Advances in Science, Technology and Engineering Systems Journal Vol. 6, No. 1, 651-657 (2021)

www.astesj.com

Special Issue on Multidisciplinary Sciences and Engineering

ASTESJ

ISSN: 2415-6698

A Novel Blockchain-Based Authentication and Access Control Model for Smart Environment

Nakhoon Choi, Heeyoul Kim*

Department of Computer Science and Engineering, Kyonggi University, Suwon 16227, Republic of Korea

A R T I C L E I N F O

Article history:
Received: 28 November, 2020
Accepted: 18 January, 2021
Online: 05 February, 2021

Keywords:
Blockchain
Access Control
Smart Environment

A B S T R A C T

With the increase of smart factories and smart cities following the recent 4th industrial revolution, internal user authentication and authorization have become an important issue. The user authentication model using the server-client structure has a problem of forgery of the access history caused by the log manipulation of the administrator and unclearness of the responsibility. In addition, users must independently manage the authentication method for each service authentication. In this paper, to solve the above problem, the researchers propose an integrated ID model based on a hybrid blockchain. The proposed model is implemented as two layers of Ethereum and Hyperledger Fabric: the former layer is responsible for integrated authentication, and the latter layer is responsible for access control. The physical pass or application for user authentication and authorization are integrated to one ID through the proposed model. In addition, the decentralized blockchain ensures the integrity and transparency of the stored access history, and it also provides non-repudiation of authority and access history.

1. Introduction

This paper is an extension of work originally presented in ICACT 2020 (International Conference on Advanced Communication Technology) [1].

With the increase of smart factories and smart cities in accordance with the 4th industrial revolution, user authentication and access control are emerging as important issues. A smart factory automates factory facilities to reduce process failure rates and is operated by a small number of employees [2]. However, there is a problem of responsibility in case of an accident. In addition, users of smart city may need different authentication means for each city service, and there is a risk of manipulation of access history.

The administrator of the centralized access control model of the existing server-client structure can modify and delete the access history. With the above problem, it is possible to avoid responsibility through internal corruption in case of an accident. In addition, the user proves the authority through different authentication methods including physical means and mobile applications for each service. This makes it inconvenient for users to issue and manage various authentication methods, and it is troublesome for users to prove their authority history and career.

*Corresponding Author: Heeyoul Kim, Department of Computer Science and Engineering, Kyonggi University, Suwon, 16227, Korea, Email: heeyoul.kim@kgu.ac.kr

To solve the above problem, the authors in [3] proposed integrated authentication using smart cards for integrated access control of various services. However, there is a problem that the administrator can modify the authority and access details arbitrarily, such as the access control list or the capability list.

In order to solve the above problem through the blockchain, the model in [4] stores the information in DHT(Distributed Hash Tables) and performs access control for information based on the Bitcoin system. However, it is not possible to store access details, and there is a problem of processing speed with low TPS(Transaction per Second) of Bitcoin. Also, it is necessary to verify practicality and suitability. These studies are inadequate for authentication and access control in smart environments.

This paper proposes an integrated authentication and access control model in smart environments that solves the problems above with the help of blockchain technology. The proposed model processes user authentication and access request through a hybrid blockchain system using private and public blockchain together. The proposed model enables users to acquire and prove various access rights through one authentication means, thus it integrates the authentication of various smart factory services and smart city services. In addition, the proposed model prevents problems such as access history manipulation that occurs in the existing central model based on the integrity and transparency of the blockchain. Moreover, the user's history recorded in the blockchain can be easily used as the proof of the user's career.

The rest of this paper presents the following. Section 2 introduces blockchain platforms utilized in the proposed model. Section describes the architecture and process of the model. Section 4 displays the results of implementing the model. Section 5 provides application scenario using the model and Section 6 contains conclusions.

2. Background

The flexible connection of vertical or horizontal organizations became important in the wake of the Fourth Industrial Revolution [5]. This paper uses blockchain platform to build reconfigurable network of existing organizations. The integrity of information is secured in the relationship of organizations through the decentralized characteristics of blockchain. Users can also benefit from all the benefits without noticing that the blockchain has been used on the surface.

