# How to divide a permission token in the delegation process of blockchain-based access control for IoT

Jeonghwa Heo
*Department of Computer Science and Engineering*
*Korea University*
Seoul, South Korea
kueun0418@korea.ac.kr

Heewoong Jang
*Department of Computer Science and Engineering*
*Korea University*
Seoul, South Korea
jormal@korea.ac.kr

Heejo Lee
*Department of Computer Science and Engineering*
*Korea University*
Seoul, South Korea
heeojo@korea.ac.kr

***Abstract*— There are several security problems arising from the characteristics of IoT, and one of them is weak access control. Traditional access control models require one centralized authority that stores all the information for access control and validates access rights. This single point of failure in IoT access control could lead to situations where a single breach can cause sensitive information leakage across the entire system. Various studies have been conducted to mitigate this security risk by introducing a decentralized architecture based on blockchain technology called BBAC. However, most BBAC models consider only a simple access control situation, which can lead to a "the Greatest privilege problem". This study proposes a novel access control model that enforces minimum privilege to an access token by the division and modification of access rights. As a result, we contributed to enhancing the practicality of the BBAC and mitigating risks that may arise in the delegation process.**

***Keywords—Blockchain, Access control model, IoT, BBAC***

## I. Introduction

With the rapid advances in Internet of Things (IoT) technologies, the use of IoT technologies has increased extensively across smart factories, smart cities, and healthcare, and Fig. 1 illustrates this phenomenon [1].

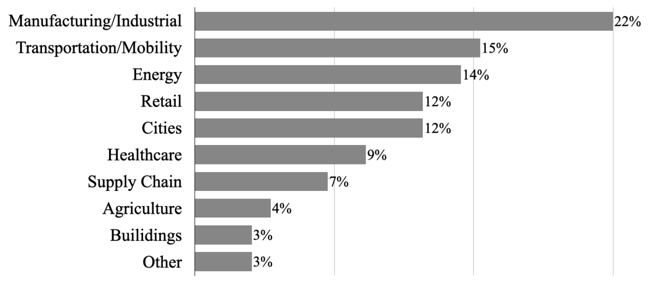

Fig. 1. Top 10 IoT Application areas 2020

IoT is currently one of the most preferred technologies. As illustrated in Fig. 2, the number of global M2M connections is expected to grow from 6.1 billion in 2018 to 14.7 billion by 2023 [2], owing to the increased use of IoT devices.

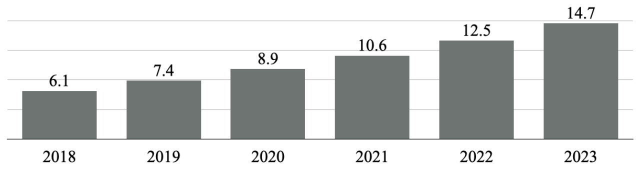

Fig. 2. The number of M2M connections 2018-2023 (Unit: Billion)

However, the continuity of IoT security incidents, such as the Mirai botnet attack [3], indicate that the security of IoT is a chronic problem. Access control is one of the major security challenge of IoT, as it involves relaying sensitive information. For example, an inadequate IoT access control model can permit unauthorized users to access medical information [4]. Considering the massive number of IoT devices installed, a well-defined IoT access control mechanism that manages various complex access rights is essential.

Several methods have been suggested for IoT access control, including role-based access control (RBAC) and attribute-based access control (ABAC). These are centralized models, in which all the access control is centralized . This centralized nature can expose IoT devices to a single point of failure. Capability-based access control (CapBAC) has emerged as a solution for providing more flexible access control through a capability token. However, CapBAC too requires a centralized entity for validating the access rights. Therefore, a single point of failure remains in trustworthy IoT environments. As an alternative, distributed capability-based access control (DCapBAC), which is a validation process occurring inside IoT devices, was suggested. However, validating access rights with limited resources of IoT devices may hinder their performance and a malicious user may cause unstable operation. Several commercial tools for IoT access control are available. However, using these tools can be seen as outsourcing IoT access control to a third party from the user's perspective, which can lead to a problem of trust. Outsourcing for access control requires very strong trust between the parties requiring the outsourcing company will never tamper with the data. Access control methods using blockchain technology have recently been proposed as an alternative solution to solve this problem completely because blockchain stores manage data in a decentralized manner.

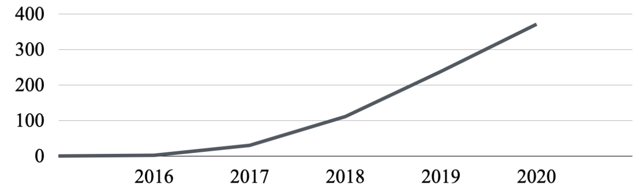

Fig. 3. Number of published papers related with BBAC

Research addressing blockchain-based access control (BBAC) models has been actively conducted since 2016. Fig. 3 illustrates the growing number of published papers related to BBAC.

Based on the characteristics of BBAC, some studies have suggested replacing existing access control methods with blockchain-based methods [5]–[10].

Recent research addresses delegating access rights to BBAC methods in IoT environments to achieve practicality. The delegation process is performed by a decentralized entity and allows an entity to use the access rights of another entity for a period of time, not permanently. However, there remains a risk of abuse of delegated rights by unauthorized users. Several studies aim to restrict such abuses by utilizing the characteristics of smart contracts [8], [10], [11]. Some researchers have suggested a BBAC model tailored to a specific area, such as smart grids and healthcare [13]–[15].

