IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 25, 2020, accepted August 12, 2020, date of publication August 24, 2020, date of current version September 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3018783

A Mechanism to Resolve the Unauthorized Access Vulnerability Caused by Permission Delegation in Blockchain-Based Access Control

JINSHAN SHI, RU LI, (Member, IEEE), AND WENHAN HOU
Inner Mongolia Key Laboratory of Wireless Networking and Mobile Computing, Hohhot 010021, China
College of Computer Science, Inner Mongolia University, Hohhot 010021, China

Corresponding author: Ru Li (csliru@imu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61862046, in part by the Inner Mongolia Autonomous Region Science and Technology Achievements Transformation Project under Grant CGZH2018124, and in part by the Science and Technology Program of Inner Mongolia Autonomous Region under Grant 2019GG376.

ABSTRACT Permission delegation in access control provides the subject with a second method to obtain object permissions in addition to permission granting. It is especially applicable when the owner and manager of the object are inconsistent. With the development of the Internet of Things, there are more and more scenes where object owners and managers are inconsistent, but the research on permission delegation in access control based on blockchain is not perfect. Therefore, once implemented in these blockchain-based access control algorithms, the permission delegation tends to have an unauthorized access. Based on the analysis of the causes for the unauthorized access vulnerability, this paper proposes a token-constrained permission delegation algorithm (TCPDA), which converts the access control policy corresponding to permissions into constraints for permission use, embeds the constraints in the permission token, and forms constraints on the transfer of tokens. Only subjects that meet the constraint conditions can receive tokens, thereby solving unauthorized access vulnerability caused by permission delegation. Since not all access control models can transform strategies into constraints and integrate them into blockchain tokens, this paper also proposes a permission delegation algorithm for decision-making entities to make desirable decisions. Finally, the security analysis shows that the two proposed schemes can overcome the unauthorized access vulnerability caused by permission delegation, and the algorithm performance is analyzed through experiments.

INDEX TERMS Access control, blockchain, permission delegation, unauthorized access vulnerability.

I. INTRODUCTION

As a distributed and decentralized computing and storage framework, the blockchain is applied to the access control model, which addresses the single point of failure and the security and trust issues among multiple agencies. These problems are mainly caused by the centralized design in the access control model.

The main reason for the single point of failure in access control is that it has a centralized authorized decision-making entity. Once this centralized authorized decision-making entity fails, it will lead to access control failure, so Ouaddah *et al.* tries to deploy this authorized decision-making entity on the blockchain [1]. The decentralized features of blockchain are employed to solve this problem. When there is a cross-domain access, multiple agencies are required to collaborate each other to achieve access control. At this stage, trust issues among multiple agencies will occur. The previous solution tends to assume that there is a third-party trusted agency acting as an authorized decision-making entity. Multiple privacy breaches in recent years indicate that third-party institutions are not credible [2], so blockchain is used to replace third-party trusted institutions. Access control is deployed on the alliance blockchain built by multiple institutions, and the blockchain maintained by multiple institutions is used to achieve trust issues among multiple institutions. In addition, the immutable and traceable features of blockchain data solve the problem of tracking access control information leakage.

Since the blockchain was originally proposed as the underlying architecture of Bitcoin, in the study of combining blockchain with access control, Zyskind *et al.* abstracted

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

those permissions into tokens in the Bitcoin blockchain and controlled permissions through transactions [3]. Later, when implemented in the blockchain, smart contracts were employed to abstract permissions to non-homogeneous currencies in Ethereum or other blockchains [4]. As the blockchain platform originally served the financial industry, the circulation was naturally supported the permissions represented by the tokens and non-homogeneous currencies mentioned above. Researchers have designed the specific transaction specifications or the smart contracts to grant permission requests features.

Permission delegation refers to the behavior that an entity passes to another a specific permission after obtaining it. Adding a permission delegation module to access control enriches the functions of the latter and provides users with another way to obtain permissions in addition to requesting authorization. It is especially applicable when the owner and manager of the object are inconsistent. Therefore, in 2016, the blockchain-based access control proposed by Ouaddah *et al.* implemented the permission delegation function by designing specific transaction specifications [1]. Among them, users can pass the transaction permission to others via this transaction specification, but the process is unknown to the permission owner. Nayak *et al.* proposed a delegation of permission based on blockchain in a cloud environment. And after receiving the entrusted permission, users cannot delegate it [5]. Xu *et al.* proposed another blockchain-based permission delegation scheme, discussed multi-hop delegation, and set the delegation depth [6], [7]. Le *et al.* considered the validity period of the permission and stipulated that the validity period of the permission received by the receiver should not exceed the validity period of the sender [8]. Pal et al considered the issue of privilege escalation in the privilege delegation scheme they designed, to avoid the possibility of increased privilege in the delegation process [9]. Although these studies have improved the security of permission delegation in blockchain-based access control from different perspectives, in their scheme, the permission delegation only needs to delegate the participation of the initiator and the permission receiver, but not the resource owner.

Therefore, in the current blockchain-based access control model, the delegation may cause permission leakage, resulting in an unauthorized access. Moreover, as it is to be completed by transactions or smart contracts, the current permission delegation requires both the consent of the user and that of the receiver, and the blockchain acts as a trusted third party as a witness and recorder. However, the owner of the permission does not participate in the authorization process, that is, when the permission is passed, the recipient does not pass the access control policy verification, so the recipient may not be the legal user of the permission. If the delegatee cannot obtain the permission, after the management agency makes a decision on the access control policy, but obtains the permission via delegation, this thereafter leads to an unauthorized access.

The motivation of this paper is as follows:

1) To the best of our knowledge, in the current blockchain-based access control, unauthorized access vulnerabilities caused by permission delegation do exist but no one has paid attention to it. Therefore, one of our motives is to propose this kind of unauthorized access vulnerability caused by permission delegation in access control based on blockchain.
2) Since this unauthorized access vulnerability was discovered, we then propose corresponding solutions to the vulnerability.

This paper proposes token-constrained permission delegation algorithm (TCPDA), which converts the access control policy corresponding to the permission into the constraint conditions used by the permission, and then embeds the constraint conditions in the permission token used to directly determine the recipient's legitimacy to address unauthorized access vulnerabilities in permission delegation. Since not all access control models can transform strategies into constraints and integrate them into blockchain tokens, this paper also proposes a permission delegation scheme for decision-making by policy decision point (PDP), which may as well be called PDP re-decision permission delegation algorithm (PRPDA). In light of this, our contributions in this paper are multifold:

1) We present a model of unauthorized access vulnerability and analyze the causes of the vulnerability.
2) We present a permission delegation scheme called TCPDA to solve the above vulnerability. And present a supplementary scheme called PRPDA.
3) We use the colored petri nets (CPN) to formally model the above two schemes and prove the safety of these two schemes.
4) We have implemented the solution and conducted experiments for the evaluation of computation time and storage costs.

The rest is organized as follows: Section II describes the related work of the study. Section III introduces the permission delegation scenarios, permission delegation model and unauthorized access vulnerabilities caused by permission delegation. Section IV proposes two permission delegation algorithms to resolve unauthorized access vulnerabilities. Section V analyzes the security of the proposed algorithm. Section VI breaks down the performance of the proposed algorithm through experiments. Section VII offers a summary and introduces our next work plan.

II. RELATED WORK

Combining blockchain and access control can solve the single-point failure problem and the security and trust problems between multiple institutions in the centralized access control model due to the centralized design. For the single point of failure, Jemel *et al.* used the blockchain for user legitimacy checks in CP-ABE access control [10]; Outchakoucht *et al.* used the blockchain for operations such

as granting, using, and circulating access permissions in SmartOrBAC [11]. To address security and trust issues among multiple organizations, Cruz *et al.* used blockchain as a trusted third party to solve the cross-organizational access control problem in Role-Based Access Control(RBAC) [12]; Alansari *et al.* used blockchain to prevent user identity attributes and access control policies in Attributes Based Access Control(ABAC) from being modified by malicious users [13]; Maesa *et al.* publicly store policies and permission exchanges on the blockchain to prevent one party from fraudulently rejecting the rights granted by the policies [14].

After solving the existing defects of access control, the researchers studied the access control model applicable to the new Internet of Things (IoT) scenarios. Zhang *et al.* proposed a IoT access control framework using smart contracts, but this access control is static and does not apply to dynamic IoT [15]. Islam *et al.* proposed a dynamic ABAC-based access control using blockchain for the IoT [16]. Zhang *et al.* also proposed a blockchain access control based on ABAC, in this scheme, a verifiable collaboration mechanism is designed to meet the needs of controlled access authorization in emergencies [17]. Guo *et al.* also proposed ABAC-based blockchain access control, but users have multiple attributes in their schemes and need to obtain them from multiple attribute authorities [18]. Rajput *et al.* proposed a blockchain-based access control system in the context of intelligent medical treatment to protect patient privacy [19].

In these researches on access control, the complexity of the IoT scenario is not fully considered. However, in the real IoT, there are many scenarios where the owner and the manager of a large number of objects are inconsistent. Taking the smart medical scene as an example, when the attending doctor A of a patient encounters an unsolvable problem and needs to refer the patient to the doctor B of another hospital, the patient's electronic medical record needs to be passed from A to B. At this time, the simplest method is to delegate permission, but there is no permission delegation function in the above scheme.

Since the permission delegation model in access control was proposed, a large number of researchers have done some in-depth researches for different purposes [20]. Permission delegation is used by researchers in different access control models. It provides users with more flexible accesses to permissions and yet brings security risks to the access control model as well. There are two main types of security risks caused by permission delegation. One is the unauthorized access, which means illegal users can obtain permissions from other users through permission transfer; the other is the data privacy leakage. This privacy leakage is not caused by unauthorized access, but by illegal use of data by legitimate users. In response to the former risk, Wainer *et al.* proposed that permissions must satisfy monotonicity during their delegation, that is, the permissions that are passed will not increase with the transfer [21]; Anggorojati *et al.* proposed a permission delegation model, in which the permission re-decides whether to grant the recipient when the permission is passed, in order to solve the problem of permission leakage caused by permission delegation [22]; Rabehaja *et al.* proposed that permission delegation must fulfil two basic conditions, namely,the user has something to delegate and the right to delegate [23]. As for the latter risk of data privacy disclosure, Moniruzzaman *et al.* designed an additional privacy model, which considers privacy policies when delegating decisions, and sets data usage policy constraints through the privacy model [24].

