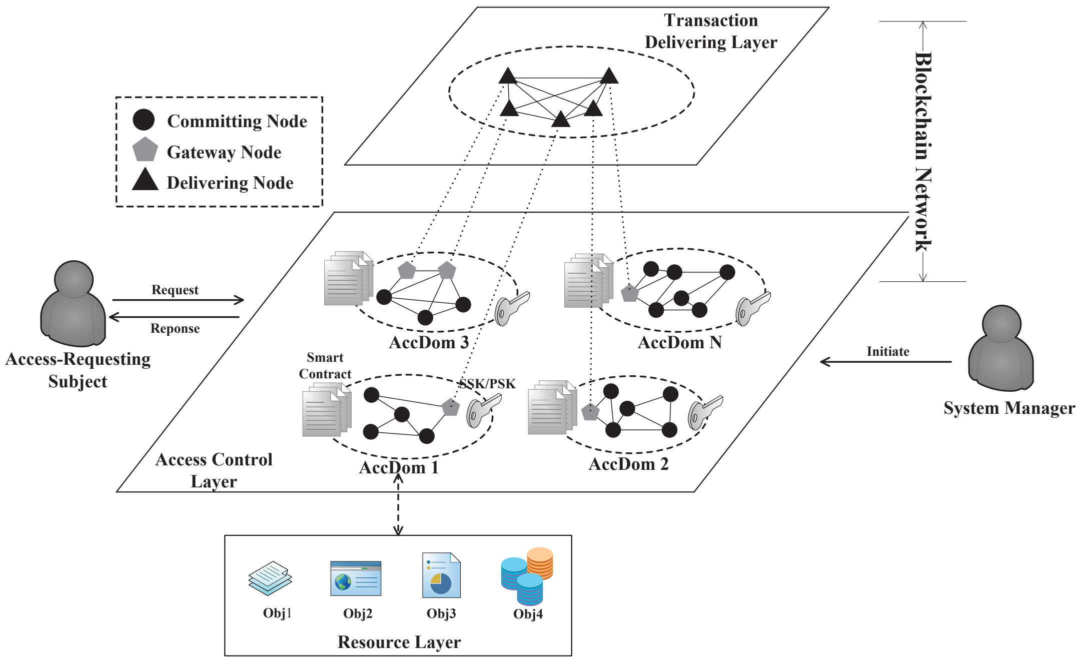


Figure 7. *Deployment architecture of BC-BLPM.*

contracts to the access domain $AccDom_x$, thus completing the instantiation of the $AccDom_x$. The steps to perform are as follows:

Step 1: The system manager m specifies an access domain identifier $AccDom_x.ID$, and selects the resource object $o \in O$ that needs to join the access domain.

Step 2: Randomly generate N-level CPK public and private key seeds PSK_{AD}/SSK_{AD} and deploy them in each access control domain $AccDom_x$.

Step 3: Then m encodes the smart contract SC and install it into the gateway node of $AccDom_x$, specifying the calling address $SC.Addr$ of the smart contract.

Step 4: As shown in Algorithm 1, by invoking the $Initiate()$ interface of the smart contract, a founding block $Genesis\_Block$ of $AccDom_x$ is generated, which is responsible for defining the metadata of the access domain, and does not contain transactions such as ART or DTT in the block. In $Genesis\_Block$, we need to add the signature $Sig\{Msg_1, Msg_2, \ldots, Msg_n\}_{SK_m}$ which is generated by the private key of m, $N - level$ CPK seed sequence $\{PSK_{AD}/SSK_{AD}\}_{n=1}^{N}$, resource object identifier $o_j.ID$, access domain identifier $AccDom.ID$, access control list ACL, and current timestamp TS to instantiate a new access domain. Note that the ACL is a simplified first-order array containing access attribute elements, representing the "optional legal operation", which plays the role of the access control matrix M_{ij} during system execution (see Section 3.2(A) for the definition of M_{ij}).

$$Genesis_{Block} \leftarrow$$

$$Sig\left\{\begin{array}{l} AccDom_x, ID, Obj_{Set}, \\ \{PSK_AD/SSK_{AD}\}_{n=1}^{N}, \\ SC, Addr, ACK, TS \end{array}\right\}_{SK_m} . \tag{5}$$

In Algorithm 1, it first needs to complete the definition of the resource object set and the access domain identification number, and package them together with the access control list and the signature of the manager to generate the proposal InitProp (as is shown in line 1 to 3). Then, through the execution of the consensus agreement, the decision result of the proposal from all nodes in the domain is obtained (see line 4 to 9); if the protocol is passed, the smart contract will generate the $Genesis\_Block$ and broadcast the event to the whole domain, otherwise will return an error message. Through the creation of $Genesis\_Block$, each node of the access domain will obtain the ledger copy of the blockchain, and will maintain the data integrity of the domain in the subsequent execution process.

Algorithm 1. *SC-Initiate ( ).*

```
Require:
    $AccDom_x, ID, \{o_1, o_2, \dots, o_n\}, ACL$
Ensure:
    $Genesis\_Block$
 1: def $Obj\_Set[n] \leftarrow \{o_1, o_2, \dots, o_n\}$;
 2: def $ID_{ACCDom} \leftarrow ACCDom_x, ID$;
 3: $InitProp \leftarrow Package(Obj\_Set \parallel ID_{AccDom} \parallel ACL \parallel m.Sig)$;
 4: for $i$ in range($AccDom_x \rightarrow Nodes.Num$) then ;
 5:     $Response \leftarrow Node_i.GetValidate(InitProp)$;
 6:     if $Response == True$ then
 7:         $ValidateNum \leftarrow ValidateNum + 1$
 8:     end if
 9: end for
10: if $ValidateNum < (AccDom_x \rightarrow NodesNum) * 2/3 + 1$ then
11:     return False
12: else then
13:     Generate $GenesisBlock$
14:     Broadcast event $ReturnBlock(GenesisBlock)$
15: end if;
```

4.2 Access Permissions Management

After the initialization of the system is completed, BC-BLPM acts as an access permissions manager, provide access decision service for external access subjects. The process is as follows: Step 1: Before sending an access request to the access control domain, the subject first needs to download the CPK private key seed of the access control domain and generate a signature private key IDSK based on its own identification information.

Step 2: The subject $s_i \in S$ issues an access request $AccReq$ to the gateway node in the access domain.

$$AccReq = Sig\{Attr, Info_{s_i}, Info_{o_j}, T_{AC}\}_{SK_{s_i}}. \quad (6)$$

$Attr$ represents access attributes, such as "read-only", "write-only" or "read-write", a single access request can only load one access attribute; $Info_{s_i} = \{s_i.ID, < f_s(s_i), f_c(s_i) >\}$ represents subject's information and $Info_(o_j) = \{o_j.ID, f(o_j)\}$ represents object's information, where $s_i.ID$, o_j.ID are the unique identifiers of subject s_i and object o_j respectively; T_{AC} represents the access time of the access-requesting subject, we stipulate that the threshold of access time is determined by the clearance level of the subject which is shown in Table 1; furthermore, $AccReq$ also adds the CPK signature of the access subject s_i to ensure the verifiability of identity information.