Limiting the rights already given to the requestor is another question to be solved in the BBAC models. Existing models [4], [16] focus only on a simple case consisting of a single delegator and delegatee. There is a need for a situation in which a subject that has permissions for several behaviors delegates only one permission to the other subjects. Consider an example case in which a subject has all permissions – create, read, update, and execute on a printer. Another subject wants to delegate the permission token to access the printer. Following the principle of minimum privilege, subjects who request access rights to the printer should only be granted to read and execute, but not to create or update.Therefore, the permission token should be broken down into two – one for read and execute rights and the other for the rest. Existing permissions must be divided into two different sub-permissions. However, previous research has failed to consider such a case that requires the division of a permission token.

We propose an enhanced BBAC model for IoT networks. Our architecture compensates for the lack of consideration in situations that require the division of rights that are already granted. Existing papers focus only on simple situations, such as the transference or delegation of one right from one entity to another. However, to increase practicality and at the same time ensure that only the minimum level of access rights is delegated, it is essential to consider the division and modification of the rights that have already been generated. This study assigns an index to sub-permission within the permission token and suggests an architecture that access right is requested or delegated in whole or in part based on the index. The contribution of this paper is as follows: We define "the Greatest privilege problem" that can occur in the delegation process on the BBAC and propose a new BBAC model that can not only prevent it and but also apply to more complicated delegation conditions. To the best of our knowledge, none of the previous works related to BBAC for IoT have considered the access right division issue.

The remainder of this paper is structured as follows: Section II provides an overview of related works on access control for the IoT domain. Section III presents the overall architecture of a BBAC system. Section IV explains the representative access right division scenario and suggestions regarding addressing the situation by proposing a delegation algorithm that is considered a complex delegation. Section V presents evaluation to prove the effectiveness of the proposed model. In Section VI, we discuss the proposed model and conclude the study in Section VII.

## II. Background

In this section, we provide a brief background on traditional access control, blockchain and smart contracts, and the delegation process. In addition, this paper explains related studies.

### A. *Traditional Access Control*

Access control has two basic principles that should be followed [17].

- Least Privilege Policy: The least privileged policy is to minimize damage caused by abuse of authority by granting only the minimum rights to perform the authorized work.
- Separate Duty Policy: This principle is to prevent the occurrence of all processes in the development - task, approval, modification, confirmation, and completion - from being conducted by one person. In other words, the separation between a person who oversees the modification process and another for confirmation.

There are four major types of access control mechanisms in IoT. Each mechanism has advantages and disadvantages in an IoT environment.

- Role-based access control (RBAC): RBAC is a mechanism that checks the role (e.g., administrator, manager, guest) of a requestor when he wants to access a right.
- Attribute-based access control (ABAC): Based on ABAC, a centralized entity makes a decision based on the special attributes of a requestor who wants to access a resource. Therefore, ABAC facilitates the management of access policies. However, both RBAC and ABAC are highly centralized and can invoke a single point-of-failure problem.
- Capability-based access control (CapBAC): In CapBAC-based schemes, subjects can perform certain activities on an object if they have appropriate capabilities. However, a centralized entity is required to validate the token.
- Distributed capability-based access control (DCapBAC): DCapBAC model [18] was proposed to provide more controlled access control. Subjects can obtain access rights based on capability, and the validation process is conducted by the requested IoT devices themselves, not by a centralized entity. Therefore, it enables flexible access control inside the IoT devices. However, owing to the low computing capability of IoT devices, they may be easily compromised by adversaries; thus, they cannot establish a fully trusted relationship between the entities that validate the access rights. In conclusion, DCapBAC is not a suitable access control method for IoT environments.

### B. *Blockchain and Smart Contracts*

Blockchain refers to distributed ledger architectures that consist of nodes that do not trust each other. As its name suggests, several timestamped blocks compose a blockchain and the blocks contain a number of transactions between nodes. Nodes validate any transactions they receive and transactions can be appended after validation. The blockchain is permanent, tamper-proof, and distributed. Any node with the right to access the blockchain can view all the records in the chain [19].

Smart contracts can be simply defined as a program that resides on a blockchain and automatically runs when the preset conditions are satisfied. Therefore, smart contracts do not require any intermediaries or the loss of time. As smart contracts are executed on a blockchain, all participants can execute its instructions and see the history of each interaction between smart contracts [20].

### C. *Delegation*

Delegation refers to a temporal process in which a resource owner transfers the right to permit access to its resource to another entity.

Delegation in IoT is conducted in a distributed manner, whereas most access rights administration operations are central. Fig. 4 shows an example of a delegation process in an IoT environment. If the delegation system has any blind points, an attacker can obtain the access rights of a resource [4], [16]. Delegation continues to receive a lot of attention, as a trustworthy delegation without a centralized third party is essential in an IoT environment that has heterogeneous and large-scale characteristics. Consequently, many delegation models in IoT that use blockchain technology to solve a single point of failure problem and improve the reliability of the models have been proposed.

