Smart contract token-based privacy-preserving access control system for industrial Internet of Things

Wang, Weizheng; Huang, Huakun; Yin, Zhimeng; Gadekallu, Thippa Reddy; Alazab, Mamoun; Su, Chunhua

Published in:
Digital Communications and Networks

Published: 01/04/2023

Document Version:
Final Published version, also known as Publisher’s PDF, Publisher’s Final version or Version of Record

License:
CC BY-NC-ND

Publication record in CityU Scholars:
Go to record

Published version (DOI):
10.1016/j.dcan.2022.10.005

Publication details:

Citing this paper
Please note that where the full-text provided on CityU Scholars is the Post-print version (also known as Accepted Author Manuscript, Peer-reviewed or Author Final version), it may differ from the Final Published version. When citing, ensure that you check and use the publisher's definitive version for pagination and other details.

General rights
Copyright for the publications made accessible via the CityU Scholars portal is retained by the author(s) and/or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights. Users may not further distribute the material or use it for any profit-making activity or commercial gain.

Publisher permission
Permission for previously published items are in accordance with publisher's copyright policies sourced from the SHERPA RoMEO database. Links to full text versions (either Published or Post-print) are only available if corresponding publishers allow open access.

Take down policy
Contact lbscholars@cityu.edu.hk if you believe that this document breaches copyright and provide us with details. We will remove access to the work immediately and investigate your claim.

Download date: 03/08/2023
Smart contract token-based privacy-preserving access control system for industrial Internet of Things

Weizheng Wang\textsuperscript{a,2}, Huakun Huang\textsuperscript{e,1,*}, Zhimeng Yin\textsuperscript{a,2}, Thippa Reddy Gadekallu\textsuperscript{c}, Mamoun Alazab\textsuperscript{d}, Chunhua Su\textsuperscript{h,**}

\textsuperscript{a} Department of Computer Science, City University of Hong Kong, Hong Kong SAR, 999077, China
\textsuperscript{b} Division of Computer Science, University of Aizu, Fukushima, 965-8580, Japan
\textsuperscript{c} School of Information Technology and Engineering, Vellore Institute of Technology, Tamil Nadu, India
\textsuperscript{d} College of Engineering, IT and Environment, Charles Darwin University, Australia
\textsuperscript{e} School of Computer Science and Cyber Engineering, Guangzhou University, China

\textbf{ARTICLE INFO}

\textbf{Keywords:}
Blockchain
Privacy preservation
Smart contract
Industrial IoT

\textbf{ABSTRACT}

Due to mobile Internet technology’s rapid popularization, the Industrial Internet of Things (IIoT) can be seen everywhere in our daily lives. While IIoT brings us much convenience, a series of security and scalability issues related to permission operations rise to the surface during device communications. Hence, at present, a reliable and dynamic access control management system for IIoT is in urgent need. Up till now, numerous access control architectures have been proposed for IIoT. However, owing to centralized models and heterogeneous devices, security and scalability requirements still cannot be met. In this paper, we offer a smart contract token-based solution for decentralized access control in IIoT systems. Specifically, there are three smart contracts in our system, including the Token Issue Contract (TIC), User Register Contract (URC), and Manage Contract (MC). These three contracts collaboratively supervise and manage various events in IIoT environments. We also utilize the lightweight and post-quantum encryption algorithm-Nth-degree Truncated Polynomial Ring Units (NTRU) to preserve user privacy during the registration process. Subsequently, to evaluate our proposed architecture’s performance, we build a prototype platform that connects to the local blockchain. Finally, experiment results show that our scheme has achieved secure and dynamic access control for the IIoT system compared with related research.