2.1. Ethereum

Ethereum [6] is a representative public blockchain using PoW(Proof of Work) [7] consensus algorithm like Bitcoin. PoW consensus algorithm is the process of selecting block creators among unreliable participants on blockchain networks. Participants compete to find a hash value that matches the target condition to elect a miner. The public blockchain introduces the economic concept of cryptocurrency, which leads to the correct consensus by compensating the cryptocurrency to the miner elected. The block generated through the consensus is distributed to the participants of the blockchain network and is connected to the previous block through the 'previous block hash' included in the block header to ensure the integrity of the previous block. All blocks and transactions are open to the public with ensuring transparency, and the decentralized network evolves continuously without administrators.

Ethereum supports the creation and distribution of smart contract [8] for the development of DApp(decentralized Application). A smart contract is written in Solidity, a Turing completeness language, and recorded in the distributed ledger with ensuring the integrity of the results according to automated execution and input [9]. Figure 1 represents the operation of the smart contract in Ethereum. The smart contract made with Solidity is compiled and stored in the form of the EVM(Ethereum Virtual Machine)-bytecode in the ledger. Calling the code through Ethereum clients such as Geth(Go-Ethereum) is executed in the EVM environment.

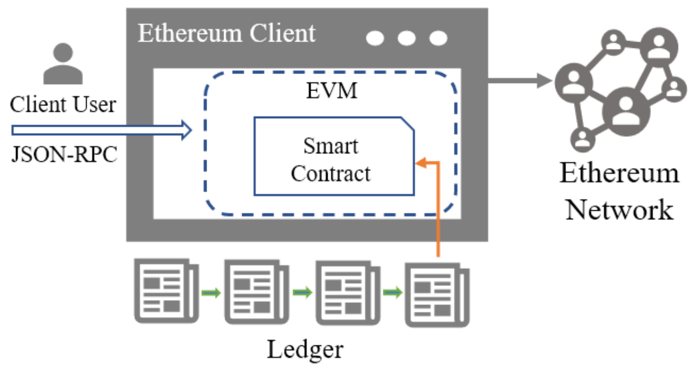


Figure 1: Operation of Smart Contract in Ethereum Client

2.2. Hyperledger Fabric

Hyperledger Fabric [10] (hereinafter referred to as "Fabric"), as part of the Linux Foundation's Hyperledger project, is a representative private blockchain platform commonly used for enterprise environments. Unlike public blockchains, it uses certificates and PKI(Public Key Infrastructure) to authenticate participants with a restricted network, and is managed through a MSP(Membership Service Provider), an authentication management system that defines the role and access rights of nodes in the network.

A peer is a node of organizations that compose the Fabric network and manages ledgers and chaincodes, and the peers are divided into endorser, committee, anchor, and leader according to their roles. Transactions for the execution of chaincodes in the network are performed through three steps: execution, ordering and validation. The chaincode plays the same role as Ethereum's smart contract and updates or queries data to the ledger. Currently, development of chaincodes is possible through Golang, Node.js, etc. Fabric does not require much resources in the consensus, unlike PoW, by ordering transactions through Ordering service to create blocks. Therefore, it has much better performance than the public blockchain platforms.

2.3. Metamask

The blockchain wallet stores the user's personal key as well as managing the balance of the blockchain account, and creates a transaction on behalf of the user and submits it to the network. In this sense, wallet is the easiest approach to using blockchain for normal users. The proposed model uses Metamask [11] for convenient use in PC or mobile environment.

Metamask is the most representative web-based Ethereum wallet extension program executed in browser, and is currently supported by Chrome, Firefox, Opera, etc., and beta test is being conducted in mobile environment. It supports Ethereum account management, digital signature generation, and transaction creation. users can easily access the smart contract of the blockchain through Metamask.

In order for users to access the Ethereum network, they must participate in the network as a blockchain node through the Ethereum client. This method lowers user accessibility, so Metamask uses Infura API [12], a cloud service for network connection to wallet users. Through the Metamask, users can access the Ethereum network in the web environment without operating the node directly, and operate the smart contract and receive the results.

3. A Blockchain-based Authentication and Access Control Model

3.1. Model Architecture

The proposed model uses a hierarchical hybrid blockchain where a public Ethereum network is connected to multiple private Fabric networks. Figure 2 shows the layered architecture of the proposed model. The public layer based on Ethereum is an integrated authentication management layer. The private layer is composed of various subsystems for smart factories and smart cities (hereinafter referred to as "Organization"), and each organization operates Fabric network separately for authorization and access control.