Prior to the introduction of access control into the blockchain, security issues caused by permission delegation have been solved by researchers. But with the combination of access control and blockchain, the access control model has undergone adaptive changes. Thus, permission delegation triggers new security problems to the access control model.

The introduction of the blockchain provides a relatively trusty computing and storage platform for access control. In the research on the combination of blockchain and access control, most researchers store permission credentials in the blockchain, and grant user permissions by passing permission credentials through transactions or smart contract distribution. At present, the permission delegation after the introduction of blockchain can be divided into two categories according to the in-depth research.

The first category is to propose permission delegation as a module in access control. The researchers including Ouaddah *et al.* [1], Tapas *et al.* [25], Alcarria *et al.* [26], Xu *et al.* [27], etc. had permission delegation function in the access control model they proposed, but they did not carry out detailed analysis and research on permission delegation.

The second is an in-depth study of permission delegation in the access control of the blockchain, which proposes a model with its own characteristics. Ali *et al.* studied the application of permission delegation in blockchain-based access control in the IoT scenario, and divided the permission delegation into an event-based one and a query-based one [28]. But this study does not focus on the security issues brought about by permission delegation. The model proposed by Xu *et al.* supports multi-hop delegation and tree-like delegation, and formalizes the delegation relationship as a delegation tree. It also specifies that an entity can only be delegated once, that is, any entity can only appear once in the delegation tree. At the same time, the delegation depth is set for the permission token, which limits the number of permission delegations [6], [7]. Pussewalage *et al.* proposed an access control delegation scheme in an e-health environment [29], in which the principal can limit the length of the delegation chain. Unlike the literature [6], there is no limit to the number of entities that can be delegated. Nayak *et al.* proposed a permission delegation in the cloud environment, which restricts users from being able to delegate permissions again after receiving them, for permissions can only be passed once [5]. Le *et al.* considered the time factor in the permission delegation model, in which the permission has a validity period, and stipulates that the period for the

receiver cannot exceed that for the sender [8]. And Pal *et al.* proposed an asynchronous permission delegation model under the blockchain, in which tokens are implemented via events, while considering the issue of privilege escalation [9]. However, in the delegation of permission, the recipients' legitimacy is not checked through the access control policy, and there are still loopholes in the unauthorized access.

In summary, in the researches of permission delegation after the combination of access control and blockchain, the security considerations brought by the introduction of the blockchain are insufficient, although researchers have proposed multiple permission delegation models.

III. PERMISSION DELEGATION MODEL AND UNAUTHORIZED ACCESS VULNERABILITY

In this section, we introduce the scenario of permission delegation in blockchain-based access control, and give a model of permission delegation. Then, we proposed an unauthorized access vulnerability based on the current permission delegation model. Finally, the threat model of permission delegation is proposed.

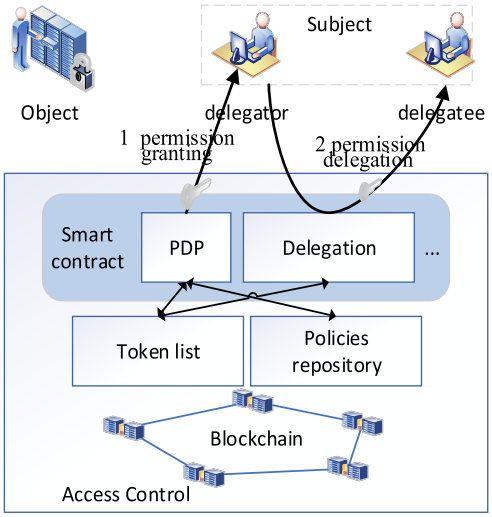


FIGURE 1. Permission delegation scenario in the access control model based on blockchain.

A. PERMISSION DELEGATION SCENARIO DESCRIPTION

This paper is meant to center around the permission delegation but not the other functions of blockchain-based access control. The permission delegation scenario is shown in Fig. 1. The access control PDP is deployed in the blockchain, and permission granting and delegation are implemented by smart contracts. The blockchain here is the consortium blockchain architecture. The resource requester requests the access permission of the resource from the access control system. After the PDP deployed in the blockchain makes a decision, the permission token is granted to the resource requester, who passes the token to the delegatee.

Definition 1 delegator: An access requester who initiates a permission delegation. Via permission delegation, a delegator transfers the permissions he owns to others. The delegator has the transferable permissions and the right to initiate permission delegation. After the permission delegation is completed, the delegator loses the delegated permission.

Definition 2 delegatee: The resource requester who receives the token and obtains the permission during the permission delegation process.

The method by which the resource requester obtains tokens in the above scenario is called permission grant, as shown in Step 1 in the Fig. 1. Assumption: The owner of the object has uploaded his own access control strategy to the access control system in the blockchain.

The process of granting permissions is as follows: (1) Request access, the subject initiates an access request to the object, the subject needs to send a request access message *RequestAccess(s, r, o, {o.p$_1$,...o.p$_i$})* to the access control system, the content of the message is: subject *s*, the object owner *r*, the object *o*, the requested permissions *{o.p$_1$,...o.p$_i$}*. (2) Make a decision, after the PDP contract deployed in the blockchain receives the subject's request access message, the PDP makes a decision according to the access control policy set by the object. If access is allowed, the permission token (*pt*) of the object is granted to the subject. Otherwise, a rejection message is returned. (3) Access object, after the subject obtains the *pt* of the object, the object can be accessed by the *pt*. The subject needs to send the access object message *GrantAccess (s, r, o, pt, {o.p$_1$,...o.p$_i$})* to the access control system in the blockchain. The access control system generates an access event after verifying that the message is legal, allows the subject to access and records the access event in the blockchain.

The method by which the delegatee obtains the token is called permission delegation, which is Step 2 in the Fig. 1. Assumption: The delegator has obtained the pt of the object.

The permission delegation process is as follows: (1) Request delegation, delegatee requests the object's access permission *pt* from the delegator who has obtained the object's target permission; (2) The delegator himself decides whether to delegate *pt* to delegatee, and if it agrees to delegate, proceed to the next step; (3) The delegator sends a message of permission delegation to the PDP. The PDP makes a decision on this delegation. If it agrees, *pt* is successfully delegated to delegatee, after which delegatee can access the object through the *pt* obtained from the delegator. The first two steps of permission delegation have nothing to do with the access control system, so the details of how the permission delegation decision is made in the third step of the access control system will be described later.

B. PERMISSION DELEGATION MODEL

The required symbols in the permission delegation model and their meanings are shown in Table 1.

A typical permission delegation model is $PDM = (o,p,s_{from},s_{to})$, where o represents the object corresponding to the permission in the permission delegation, and $o \in O$. p represents the delegated permission, which is usually

TABLE 1. Symbols, relationships, and meanings in the permission delegation model.

SYMBOL	RELATIONSHIPS	MEANINGS
S	$S=\{s_1,s_2,\ldots,s_n\}$	Subject set in access control
O	$O=\{o_1,o_2,\ldots,o_n\}$	Object set in access control
P	$P=\{p_1,p_2,\ldots,p_n\}$	A set of permissions that an object can grant to a subject
S_{legal}	$S_{legal}\subseteq S$	Subject set that can access the object according to the object's access control policy
p	$p\in P$	Object permission
s_{from}	$s_{from}\in S_{legal}$	Delegator
s_{to}	$s_{to}\in S$	Delegatee

abstracted as a permission token in a blockchain-based access control model. This token can represent multiple permissions. s_{from} and s_{to} are the participants in the permission delegation process, and the permissions in the model are passed from s_{from} to s_{to}. It is assumed here that the permission p obtained by s_{from} is determined and granted through an access control policy, hereby $s_{from} \in S_{legal}$. In the existing permission delegation model, it is considered that the object trusts the subject s_{from}, so the permission is granted to the subject s_{from}, and the subject s_{from} trusts s_{to}, hence s_{from} passes p to s_{to}, then we can get $s_{to} \in S$.

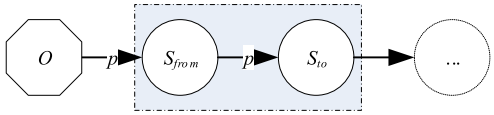


FIGURE 2. Permission delegation model.

The permission delegation model is shown in Fig. 2, in which the object is represented by an octagon; the subject, a circle; and the granting and transferring of the permission, arrows. This paper focuses on the process in which s_{from} passes the permission p to s_{to} in the model, shown as the shaded part in the figure. When permission is passed multiple times, the legality of s_{from} cannot be guaranteed except for the first time.

At the same time, there is a default condition here. After obtaining the permission token, the subject can access only the permission credentials when accessing the object without the need to verify the validity again.

This paper is centered on the unauthorized access vulnerabilities in the process of permission delegation. For clarity of description, this paper only addresses single-hop permission delegation, but the solution proposed in this article is also applicable to multi-hop one.

C. UNAUTHORIZED ACCESS VULNERABILITY

The unauthorized access vulnerability is shown in the following permission delegation process: in the above permission delegation model, the access permission p of the object o is granted to the subject that meets the access control policy. These subjects like the above are called the legitimate subjects as $S_{legal} = \{s_i, s_{i+1},\ldots,s_j\}$, where $S_{legal} \subseteq S$. It should be noted that the object does not pre-specify the members in S_{legal}. These legitimate users are screened out by the access control policy of the object permission. They have the right to request the access permission of the object from the access control system, but they may not have obtained this permission, the elements in S_{legal} are not always the same. As shown in Fig. 3, the circle in the figure represents the subject. Some subjects get permission p, we call it s_{from}, and we can know that $s_{from} \in S_{legal}$. Then a permission delegation event occurs, and s_{from} passes permission p to s_{to}. The vulnerability occurs at this step. Thus, the permission delegation mechanism in access control only verifies that s_{from} and s_{to} agree to the permission p but does not verify whether s_{to} is an element in the set S_{legal}, so s_{to} may not satisfy the access control policy of permission p. But at this time p has been passed from s_{from} to s_{to}, so s_{to} has obtained permission p without satisfying the object o access control policy. Therefore, the delegation of permission caused an unauthorized access.