Algorithm 2. *SC-Read_only ( ).*

```
Require:
    $AccReq$
Ensure:
    $Decisionresultinstringform$
 1: def $MaxSbjLevel \leftarrow AccReq.f_s(s_i)$,
    $CurSbjLevel \leftarrow AccReq.f_c(s_i)$ and
    $ObjLevel \leftarrow AccReq.f(o_j)$
 2: if $AccReq.Attr \notin ACL$ or $T_AC > Threshold$ then
 3:     return $string(\text{"}ERROR\text{"})$
 4: else then
 5:     $Decision \leftarrow \rho_r(MaxSbjLevel, CurSbjLevel, ObjLevel)$

 6:     if $Decision ==$ "$YES$" then
 7:         return $string(\text{"}PERMIT\text{"})$
 8:     else if $Decision ==$ "$NO$" then
 9:         return $string(\text{"}DENY\text{"})$
10:     else return $string(\text{"?"})$
11:     end if
12: end if;
```

Table 1. *Correspondence between clearance level and access time.*

Clearance level interval	Label	Threshold (h)
[0, 10)	Normal	0.5
[10, 30)	secret	1
[30, 50)	confidential	10
[50, ∞]	Top secret	24

Step 3: After receiving the access request, the gateway nodes first verify the correctness of signature Sig. If the verification fails, an error message is sent to s_i to end the interaction process. When the verification is passed, the gateway nodes broadcast $AccReq$ to the remaining nodes in the access domain.

Step 4: After receiving the message, all the nodes in the domain extract the Attr element and call the policy interface $Read\_only()$, $Write\_only()$ or $Write\_Read()$ of the smart contract according to different access attribute values to get the decision result and send it back to the gateway node. Take the $Read\_only()$ interface as an example. Its implementation process is shown in Algorithm 2. It first defines the security level, determines the validity of the access attribute and the access request time (line 1 to 3), if

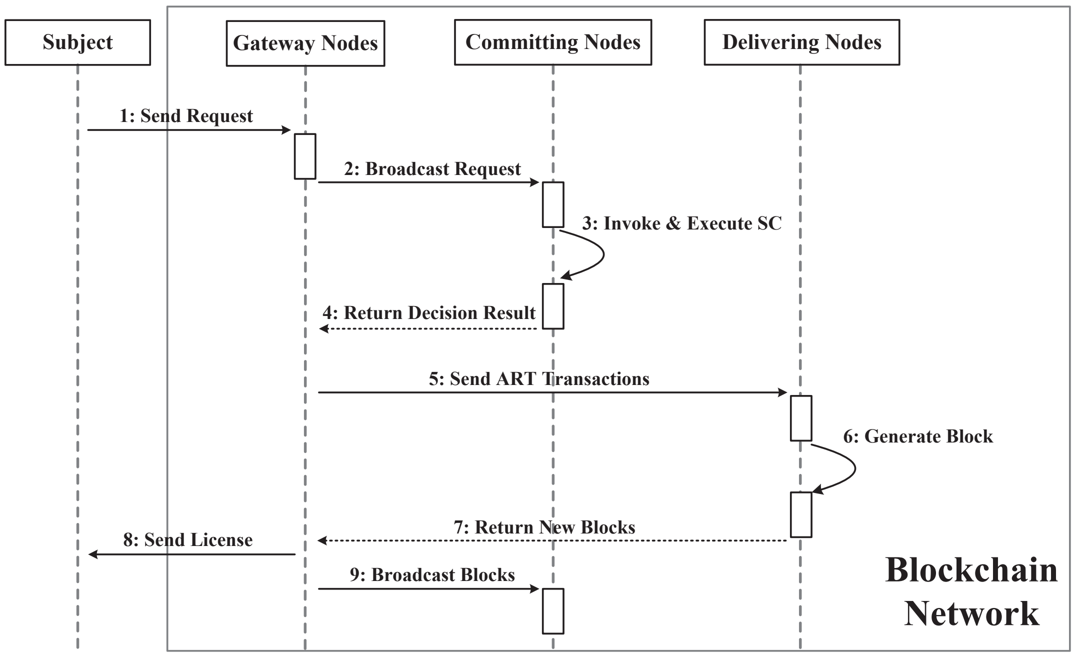


Figure 8. *Implementation process of access permissions management phase.*

the decision is passed, the result of the access requests will be obtained by the execution of the state transition rule ρ_r (lines 5 to 10).

Step 5: The gateway nodes collect the proposal response and executes the consensus protocol. If passed, the AccReq and the decision result are packaged into an ART transaction and sent to the delivering nodes.

Step 6: The delivering nodes pack different types of transactions in a unit time into blocks, and return the blocks to the gateway nodes of the corresponding access domain. After receiving the block, the gateway nodes link the block into the public ledger of blockchain and synchronize the block to other nodes in the domain.

Step 7: Finally, the gateway nodes generate a temporary access token $Licence_{(s_i,o_j)}$ and send it to the subject. $Licence_{(s_i,o_j)}$ consists of the subject's and object's information $Info_{s_i}$ and $Info_{o_j}$, the access attribute $Attr$, the access time limit LC and the current timestamp TS, and the signature of system manager is added as well.

$$\begin{aligned} &License_{(s_i,o_j)} = \\ &\quad Sig\{Info_{s_i},Info_{o_j},TS,LC,Attr\}_{SK_m} \end{aligned}. \tag{7}$$

The implementation process of the access permissions management phase is shown in Figure 8.

4.3 Data Transfer

During the access period, the subject $s_i \in S$ may transmit the data resources in the accessed object to other objects, thereby realizing data sharing between two different resource objects. The data transfer phase can be divided into intra-domain data transfer mode and inter-domain data transfer mode according to the different access domains involved.

4.3.1 Intra-domain Data Transfer Mode

In this mode, the subject sends the data resources in the accessed object to another resource object, and both of two objects are in the same access domain. Proceed as follows:

Step 1: The subject $s_i \in S$ creates a data transfer request $DataTransReq$ and sends it to the gateway nodes of the access domain where the object o_j is located, Where TS' represents the current timestamp, $DataBuf$ is a temporary "key-value" array that stores transferred data, the temporary access token $Licence_{(s_i,o_j)}$ can be used as the identifier of the access period, $Info_{o_k}$ is the composition information of the data-receiving object o_k, and finally, we need to

Algorithm 3. *SC-Data_Sending ( ).*

```
Require:
    $DataTransReq, Licence$
Ensure:
    Decision result in string form
 1: def $CurSbjLevel \leftarrow DataTransReq.f_c(s_i)$, $SObjLevel \leftarrow DataTransReq.f(o_j)$ and $RObjLevel \leftarrow DataTransReq.f(o_k)$
 2: if $DataTransReq.TS' > Licence.TS + Licence.LC$ then
 3:     return $string(“DENY”)$
 4: else then
 5:     $Decision \leftarrow \rho_{sd}(CurSbjLevel, SObjLevel, RObjLevel)$
 6:     if $Decision ==$ “$YES$” then
 7:         return $string(“PERMIT”)$
 8:     else if $Decision ==$ “$NO$” then
 9:         return $string(“DENY”)$
10:     else return $string(“?”)$
11:     end if
12: end if;
```

add the signature of s_i to the request.

$$DataTransReq =$$

$$Sig\left\{\begin{array}{c} sd, Info_{s_i}, Info_{o_j}, \\ DataBuf, Info_{o_k}, TS' \end{array}\right\}_{SK_{s_j}}. \tag{8}$$

Step 2: After receiving the request DataTransReq, the gateway nodes broadcast the request to the remaining nodes in the domain. This process is the same as step 2 of Section 4.2. All nodes in the access domain invoke the interface $Data\_Sending()$ of the smart contract relying on the access attribute sd to get the decision result, and send it back to the gateway nodes. The invoking process of $Data\_Sending()$ is shown in Algorithm 3. After the algorithm first completes the definition of the security level variables, it needs to determine whether the request time expires in the temporary license (line 1 to 3); if the request time is not expired, the decision result of the data transfer request will be obtained by executing the state transition rule ρ_{sd} (line 4 to 10). Finally, the decision result is returned.