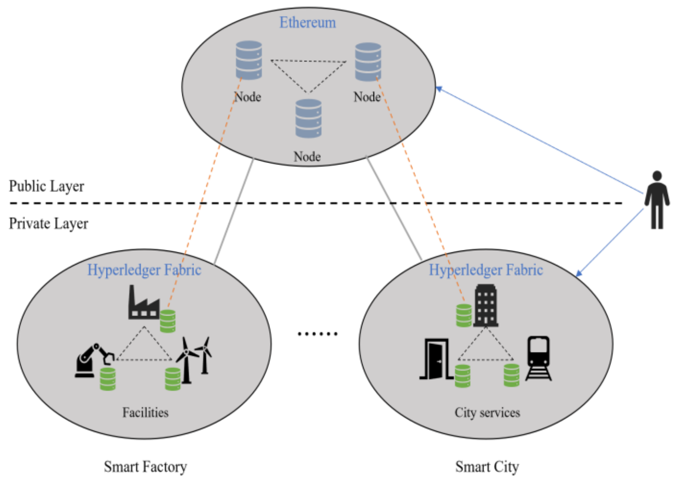


Figure 2: Layered Architecture of the Proposed Model

The proposed model uses a hierarchical hybrid blockchain where a public Ethereum network is connected to multiple private Fabric networks. Figure 2 shows the layered architecture of the proposed model. The public layer based on Ethereum is an integrated authentication management layer. The private layer is composed of various subsystems for smart factories and smart cities (hereinafter referred to as "Organization"), and each organization operates Fabric network separately for authorization and access control.

3.1.1. Public Layer - Integrated Authentication Management

Through this public layer, users create their own IDs based on their Ethereum accounts. The ID of a user is composed of the user's Ethereum account, a secret computed by hashing the user's personal information, a set of tokens used for user authentication by organizations, and a contact method (e.g., E-Mail) as shown in Figure 3. The proposed model provides a smart contract named IAM (Integrated Authentication Manager) utilized by the users to create and control their IDs. The functions in the IAM is described in Table 1, and these functions except for query functions have an access control mechanism on the IDs based on the function modifier in Solidity. In other words, it is guaranteed that only the legitimate owner of the ID can modify the secret and register tokens via these functions.

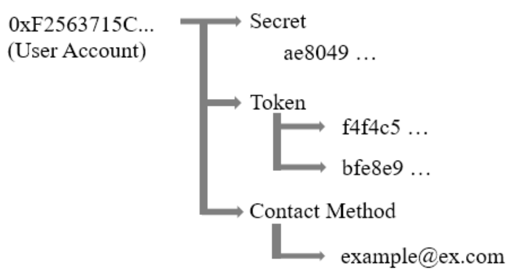


Figure 3: The structure of a user's ID

Table 1: Description of Functions in the IAM Smart Contract

Function	Purpose
createId(secret)	To create an ID and register the secret based on the user's Ethereum account
modifySecret(secret)	To modify the secret registered
regToken (token, tkName)	To register tokens and their aliases for the authentication by organizations
queryUser(secret)	To find the registered user by given secret and to retrieve user information
queryByToken(token)	To verify whether the token was successfully registered by the user

3.2.2. Private Layer – Access Control Management

In the private layer, there may be several Organizations managing smart environments respectively. Figure 4 represents the architecture of an Organization. Several nodes compose a Hyperledger Fabric network for the Organization. Among them, the admin node provides the ordering service for consensus and is handled by the administrator. The Ethereum sync node is connected to the Ethereum in the public layer and responsible for delivering the user's ID information for authentication.

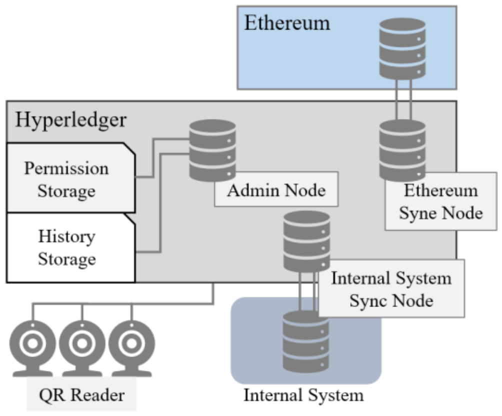


Figure 4: Architecture of an Organization in the Private Layer

Our model provides role-based access control (RBAC), and the role and access rights of each user are recorded in the Fabric blockchain via the chaincode named PM (Permission Manager). Table 2 shows the functions provided by the PM, and only the administrator can execute these functions for creation and modification of the authority. The access history is written by another chaincode called by the client application. The information stored in the Fabric blockchain is transparent and open to the legitimate participants in the Organization.