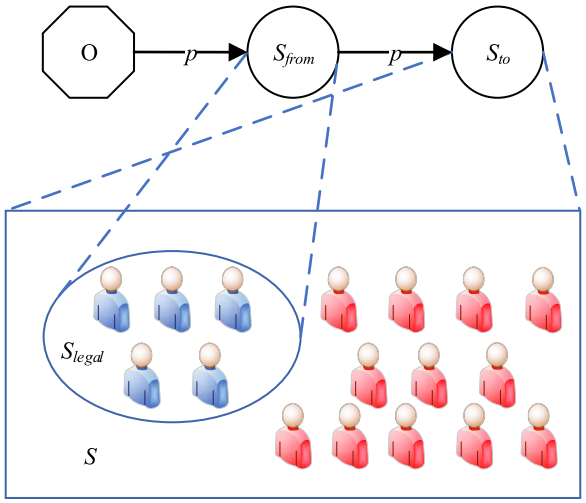


FIGURE 3. Schematic diagram of unauthorized access vulnerability in permission delegation.

This vulnerability is caused by the fact that the permission p is passed without consideration of whether s_{to} has permission to use p. In order to avoid this vulnerability, the access control model must ensure that any subject s that owns p belongs to S_{legal}.

Therefore, s_{to} must be verified during permission delegation. If $s_{to} \notin S_{legal}$ exists, unauthorized access occurs. Such permission delegation is illegal.

In the permission delegation model above, the transfer of permissions is essentially the transfer of trust. For example, an object trusts subject Alice, so the object grants permissions to subject Alice. Because subject Alice trusts subject Bob, subject Alice passes the object's permissions to subject Bob. But it is possible that the object neither knows Bob nor wants to grant Bob permission.

The current permission delegation model has an unauthorized access vulnerability. This is because it is unreasonable to directly apply a chain of trust to permission delegation. It can be simply understood as "I trust you, but I do not trust your friends." It is undesirable to pass permissions blindly to an unknown user in the security realm.

D. THREAT MODEL

The focus of our consideration is the unauthorized access vulnerability mentioned above, so this paper considers that the attacker uses the unauthorized access vulnerability proposed in Section III.C to attack the access control system to obtain the access permissions of the object. It is assumed here that except for the permission delegation module in the access control system, the attacker cannot legally obtain the permission of the target object by requesting a grant. The attacker's attacking method is to obtain the permission from the user who has obtained the target object permission, and these users cannot distinguish whether the delegatee is legal or not, so the permission may be delegated to illegal subjects during the permission delegation process. The security goal is that the object permissions will not be acquired by illegal subjects during the process.

IV. VULNERABILITY SOLUTIONS

In this section, the token-constrained permission delegation scheme is mainly elaborated. In addition, considering that not all access control methods can integrate policies into tokens, we propose the PDP re-decision permission delegation scheme as a supplement program.

A. TOKEN-CONSTRAINED PERMISSION DELEGATION SCHEME

The unauthorized access vulnerability caused by permission delegation is due to the fact that the permission receiver is not verified by the access control policy, so the idea of solving the vulnerability is to verify directly whether the recipient is authorized to use the permission during the delegation. In blockchain-based access control, permissions are usually granted to the subject in the form of a token, which is stored in the blockchain as a non-homogeneous token, and serves as a credential for the permission of the subject.

Therefore, when the permission is delegated, the idea is as follows of verifying whether the delegatee is authorized to use the permission: Convert the access control policy corresponding to the permission into the constraints for permission usage, and then to embed the constraints for permission usage into permission tokens. The passing of tokens constitutes a constraint chain, and only subjects that meet the constraints can receive tokens. For example, in a system that combines ABAC or RBAC with a blockchain, they can add permission access constraints with Roles or Properties to the token.

Therefore, the permission delegation scheme with constraints is shown in Fig. 4, where dele_SC is a permission delegation smart contract and token_list is a list of tokens in the access control system. Delegator passing his permission token to the delegatee needs the permission delegation contract (or module) in the blockchain to judge whether this transfer meets three basic conditions of determining (1) first whether the permission token is allowed to be passed, (2) then if the delegator is authorized to pass permissions, and (3) whether the delegatee is authorized to use the permission token.

1) DESIGN OF PERMISSION TOKEN STRUCTURE

Integrating the access control policy in the permission token requires designing a new token structure, which needs to include usage restrictions for the permissions of the token.

The permission token is expressed as $PT = f(s_i) \rightarrow \{o_j,\{p_1 \ldots p_n\},N,C\}$, where $f()$ is a mapping function from the subject to the permission token, implemented by the blockchain, indicating that the subject is the owner of the permission token. s_i is the subject in the access control, here represents the owner of the permission token. o_j represents the object corresponding to the permission token. $\{p_1 \ldots p_n\}$ represents the access permission of the object o_j represented by the token, where the permission can be multiple. N represents the features of the token, including transferability, valid time, transferable hops, etc. C represents the constraints formed by all the access strategies of $\{p_1 \ldots p_n\}$, and only subjects that meet these constraints can receive the token. Among them, C can be empty. When the permission is not allowed to pass or the access control policy of the permission cannot be integrated in the permission token, C is empty.

2) AN APPROACH TO POLICY INTEGRATION

When the subject requests the object's permission $\{p_1 \ldots p_n\}$, the object's PDP generates an access control policy (ACP) corresponding to $\{p_1 \ldots p_n\}$, and decides whether to grant permission $\{p_1 \ldots p_n\}$ to the subject. When the PDP determines that the subject can use $\{p_1 \ldots p_n\}$, it will generate a permission token PT in the blockchain. The fourth element C of the permission token PT needs to be generated by the ACP.

The description of the access control policy refers to the XACML policy language model. An ACP corresponding to a permission can be expressed as ACP $= fp(\{p_1 \ldots p_i\}) \rightarrow \{policy_1 \ldots policy_n\}$, where $policy_i$ is the policy for managing access permissions. A permission may have multiple policies. The policy is expressed as $policy = \{target, combination, \{rule_1 \ldots rule_m\}, \{responsibility\}\}$, where *target* is the object to which the policy is applied, which can be simply understood as the permission corresponding to the policy; *combination* represents a combination algorithm that combines multiple rules together. *rule* is the one corresponding to target. *responsibility* is a collection of responsibilities. The storage space occupied by a complete ACP is relatively large, so the ACP needs to be converted to a constraint C condition that is only for the current permissions. Therefore, the specific policy integration algorithm is shown in Algorithm 1. First, obtain the type of access control strategy used according to the object. If it is ABAC,

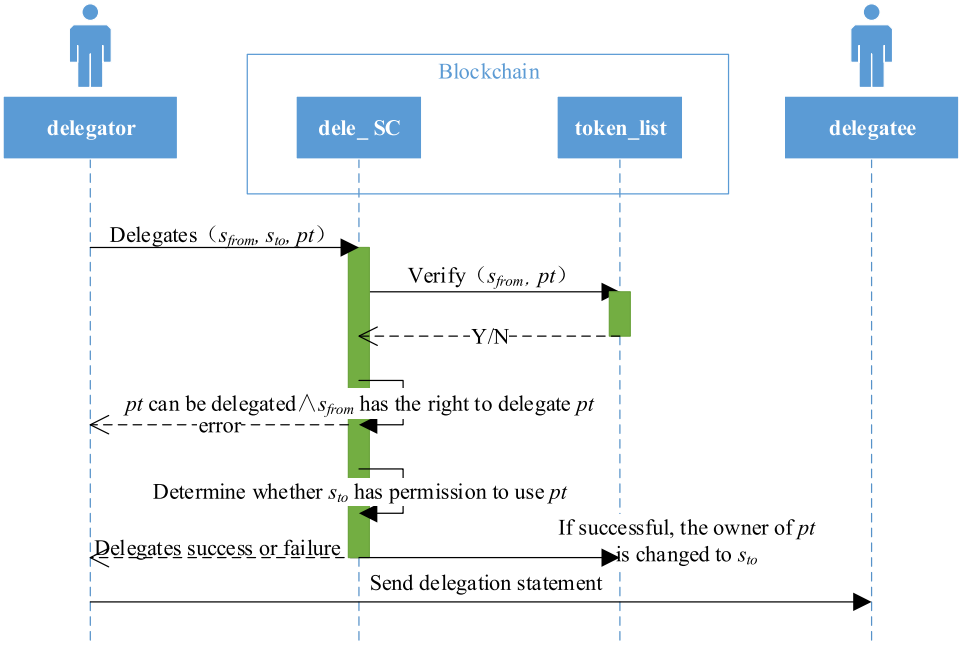


FIGURE 4. Token-constrained permission delegation scheme.

extract the ACP rules in ABAC from <subject attributes, object attributes, environment attributes, permission> into < subject attributes, environment attributes> and combine them into policies according to the *combination*, and finally combine all policies into a complete constraint C; if it is RBAC, extract all allowed roles, and then combine them into a complete constraint C.

3) ALGORITHM DESIGN

A more detailed delegation step can be expressed by Algorithm 2, The following three conditions need to be met during delegation: (1) First, it is necessary to determine whether the permission token is allowed to be passed, and here is to judge according to the N in pt. (2) Then it is necessary to determine whether the subject has the right to pass the permission token, here we need to verify whether the owner of pt is the subject s_{from} (3) It is also necessary to determine whether the delegatee has the right to use the permission token, and here we need to extract the constraint C in the pt, and then use C to determine whether the delegatee s_{to} can receive the pt. This algorithm is deployed in Smart contract *dele_SC*, and the message format and verification process are not reflected in the algorithm. The specific algorithm is as follows:

It should be noted that when the user updates the access control policy, the permission token becomes invalid immediately and needs to be reapplied.