Step 3: After receiving the decision result, the gateway nodes generate a transaction DTT and send it to the delivering nodes. The delivering nodes then package the transaction into a block and send it back to the access domain.

4.3.2 Inter-Domain Data Transfer Mode

In this mode, the subject sends data resources in the accessed object to another object belong to different access domain. The biggest difference between this phase and the inter-domain data transfer phase is that the block containing DTTs needs to be updated in the sidechains of the two different access domains.

The execution step 1-2 of the mode is the same as the corresponding step of the intra-domain data transfer mode. After the gateway nodes send DTT to the delivering node, the delivering nodes need to determine the $SObj\_ID$ and $RObj\_ID$ fields of the DTT. If the objects displayed in the two fields have different access domain identifiers, the DTT will be packaged into two identical blocks, and the blocks are sent to different access domains according to the iden-tifiers in the fields. The two access domains finally perform an update operation of its respective public ledger after receiving the block.

The implementation process of the data transfer phase is shown in Figure 9.

4.4 Resource Management

With the development of actual business, such as the adjustment of a business department in the government department or the change of business data, the resource management mode of BC-BLPM will also be adjusted accordingly.

4.4.1 Resource Addition and Revocation

The new resource objects may need to be added to the BC-BLPM along with the extension of the business logic, and the old need to be periodically cleared out. This process is performed by the system manager $m \in S^*$. It is similar to the initialization phase that m needs to provide a new resource object set Obj_Set_{New} or a set to be cleared Obj_Set_{Rev}, and generates a new block Update_Block by re-executing the $Initiate()$ interface of the smart contract SC. The block is used to record the addition or undo behavior of the resource object.

$$Update\_Block \leftarrow$$

$$Sig\left\{\begin{array}{c} AccDom_x.ID, TS, \\ Obj\_Set_{New}/Obj\_Set_{Rev} \end{array}\right\}_{SK_m}. \tag{9}$$

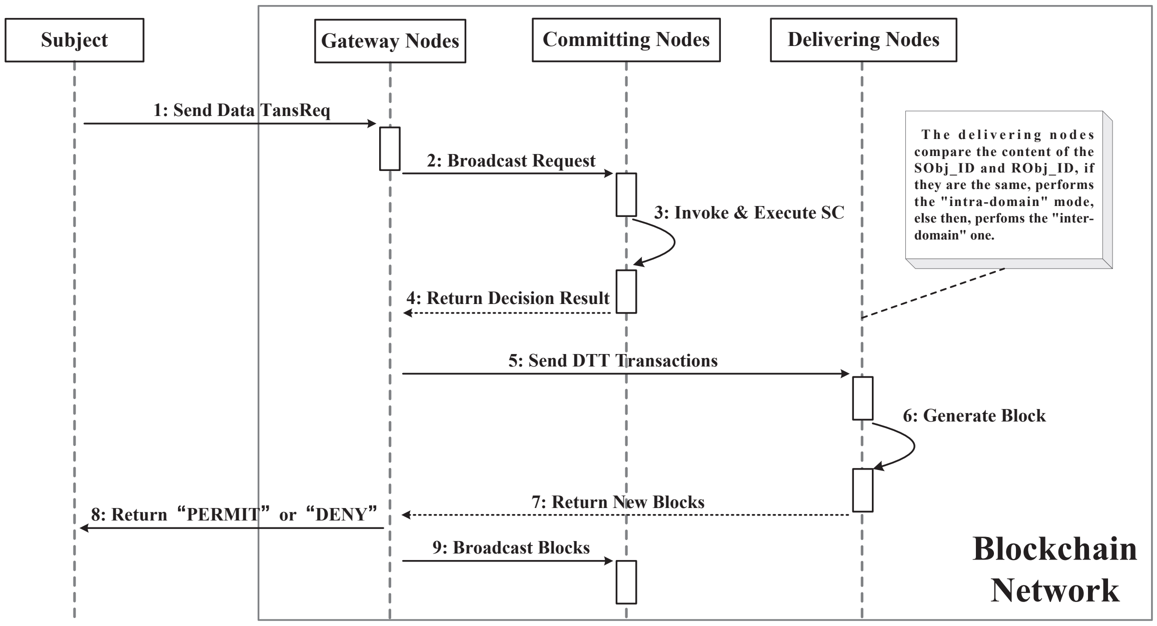

Subject
Gateway Nodes
Committing Nodes
Delivering Nodes
1: Send Data TansReq
2: Broadcast Request
3: Invoke & Execute SC
4: Return Decision Result
5: Send DTT Transactions
6: Generate Block
7: Return New Blocks
8: Return "PERMIT" or "DENY"
9: Broadcast Blocks
The delivering nodes compare the content of the SObj_ID and RObj_ID, if they are the same, performs the "intra-domain" mode, else then, perfoms the "inter-domain" one.
Blockchain Network

Figure 9. *Implementation process of access permissions management phase.*

4.4.2 Node Revocation

When a node in the access domain is found to be maliciously attacked or invalidated, the system manager $m \in S^*$ can issue a null ART transaction (only fill in the node ID and the content of the "Node Identity Revocation" in the Obj_ID and $AccReq\_Attr$ fields respectively) and record it separately in the sidechain of the access domain where the node is located to represent the revocation process of the node.

V. EXPERIMENT AND ANALYSIS

Here we set up a simple intranet environment and implemented the BC-BLPM prototype system based on Hyperledger Fabric V-1.1.0 [35]. As an enterprise-level permissioned blockchain infrastructure, Hyperledger Fabric provides PKI-based node and user authentication services, Gossip-based and gRPC-based distributed communication services, channel-based "multi-blockchain" services, and Chaincode-based smart contract encoding services, which are available for the functional requirements of the BC-BLPM infrastructure. We apply Channel to divide the access domain and encode the smart contracts based on Chaincode logic. On this basis, we separately evaluate the implementation effect and system performance of BC-BLPM, and analyzes the security of the model. In addition, the final part of this section compares our model with the existing multi-level security system, showing the similarities and differences between the BC-BLPM and the existing multi-level security model.

5.1 Experimental Environment

The experiment was carried out in 8 desktop computers configured with 3.40GHz Inter Core i5 CPU, 8.00 GB memory and Ubuntu 16.04 OS. The computer set up virtual routing through the router to form a small intranet structure with multiple access domains. The experimental topology is shown in Figure 10. Then we use the sandbox container tool Docker [36] to deploy virtual fabric nodes in each physical computer. The fabric node is divided into two roles: Peer and Orderer, where Peer is a common peer node responsible for generating and verifying transactions; Orderer is a transaction sorting and delivering node that is responsible for packaging transactions into blocks and distributing them to different Channels. We set VLAN 0 as the transaction delivering layer, deploy 4 virtual Orderers in the domain, and deploy the high-performance cluster data processing plugin Apache Kafka [37] to improve the efficiency of transaction sorting and delivering. The remaining VLANs are deployed single channel as separate access domains, that is, separate sidechains in each VLAN.