Table 2: Description of Functions in the PM Chaincode

Function	Purpose
userReg(secret)	To register a new user identified by the secret
permSet(account, role)	To set the role and authorization of the user having the given account
queryPerm(account)	To check and verify the user's role and access rights

3.2. Process Description

3.2.1. Creating an Integrated ID

A new user joining in the proposed model first performs the process of creating an integrated ID. The user first generates a secret by hashing his personal information as follows:

$$secret = Keccak\text{-}256(Birth,\ Name,\ Phone) \qquad (1)$$

Formula 1 shows the creation of the user's secret, and Keccak-256 is a hash function adopted in the SHA-3 competition of the NIST (National Institute of Standards and Technology) [13].

Then, with assuming that the user already has an Ethereum account, he makes a transaction via Metamask to request the smart contract IAM for the creation of his integrated ID. Since the transaction is signed by his Ethereum private key, the IAM can identify and authenticate the requester. The secret can be seen by anyone after recorded in the blockchain. However, his personal information is not exposed due to the one-wayness of the hash function.

3.2.2. Registering User's Access Rights in Organization

When a user who has completed creating his ID newly joins in a specific Organization, his access right is registered by the process shown in Figure 5. The detailed process is as follows.

1) A new user Alice provides his personal data used for computing secret to the Organization.
2) The internal system of the Organization computes secret of the user by Equation (1), and then send it to the Ethereum sync node in the Fabric network.
3) The Ethereum sync node approaches the smart contract IAM to obtain the user's ID with the received secret.
4) The admin node issues a user-specific validation token T_{Alice}. The purpose of this token is to verify the ownership of the ID obtained in step 3). This token is the hash value of a seed generated per each user, and it is signed by the private key of the admin node (Equation (2)). Then, the token is sent to the user.

$$T_{Alice} = Sign(Keccak\text{-}256(seed_{Alice}),\ Org\ admin) \qquad (2)$$

5) Alice registers the received token in his integrated ID through the smart contract IAM. This step only can be done by the creator of the ID because the IAM compares the requester of this register transaction with the owner of the target ID.
6) If the registration of T_{Alice} is successfully accepted by the IAM contract, the Ethereum sync node can recognize it and is convinced of Alice's ID ownership.
7) The administrator registers both the authority information of Alice and his Ethereum account in the permission storage of the organization.

3.2.3. Access Control

Our model provides a role-based access control with registered users' authorization information. Figure 6 shows the process when Alice requests access to the manufacturing control system in a smart factory. A QR reader attached to the control system is used to interact with the user. Our model determines whether the user's role has appropriate access right or not as follows.

1) Alice generates a QR code which includes both a timestamp for preventing replay attack and a digital signature generated by his Ethereum private key. Our model uses ECDSA [14] based on elliptic curve 'secp256k1' as well as Ethereum. Let G be the base point of the curve E where n is the order of G. Alice's private key is d, and his public key is $Q=dG$. The signature $(r,\ s,\ v)$ is computed as follows.