This algorithm is advantageous in that the complexity of the algorithm is only *O(1)*. The permission delegation does not need to use the PDP to verify the validity of the delegatee. Instead, it directly determines whether the delegatee is legal according to constraint C in the permission token, which reduces the overhead of the decision. The average algorithm complexity is required for this decision to be executed by the PDP, which is *O (nlogn)* in most cases, and it is considerable for access control models with a large number of policies.

However, the disadvantage of this algorithm is that embedding the policy in the permission token tends to enlarge the permission token, which is one of the worst disadvantages for the blockchain. The specific impact is given in the experiment in Section VI of the experimental data and analytical results.

B. PDP RE-DECISION PERMISSION DELEGATION SCHEME

Not all access control methods can integrate policies into tokens, so the second solution is to make a decision on the legitimacy of the delegatee by the PDP when the permission is delegated. Only the legitimate delegatee can receive permissions.

The difference between this solution and the previous one is that the constraint condition C of the permission token pt is empty, and the delegation process needs the PDP to determine whether the delegatee has the right to use the token. The specific process is shown in Fig. 5. The delegator sends a request of permission delegation to the permission delegation smart contract dele_SC. After basic verification of the delegation request, dele_SC sends the delegatee and pt to the PDP smart contract PDP_SC. PDP_SC decides whether the delegatee is legal or not. PDP_SC looks up the corresponding policy in the access control policy set, and then judges whether the delegatee has the right to use pt according to the policy.

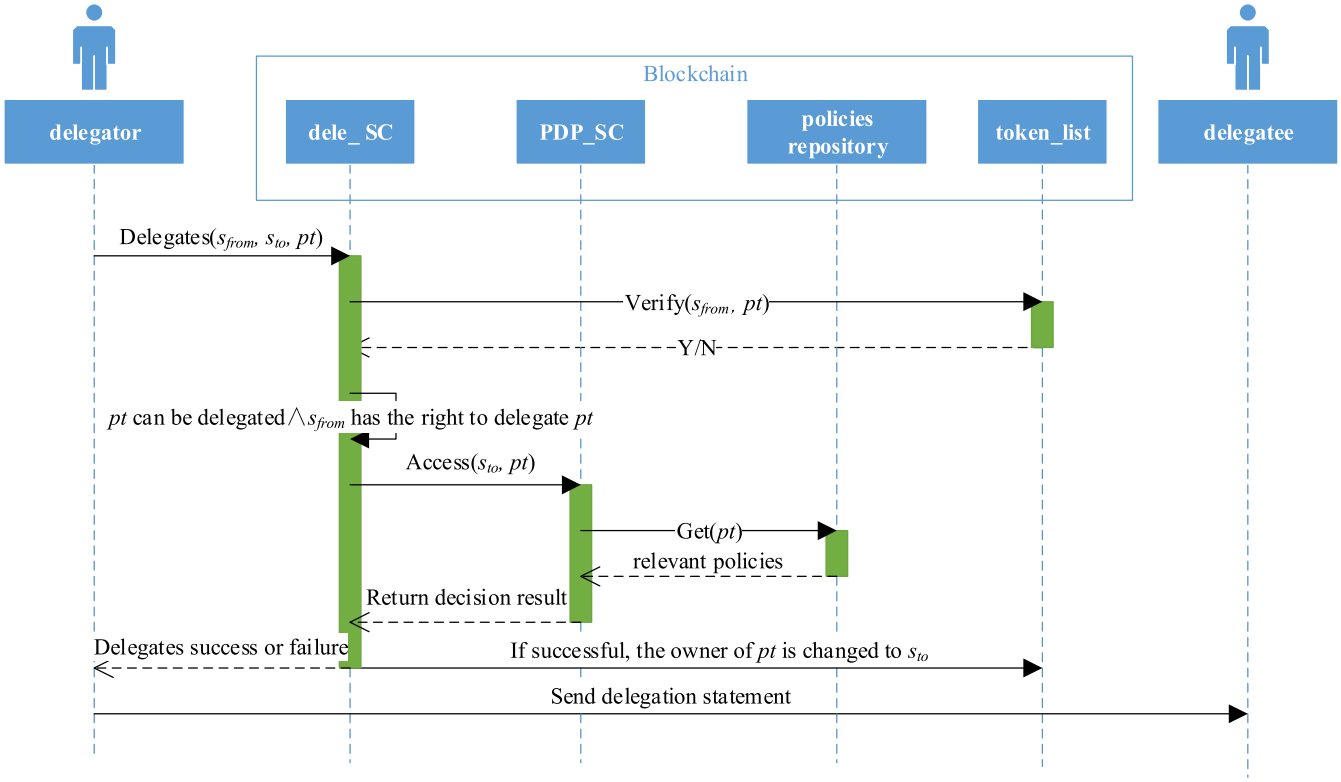


FIGURE 5. PDP re-decision permission delegation scheme.

The more detailed permission delegation steps are shown in Algorithm 3, where Step 5 is performed in the PDP. The PDP receives not the permission token, but the permission $\{p_1 \ldots p_n\}$ represented by the token. The PDP then determines whether the delegatee has the right to use $\{p_1 \ldots p_n\}$.

This is one of the simplest solutions. The advantage of Algorithm 3 is that it occupies less storage space on the blockchain, and the preciousness of the blockchain storage space will not be repeated here. Its disadvantage is that this algorithm has a high time complexity. In Step 5 of the algorithm, the PDP needs to make a decision, so it is necessary to find the policy corresponding to the permission $\{p_1 \ldots p_n\}$ or the permission corresponding to the delegatee. The minimum time complexity of its search algorithm is $O(nlogn)$. If there are multiple strategies, strategy combination is required.

V. SECURITY ANALYSIS

Both solutions proposed in this paper need to meet the threat model and security objectives.

Proposition 1: The permission delegation scheme with constraints and PDP re-decision permission delegation scheme proposed in this paper can counter the unauthorized access vulnerability caused by permission delegation in blockchain-based access control, and the object permissions will not be obtained by illegal delegatee during the permission delegation process.

Safety analysis. As for the permission delegation scheme with constraints, the access control strategy corresponding to the permissions is transformed into the constraints of permission usage, and then the constraints of permission usage are embedded in the permissions token. During the process, the permission delegation smart contract will parse the permission token, extract the access control strategy of the token, and then determine whether the delegatee is legal according to the strategy. Therefore, in the delegation process, the validity of the delegatee is verified through the constraints of the token to ensure that the object permission will not be delegated to illegal subjects.

With regard to the permission delegation scheme for PDP re-decision, in the permission delegation process, the delegation smart contract will send the delegated permission and delegatee to the PDP of the access control system, and the PDP will make the decision. The process of PDP decision-making is equivalent to the process of delegatee requesting permission, and then PDP returns the decision result to the delegation smart contract which in turn decides whether to complete the delegation based on the result. Therefore, in the permission delegation process, the legality of each delegatee is verified through PDP to ensure that the object permission will not be delegated to illegal subjects.

In order to prove whether the solution proposed in this paper meets the threat model and security objectives in Section III.D, it is necessary to formally model the solution, and then analyze whether it meets the security requirements

Algorithm 1 Policy Integration Algorithm

Input:
object o, permission $p[p_1 \ldots p_n]$
Output:
constraints C
1: $o.acst \leftarrow$ GetAccessControlStrategyType(o);
2: **IF**($o.acst ==$ ABAC) **THEN**
3: **FOR** $i \leftarrow 1$ to $p.lenght$ **DO**
4: $p_i.ACP \leftarrow$ GetACP(o, p_i);
5: **FOR** $j \leftarrow 1$ to $p_i.ACP.lenght$ **DO**
6: $p_i.ACP.policy_j \leftarrow$ GetPolicy($p_i.ACP$, j);
7: **FOR** $k \leftarrow 1$ to $p_i.ACP.policy_j.rule.lenght$ **DO**
8: ruleCom[$s.att$, $e.att$] $\leftarrow$ CompressionRules($p_i.ACP.policy_j.rule$);
9: **END**
10: CombinationRules(ruleCom, $p_i.ACP.policy_j.combination$);
11: **END**
12: **END**
13: $C \leftarrow$ combine compressed policies and simplify policies; return C;
14: **ELSE IF**($o.acst ==$ RBAC) **THEN**
15: **FOR** $i \leftarrow$1 to $p.lenght$ **DO**
16: $p_i.ACP \leftarrow$ GetACP(o, p_i);
17: extract allowed roles;
18: **END**
19: $C \leftarrow$ rules after combination; return C;
20: **ELSE**
21: This strategy type does not support integration
22: **END IF**

Algorithm 2 Token-Constrained Permission Delegation Algorithm

Input:
delegator s_{from}, delegatee s_{to}, permission token pt
Output:
true or false
1: PermissionDelegation(s_{from}, s_{to}, pt);
2:**IF** pt belongs to the $s_{from} \wedge$ the pt can be delegated $\wedge$ the s_{from} has the right to delegate the pt **THEN**
3: $pt.C \leftarrow$ ExtractConstraints(pt);
4: **IF** Decision(s_{to}, $pt.C$) **THEN**
5: Change the owner of the pt from delegator s_{from} to delegatee s_{to}
6: return true
7: **END IF**
8: return false
9: **END IF**

Algorithm 3 Pdp Re-Decision Permission Delegation Algorithm

Input:
delegator s_{from}, delegatee s_{to}, permission token pt
Output:
true or false
1: PermissionDelegation(s_{from}, s_{to}, pt);
2: **IF** pt belongs to the $s_{from} \wedge$ the pt can be delegated $\wedge$ the s_{from} has the right to delegate the pt **THEN**
3: $\{p_1 \ldots p_n\} \leftarrow$ GetPermission(pt)
4: Access(s_{to}, $\{p_1 \ldots p_n\}$)
5: $result \leftarrow$ PDPDecision(s_{to}, $\{p_1 \ldots p_n\}$);
6: **IF** $result ==$ true **THEN**
7: Change the owner of the pt from delegator s_{from} to delegatee s_{to}
8: return true
9: **END IF**
10: return false
11: **END IF**

through a mathematical model. Here, the CPN are used for modeling. The CPN is a mathematical modeling tool with strict mathematical definition and graphical representation. And the relevant concepts of CPN can be referred to the literature [30].