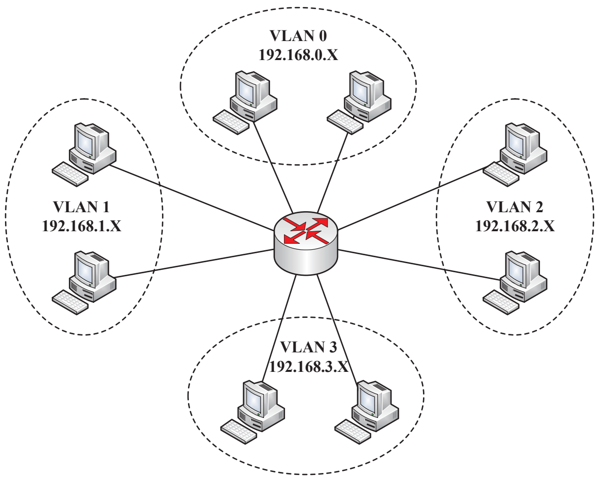


Figure 10. *Experimental topology.*

The parameter settings of Fabric are shown in Table 2. We specify that the average write-set length of each transaction is 20 bytes, each block contains 30 transactions, and the block size does not exceed 512 KB; Set the total capacity of the transaction pool to 1000TX (TX means transaction unit), in which "queued" queue is allocated 300TX and "pending" queue is allocated 700TX; GoLevelDB [38] is adopted as the blockchain state database for storing configuration information of blockchain ledgers in the local environment with multiple sidechains; the communication bandwidth between the nodes is 2 Gbps, and the communication link adopts the TLS security protocol; the consensus module adopts the PBFT protocol, that is, in each round of transactions execution, the number of honest nodes participating in transaction verification must be no less than 2/3 of the total number of nodes.

Table 2. *Parameter settings of Fabric.*

parameter	value
TX size	20 bytes
State database	GoLevelDB
Network transmission bandwidth	2 Gbps
Consensus module	PBFT
Block height	30 Transactions Per Block
Block size	512KB
Transaction pool capacity	1000TX
Transaction timeout	1000ms
Whether to adopt the TLS protocol?	Yes

We encode the policy execution interfaces required for BC-BLPM smart contracts through the Chaincode. As a smart contract paradigm for Fabric, Chaincode can be implemented in a variety of programming languages such as golang, java, node.js, etc. It is designed as a virtual container deployed in Docker, capable of communicating with Peer nodes via gRPC protocol, providing blockchain transaction logic based on the state machine model, and using the shim.ChaincodeStubInterface structure to achieve interaction with the blockchain ledgers.

In the experiment, we use the benchmark test tool Hyperledger-Caliper [39] to simulate the client nodes in the network and generate transaction requests to the blockchain network, each virtual client node plays the role of "external access-requesting subject". A unified X.509 format certifi-cate is issued by the same certificate authority, and the subject clearance level is written to the "Extend" field of these certificates. The "resource object" is uniformly represented by a lightweight data file in the JSON format, and the security level label is represented by the unified key value of the file header "SecLevel-Value".

We mainly examine the performance of BC-BLPM through the following indicators:

- Throughout: The number of transactions generated in the blockchain network per unit time, measured in units of TPS (Transactions Per Seconds).
- Latency: The time it takes for a message to be transmitted between nodes when a client completes a transactional interaction. We divide it into Average Latency (Avg Latency), Maximum Latency (Max Latency) and Minimum Latency (Min Latency).
- Hit ratio: The percentage of successfully generated transactions as a percentage of the total transaction requests.

5.2 Effectiveness Evaluation

In this experiment, we evaluate the execution results of different access control policies by setting different test cases. We set up two client nodes named Cli1 and Cli2, their current clearance levels are 3 and 2 respectively; three data files named Jfile1, Jfile2, and Jfile3 are set, and their security levels are 3, 2, and 1 respectively. Jfile1 and Jfile2 are located in VLAN 1, and Jfile3 is located in VLAN 2. The resources and smart contracts are initialized in the two access domains respectively, where $ACL = [r, a, w, sd]$ in VLAN 1 and $ACL = [r, a, sd]$ in VLAN 2. The security level satisfies the partial order relationship, e.g. $3 \rhd 2 \rhd 1$.

Through the data-flow API provided by Hyperledger-Caliper, 5000 transaction request packets containing the same access attribute are sent for each group of experiments, and the sending rate is set to 50 TPS to check the execution effect of the policy. The experimental results are shown in Table 3.

For the results of Table 3, the Chaincode-based policies can automatically identify the request type and execute the corresponding blockchain transaction logic, and the hit ratio is generally above 85%. At the same time, the sd transaction across two access domains is significantly higher than the transaction in the same domain (as is shown in line 6, 12, and 13), which is determined by the characteristics of the inter-domain data transfer mode.

5.3 Security Evaluation

Security as a basic evaluation indicator for an access control system is no exception in our model. BC-BLPM requires a multi-level secure access control environment that requires the ability to withstand external or internal malicious attacks and data leakage. Therefore, we use the basic principle of information security level protection—CIA (i.e., confidentiality, integrity, and availability) as an indicator of the security. Among them, confidentiality means that only authorized users can access data; integrity means that data cannot be tampered with and destroyed by illegal authorized entities during transmission; availability means that the access of data by legitimate users is not maliciously rejected.

5.3.1 Confidentiality

The access control policy of BC-BLPM is implemented by BLP-based security policy primitives. The policy based on the Bell-LaPadula model limits the flow of authority data, and can effectively prevent high-confidentiality data from flowing to users with low-confidentiality level. The security of the BC-BLPM state transition rule needs to be verified, thus ensuring the overall confidentiality of the system.

Proof. Rule 1 maintains secure

Assuming that the initial state $v = (b, M, f)$ is secure, when the access request $R_k = (s_i, o_j, r)$ is executed, the new state is $v^* = v$ or $v^* = (b^*, M, f^*)$. If $v^* = v$, the state v^* maintains secure because of the security of v. If $v^* = (b^*, M, f^*)$:$v^* - v = ((s, o, r), M, f^*)$ where $f^* = f \cup \{max(f_c(s), f(o))\}$ and $f_s(s_i) \rhd f(o_i)$, so that $v^* - v$ satisfies the simple-security property. Since v also satisfies simple-security property, v^* satisfies simple-security property. Since $f^* - f = max(f_c(s), f(o))$, where $(f^* - f) \rhd f(o)$, so v^* satisfies the *-property. From the above, we prove the security of Rule 1.