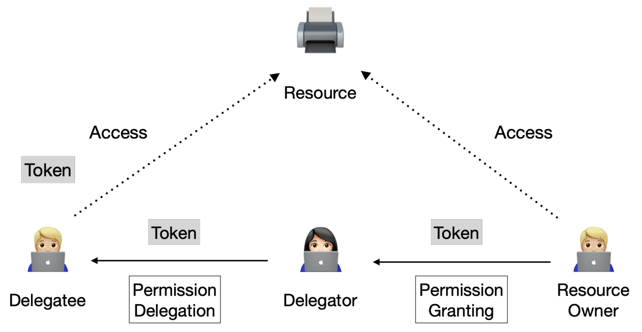

Fig. 4. Process of Delegation

### D. *Related Works*

Considering the advantages of blockchain technology, majority of the research has suggested a blockchain-based access control (BBAC) model for IoT. For example, Pal et al. [9] proposed a delegation access control rights architecture using a public blockchain and a private chain to strengthen privacy. Xu et al. [21] implemented BlendCAC, a capability delegation mechanism based on a blockchain network. In this structure, when an entity sends a request to obtain access rights, the entity designated as a domain owner issues a capability token. In addition, Gauhar. A et al. [11] presented a decentralized architecture for permission delegation and access control in the IoT, where the delegation process is based on both queries and events. Zhang [22] suggested a smart contract-based framework consisting of multiple-access control contracts as a trustworthy and distributed access control model in an IoT environment.

In some studies, a BBAC model has been customized for a specific IoT area. For example, Feng [13] suggested a consortium blockchain-based access control framework with dynamic ordered node selection for 5G-enabled industrial IoT, and compared the framework implementation based on the Practical Byzantine Fault Tolerance (PBFT) consensus. In addition, Bera et al. [14] designed a BBAC protocol for an IoT-enabled smart-grid system called DBACP-IoTSG. Saha et al. [15] proposed a BBAC mechanism for IoT-enabled healthcare applications which emphasizes on supporting various functionality features while providing better security and maintaining a low level of communication and computation costs.

A research has been conducted on addressing the issue of delegation of arbitrary access rights to other subjects. Saha et al. [16] suggested a flexible delegation model for IoT environments using blockchain. Shi et al. [4] emphasized that, for the delegation process, an unauthorized access vulnerability exists, where an unauthorized user can obtain rights by abusing this process. For example, a resource request that obtains permission from the access control system can delegate permission to another subject that is not verified by the access control system. This situation is possible because the object does not predefine all legitimate users. The object only verifies whether the delegator and delegatee agree to the permission. The author suggested a token-constrained permission-delegation algorithm as a solution.

However, previous papers discussing BBAC models mostly consider a single delegator and a single delegatee, ignoring the possibility of access rights division, which exists in several delegatees. This can cause "the greatest privilege problem" that a subject can get more privileges than necessary. To address this complex delegation situation, the ability to divide access rights is a requirement for managing practical cases in the real world.

## III. Blockchain-based Access Control System

In this section, we present a blockchain-based access control (BBAC) model and introduce the permission delegation case. Based on this model, this paper suggests a model for dividing access rights that have already been generated.

### A. *Overall architecutre of BBAC*

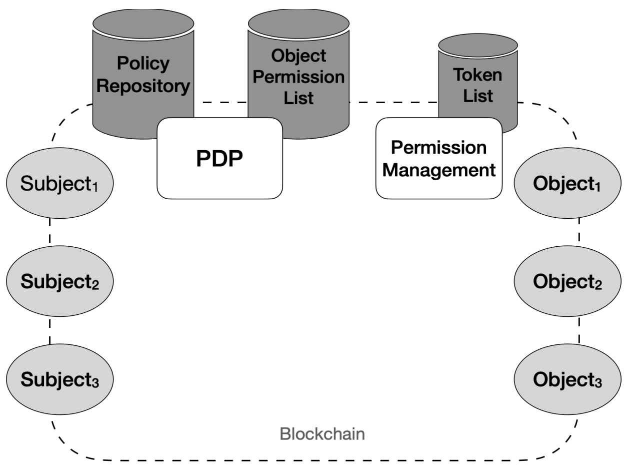

Fig. 5. Overall Architecture of BBAC

Fig. 5 shows the overall structure of the proposed architecture. The BBAC system architecture consists of subjects, objects, token lists, object permission lists, a policy repository, and smart contracts, such as the policy decision point (PDP) and permission management (PM). Smart contracts are deployed in the blockchain. This blockchain is based on a consortium blockchain architecture, which requires permission to participate in the blockchain network. The PDP validates the permission to access a resource based on the policy repository with a preset policy list and conditions. In addition, the PDP utilizes an object permission list containing information on all object permissions that are already granted by the PDP. Objects, actions, features, and constraints are recorded in the list. With the object permission list, we can determine the type of actions that can be conducted by the object and each permission's features, including whether it is transferable, and if it is, how many times it can be moved. Through these actions, the object permission list assigns indexes that are useful for delegating a permission token. In addition, based on the constraint, we can grant a permission token considering the limited situation based on the policy that all objects have already been set. Notably, this study does not address the constraints and methods of how objects record their policies in detail to focus on the delegation issue. The token list is managed by the PM and includes all the information of tokens that has been given to a requestor. In the token list, the object, token id, and permission indexes are stored, and based on this information, the PM manages what sub-permissions are included in a permission token based on the index.