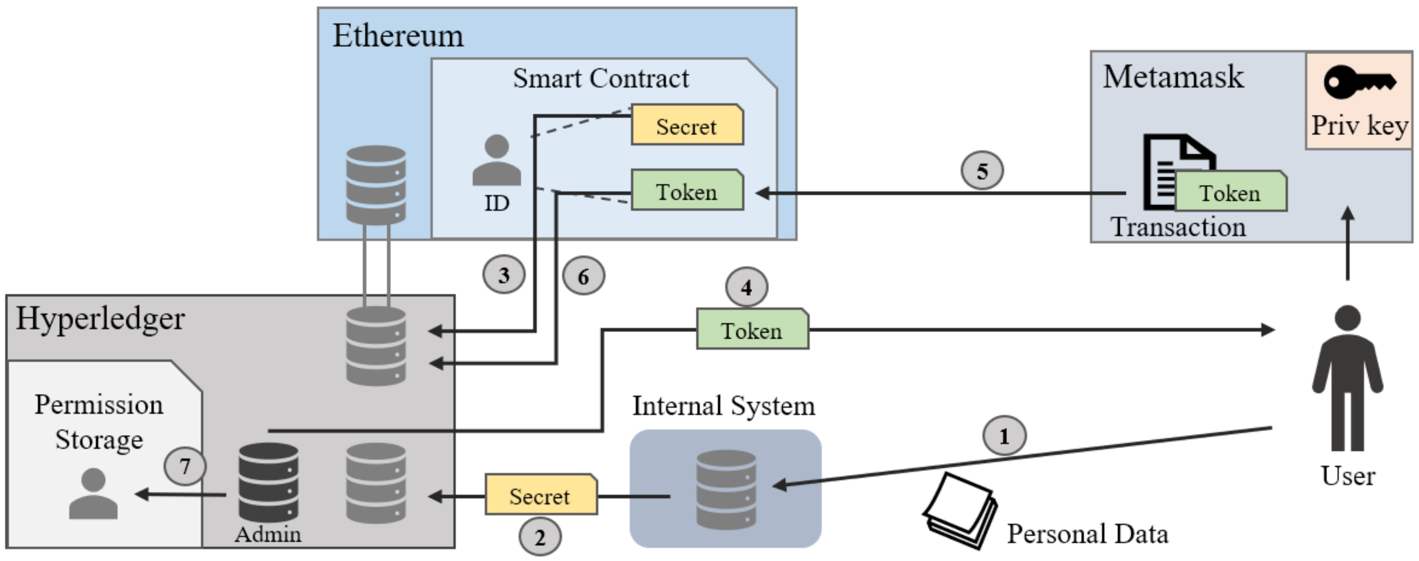


Figure 5: The Process of Registering User's Access Rights in an Organization

① Calculate the hash value $h = hash(m)$ of the message m to be signed, and let z be the l leftmost bits of h where l is the bit length of n.
② Generates a random number k within $[0 \sim n\text{-}1]$.
③ Compute $R = kG$ and obtain the x-coordinate r of point R.
④ Compute $s = k^{-1}(z + rd) \bmod n$.
⑤ Compute $v = 27 + (y \% 2)$ (y: y-coordinate of Q).
⑥ (r, s, v) is the digital signature value.

2) The QR reader scans the QR code generated by Alice, and as a Fabric client it delivers the timestamp, signature, and the *object_serial* (ID of the QR reader) to the Fabric node.
3) The Fabric node verifies received signature (r, s, v) with the help of Ethereum's *ecRecover()* method. If it is valid, as a result of this method call, the node discovers Alice's Ethereum account.
4) The authority information of the discovered Ethereum account is retrieved from the permission storage.
5) If Alice's role has been approved for access to the subject to which the *object_serial* is assigned, this request is granted.
6) Alice's access attempt is recorded in the history storage.

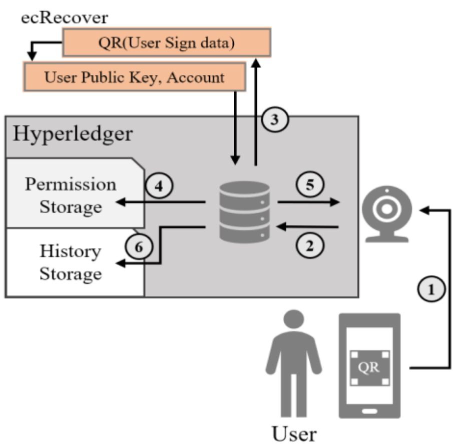


Figure 6: The Process of Access Control in an Organization

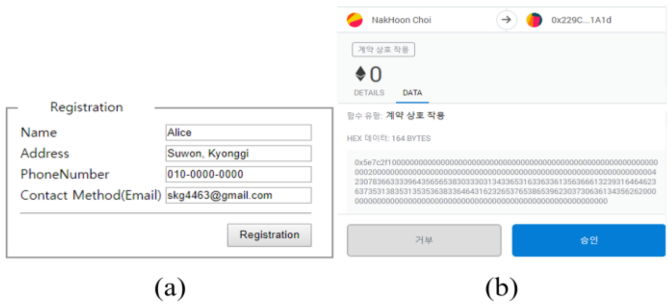
4. Implementation and Experiment

The proposed model was implemented, and its soundness was confirmed through an experiment. Table 3 describes the implementation environment of each component. The web dApp runs on the Chrome browser with Metamask installed, and it registers the user's secret for creating ID and the Organization's token for registering the user's access rights. The smart contract IAM and the chaincode PM provides the functions described in Section 3.1. QR generator is an application where the user's Ethereum private key is stored, and it generates a QR code containing the user's signature to obtain access right. QR reader is a Fabric client attached to the protected resources.