Proof. Rule 2 maintains secure

Assuming that the initial state $v = (b, M, f)$ is secure, when the access request $R_k = (s_i, o_j, a)$ is completed, the new state is $v^* = v$ or $v^* = (b^*, M, f)$. If $v^* = v$, the state v^* maintains secure because of the security of v. If $v^* = (b^*, M, f)$:$v^* - v = ((s, o, a), M, f)$, where $f_s(s_i) \rhd f(o_i)$, so that $v^* - v$ satisfies the simple-security property. Since v also satisfies simple-security property, v^* satisfies simple-security property. Since $f_c(s_i) \rhd f(o_k)$ in Rule 2, $v^* - v$ satisfies the *-property. And since v also satisfies the *-property, v^* satisfies the *-property. From the above, we prove the security of Rule 2.

Proof. Rule 3 maintains secure

Assuming that the initial state $v = (b, M, f)$ is secure, when the access request $R_k = (s_i, o_j, w)$ is executed, the new state is $v^* = v$ or $v^* = (b^*, M, f^*)$. If $v^* = v$, the state v^* maintains secure because of the security of v. If $v^* = (b^*, M, f^*)$:$v^* - v = ((s, o, w), M, f^*)$, where $f^* = f \cup \{f(o)\}$ and $f_s(s_i) \rhd f(o_i)$, so that $v^* - v$ satisfies the simple-security property. Since v also satisfies simple-security property, then v^* satisfies simple-security property. Since $f^* - f = f(o)$, then $v^* - v$ satisfies the *-property. Since v also satisfies the *-property, v^* satisfies the *-property. From the above, we prove the security of Rule 3.

Proof. Rule 4 maintains secure

Assuming that the initial state $v = (b, M, f)$ is secure, when the access request $R_k = (s_i, o_j, o_k, sd)$ is executed, the new state is $v^* = v$ or $v^* = (b^*, M, f^*)$. If $v^* = v$, the state v^* maintains secure because of

Table 3. *Execution results of different access control policies.*

No.	**Request**	Result	Number of transactions	Hit ratio (%)	Min Latency (s)	Max Latency (s)	Avg Latency (s)
1	(Cli1, Jfile2, r)	Permit	4735	94.70	0.13	1.29	0.74
2	(Cli1, Jfile3, a)	Deny	4814	96.28	0.09	1.38	0.89
3	(Cli1, Jfile1, w)	Permit	4792	95.84	0.11	1.34	0.72
4	(Cli1, Jfile1, Jfile2, sd)	Deny	4686	93.72	0.28	1.56	0.81
5	(Cli1, Jfile2, Jfile1, sd)	Permit	4715	94.30	0.16	1.35	0.70
6	(Cli1, Jfile1, Jfile3, sd)	Deny	4256	85.12	1.54	3.46	2.62
7	(Cli2, Jfile1, r)	Deny	4824	96.48	0.13	1.70	0.83
8	(Cli2, Jfile2, a)	Permit	4659	93.18	0.17	1.42	0.87
9	(Cli2, Jfile2, w)	Permit	4785	95.70	0.27	1.69	0.73
10	(Cli2, Jfile3, w)	Deny	4791	95.82	0.21	1.42	0.75
11	(Cli2, Jfile2, Jfile1, sd)	Permit	4627	92.54	0.18	1.69	0.87
12	(Cli2, Jfile3, Jfile2, sd)	Permit	4338	86.76	1.75	3.89	2.19
13	(Cli2, Jfile3, Jfile1, sd)	Permit	4308	86.16	1.86	4.56	2.34

the security of v. If $v^* = (b^*, M, f^*)$:$v^* - v = ((s_i, o_j, o_k, sd), M, f^*)$, where $f^* = f \cup \{f(o_k)\}$ and $f_s(s_i) \triangleright f(o_k)$, then $v^* - v$ satisfies the simple-security property. Since v also satisfies simple-security property, v^* satisfies simple-security property. Since $f^* - f = f(o_k)$, where $f(o_k) \triangleright f(o_j)$, $v^* - v$ satisfies the *-property and data-flow property. Since v also satisfies the *-property and data-flow property, v^* satisfies the *-property and data-flow property. From the above, we prove the security of Rule 4.

5.3.2 Integrity

In the blockchain system, the blocks are linked to the end of the ledger in a time-sequential sequence by time stamping, and they are connected to each other by Hash values. BC-BLPM uses the blockchain to store the access records of the access-requesting subject. Each time a subject completes the access process in a specific access domain (i.e., each time a smart contract is executed), a new data block is linked in the blockchain maintained by the domain.

Since each block has a Hash value containing the complete block volume data and the PreHash value of the pre-order block, if an attacker illegally tampers with the data in a single block, it will cause the Hash value to change, so that the attacker must continue to change the pre-order Hash value of the block. And due to the full redundancy backup feature of the blockchain data, tampered blocks must be broadcast to the entire network nodes to make it effectively. After receiving the new blocks, these nodes will be able to detect that the block is invalid by traversing its own blockchain data backup, thus resisting forgery. In this way, the attacker must invade all distributed nodes that maintain the blockchain and continue to repeat the tampering process of the block data, which will greatly increase the cost of the attack. According to the analysis in Ref. [40], the probability of changing the content of a block in the blockchain is: assume that n is the number of blocks that the honest node finds during the average time (i.e., the execution time of the consensus protocol), l is the number of blocks found by an external malicious attacker during that time. The competitive relationship between honest nodes and malicious attackers is a binomial random walk process, where l is a negative binomial variable, equivalent to the generation process of n blocks (blocks discovered by honest nodes). The number of successfully generated blocks (generated by the attacker) whose success rate is set to q. Assuming that the probability of an honest node acquiring the next block is p, and $q < p$, the probability of a malicious attacker tampering with the block is equal to:

$$\rho_a = \sum_{t=0}^{\infty} P(l) a_{n-1-l} \sum_{l=0}^{n-1} \left(\begin{array}{c} l+n-1 \\ l \end{array} \right) p^n q^l$$

$$\left(\left(min\left(\frac{q}{p}, 1\right) \right) \right)^{n-1} + \left(\begin{array}{c} l+n-1 \\ l \end{array} \right) \tag{10}$$

$$\sum_{l=n}^{n-1} p^n q^l = \begin{cases} 1 - \sum_l^{1+n-l} \left(p^n q^l - p^l q^n\right) & q < p \\ l & q \geq p \end{cases}.$$

In summary, when n is 2, the tampering rate is less than 10%; when n is 4, the change tampering rate is less than 1%; n is 6 is lower than 0.1%. Therefore, as long as the length of the blockchain continues to grow, the tamper resistance of the blockchain will become stronger. In addition, since the data on the blockchain is publicly visible to all subjects, the access result is recorded on the blockchain, so that the policy is made public and transparent, and it is convenient for the third-party authorities to verify and audit the access records.