In this case, the access requestor is a delegator that initiates permission delegation. If the permission that the delegator owns is transferable, it can send permission to other people. Those who obtain permission from the delegator through the delegation process are defined as delegatees.

In the access control system, the object owner has already uploaded the object's access control strategy, which he preset, to the policy repository, and the resource requester sends a request for permission to access the resource. The object owner can set their access control policy based on various methods, such as ABAC and RBAC.

If the PDP decides to grant access rights to the requestor, it sends the requester a permission token. The permission token has a transferability attribute that can be passed on to the delegatee by the resource requester.

## B. Permission Granting Process Model

When a subject requests rights to access a resource, the PDP validates such a request based on the preset policy repository, which is called the permission-granting process in Fig. 6.

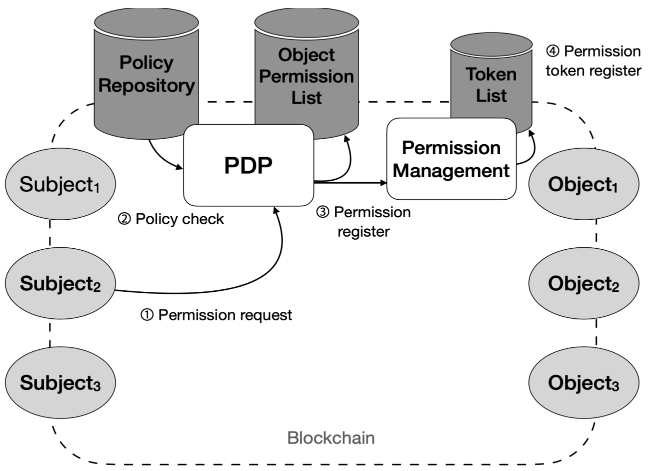

Fig. 6. Permission Granting Process

The process is as follows. The subject sends a message *RequestAccess(s, r, o, {o.p₁, o.p₂,… o.pᵢ})* to the access control system. The message content is as follows: the subject *s*, the object owner *r*, the object *o*, and the requested permissions *{o.p₁, o.p₂,… o.pᵢ}* such as create, read, update, and delete. If the PDP decides to grant permission to the subject based on the access control policy recorded in the policy repository, the PDP generates and gives a permission token (*pt*) to the subject and records the token information in the object permission list. Permission token is denoted as $pt = \{o, id, P\}$, indicating $o$ as an object, $id$ as token ID, and $P$ as a list of permission indexes. Otherwise, the contracts just leave a reject message to the requestor.

The PDP transfers the information of the token granted to the PM to log the history in the token list. By receiving an access right token, the requestor can transfer the access rights specified in the token as whole or in part.

## C. Permission Delegation Process Model

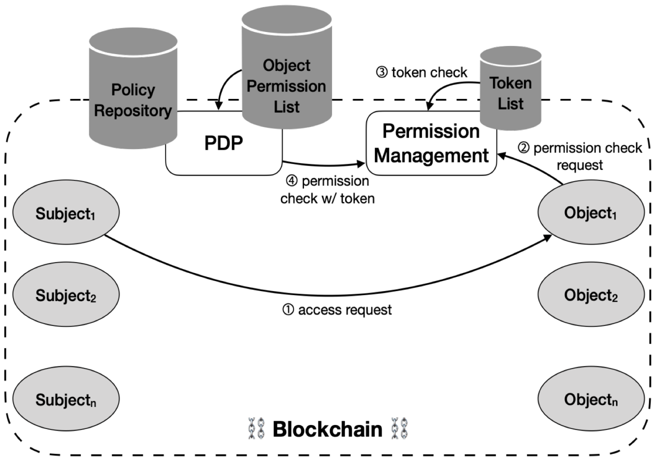

Fig. 7. Permission Delegation Process

The permission delegation process model shown in Fig. 7 is a method in which a delegatee obtains a permission token from a delegator. The process is as follows. (1) A delegatee sends a request to delegate the permission token that the delegator owns. (2) The delegator then receives the request from the delegatee to give permission. If the delegator decides to delegate, the process continues until the next stage.

Note: ** The first and second steps of permission delegation have no relationship with the access control system.

(3) A subject (*delegator*) who owns the permission token asks the PM to transfer the token to another subject (*delegatee*) who wants to obtain access rights included in the token of the existing subject (*delegator*). (4) The PM delivers information-checking features and constraints to the PDP. After receiving the information, the PDP decides whether to accept the request based on the object permission list. (5) Then PM records the generated delegation information to the token list.

The difference between the permission granting and delegation processes is that the receiver receives the subject's request. In the delegation process, the PM is a receiver while the subject sends a request to the PDP to obtain permission.

## D. Permission Check Process Model

The permission check process is for when a subject who has a token tries to access an object. This process is essential for validating the permission-token.

First, the subjects should send a message requesting access to the object *GrantAccess (s, r, o, pt, {o.p₁, o.p₂,… o.pᵢ})* to the access control system in the blockchain. Then, the object requests the PDP whether the subject's token is valid to access the object based on the token list and information based on the object permission list. Through these check phrases, if the token through which the subject tries to access the object is valid, then the subject can access the object. Otherwise, access to the object by the subject is denied.