According to the permission delegation model proposed in this paper, the definition and description of the color set are shown in Table 2.

The top layer model is shown in Fig. 6. There are 8 transitions and 11 places in the top layer model, of which there are 2 substitution transitions. We simulated the delegation message sending process, part of the message verification process, and the *pt* delegation process through the model. The transition *TestA*, indicating that the delegation contract deployed in the blockchain has received the delegation message sent by the delegator, and has performed a preliminary verification. Here, it is verified whether the owner of the *pt* to be delegated is the delegator. Next, the transition *TestB*, which means verifying whether *pt* is allowed to be entrusted. The transition *TestC1* verifies whether the delegatee is registered in the access control system. The transition *DifferentSituation*, which means facing different access control strategies and choosing different algorithm verification. The transition *TestC2* means its own special verification method in PRPDA, and the transition *TestC3* means its own special verification method in TCPDA. After that, the two transitions of *RemovePT* and *AddNewPT* are fired to complete the process of token owner replacement.

The *TestC2* layer is shown in Fig. 7, with 5 transitions and 9 places. For the situation where the strategy cannot be integrated as a constraint. Therefore, it is necessary for the PDP to determine whether delegatee has the right to use *pt*, so the transition *SendRequest* fires and sends a request to the PDP. After the PDP receives the request message, the transition *GetACP* fires to find the policy corresponding to *pt*, and then makes a decision through the transition *PDPDecision*. Modeling is not a complete realization of the system, so the decision here is abstracted without

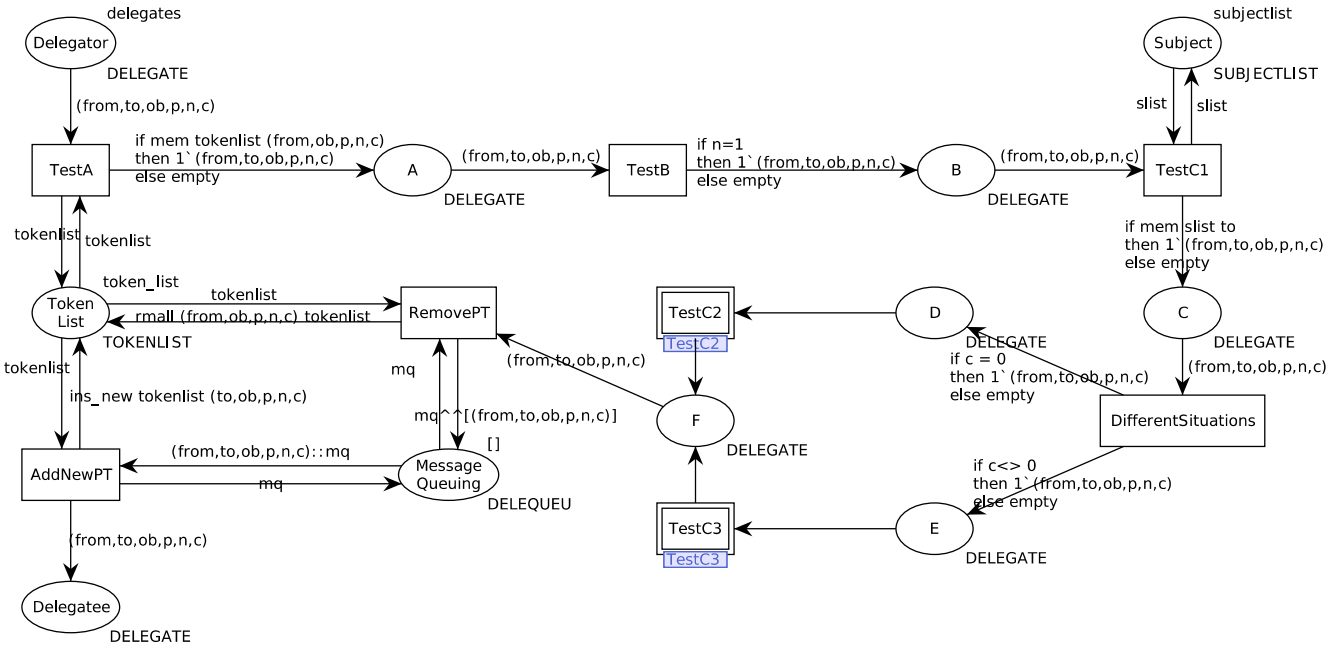


FIGURE 6. The top layer CPN model of permission delegation scheme.

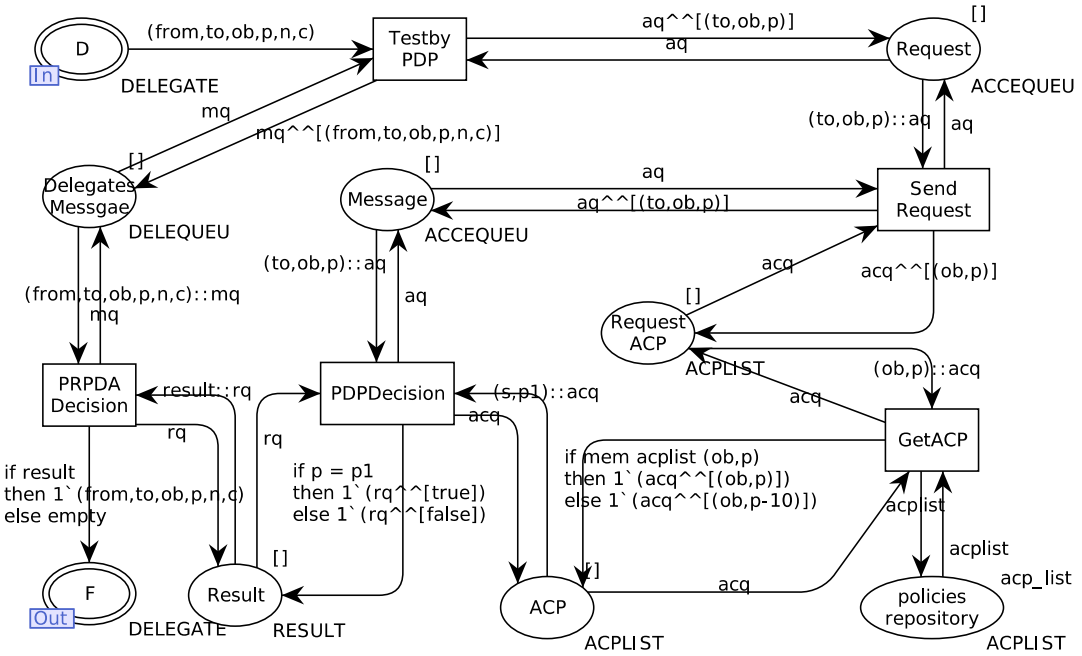


FIGURE 7. The TestC2 layer.

affecting the security of the system. Finally, the transition *PRPDADecision* is fired and the decision message is passed to the delegation contract.

The *TestC3* layer is shown in Fig. 8, with 2 transitions and 4 places. Since the delegatee's constraints are integrated in *pt*, the policy in constraint *C* can be directly extracted. The transition *TestByToken* converts the constraints in *pt* into policies, where $c + 2$ means to convert constraints into policies. Then make a decision through the transition *TCPDADecision*.

Proof: Through modeling, the security problem of the permission delegation mechanism is transformed into the accessibility problem in the CPN model. For any occurrence sequence $R(M)$, when M[*RemovePT* $>$ appears, that is, the transition *RemovePT* can be fired, it means that the token in the CPN must have undergone one of the following two transition occurrence sequences. One is the sequence using the TCPDA scheme, and the other is the sequence using the PRPDA scheme.

The occurrence sequence of using TCPDA is as follows: (1) M0[*TestA* $>$ M1, which means the transition *TestA* is fired, and will filter out two types of illegal requests, one is illegal *pt* requests, that is, there is no corresponding *pt* in the place *TokenList*; the other type is that the owner of *pt* is

TABLE 2. The color sets of formal models.

VARIABLE	COLOR SET DESCRIPTION
colset S,O,P,N,C = int	Subject, objcet, permission, *N* in *pt*, *C* in *pt*
colset TOKEN = product S*O*P*N*C;	Permission token
colset TOKENLIST = list TOKEN;	List of *pt* of granted principals stored in the blockchain
colset DELEGATE = product S*S*O*P*N*C;	Delegation message
colset ACCESS = product S*O*P;	Decision message sent to PDP
colset SUBJECTLIST = list S;	The subject list, here is delegatee
colset ACP = product S*P;	Access control policy of PDP request in blockchain
colset ACPLIST = list ACP;	Policies repository
colset DELEQUEU = list DELEGATE;	Message queue
colset ACCEQUEU= list ACCESS;	Message queue
colset CQUEU = list C;	Message queue
colset RESULT =list BOOL;	Message queue

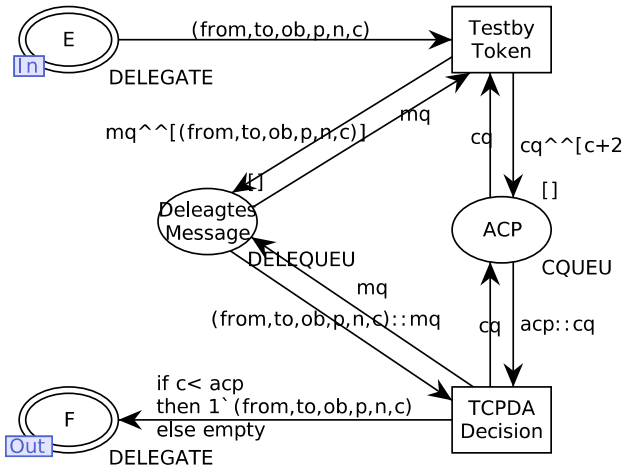


FIGURE 8. The TestC3 layer.

illegal, that is, the owner of the *pt* to be delegated is not the sender of the delegation request message. (2) M1[*TestB* > M2, will filter out all *pt* that are not allowed to be delegated, which means that all delegation messages expressed as $c = 0$ in the model are illegal. (3) M2[*TestC1* > M3, will filter out all delegatee that are not registered in the access control system. (4) M3[*DifferentSituations* > M4, the fire route is E(*DifferentSituation*, *E*) < if $c <> 0$ then 1‘(*from,to,ob,p,n,c*) else empty >= 1‘(*from,to,ob,p,n,c*), choose to use TCPDA algorithm. Next, verify in the *TestC3* layer. (5) M4[*TestbyToken* > M5, extract the access control policy in *pt*. (6) M5[*TCPDADecision* > M6, use the access control policy to verify whether the delegatee has the right to receive *pt*, and filter out illegal requests of delegatee.