The relationship between the probability that the attacker successfully tampered with the block ρ_a and the distance n of the block changes as shown in Figure 11. Increase and gradually decrease. Conversely, when the probability that the attacker obtains ownership of the next node is greater than or equal to 0.5, the attacker can successfully tamper with the next block. That is to say, only when the attacker obtains more than 50% of the nodes on the blockchain can he control the data trend of the entire blockchain. Because there are many nodes in the blockchain, an attacker who wants to fully control 50% of the entire network will have to pay a huge cost, so it is difficult to break through, so a blockchain with a certain chain length can achieve very good anti-attack effects. In addition, in addition

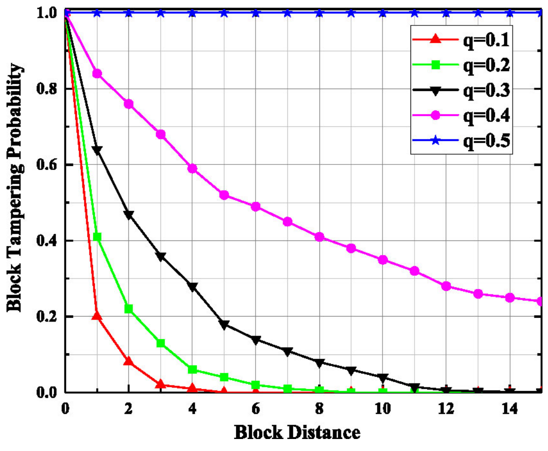


Figure 11. *Attacker's probability of success.*

to the transaction signature TX Sig verification of the user identity in the BC-BLPM transaction, the CPK signature value of the subject also provides a verification mechanism for the integrity of the subject identification information for the BC-BLPM transaction. The implementation of the CPK signature mechanism relies on a new key pair generated based on the identity hash. The generation of this key pair needs to strictly meet anti-collision requirements, that is, different identity identification information hash generates different key pairs. The following is the proof of anti-collision of CPK seed key sequence selection:

Proof. CPK seed key sequence selection has anti-collision characteristics

In the CPK system, the user ID first obtains a fixed-length hash value through a hash function, and then applies the row mapping algorithm to the hash value, and finally obtains the row coordinate selection sequence of the required length. At this time, the sequence S is selected. The length is $len \times k$. Suppose the length of the user identification is len_{ID} bits, the block length of the block cipher E with the encryption key K is len_B bits, and the length of the required sequence S of the key is len_S bits. Since the length of the user ID is len_{ID}, the size of the public and private key space must reach at least $2^(len_{ID})$. Therefore, it is reasonable to limit $len_ID < len_S$, and $len_B \mid len_S$ to get the output result of two sequences S.

(a) When $len_{ID} \leq len_B$, because of $len_B \mid len_S$, so $len_B \leq len_S$, calculate $MAP_{i+1} = E(MAP_{i+1}, K)(i = 0, 1, \ldots, len_S/len_B - 1)$, the output sequence $S = MAP_0 \parallel MAP_1 \parallel \cdots \parallel MAP_{len_S/len_B - 1}$.

(b) When $len_ID > len_B$, by filling the user ID with 0 bits to $\lceil len_ID/len_B \rceil \times len_B$ bits, you can get different $\lceil len_{ID}/len_B \rceil$ Group $ID_0, ID_1, \ldots, ID_(\lceil len_ID/len_B \rceil - 1)$, of which $ID_i(i = 0, 1, \ldots, \lceil len_ID/len_B \rceil - 1)$ The length is len_B, calculate $MAP_{i+1} = E(MAP_{i+1}, K)(i = 0, 1, \ldots, \lceil len_{ID}/len_B \rceil - 1)$, the output sequence $S = MAP_0 \parallel MAP_1 \parallel \cdots \parallel MAP_{\lceil len_ID/len_B \rceil - 1}$; if $\lceil len_{ID}/len_B \rceil < len_S/len_B$, calculate $MAP_{i+1} = E(MAP_(i + 1), K)(i = \lceil len_{ID}/len_B \rceil - 1, \ldots, len_S/len_B - 1)$, which can be finally obtained Output sequence $S = S \parallel MAP_{len_{ID}/len_B} \parallel \cdots \parallel MAP_{len_S/len_B - 1}$.

The above two output results completely avoid the collision caused by intercepting the intermediate sequence to generate the sequence S that is ultimately required.

5.3.3 Availability

Many malicious attacks from outside or inside can restrict the availability of access control systems, such as denial of service (DoS) attacks, content tampering attacks, identity spoofing attacks, deleting attacks, and replay attacks, etc. The decentralized and anti-forgery features of the blockchain are significant for defending against content tampering attacks, identity spoofing attacks, and deleting attacks. Therefore, we mainly analyze the ability to resist DoS attacks and anti-replay attacks.

In a replay attack, an attacker may attempt to spoof the object by intercepting the message sent by the subject to the object and sending it to the object again. BC-BLPM acts as an intermediary for message interaction between the subject and the object. After the completion of an interaction process, the blockchain network can perform distributed verification on the legitimacy of the message. Any duplicate message will not be verified by the honest nodes. In addition, all verification processes will be publicly and transparently recorded in the blockchain, and anyone can detect duplicate illegal messages by verifying the information in the records.

In a DoS attack, an attacker may attempt to consume the resources of the victim by brute force means (such as a botnet attack), preventing legitimate subjects from accessing resources and requesting services. In BC-BLPM, an adversary can initiate a DoS attack in two ways: 1) the visitor sends a malicious access request to a device that exceeds its access rights; 2) the visitor sends a large number of false access requests to the system. Since the BC-BLPM needs to execute the smart contract implemented according to the multi-level security access control policy to output the decision result for the access request, the first attack mode cannot afford effect. In the second mode, since the visitor needs to submit the access time variable T_{AC} in the access request, our model will judge the variable according to the access time threshold corresponding to the clearance level of the subject. Once the threshold is exceeded, the access request will be discarded, which greatly reduce the possibility of DoS attacks.

5.4 Performance Evaluation

As a blockchain-based implementation, the performance of BC-BLPM may be limited by the number of nodes, secure transport protocols, transaction request rates, concurrency rates, and inter-domain transactions, etc. Here we set the above parameters to study the effect of the relationship between the various parameters on the performance of BC-BLPM.

Figure 12 shows the impact of different number of nodes on system throughput. We first set up a test case containing a single client node and multiple nodes, using Hyperledger-Caliper to provide 5000 transaction request packets (such as "read-only" access requests) for the prototype system, and test the variation trends of system throughput for different node counts. It can be obtained from the experimental results that as the number of nodes increases, the system throughput generally shows an upward trend. On the other hand, from the perspective of transaction consistency check, the increase in the number of nodes means that the number of participating nodes in each round of consensus increases. Taking the PBFT protocol as an example, the access domain consisting of 9 nodes requires at least 7 honest nodes to participate in transaction verifying, while 24 nodes require at least 17 honest nodes. This change decentralizes the transaction verification process at the expense of the efficiency of the system operation, effectively avoids the problem of single point failure, and improves the fault tolerance of transaction verification.

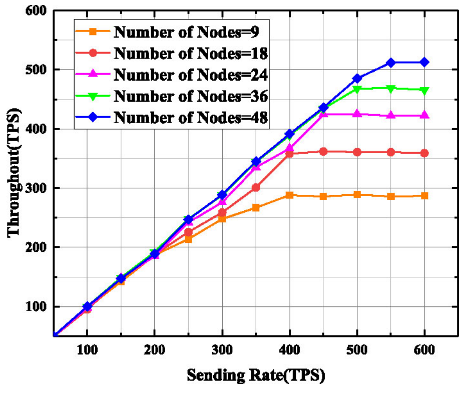


Figure 12. *System throughput changes under different node counts.*

In the actual operation of our prototype, a common scenario is that multiple client nodes issue access requests to network resources in parallel. Here we establish a 9-nodes access domain, and provide 5000 client-wide transaction request packets for each client. The experimental results are shown in Figure 13 and Figure 14. Among them, Figure 10 shows the trend of the throughput of different transaction flow rates with the increase of the client nodes. In this scenario, the output of each set of experiments reaches the maximum value in the interval of input 10 to 100 respectively, and the input range of 800 to 1000 shows a downward trend. It can be seen from the results of Figure 12 that as the number of concurrent requests of the client increases, the system delay and the transaction hit ratio also change, and the peak value is reached when the number of client nodes is equal to 1000, which directly leads to the deterioration of the system throughput.