## IV. Division of Access Rights

In this section, we explain how access rights that have already been granted to a subject are divided, which is the main contribution of this study. The division of access rights is the modification of the access token that has already been generated, and is transferable to a part of the rights that a delegator owns.

### *A. the Greatest Privilege Problem*

There is a risk that more privileges will be moved than the appropriate privilege level because the division of a permission token is not possible in the previous models. This paper refers to this limitation as "the Greatest privilege problem in access control." Consider the following sample case, where Alice owns a permission token granted by object A's owner and includes a total of four rights: read, update, execute, and delete. Bob wants to delegate a permission token to Alice to read Object A's material, and Alice agrees with Bob. If dividing a permission token is impossible, Alice can only delegate the token with all four access rights to Bob. This situation is in contrast to the basic principles of access control. In addition, many IoT devices address private personal information such as health and location. Therefore, there is a risk that the delegatee can exploit the permission token that contains more privileges than required.

Therefore, the existing model, where dividing the permission token is impossible, is not practical in real-life use because the access control model cannot apply the minimal right principle.

### *B. Delegation Algorithm*

To address the greatest privilege vulnerability in access control, we propose an enhanced delegation algorithm.

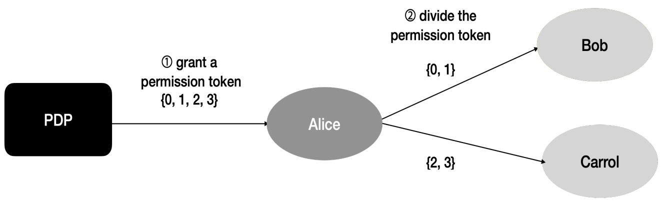
② divide the permission token
Bob
① grant a permission token {0, 1, 2, 3}
{0, 1}
PDP
Alice
{2, 3}
Carrol

Fig. 8. Example Scenario of Permission Division

To clarify this explanation, we provide one specific scenario, as shown in Fig. 8. Consider the following situation where there are a total of three different entities involved. First, PDP has been granted to a permission token that allows four different functions for object A to Alice (the delegator in this case), and the functions are create, read, update, and delete. Each of the functions was assigned an index from 0 to 3 in order and recorded in the permission token that Alice owns and in the token list. Through the permission-granting process, Alice can conduct all functions for object A. Alice wants to delegate read and write permissions to Bob (delegatee$_1$ in this case) and the rest to Carlo (delegatee$_2$ in this case), therefore, requiring split of access rights.

Even though there exists only Bob as a delegatee, Alice can delegate only a part of the rights within the permission token, instead of all the rights. Consequently, Alice can maintain the rest of their rights and prevent the delegatee from delegating excessive rights.

### *C. Algorithm Design*

In this section, we explain the details of the delegation algorithm considering the division of permission tokens. When the delegator delegates the token to delegatees, the delegator sends a new permission token to each delegatee, modifying the indices that indicate the permission that the delegatees want to obtain. The details are described in the following algorithm.

**Algorithm 1. Permission division**

```
INPUT: the ID of permission $pid$, index lists $dlist$ for delegation in permission token, and the addresses of subjects $s_{from}$, $s_{to}$ which indicate delegator and delegatee.

OUTPUT: the token delegation

if $not\ permissionIdxExists\ (s_{from}, pid, dlist)$
then revert

$obj \leftarrow tokenList[s_{from}][pid].object$

if $not\ checkFeature(obj.id, pid)$ then revert

$tokenList[s_{from}][pid] \leftarrow tokenList[s_{from}][pid] - dlist$
$tokenList[s_{to}][pid] \leftarrow tokenCreate(obj,pid,dlist)$

if $tokenList[s_{from}][pid] = \emptyset$ then del
$tokenList[s_{from}][pid]$

Delegation done.
```

Fig. 9. Permission Division Algorithm

Fig. 9 shows an algorithm for the separation of permission-s that have already been generated. The input for this algorithm consists of the ID of permission *pid*, index list *dlist* for delegation in the permission token, and the addresses of subjects who send and receive the permission token $s_{from}$ $s_{to}$. First, the algorithm checks whether $s_{from}$'s permission index exists in the index list. If it does not exist, the algorithm simply reverts the input Otherwise, brings the object for which the permission token is for. In addition, the algorithm checks feature *F* of the object and permission. Based on *F*, if the permission token cannot be delegated, the algorithm reverts to the input. In the case that the permission token has transferability, all permission ids that will be delegated are removed from the *dlist* and the delegator's permission token is updated, excluding the permission ids delegated to the delegatee. However, for the delegatee, a new permission token is created, including the permission ids delegated from the delegator. After the delegation process, if there does not exist any permission available to the delegator, the delegator's permission token is deleted. Consequently, the permission token is separated and delegated.

The sequence diagram of our model is shown in Fig. 10. For ease of understanding, we compared the existing model from [4], which is described in Fig. 11, which cannot partially delegate a permission token. According to the previous model, to separate a permission token generated, as modification of the permission is impossible, the original permission should be deactivated, and the regeneration of permission tokens reflected in the modification should be conducted. This overall permission modification process is time consuming.

In our model, the time required to modify the content of the permission token during the delegation process was reduced. Our model is suitable for modifying permission token content by managing permission lists and token lists separately and allows partial delegation. It seems that the new method is time consuming; however, our evaluation shows the opposite results.