The occurrence sequence of using PRPDA is as follows: the first three steps are the same as the sequence of TCPDA, in the fourth step, M3[*DifferentSituations* > M4, the fire route is E(*DifferentSituation*, *D*) < if $c = 0$ then 1‘(*from,to,ob,p,n,c*) else empty >= 1‘(*from,to,ob,p,n,c*), choose to use the PRPDA algorithm, and enter the *TestC2* layer for verification. (5) M4[*TestbyPDP* > M5’, extract the permission and delegatee corresponding to *pt* from the delegation message. (6) M5’[*SendRequest* > M6’, send delegatee and delegated permission to the PDP. (7) M6’[*GetACP* > M7’, the PDP requests the access control policy corresponding to the permission. (8) M7’[*PDPDecision* > M8’, the PDP makes a decision. (9) M8’[*PRPDADecision* > M9’, filter out illegal requests of delegatee according to the decision of PDP.

According to the above-mentioned two transition occurrence sequences, as long as M[*RemovePT* > appears, the token entered must undergo all security checks, and there will be no unauthorized access loopholes.

At the same time, through the simulation performed by cpn tools, the initial value in the model in the initial state is shown in Table 3. There are 6 types of delegation request messages representing different situations in the delegates. The message that finally completes the delegation through simulation is (1,2,10,1,1,1) and (1,4,10,3,1,0), meet the experimental expectations. Therefore, the two schemes proposed in this paper meet the safety goals.

TABLE 3. Model initialization.

NAME	VALUE	MEANING
delegates	[(1,2,10,1,1,1), (1,3,10,2,0,1), (1,4,10,3,1,0), (1,4,10,4,1,0), (1,5,10,4,1,0), (1,6,10,3,1,1)]	6 types of delegation request messages representing different situations
token_list	[(1,10,1,1,1), (1,10,2,0,1), (1,10,3,1,0), (1,10,4,1,0), (1,10,5,1,0)]	Existing *pt* in the system
acp_list	[(10,1), (10,2), (10,3)]	Access control policy
subjectlist	[2, 3, 4]	Delegatee registered in the system

VI. EXPERIMENT AND ANALYSIS

In this section, we evaluate the performance overhead of the two proposed schemes from the theoretical and experimental perspectives, and the experimental results verify the correctness of the theoretical analysis.

A. ALGORITHM TIME OVERHEAD

1) ALGORITHM TIME OVERHEAD ANALYSIS

Although the permission delegation algorithm and the blockchain consensus algorithm are not on the same layer according to the access control architecture, currently due to the limitations of the blockchain itself, it is not significant to discuss the response time of the permission delegation algorithm. Although some researchers have proposed millisecond-level consensus algorithms, for example, Sujit

Biswas *et al.* proposed a lightweight consensus algorithm PoBT for commercial blockchains [31]. Under the same conditions, PoBT is one-third faster than Fabric's Endorsement, but the current mainstream blockchain consensus time is still longer. Table 4 shows the consensus time of the current mainstream consensus algorithms [32]. The response time of the permission delegation in the actual blockchain environment is mainly determined by the consensus time of the blockchain platform where it is located. At this stage, the formula is given here of response time of permission delegation considering the influence of consensus algorithm.

TABLE 4. Time scale of blockchain consensus algorithm.

CONSENSUS ALGORITHM	TIME SCALE
Proof of Work (PoW)	>100s
Proof of Stake (PoS)	<100s
Delegated Proof of Stake (DPoS)	<100s
Practical Byzantine Fault Tolerance (PBFT)	<10s
RAFT	<10s

Considering the permission delegation scenario, the following assumptions can be drawn: Assuming that the permission delegation event reaches the blockchain consensus queue according to a Poisson process with a rate of λ, the blockchain consensus time can be regarded as an exponential random variable with a rate of μ, and it can be known that the consensus system of blockchain has only one service line, so the permission delegation considering the consensus time is a limited capacity $M/M/1$ queuing model.

It can be seen that considering consensus time the average response time function of TCPDA is:

$$T_{TCPDA} = t_{TCPDA} + \frac{1 + N\left(\lambda/\mu\right)^{N+1} - (N+1)\left(\lambda/\mu\right)^{N}}{(1 - P_N)(\mu - \lambda)\left(1 - \left(\lambda/\mu\right)^{N+1}\right)} \tag{1}$$

Among them, t_{TCPDA} refers to the execution time of TCPDA in the Ethereum virtual machine; N is the system capacity, and P_N, the probability that the consensus queue is full.

It can be seen that considering consensus time the average response time function of PRPDA is:

$$T_{PRPDA} = t_{PRPDA} + \frac{1 + N\left(\lambda/\mu\right)^{N+1} - (N+1)\left(\lambda/\mu\right)^{N}}{(1 - P_N)(\mu - \lambda)\left(1 - \left(\lambda/\mu\right)^{N+1}\right)} \tag{2}$$

t_{TCPDA} refers to the execution time of TCPDA in the Ethereum virtual machine; N is the system capacity, and P_N, the probability that the consensus queue is full.

2) ALGORITHM RESPONSE TIME WITHOUT CONSIDERATION OF CONSENSUS TIME

This experiment is a comparison of the time required for a permission delegation algorithm without consideration of the time required for consensus. The experiment is performed on the Ethereum blockchain. The experimental environment is shown in Table 5.

TABLE 5. Experimental environment parameters.

PARAMETERS	VALUE
Solidity version	0.5.3
Remix	0.8.8
Environment	JavaScript VM
CPU	intel(R)core(tm)i7- 4790
Memory	Memory size 8G

The response time in the above two schemes is determined by the execution time of the algorithm and the consensus time of the blockchain. The advantages and disadvantages are analyzed of both algorithms without considering the consensus time of the blockchain.

Besides the algorithms proposed above, the additional comparison objects are permission delegation schemes that do not use these two algorithms and the method proposed in [9]. For convenience, we will write the permission delegation scheme without TCPDA algorithm as WPDA. WPDA stands for a permission delegation scheme that does not verify the legitimacy of the delegatee, so it will cause the problem of unauthorized access. As a reference benchmark through comparison WPDA can reflect the price paid by TCPDA to increase security. Reference [9] represents a permission delegation scheme that stores permissions and permission delegations in the blockchain in the form of events rather than tokens. Although the permission delegation scheme proposed in [9] verifies the validity of the authentication delegatee, the verifier, also the delegator, does not know the access control policy set by the resource owner on the resource. Therefore, the program still has unauthorized access vulnerabilities. This scheme has the most security considerations in the current permission delegation schemes, considering the issue of permission proliferation, so it is used as a comparison object, which is represented by Pal_PDA in the experiment.

Fig. 9 shows the average response time in ten experiments of the permission delegation algorithm. The permission delegation contract is deployed in the remix JavaScript VM environment, so there is no effect of the blockchain consensus time on smart contract in the experimental data. The response time of TCPDA is 0.649s longer than Pal_PDA and 0.7s longer than WPDA. This is because TCPDA needs to extract the constraints in the permission token and use these constraints to verify the delegatee, so TCPDA needs to spend more time. PRPDA takes the longest time because the algorithm needs to pass the permission token and delegatee to the PDP during permission delegation, while other algorithms do not need to complete the decision-making process. WPDA and Pal_PDA take less time because they do not verify the legality of the delegatee, but they are not secure.

The data in the blockchain can only be used after consensus. At the same time, it is necessary to ensure that the data delegated by the permission is in the main chain

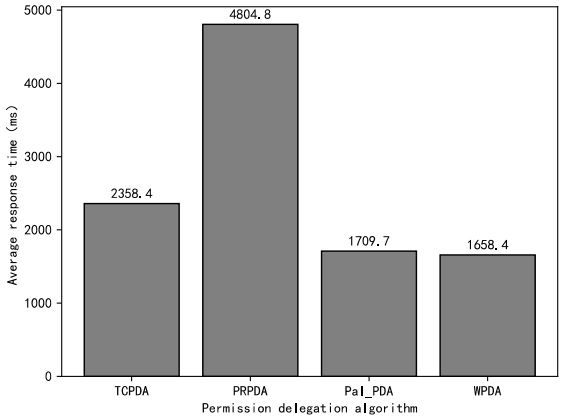


FIGURE 9. Average response time of permission delegation without considering consensus time.

of the blockchain. As security requirements increase, it is necessary to wait for the subsequent generation of more subsequent blocks. So, the consensus algorithm of the chain has a particularly profound impact on the permission transfer algorithm.

3) SUMMARY

In summary, although the two algorithms proposed in this paper has a longer time cost compared with the original permission delegation, the time required by these two algorithms is much shorter than that of the consensus algorithm. PRPDA as a supplement to TCPDA requires more time overhead. At the same time, the permission delegation response time will not increase due to multiple passes of permissions.

B. ALGORITHM STORAGE OVERHEAD

This section mainly analyzes the data overhead stored in the blockchain during the permission process. In order to solve the problems raised by the threat model, TCPDA and PRPDA add token storage data, compared with other existing solutions. The related expenses of permission delegation are mainly divided into two parts, token storage and delegation event storage. For the same access times in the same scenario, the frequency of permission delegation will affect the number of tokens in this system, and the smaller the number of tokens, the less storage overhead we propose. Next, the analysis is justified through experiments and comparison with existing solutions.