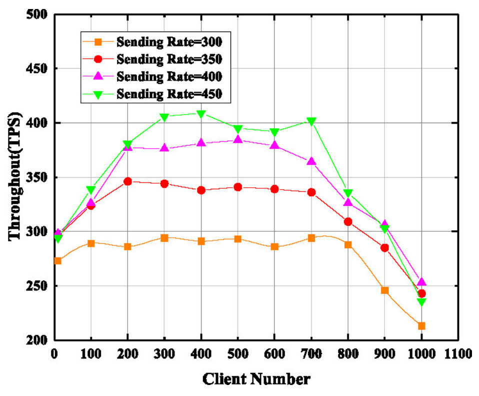


Figure 13. *System throughput changes under different client numbers.*

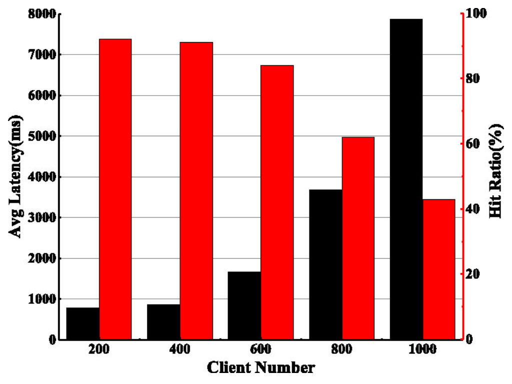


Figure 14. *System average latency and transaction hit ratio change under different client numbers.*

The data transfer process across different access control domains undoubtedly have an impact on the performance of our prototype. We demonstrate this effect through a simple experimental scenario: setting up 1 client node and 2 different access domains with 9 nodes each, providing 5000 access transaction request packets for each client. In addition, on the basis of keeping the total number of transaction requests unchanged, different proportions of inter-domain data transfer requests are randomly inserted. The experimental results are shown in Figure 15. It can be seen that under the premise of the transaction sending rate unchanged, as the proportion of inter-domain transactions increases, the overall system throughput drops significantly. This is because the delivering nodes additionally generate a transaction block for the target domain when processing the cross-domain transaction. Assuming that the transmission distance from the delivering nodes to the gateway nodes of different access domains are equal, which results in the necessity for two access domains to perform the correctness verification of the block almost simultaneously. This incurs additional time overhead.

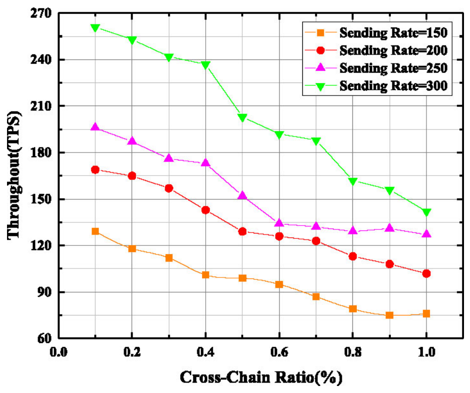


Figure 15. *System throughput changes under different inter-domain (cross-chain) Transaction ratios.*

The experimental results show that the performance indicators of BC-BLPM are limited by the number of nodes, transaction request rate, transaction concurrency rate and cross-domain transactions. Therefore, in the actual production environment, it is necessary to comprehensively consider the above factors according to the actual system construction requirements, such as increasing the trustworthiness of transaction verification by increasing the number of nodes in the access domain, or improving the system throughput by controlling the request rate and the number of concurrent

Table 4. *Scheme comparison.*

	VR-BLP [12]	CPCE/MLS [11]	MLS-AC [41]	Ref. [42]	BC-BLPM
Decentralization	NO	NO	NO	NO	YES
Auditability	NO	YES (with terminal controller)	NO	YES (with access record center)	YES (with Block-based public ledger)
Scalability	YES	NO	NO	NO	YES
Environment	IoT	Pervasive computing environment	Mobile devices	Universal multi-level security environment	Universal multi-level security environment
communication and computational overhead	Medium	Low	Medium	Unknow	High
Whether to support inter-domain operations?	YES	NO	NO	YES	YES

interactions.

5.5 Comparison with Existing Schemes

In some high-confidence scenarios, especially in government departments and enterprises that put information security issues first, multi-level security issues have been the focus of researchers' attention. BC-BLPM is a multi-level secure access control model using blockchain technology, which has the advantages of decentralization, auditability and scalability. By comparing with the researches of the MLS system in recent years, the comparison results are shown in Table 4. It can be seen that compared with the non-decentralized MLS system, the biggest advantage of the proposed architecture is that it can be audited through distributed deployment and blockchain-based management. Although there are disadvantages in commu-nication and computational overhead (inter-domain data transfer, multi-client transaction request and other operations generate additional time overhead, affecting the system throughput), the characteristics of decentralization determine that the architecture can provide reliable MLS-based access control services in an untrusted network environment, which is critical for multi-level security systems.

VI. CONCLUSION

We focus on the centralized single-point failure problem of traditional MLS systems, combine BLP-based security policy primitives and blockchain technology, propose a blockchain-based multi-level security access control model BC-BLPM and implement decentralized multi-level secure mechanism. Based on the principle of confidentiality, integrity and availability of multi-level security, the model further introduces attributes such as auditability and scalability. We use "multi-blockchain" architecture to divide the network into different access domains, so that the resource objects of different organization or departments are logically isolated from each other, improving the efficiency of resource maintenance. Besides, the integrity of the data stored in sidechains is maintained by all nodes in the domain and cross-domain data interaction is permitted. The BLP-based access control policy execution interface is designed by using blockchain smart contract, so that any access-requesting subject must automatically and forcibly execute the policy, thereby further improving resource security. The experimental and analytical results show that the model has high reliability in terms of execution performance and system performance, and has the ability to deploy in the actual production environment. In the future, we consider studying the dynamic adjustment of the security label to further enhance the security and flexibility of our model.

References

[1] D. McCullough, "Specifications for multi-level security and a hook-up," in *1987 IEEE Symposium on Security and Privacy*. IEEE, 1987, pp. 161–161.

[2] D. E. Bell and L. J. LaPadula, "Secure computer systems: Mathematical foundations," MITRE CORP BEDFORD MA, Tech. Rep., 1973.

[3] K. J. Biba, “Integrity considerations for secure computer systems,” MITRE CORP BEDFORD MA, Tech. Rep., 1977.

[4] D. D. Clark and D. R. Wilson, “A comparison of commercial and military computer security policies,” in *1987 IEEE Symposium on Security and Privacy*. IEEE, 1987, pp. 184–184.

[5] D. F. Brewer and M. J. Nash, “The chinese wall security policy.” in *IEEE Symposium on Security and Privacy*, vol. 1989. Oakland, 1989, p. 206.