Table 3: Development Environment

Web dApp	Environment	Chrome, Metamask
	Language	JavaScript
Smart Contract IAM	Environment	Ethereum Ropsten Testnet
	Language	Solidity ^0.5.0
Chaincode PM	Environment	Hyperledger Fabric 2.1
	Language	Node.js
QR Generator	Environment	Android Emulator
	Language	Java
QR Reader	Environment	Raspberry Pi
	Language	Node.js

Figure 7 shows the screenshots when a user creates his integrated ID via the web dApp. As can be seen in Figure 8(a), a user inputs his personal data to compute secret. Then, the installed Metamask generates and transmits an Ethereum transaction requesting creation of the user's integrated ID.


Registration
Name: Alice
Address: Suwon, Kyonggi
PhoneNumber: 010-0000-0000
Contact Method(Email): skg4463@gmail.com

NakHoon Choi → 0x229C...1A1d
계약 상호 작용
♦0
함수 유형: 계약 상호 작용
HEX 데이터: 164 BYTES

```
0x5e7c2f100000000000000000000000000000000000000000000000000000000
0002000000000000000000000000000000000000000000000000000000000004
2307836633339643565653830333031343365316336336135636661323931646462
3637353138353135353638336464316232653765386539623037306361343562620000
00000000000000000000000000000000000000000000000000000
```

Figure 7: Screenshots when Creating Integrated ID. (a) Input of User Data for Computing Secret, (b) Generation of Ethereum transaction with Metamask

Figure 8 shows the logs when an organization registers a user Alice's role after verifying authentication. With given Alice's personal data, the organization computes Alice's secret (Figure 8(a)). Then, the organization checks whether Alice's integrated ID exists or not in the smart contract IAM by calling the *queryUser(secret)* function. If the ID exists, a validation token is issued for Alice (Figure 8(b)). After Alice successfully registers his token in his ID, the organization's Ethereum sync node can detect this event by implementing an event listener for IAM. As can be seen in Figure 8(c), the organization discovers Alice's Ethereum account from this event, and then assigns Alice's role and access right. In this experiment, the role 'level_2' is assigned to Alice and he is granted to access 'object 001'.

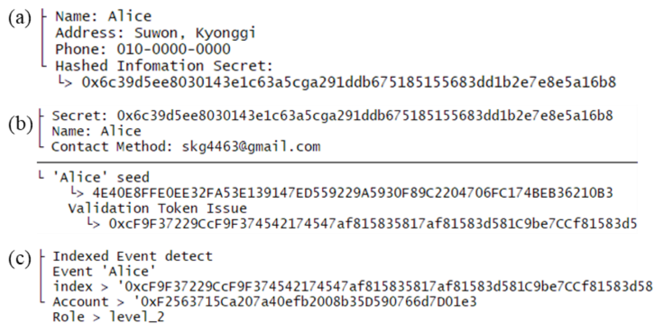
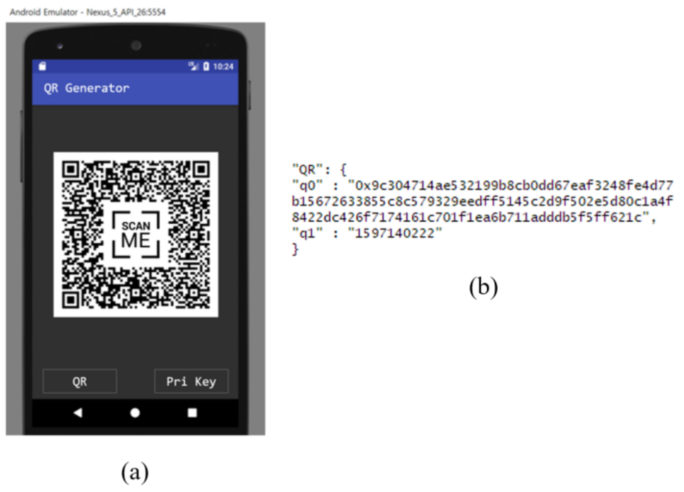