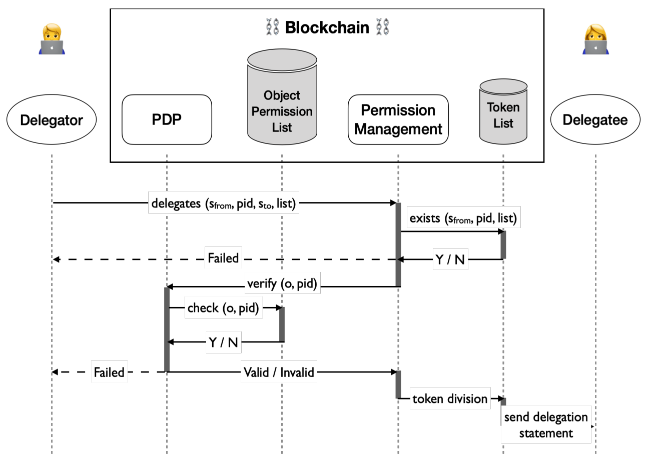

Fig. 10. The Diagram of Our Model

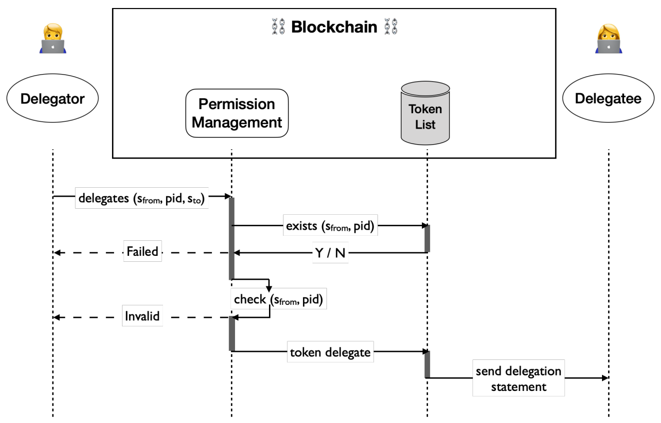

Fig. 11. The BBAC Exisiting Model

## V. Evaluation

In this section, we describe the evaluation method used to demonstrate the effectiveness of the BBAC model and the experimental results. We then verify that the model is meaningful compared to the legacy model.

To prove the performance of our enhanced BBAC model, we measured the time and gas cost to improve the effectiveness of our model compared to that in [4]. We evaluated two different scenarios: generation of a simple delegation and access rights division. To explain this easily, we refer to [4] and this paper as *legacy* and *new*, respectively. The implementation of *legacy* is based on [4].

### A. Experimental Environments

The experimental environment is presented in Table I. We used Hardhat to evaluate the execution time of each smart contract [23]. Hardhat is a development environment that provides various features related to ethereum software, such as compilation, deployment, testing, and debugging.

TABLE I. Experimental Environments

| Parameter | Value |
|---|---|
| Solidity Version | 0.8.4 |
| Lines of Code | 253 |
| CPU | Intel(R) Core(TM) i5-8279U CPU @ 2.40GHz |
| Memory | 16GB 2133MHz LPDDR3 |
| Web3js Version | 1.3.6 |
| Hardhat Version | 2.3.3 |

### B. Scenario 1 - Simple Delegation

The first case involved simple delegation. Resource owners, objects, delegators, and delegatees exist. We assume that Alice is a delegator and Bob is a delegatee, and that there are two sub-permissions in a permission token that index 0 and 1 – *list* [0, 1]. We conducted an experiment divided into the following two steps:

*1) Permission Granting Process*: The owner grants a permission token created, which permits access to an object, to Alice. In this step, a process confirming whether Alice has successfully obtained the new permission token is included by checking for the existence of *the list* [0,1].

*2) Permission Delegation Process*: Alice($s_{from}$) has a permission token granted by the owner of the object by the permission granting process. Then, Bob($s_{to}$) requested Alice to delegate the permission token with *dlist* [0, 1], which is a whole list that is included in the permission token Alice owns, and Alice agreed to the request.

This study checked the time and gas consumption of the permission granting and delegation processes. The results are presented in Table II. Cost comparison between *legacy* and *new* scenarios for Scenario 1.

TABLE II. Cost Comparision between legacy and new for Scenario 1

| Process | Standard | *legacy* | *new* |
|---|---|---|---|
| Permission Granting | time | 135ms | 75ms |
| | gas | 267,158 | 300,475 |
| Permission Delegation | time | 166ms | 156ms |
| | gas | 240,431 | 180,776 |
| Total | time | 301ms | 231ms |
| | gas | 507,589 | 481,251 |

At the permission granting processes, each of the time costs of *legacy* and *new* are 135ms and 75ms, respectively, reduced by 45%. On the other hand, the gas consumption of *legacy* and *new* is 267158 and 300475, respectively, which is an increase of 12%.

Next, in the permission delegation processes, the time cost of *legacy* and *new* is 166ms and 156ms, respectively, which are reduced by 7%. In addition, the gas consumption of *legacy* and *new* is 128831 and 127524, respectively, which is reduced by 2% in our enhanced BBAC model.