[6] N. Zhang, M. Li, *et al.*, “Mushi: Toward multiple level security cloud with strong hardware level isolation,” in *MILCOM 2012-2012 IEEE Military Communications Conference*. IEEE, 2012, pp. 1–6.

[7] N. Virvilis, D. Gritzalis, *et al.*, “Trusted computing vs. advanced persistent threats: Can a defender win this game?” in *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*. IEEE, 2013, pp. 396–403.

[8] N. Szabo, “Smart contracts,” http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html, 2006.

[9] D. E. Bell and L. J. La Padula, “Secure computer system: Unified exposition and multics interpretation,” MITRE CORP BEDFORD MA, Tech. Rep., 1976.

[10] W. Zeng, M. Koutny, *et al.*, “Formal verification of secure information flow in cloud computing,” *Journal of Information Security and Applications*, vol. 27, 2016, pp. 103–116.

[11] Z. Tan, D. Liu, *et al.*, “Implementation and performance analysis of multilevel security system in pervasive computing environment,” *The Journal of Supercomputing*, vol. 66, no. 3, 2013, pp. 1243–1259.

[12] J. Dong, H. Zhu, *et al.*, “Task-oriented multilevel cooperative access control scheme for environment with virtualization and iot,” *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[13] L. Freitas and P. Watson, “Formalizing workflows partitioning over federated clouds: multi-level security and costs,” *International Journal of Computer Mathematics*, vol. 91, no. 5, 2014, pp. 881–906.

[14] Y. Shen and L. Xiong, “Lattice based blp extended model,” in *2009 Second International Conference on Future Information Technology and Management Engineering*. IEEE, 2009, pp. 309–312.

[15] S. Veloudis and N. Nissanke, “A novel permission hierarchy for rbac for dealing with sod in mac models,” *The Computer Journal*, vol. 59, no. 4, 2016, pp. 462–492.

[16] Z. Tang, X. Ding, *et al.*, “A self-adaptive bell–lapadula model based on model training with historical access logs,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, 2018, pp. 2047–2061.

[17] S. Nakamoto and A. Bitcoin, “A peer-to-peer electronic cash system,” *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, vol. 4, 2008.

[18] S. Raju, S. Boddepalli, *et al.*, “Identity management using blockchain for cognitive cellular networks,” in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[19] N. Alexopoulos, J. Daubert, *et al.*, “Beyond the hype: On using blockchains in trust management for authentication,” in *2017 IEEE Trustcom/BigDataSE/ICESS*. IEEE, 2017, pp. 546–553.

[20] A. Dorri, S. S. Kanhere, *et al.*, “Blockchain for iot security and privacy: The case study of a smart home,” in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom workshops)*. IEEE, 2017, pp. 618–623.

[21] S. Alansari, F. Paci, *et al.*, “A distributed access control system for cloud federations,” in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 2131–2136.

[22] L. S. Sankar, M. Sindhu, *et al.*, “Survey of consensus protocols on blockchain applications,” in *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE, 2017, pp. 1–5.

[23] R. S. Sandhu, “Role-based access control,” in *Advances in computers*. Elsevier, 1998, vol. 46, pp. 237–286.

[24] V. C. Hu, D. Ferraiolo, *et al.*, “Guide to attribute based access control (abac) definition and considerations (draft),” *NIST special publication*, vol. 800, no. 162, 2013.

[25] R. Sandhu and J. Park, “Usage control: A vision for next generation access control,” in *International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security*. Springer, 2003, pp. 17–31.

[26] G. Zyskind, O. Nathan, *et al.*, “Decentralizing privacy: Using blockchain to protect personal data,” in *2015 IEEE Security and Privacy Workshops*. IEEE, 2015, pp. 180–184.

[27] A. Ouaddah, A. Abou Elkalam, *et al.*, “Fairaccess: a new blockchain-based access control framework for the internet of things,” *Security and Communication Networks*, vol. 9, no. 18, 2016, pp. 5943–5964.

[28] M. Ma, G. Shi, *et al.*, “Privacy-oriented blockchain-based distributed key management architecture for hierarchical access control in the iot scenario,” *IEEE Access*, vol. 7, 2019, pp. 34 045–34 059.

[29] J. P. Cruz, Y. Kaji, *et al.*, “Rbac-sc: Role-based access control using smart contract,” *IEEE Access*, vol. 6, 2018, pp. 12 240–12 251.

[30] S. Wang, Y. Zhang, *et al.*, “A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems,” *IEEE Access*, vol. 6, 2018, pp. 38 437–38 450.

[31] A. Azaria, A. Ekblaw, *et al.*, “Medrec: Using blockchain for medical data access and permission management,” in *2016 2nd International Conference on Open and Big Data (OBD)*. IEEE, 2016, pp. 25–30.

[32] D. D. F. Maesa, P. Mori, *et al.*, “A blockchain based approach for the definition of auditable access control systems,” *Computers & Security*, vol. 84, 2019, pp. 93–119.

[33] X. Nan, “Identity authentication based on cpk,” *National Defense Industry Press, Beijing*, 2006, pp. 57–58.

[34] D. D. F. Maesa, P. Mori, *et al.*, “Blockchain based access control,” in *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, 2017, pp. 206–220.

[35] “Hyperledger fabric v1.1.0,” https://github.com/hyperledger/fabric/tree/release-1.1, 2018.

[36] Docker, “Docker enterprise application container platform

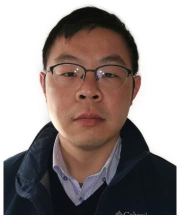
— docker," https://www.docker.com, 2013.
[37] "Apache kafka," https://kafka.apache.org, 2017.
[38] "Goleveldb," https://github.com/syndtr/goleveldb, 2018.
[39] "Hyperledger-caliper," https://github.com/hyperledge-archives/caliper, 2018.
[40] I. Bentov, C. Lee, *et al.*, "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, 2014, pp. 34–37.
[41] Z. Zulkefli, M. M. Singh, *et al.*, "Advanced persistent threat mitigation using multi level security–access control framework," in *International Conference on Computational Science and Its Applications*. Springer, 2015, pp. 90–105.
[42] H. Zhang, J. Wang, *et al.*, "An access control model for multi-level security in multi-domain networking environments," in *2017 9th International Conference on Modelling, Identification and Control (ICMIC)*. IEEE, 2017, pp. 809–814.

Biographies

Xiang Yu is a lecturer of the National University of Defense Technology, and also a PhD candidate at College of Computer Science of National University of Defense Technology. He received the B.E degree and M.E degree from Electronic Engineering Institute in 2007 and 2012. His research interests include blockchain technology, trusted computing, and cloud computing security. Email: yuxiang17@nudt.edu.cn

Zhanxiang Shu is currently working toward the M.E degree at National University of Defense Technology. He received the B.E degree from Electronic Engeerning Institute in 2017. His research interests include blockchain and cyberspace security. Email: szxstudy@sina.com

Qiang Li is a professor of the National University of Defense Technology. He received the M.E degree from Southwest Jiaotong University. His research interests include cyberspace security, software engineering, and software security testing. Email: lychfeei@163.com


Jun Huang is an associate professor at National University of Defense Technology. He received a B.E degree in automation, a M.E degree in military communication, a PhD in information security all from Electronic Engineering Institute. His research interests are in wireless network security. Email: 151244199@qq.com