1. Introduction

The Industrial Internet of Things (IIoT) is a vital component in the progress of Industry 4.0 development, which has attracted wide attention in many fields, e.g., financial sector, manufacturing and transportation industry [1–3]. However, since there are extensive interactions among heterogeneous IIoT devices, how to achieve a balance between secure and efficient access control becomes a significant challenge in the industrial management system. Although a perfect access control architecture does not exist for IIoT, the prototype of access control itself has already owned several mature models, such as the Role-Based Access Control (RBAC) model [4], Attribute-Based Access Control (ABAC) model [5], and Capability-Based Access Control (CapBAC) model [6]. The RBAC is a many-to-many access control model defined between privileges and roles, which is convenient for system management. Especially in large corporations with thousands of users and permissions, RBAC is suitable for security administration. Although RBAC is not the same as Discretionary Access Control (DAC) and Mandatory Access Control (MAC), it still can enforce these policies without any complication. The ABAC adopts attributes in the form of a structured language, i.e., Extensible Access Control Markup Language (XACML), to define access control rules. During the authorization process, attributes are used to describe all the entities that must be referred to. Once an access request is made, the attributes of requests and pre-defined rules will be evaluated by the

---

\* Corresponding author.
** Corresponding author.
\textsuperscript{1} Huakun Huang is supported by the National Natural Science Foundation of China under Grant 62001126.
\textsuperscript{2} Weizheng Wang and Zhimeng Yin are supported by NSFC (Project No. 62102332) and CityU APRC 9610491.

https://doi.org/10.1016/j.dcan.2022.10.005
Received 4 June 2021; Received in revised form 25 July 2022; Accepted 6 October 2022
Available online 14 October 2022
2352-8648/© 2023 Chongqing University of Posts and Telecommunications. Publishing Services by Elsevier B.V. on behalf of KeAi Communications Co. Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
ABAC mechanism to authorize or not. The CapBAC utilizes unforgeable mediums to strengthen the security of systems. Typically, the object issues an unforgeable reference, which represents its own identity. Furthermore, the system must verify any attempt to access this object according to an Access Control List (ACL). Only if the verification is successful, requesters will have permission to operate the object.

Based on the existing models, researchers have developed many practical access control frameworks for IIoT. Two famous frameworks are XACML [7] and Open Authorization (OAuth) [8]. XACML defines access control policies and access control decisions by using the XML language. In this framework, the Policy Enforcement Point (PEP) checks the Policy Decision Point (PDP) before deciding to release the target resource from the resource server. OAuth is another famous framework that allows secure authorizations as a token-based method for Web, mobile, and desktop applications. Some researchers also tried to incorporate different policies into the hardware layer [9]. Nevertheless, due to the heterogeneous and complex components in IIoT environments, it is impossible to propose a unified protocol for the hardware layer.

In detail, Fig. 1 describes a universal framework (i.e., the centralized client-server mode) of the aforementioned typical access control systems. Due to the centralization feature, a single point of failure caused by an intrusion device may even lead to the collapse of the whole IIoT system. On the other hand, a thousand or even a million devices connected bring an obstruction to IIoT systems, most of which are approaching overload. Hence, in the current context of IIoT, a centralized and resource-constrained access control system cannot satisfy the secure and dynamic demands of IIoT anymore.

In this article, we propose a privacy-preserving access control method based on the smart contract’s non-fungible token (NFT) named ERC721 [10]. Each participant can execute the deployed smart contract to invoke and revoke access control in the blockchain network. Besides, IIoT owners are able to mint, transfer or burn tokens, which represent the licenses of IIoT devices. Then, the users can register for requesting tokens. To protect user privacy, we also utilize lightweight cryptography named NTRU, i.e., Nth-degree Truncated Polynomial Ring Units to encrypt users’ registration information. Finally, a malicious registration attack is used, so a time function is introduced in the contract by calculating the interval to block malicious requesters. The major contributions are summarized as follows.

1. We establish a smart contract token-based access control architecture for IIoT. The access control policy is also transformed into the dynamic ERC721 in the blockchain.
2. We innovatively apply the post-quantum NTRU to protect users’ registration information. Simultaneously, a time interval check function is also imported to prevent malicious registration attacks.
3. We develop a proof-of-concept implementation in the blockchain and build a real testbed to evaluate the proposed scheme. The results show that the used event feature of Ethereum can hugely save gas and storage in smart contracts.