(a)
```
├ Name: Alice
│ Address: Suwon, Kyonggi
│ Phone: 010-0000-0000
└ Hashed Infomation Secret:
   └> 0x6c39d5ee8030143e1c63a5cga291ddb675185155683dd1b2e7e8e5a16b8
```
(b)
```
├ Secret: 0x6c39d5ee8030143e1c63a5cga291ddb675185155683dd1b2e7e8e5a16b8
│ Name: Alice
└ Contact Method: skg4463@gmail.com
──────────────────────────────────────────────────────────────
└ 'Alice' seed
    └> 4E40E8FFE0EE32FA53E139147ED559229A5930F89C2204706FC174BEB36210B3
   Validation Token Issue
      └> 0xcF9F37229CcF9F374542174547af815835817af81583d581C9be7CCf81583d5
```
(c)
```
├ Indexed Event detect
│ Event 'Alice'
│ index > '0xcF9F37229CcF9F374542174547af815835817af81583d581C9be7CCf81583d58
└ Account > '0xF2563715Ca207a40efb2008b35D590766d7D01e3
  Role > level_2
```
Figure 8: Logs when Registering Alice's Access Right in an Organization. (a) Computation of Alice's Secret by the Organization, (b) Issuance of a Validation Token for Alice, (c) Detection of Token Registration and Assignment of Alice's Role

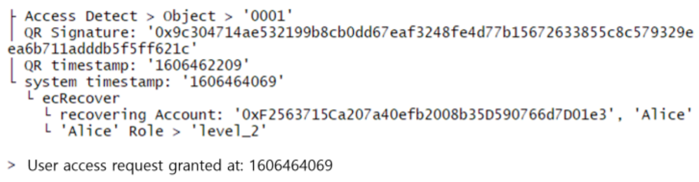
Figure 9 shows an example QR code generated when Alice wants to access a protected resource in the organization. The QR generator installed in Alice's smartphone generates this QR code by using of the stored private key (Figure 9(a)). As can be seen in Figure 9(b), the QR code contains two values (q0 and q1), where q0 is Alice's ECDSA signature and q1 is the Unix timestamp. Then, Alice provides this QR code to the QR reader attached to the resource.



```
"QR": {
"q0" : "0x9c304714ae532199b8cb0dd67eaf3248fe4d77
b15672633855c8c579329eedff5145c2d9f502e5d80c1a4f
8422dc426f7174161c701f1ea6b711adddb5f5ff621c",
"q1" : "1597140222"
}
```

Figure 9: An Example QR Code Generated for Access. (a) Screenshot of QR Generator, (b) The Information stored in the QR Code

The QR reader transmits both its *object_serial* ('0001' in Figure 10) and the signature obtained by scanning the QR code to the Fabric node. The node recovers Alice's Ethereum account by using of the *ecRecover* function. Also, it verifies the validity of both ECDSA signature and timestamp. Then, it can get Alice's registered role from the permission storage with the account, as shown in Figure 10. Finally, the node decides whether to grant or deny Alice's access request based on his role, and then it informs the QR reader of the decision result.



```
├ Access Detect > Object > '0001'
│ QR Signature: '0x9c304714ae532199b8cb0dd67eaf3248fe4d77b15672633855c8c579329e
ea6b711adddb5f5ff621c'
│ QR timestamp: '1606462209'
└ system timestamp: '1606464069'
   └ ecRecover
      └ recovering Account: '0xF2563715Ca207a40efb2008b35D590766d7D01e3', 'Alice'
      └ 'Alice' Role > 'level_2'

>  User access request granted at: 1606464069
```

Figure 10: Logs when Alice's Access Request is Granted

5. Application Scenario using the Proposed Model

To help understanding the proposed model and claim its usefulness, we provide an application scenario using the model from a user point of view in this section. Let us assume that a new user Alice having an Ethereum account Eth_A decided to use the proposed model. He visits the web dApp and creates his own integrated ID (ID_A) by registering both his email address and secret based on his personal data.

Since then, Alice has moved into the smart city K-city (Organization #1) to seek employment. The K-city already has adopted the model extensively, and it participates as an organization. He visits the government office of the city to register himself. The office checks the existence of ID_A and then issues a token T_{Alice}^{Org1}. Alice registers this token in his ID_A by using of his Ethereum private key. Then, the office assigns the role 'citizen' to Alice and registers his access right to use various public facilities in the city.

Now Alice can prove his identity and use various facilities with his smartphone where the QR generator application is installed. Suppose Alice visits a public library to get hiring information. He just generates a QR code and shows it to the QR reader attached to the gate of the library. Then, his access is allowed, and he can enter the library.