1) EXPERIMENTAL ENVIRONMENT SETUP

The storage part of the blockchain can be regarded as a monotonically increasing database. The data on the blockchain stored can only be increased but not reduced. Therefore, the storage overhead in the permission delegation algorithm needs to be analyzed in detail. The following experiments mainly analyze the storage overhead of the algorithm.

Since there is currently no standard data set disclosed in permission delegation, the experiments in this section use Python to generate data to simulate the storage overhead of the permission delegation algorithm proposed above.

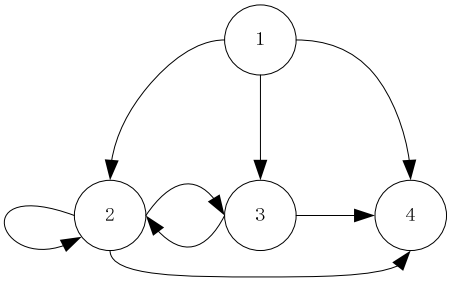


FIGURE 10. State transition diagram of permission token life process.

The simulation experiment data are generated by using a Markov chain. The structure of the Markov chain is shown in Fig. 10. There are four states, where State 1 indicates that the permission token is generated and granted to the subject; State 2 shows that the subject has an access using the permission token Resources; State 3 signifies that the permission delegation occurs, and the permission token is passed to the delegatee by the delegator; lastly, Status 4 means that the permission token is invalid. The transition between states represents the action of the permission token in the access control system. In state 1, the subject gets the *pt*. When the subject uses the *pt*, it enters state 2. When the subject delegates *pt* to delegatee, it enters state 3. And when the subject's *pt* is invalid, it enters state 4.

The state transition matrices of different Markov chains represent different probabilities of permission delegation events. The probability of sending permission delegation events is set to 6 levels from low to high, and the state transition matrices are specifically constructed as follows:

$$\mathrm{M}_a = \begin{bmatrix} 0 & 0.9 & 0.09 & 0.01 \\ 0 & 0.3 & a & 0.7-a \\ 0 & 0.99 & 0 & 0.01 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Among them, the value of a is [0.1, 0.2, 0.3, 0.4, 0.5, 0.6], which indicates the different situations in which the permission delegation occurs from low to high probability.

2) ALGORITHM STORAGE OVERHEAD

Fig. 11 shows the storage overhead of TCPDA and PRPDA under different probability delegation events. TCPDA has the largest storage overhead. When $a = 0.6$, the storage overhead is 3.05% higher than the scheme without delegate, 15.51% higher than that of WPDA and PRPDA, and 20.16% higher than that of Pal_PDA. This is because the algorithm integrates the access policy for permissions into the permission token. However, as WPDA and PRPDA has no storage constraints, its permission token is significantly smaller than that of TCPDA. Pal_PDA requires the least storage space. This is because the scheme stores the least amount of data in the

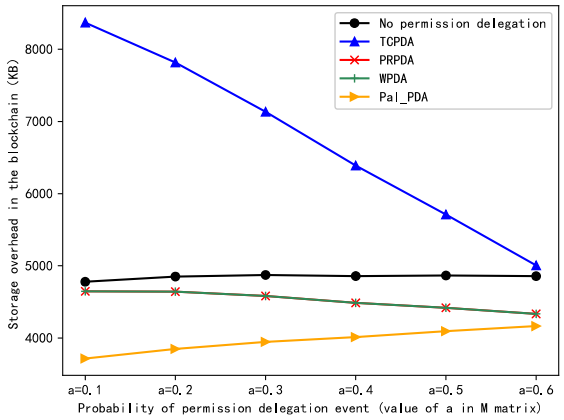


FIGURE 11. Comparison of storage space usage of different permission delegation algorithms.

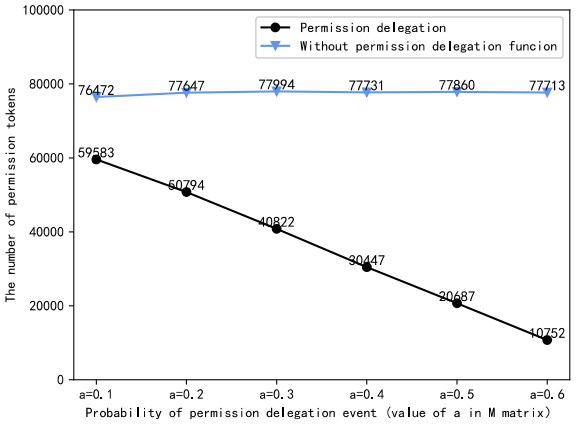


FIGURE 12. Impact of permission delegation on the number of permission tokens.

token. At the same time, the TCPDA token stores the data that Pal_PDA does not, such as the validity time of the permission, the number of times that it can be delegated, and delegatee constraints. The storage overhead of TCPDA is always greater than that of the scheme without a delegate. This is because TCPDA has more data stored in the permission token in order to counter the unauthorized access vulnerability. The data of PRPDA and WPDA are overlapped because the cost of token and permission delegation of the two algorithms is the same.

As the probability of permission delegation increases. the storage overhead of TCPDA will decrease accordingly. At the same time, TCPDA is the algorithm with the largest decrease, from 8370 KB at $a = 0.1$ to 5005 KB at $a = 0.6$, and the storage overhead is reduced by 40.2%. This is because the TCPDA's permission token size is incredibly different from the storage size of permission delegating event. However, the storage overhead of TCPDA is always larger than that of unauthorized delegation.

Under the same conditions, PRPDA, WPDA and Pal_PDA are smaller than the scheme without delegation. This is because the tokens of these schemes are smaller, and the existence of permission delegation reduces the number of tokens in the system.

As shown in Fig. 12, under different probabilities of occurrence of permission delegation events, the comparison of the number of permission tokens is generated by using the permission delegation and the non-use one. The horizontal axis represents the probability of permission delegation, and the vertical axis is the number of permission tokens that exist in the blockchain when the permission token is accessed 100,000 times.

It can be seen that as the probability of permission delegation increases, the number of permission tokens decreases that need to be generated in the access control system. The storage of permission tokens in the blockchain requires corresponding storage space, so permission delegation may reduce the overhead of access control on the blockchain storage.

Whether the storage overhead is reduced depends on the size of the permission token, and the size of the permission token varies in different schemes.

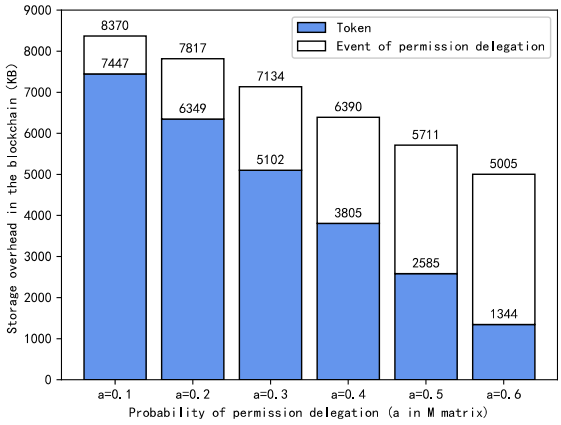


FIGURE 13. Storage overhead ratio of TCPDA.

Next, the storage overhead ratio of TCPDA is analyzed. Fig. 13 shows the ratio of storage overhead of a permission token and a permission delegation under different probabilities of TCPDA delegation events. From the experimental data, it can be seen that when $a = 0.1$, the storage cost of permission delegation accounts for 11.03% of the total cost. When $a = 0.6$, the storage cost of permission delegation already accounts for 73.15% of the total storage cost. After $a > 0.4$, the permission delegation part of the storage overhead starts to have more storage overheads than the permission token.

As shown in Fig. 14, when $a = 0.1$, the storage cost of permission delegation accounts for 19.87% of the total cost, and when $a = 0.6$, the storage cost of permission delegation accounts for 84.49% of the total cost. The permission delegation storage ratio of PRPDA is higher than that of TCPDA as the permission token of PRPDA is smaller than that of TCPDA, and the permission delegation data dominates.

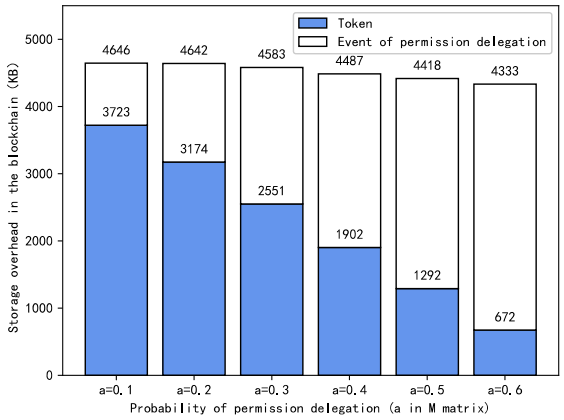


FIGURE 14. Storage overhead ratio of PRPDA.

3) SUMMARY

In summary, TCPDA counters the unauthorized access vulnerability caused by permission delegation, but it increases the storage overhead. With the increase of the probability of permission delegation, the storage overhead of TCPDA decreases accordingly. And that of PRPDA is small while the time overhead is large.

VII. CONCLUSION

It is not appropriate to use the user's trust chain directly in permission delegation of access control. It can be simply expressed as "I trust you but I do not trust your friends." Therefore, in the current blockchain-based access control, the delegation is not secure, determined only by the owner of the permission token. The token may be passed to an unauthorized user to obtain another one. So it is proposed to employ a permission-based unauthorized delegation vulnerability based on blockchain. The key to this vulnerability is that after the permission is granted via the object, subject A wants to pass the permission to another subject B. It must ensure that subject B who receives the permission has the right to use the permission. The delegatee passes the token after verification, and may pass the token to an illegal user. Then, two permission algorithms, TCPDA and PRPDA, are proposed to solve the problem. The use of CPN to model the delegation process proves the safety of the above two algorithms. Finally, experiments and analysis are carried out on the above two algorithms. The experimental results show that although both of these two algorithms have increased in time and space overhead, the time required by the permission delegation algorithm is much shorter than the time overhead of the consensus algorithm, and the storage overhead is gradually reduced as permission delegation events increase with a greater probability.