In summary, there was a loss of approximately 3300 gases in the permission granting process, while there was 6000 gas gains per permission delegation process. In other words, the more frequently the permission token is modified, the better it is in terms of the gas. As a result, through the evaluation, we can prove that our model is good for environments that are required to easily edit the contents of the permission token that has already been generated.

### C. *Scenario 2 - Division of Permission*

The second case pertains to the division of access rights. The difference with simple delegation is that there exists more than one delegatee to consider the split of the permission token.

For the experiment, we assume that there is a resource owner, an object, and a delegator, similar to a simple division. However, there was a significant difference from the former. For this case, there are two *delegatees* ($s_{to}$) - Bob and Carrol, who want to delegate the part of the permission token of Alice($s_{from}$). The resource owner granted the permission token to Alice, so Alice was able to conduct four different rights - create, read, update, and delete. Each of the permission rights indicated from 0 to 3 – *list* [0, 1, 2, 3]. Bob wants to get permission including index 0 and 1 – *dlist* [0, 1] , while Carrol wants to get permission for the remaining – *dlist* [2, 3]. As a result, Bob is able to conduct action limited to create and read on the object and Carrol has right to update and delete on the object.

In this case, we applied both *legacy* and *new*, and checked the time and gas consumption of each case. For the comparison, we assume that Alice's existing permission token is deactivated; then, two permission tokens – one for creation and reading, and the other for update and delete – are regenerated. Each permission token was delegated to Bob and Carrol in a *legacy*. However, according to our new algorithm, a split of the permission tokens is conducted in the delegation process. Therefore, simply sending different permission indices between Bob and Carrol is required, and the process is short compared to the former case.

As a result, when using *legacy*, it requires a total of 1428031 gases to deactivate the permission that has already existed, two times the permission creation and delegation process. However, the *new* passes only two delegation processes for the permission division and requires only 1250272 gas, which is reduced by 13%. With respect to time, the cost was reduced by 13%, from 339ms to 296ms. In other words, our model has an advantage in terms of the division of permissions compared to [4].

## VI. Discussion

In this section, we address several points related to the enhanced BBAC model proposed in this study.

### A. *Comparison*

First, we evaluated our model and compared it with only [4]. This is because we motivated the study and improved the existing mechanism to be more practical in the division of permission tokens in the delegation process. Therefore, we considered the most important and meaningful comparison with [4] instead of including others.

### B. *Increased Gas*

In our model, the permission-granting gas consumption is increased compared to that in [4]. Additionally, the delegation process requires less gas. However, our enhanced model is optimized for cases in which delegation requests occur frequently.

### C. *Permission Check Process*

Compared with the existing token-constrained BBAC, such as [4], [21], [24], the permission check process is more complex. Most token-constrained BBAC check the request of a subject to an object based on the existence of a permission token. However, in our model, the permission check process adds the permission index list to divide the permission token. Therefore, the gas cost required for the permission-check process increases. However, this is an inevitable consequence; it is a trade-off relationship with the implementation of the access rights division, which is required to modify the parameter structure.

### D. *Time Complexity*

The time complexity problem of BBAC remains. According to [4], the time complexity of the BBAC algorithm is high. However, there is room for improvement, such as using miners of high quality or choosing a consensus algorithm called RAFT to accelerate the block speed [13], [25].

### E. *Single Point of Failure Problem*

Because this model is based on a consortium blockchain, the single point of failure problem cannot be totally ignored. This is because being based on a consortium blockchain implies that there is a separate entity that determines which subjects or objects will participate in the corresponding blockchain. Additionally, it is difficult to completely rule out the possibility that an entity may act as a single point of failure. However, when BBAC is introduced, the possibility of a single point of failure is significantly lowered, and the transparency of access control can be increased simultaneously. Therefore, the introduction of BBAC is meaningful for resolving a single point of failure problem.

### F. *Improvement for BBAC*

As BBAC has recently been discussed actively, except for the division of permission, there are various parts of BBAC that must be improved, including the time complexity problem. However, this study focuses on permission division to improve feasibility. Other challenges for BBAC will be addressed in future work.

## VII. Conclusion

In this study, we propose a blockchain-based access control (BBAC) model for IoT and emphasize the risk caused by the greatest privilege vulnerability. To address this risk, we propose a new delegation algorithm that redefines the structures to facilitate permission division with reference to [4]. This study proves the possibility and effectiveness of our enhanced BBAC model. In effect, although even after adding more parameters to implement the division of access rights, the gas cost for the overall process has reduced. In particular, our model is appropriate for cases that require addressing many delegation requests.

## VIII. Acknowledgments

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-01697 Development of Automated Vulnerability Discovery Technologies for Blockchain Platform Security, No. 2019-0-01343 Regional Strategic Industry Convergence Security Core Talent Training Business, and No. IITP-2021-2020-0-01819 ICT Creative Consilience program).

## References

[1] P. Scully, “Top 10 IoT applications in 2020 - Which are the hottest areas right now?,” *IoT Analytics*. 2020, [Online]. Available: https://iot-analytics.com/top-10-iot-applications-in-2020/.

[2] CISCO, “Cisco Annual Internet Report (2018–2023)”, *CISCO*, 2018.