Before long, Alice is hired by a smart factory F-factory (Organization #2). Similarly, in the enrollment process, he can easily register himself with a newly issued token T_{Alice}^{Org2}, without creating a new identity to be used in the factory. Then, the role 'engineer' is assigned to Alice and his access right for the control panel in the factory is registered. As a result, he can control the panel by generation a valid QR code whenever he needs to access it while at work.

Suppose someday Alice commits a mistake in operating some equipment in F-factory, which causes enormous damage to the factory. He gets into a panic, and he may try to avoid suspicion by deleting or manipulation evidences such as CCTV videos. However, since the access history of the users is stored in the immutable blockchain ledger of the F-factory nodes, he cannot hide the fact that he accessed the equipment at that time. Thus, the F-factory can identify Alice's behavior and call him to account.

In another case, suppose an attacker Trudy pretending as Alice tries to access the control panel. He cannot generate a valid QR code because he cannot discover Alice's private key. The attacker may eavesdrop Alice's QR code and try to use it later. However, the timestamp in the QR code prevents this kind of replay attack. Moreover, even if a security incident occurs, the factory can perform a security audit with the immutable access history.

6. Conclusion

Currently many companies and organizations use a centralized model for authentication and access control, so there is internal corruption caused by manipulating access history and it is possible to avoid responsibility in case of an accident. This paper proposes an access control model for smart factories and smart cities using blockchain, a decentralization platform, to solve the problems of existing centralized model and to ensure the integrity and immutability of access history.

The proposed model allows a user to prove their access rights through one integrated ID based on his Ethereum account without creating a new ID separately for each organization. In the aspect of an organization, it is needless to establish separate authentication mechanism, and the organization can easily authorize the users belong to it to access its protected resources. In addition, the proposed model records the access history in the internal blockchain of the organization not only to prevent arbitrary modification and deletion but also to provide non-repudiation.

The proposed model provides a clue to how various smart environments cooperate closely. Both the concept of integrated ID and the hybrid blockchain combining public blockchain and private blockchain can help secure, pragmatic, and user-friendly smart systems. In addition, flexible scalability in a smart environment is secured through a blockchain, and reliability of services and information provided to users is guaranteed.

Acknowledgement

This research was supported by Ministry of Science ICT Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education(2018R1C1B6002903). This work was supported by Kyonggi University's Graduate Research Assistantship 2020.

References

[1] N. Choi, H. Kim, "Hybrid Blockchain-based Unification ID in Smart Environment," International Conference on Advanced Communication Technology, ICACT, **2020**, 166–170, 2020, doi:10.23919/ICACT48636.2020.9061430.
[2] G. Büchi, M. Cugno, R. Castagnoli, "Smart factory performance and Industry 4.0," Technological Forecasting and Social Change, **150**(October 2019), 119790, 2020, doi:10.1016/j.techfore.2019.119790.
[3] G. Moukhliss, R. Filali Hilali, H. Belhadaoui, "A smart card digital identity check model for university services access," ACM International Conference Proceeding Series, **Part F1481**, 3–6, 2019, doi:10.1145/3320326.3320401.
[4] G. Zyskind, O. Nathan, A.S. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," Proceedings - 2015 IEEE Security and Privacy Workshops, SPW 2015, 180–184, 2015, doi:10.1109/SPW.2015.27.
[5] A. Mushtaq, I.U. Haq, "Implications of blockchain in industry 4.O," 2019 International Conference on Engineering and Emerging Technologies, ICEET 2019, 1–5, 2019, doi:10.1109/CEET1.2019.8711819.
[6] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project Yellow Paper, **151**(2014), 1–32, 2014.
[7] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Manubot, 2019.
[8] N. Szabo, "Smart contracts: building blocks for digital markets," EXTROPY: The Journal of Transhumanist Thought,(16), **18**(2), 1996.
[9] K. Christidis, M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," IEEE Access, **4**, 2292–2303, 2016, doi:10.1109/ACCESS.2016.2566339.
[10] C. Cachin, "Architecture of the hyperledger blockchain fabric," in Workshop on distributed cryptocurrencies and consensus ledgers, 2016.
[11] Metamask Inc, Metamask, Website Https://Metamask.Io/,.
[12] The Infura Inc, INFURA, Website Https://Infura.Io/,.
[13] S. Chang, R. Perlner, W.E. Burr, J.M. Kelsey, L.E. Bassham, "Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition NISTIR 7896 Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition."
[14] D.R.L. Brown, "SEC 1: Elliptic curve cryptography," Certicom Research, v2, 2009.