Future work will investigate the division of permissions in access control. At present, if some of the permissions are intended to pass on to others during the permission delegation process, restrictions are needed to add to the permission token so that the receiver can only use some permissions. Given some restrictions and inconveniences of the method, an alternative of dividing permissions is proposed to allow users to perform more fine-grained operations on permissions they own.

REFERENCES

[1] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "FairAccess: A new blockchain-based access control framework for the Internet of Things," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 5943–5964, 2016, doi: 10.1002/sec.1748.

[2] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "Towards a novel privacy-preserving access control model based on blockchain technology in IoT," in *Europe and MENA Cooperation Advances in Information and Communication Technologies* (Advances in Intelligent Systems and Computing). Cham, Switzerland: Springer, 2017, pp. 523–533, doi: 10.1007/978-3-319-46568-5_53.

[3] G. Zyskind, O. Nathan, and A. S. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Proc. IEEE Secur. Privacy Workshops*, San Jose, CA, USA, May 2015, pp. 180–184, doi: 10.1109/SPW.2015.27.

[4] N. Rifi, E. Rachkidi, N. Agoulmine, and N. C. Taher, "Towards using blockchain technology for IoT data access protection," in *Proc. IEEE 17th Int. Conf. Ubiquitous Wireless Broadband (ICUWB)*, Salamanca, Spain, Sep. 2017, pp. 1–5, doi: 10.1109/ICUWB.2017.8251003.

[5] S. Nayak, N. C. Narendra, A. Shukla, and J. Kempf, "Saranyu: Using smart contracts and blockchain for cloud tenant management," in *Proc. IEEE 11th Int. Conf. Cloud Comput. (CLOUD)*, San Francisco, CA, USA, Jul. 2018, pp. 857–861, doi: 10.1109/CLOUD.2018.00121.

[6] R. Xu, Y. Chen, E. Blasch, and G. Chen, "BlendCAC: A smart contract enabled decentralized capability-based access control mechanism for the IoT," *Computers*, vol. 7, no. 3, p. 39, Jul. 2018, doi: 10.3390/computers7030039.

[7] R. Xu, Y. Chen, E. Blasch, and G. Chen, "BlendCAC: A BLockchain-enabled decentralized capability-based access control for IoTs," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Halifax, NS, Canada, Jul. 2018, pp. 1027–1034, doi: 10.1109/Cybermatics_2018.2018.00191.

[8] T. Le and M. W. Mutka, "CapChain: A privacy preserving access control framework based on blockchain for pervasive environments," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Taormina, Italy, Jun. 2018, pp. 57–64, doi: 10.1109/SMARTCOMP.2018.00074.

[9] S. Pal, T. Rabehaja, M. Hitchens, V. Varadharajan, and A. Hill, "On the design of a flexible delegation model for the Internet of Things using blockchain," *IEEE Trans. Ind. Informat.*, vol. 16, no. 5, pp. 3521–3530, May 2020, doi: 10.1109/TII.2019.2925898.

[10] M. Jemel and A. Serhrouchni, "Decentralized access control mechanism with temporal dimension based on blockchain," in *Proc. IEEE 14th Int. Conf. E-Business Eng. (ICEBE)*, Shanghai, China, Nov. 2017, pp. 177–182, doi: 10.1109/ICEBE.2017.35.

[11] A. Outchakoucht, H. ES-SAMAALI, and J. Philippe, "Dynamic access control policy based on blockchain and machine learning for the Internet of Things," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 7, 2017, doi: 10.14569/IJACSA.2017.080757.

[12] J. P. Cruz, Y. Kaji, and N. Yanai, "RBAC-SC: Role-based access control using smart contract," *IEEE Access*, vol. 6, pp. 12240–12251, 2018, doi: 10.1109/ACCESS.2018.2812844.

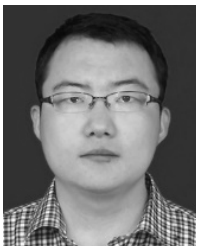
[13] S. Alansari, F. Paci, and V. Sassone, "A distributed access control system for cloud federations," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Atlanta, GA, USA, Jun. 2017, pp. 2131–2136, doi: 10.1109/ICDCS.2017.241.

[14] D. D. F. Maesa, P. Mori, and L. Ricci, "Blockchain based access control," in *Proc. IFIP Int. Conf. Distrib. Appl. Interoperable Syst.* Cham, Switzerland: Springer, 2017, pp. 206–220, doi: 10.1007/978-3-319-59665-5_15.

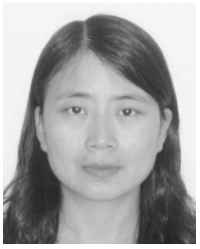
[15] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, "Smart contract-based access control for the Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1594–1605, Apr. 2019, doi: 10.1109/JIOT.2018.2847705.

[16] M. A. Islam and S. Madria, "A permissioned blockchain based access control system for IOT," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Atlanta, GA, USA, Jul. 2019, pp. 469–476, doi: 10.1109/Blockchain.2019.00071.

[17] Y. Zhang, B. Li, B. Liu, J. Wu, Y. Wang, and X. Yang, "An attribute-based collaborative access control scheme using blockchain for IoT devices," *Electronics*, vol. 9, no. 2, p. 285, Feb. 2020, doi: 10.3390/electronics9020285.
[18] H. Guo, E. Meamari, and C.-C. Shen, "Multi-authority attribute-based access control with smart contract," in *Proc. Int. Conf. Blockchain Technol. (ICBCT)*, Honolulu, HI, USA, 2019, pp. 6–11, doi: 10.1145/3320154.3320164.
[19] A. R. Rajput, Q. Li, M. T. Ahvanooey, and I. Masood, "EACMS: Emergency access control management system for personal health record based on blockchain," *IEEE Access*, vol. 7, pp. 84304–84317, 2019, doi: 10.1109/ACCESS.2019.2917976.
[20] A. Ali, U. Habiba, and M. A. Shibli, "Taxonomy of delegation model," in *Proc. 12th Int. Conf. Inf. Technol.-New Generat.*, Las Vegas, NV, USA, Apr. 2015, pp. 218–223, doi: 10.1109/ITNG.2015.41.
[21] J. Wainer and A. Kumar, "A fine-grained, controllable, user-to-user delegation method in RBAC," in *Proc. 10th ACM Symp. Access Control Models Technol. (SACMAT)*. New York, NY, USA: Association Computing Machinery, 2005, pp. 59–66, doi: 10.1145/1063979.1063991.
[22] B. Anggorojati, P. N. Mahalle, N. R. Prasad, and R. Prasad, "Capability-based access control delegation model on the federated IoT network," in *Proc. 15th Int. Symp. Wireless Pers. Multimedia Commun.*, Taipei, Taiwan, Sep. 2012, pp. 604–608.
[23] T. Rabehaja, S. Pal, and M. Hitchens, "Design and implementation of a secure and flexible access-right delegation for resource constrained environments," *Future Gener. Comput. Syst.*, vol. 99, pp. 593–608, Oct. 2019, doi: 10.1016/j.future.2019.04.035.
[24] M. Moniruzzaman and K. Barker, "Delegation of access rights in a privacy preserving access control model," in *Proc. 9th Annu. Int. Conf. Privacy, Secur. Trust*, Montreal, QC, Canada, Jul. 2011, pp. 124–133, doi: 10.1109/PST.2011.5971974.
[25] N. Tapas, G. Merlino, and F. Longo, "Blockchain-based IoT-cloud authorization and delegation," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Taormina, Italy, Jun. 2018, pp. 411–416, doi: 10.1109/SMARTCOMP.2018.00038.
[26] R. Alcarria, B. Bordel, T. Robles, D. Martín, and M.-Á. Manso-Callejo, "A blockchain-based authorization system for trustworthy resource monitoring and trading in smart communities," *Sensors*, vol. 18, no. 10, p. 3561, Oct. 2018, doi: 10.3390/s18103561.
[27] R. Xu, Y. Chen, E. Blasch, and G. Chen, "Exploration of blockchain-enabled decentralized capability-based access control strategy for space situation awareness," *Proc. SPIE*, vol. 58, no. 4, 2019, Art. no. 041609, doi: 10.1117/1.OE.58.4.041609.
[28] G. Ali, N. Ahmad, Y. Cao, M. Asif, H. Cruickshank, and Q. E. Ali, "Blockchain based permission delegation and access control in Internet of Things (BACI)," *Comput. Secur.*, vol. 86, pp. 318–334, Sep. 2019, doi: 10.1016/j.cose.2019.06.010.
[29] H. S. G. Pussewalage and V. A. Oleshchuk, "Blockchain based delegatable access control scheme for a collaborative E-Health environment," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Halifax, NS, Canada, Jul. 2018, pp. 1204–1211, doi: 10.1109/Cybermatics_2018.2018.00214.
[30] K. Jensen and L. M. Kristensen, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. Berlin, Germany: Springer, 2009.
[31] S. Biswas, K. Sharif, F. Li, S. Maharjan, S. P. Mohanty, and Y. Wang, "PoBT: A lightweight consensus algorithm for scalable IoT business blockchain," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2343–2355, Mar. 2020, doi: 10.1109/JIOT.2019.2958077.
[32] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, "A review on consensus algorithm of blockchain," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Banff, AB, Canada, Oct. 2017, pp. 2567–2572, doi: 10.1109/SMC.2017.8123011.



JINSHAN SHI received the B.S. degree in computer science and technology from Inner Mongolia University, Hohhot, China, in 2009, where he is currently pursuing the Ph.D. degree in computer science and technology. His research interests include access control issues under the Internet of Things and using blockchain to solve problems in access control.



RU LI (Member, IEEE) received the M.S. degree in computer science from Peking University, in 2008, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, in 2005. She is currently a Professor and a Ph.D. Supervisor with Inner Mongolia University. She has published more than 50 papers in international conferences and journals. Her research interests include blockchain, wireless networks, future internet, and so on.

WENHAN HOU received the bachelor's degree in software engineering from Yantai University, China, in 2018. He is currently pursuing the master's degree with Inner Mongolia University. His research interests include blockchain and access control.