[3] Z. K. Zhang, M. C. Y. Cho, C. W. Wang, C. W. Hsu, C. K. Chen, and S. Shieh, “IoT security: Ongoing challenges and research opportunities,” *Proc. - IEEE 7th Int. Conf. Serv. Comput. Appl. SOCA 2014*, pp. 230–234, 2014, doi: 10.1109/SOCA.2014.58.

[4] J. Shi, R. Li, and W. Hou, “A Mechanism to Resolve the Unauthorized Access Vulnerability Caused by Permission Delegation in Blockchain-Based Access Control,” *IEEE Access*, vol. 8, pp. 156027–156042, 2020, doi: 10.1109/ACCESS.2020.3018783.

[5] S. Ding, J. Cao, C. Li, K. Fan, and H. Li, “A Novel Attribute-Based Access Control Scheme Using Blockchain for IoT,” *IEEE Access*, vol. 7, pp. 38431–38441, 2019, doi: 10.1109/ACCESS.2019.2905846.

[6] H. Liu, D. Han, and D. Li, “Fabric-iot: A Blockchain-Based Access Control System in IoT,” *IEEE Access*, vol. 8, pp. 18207–18218, 2020, doi: 10.1109/ACCESS.2020.2968492.

[7] O. Novo, “Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT,” *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1184–1195, Apr. 2018, doi: 10.1109/JIOT.2018.2812239.

[8] A. Outchakoucht and J. P. Leroy, “Dynamic Access Control Policy based on Blockchain and Machine Learning for the Internet of Things,” *International Journal of Advanced Computer Science and Applications*, 2017. [Online]. Available: www.ijacsa.thesai.org.

[9] S. Pal, T. Rabehaja, A. Hill, M. Hitchens, and V. Varadharajan, “On the Integration of Blockchain to the Internet of Things for Enabling Access Right Delegation,” *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2630–2639, Apr. 2020, doi: 10.1109/JIOT.2019.2952141.

[10] O. J. A. Pinno, A. R. A. Grégio, and L. C. E. De Bona, “ControlChain: A new stage on the IoT access control authorization,” in *Concurrency Computation* , Jun. 2020, vol. 32, no. 12, doi: 10.1002/cpe.5238.

[11] 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.

[12] Q. Wang, N. Li, and H. Chen, “On the Security of Delegation in Access Control Systems,” *Eur. Symp. Res. Comput. Secur.*, vol. 5283, pp. 317–332, 2008.

[13] Y. Feng, W. Zhang, X. Luo, and B. Zhang, “A Consortium Blockchain-based Access Control Framework with Dynamic Orderer Node Selection for 5G-enabled Industrial IoT,” *IEEE Trans. Ind. Informatics*, vol. 3203, no. c, pp. 1–9, 2021, doi: 10.1109/TII.2021.3078183.

[14] B. Bera, S. Saha, A. K. Das, and A. V. Vasilakos, “Designing blockchain-based access control protocol in iot-enabled smart-grid system,” *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5744–5761, 2021, doi: 10.1109/JIOT.2020.3030308.

[15] S. Saha, A. K. Sutrala, A. K. Das, N. Kumar, and J. J. P. C. Rodrigues, “On the Design of Blockchain-Based Access Control Protocol for IoT-Enabled Healthcare Applications,” *IEEE Int. Conf. Commun.*, vol. 2020-June, pp. 1–6, 2020, doi: 10.1109/ICC40277.2020.9148915.

[16] 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. Informatics*, vol. 16, no. 5, pp. 3521–3530, 2020, doi: 10.1109/TII.2019.2925898.

[17] R. S. Sandhu and P. Samarati, “Access control: Principles and Practice,” *IEEE Commun. Mag.*, vol. 32, no. September, pp. 40–48, 1994.

[18] J. L. Hernández-Ramos, A. J. Jara, L. Marín, and A. F. Skarmeta Gómez, “DCapBAC: embedding authorization logic into smart things through ECC optimizations,” *Int. J. Comput. Math.*, vol. 93, no. 2, pp. 345–366, 2016, doi: 10.1080/00207160.2014.915316.

[19] G. Das D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, “Everything You Wanted to Know About the Blockchain: Its Promise, Components, Processes, and Problems,” *IEEE Consum. Electron. Mag.*, vol. 7, no. 4, pp. 6–14, 2018, doi: 10.1109/MCE.2018.2816299.

[20] V. Buterin, “A next-generation smart contract and decentralized application platform,” *white paper*, 2014.

[21] R. Xu, Y. Chen, E. Blasch, and G. Chen, “BlendCAC: A BLockchain-ENabled Decentralized Capability-based Access Control for IoTs,” *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, pp. 1027-1034, doi: 10.1109/Cybermatics_2018.2018.00191.

[22] 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, 2019, doi: 10.1109/JIOT.2018.2847705.

[23] Nomic Labs LLC, “No Title.” https://hardhat.org/.

[24] Y. E. Oktian and S. G. Lee, “BorderChain: Blockchain-Based Access Control Framework for the Internet of Things Endpoint,” *IEEE Access*, vol. 9, pp. 3592–3615, 2021, doi: 10.1109/ACCESS.2020.3047413.

[25] D. Kim, I. Doh and K. Chae, "Improved Raft Algorithm exploiting Federated Learning for Private Blockchain performance enhancement," *2021 International Conference on Information Networking (ICOIN)*, 2021, pp. 828-832, doi: 10.1109/ICOIN50884.2021.9333932.