The rest of this paper is organized as follows. In Section 2, we review the related literature about access control systems based on blockchain. The relevant preliminaries are presented in Section 3. In Section 4, we present each composition and stage of our proposed model step-by-step. Simultaneously, a comprehensive security analysis is given. In Section 5, we implement a prototype of our system to evaluate the performance as compared with other work, and list potential applications. At last, Section 6 concludes this article.

2. Related work

This section introduces some recent research about access control systems implemented on blockchain platforms [11–13].

In the initial development of blockchain-based access control, the core theme is to implement a new chain. Ouaddah et al. foremost put forward the concept of blockchain-based access control [14]. Their main contribution exists in proposing a novel decentralized blockchain—FairAccess, which provides users with privacy-preserving authorizations, and participants can freely manage their data. Shortly afterward, Ouaddah et al. [9] continued to improve FairAccess and imported the smart contract, which enables their framework to express fine-grained context-aware access control policies. However, Pinno et al. [15] pointed out that FairAccess lacks flexibility and efficiency during authorizations. Hence, Pinno et al. designed a novel blockchain to enable system compatibility with a wide variety of access control models employed by IIoT during the authorization process. Zhang et al. [16] proposed efficient policy-hiding attribute-based access control in terms of security and privacy about smart health. Since the attribute values in access policies are concealed in ciphertexts, the attribute privacy can be well preserved. Zhou et al. [17] targeted solving access control problems in M2M communications. They devised a contract-based mechanism that incentives machine-type communication devices to maximize the utility of base stations under information asymmetry. Hussein et al. [18] proposed a community-driven access control framework for distributed IIoT. The community defines the notion of rights, which drives all the IIoT nodes to fulfill a common mission.

To satisfy specified requirements, research mentioned above is devoted to customizing new blockchains. However, the new-generated blockchain usually lacks sufficient nodes to maintain the stability of this blockchain network. Therefore, such situations are possible to incur a potential adversary who can conduct Sybil attacks to generate fake nodes. Once the major nodes are mastered, the computing power of the blockchain system is centralized by this adversary who will endanger the execution of the system. Ding et al. [19] suggested a novel attribute-based access control scheme for IIoT devices. Instead of using an ACL, a set of attributes are predefined and recorded in the specified blockchain system. The nodes that maintain the blockchain network will accept or decline users’ requests according to the data inquiry result from the blockchain. With the aid of blockchain, Maesa et al. [20] published the policies to represent the rights to access resources in a simplified manner. The publicly visible rights can also be transferred among

![Fig. 1. Illustration of the typical access control system for industrial Internet of Things.](image-url)
different users. This proposal allows distributed auditability because no party can inadvertently declare the rights authorized by an enforceable policy. Schuster et al. [21] introduced a novel method called "Environment Situation Oracles" (ESOs), which encapsulates the implementation details of the IIoT environment. According to the requirements, the ESOs can dynamically adjust the IIoT circumstance for heterogeneous access control frameworks without leaking any sensitive details. Simultaneously, the ESOs can also be deployed at any layer of the IIoT stack.

However, the decentralization feature is hard to be assured in the new blockchain due to its insufficient computation power. To establish a practical and reliable system quickly, many researchers build decentralized access control systems on a mature platform—Ethereum. Furthermore, with the aid of smart contracts built in Ethereum, the access control design becomes simpler. Novo [22] proposed a single smart contract-based decentralized access control architecture for IIoT. In this architecture, other than management hub nodes and IIoT devices, the remaining entities are also covered by the blockchain. The solution involves a single smart contract that defines all the operation rules. Hence, every request from participants will be judged by the contract for the first time. Then Xu et al. [23] suggested a capability delegation mechanism called BlendCAC for access permission propagations. In the BlendCAC, there is not a centralized authority that supervises the IIoT devices. IIoT devices manage their resources by themselves. Although the conception of identity tokens was proposed for the registration process, propagation process, and revocation process, security and privacy have not been well considered. Zhang et al. [24] specified smart contracts in a further step. To achieve distributed and trustworthy access control for IIoT, they included multiple access control contracts. Unfortunately, more contracts would increase the gas cost and pressure on the IIoT system. Subsequently, Putra et al. [25] introduced a lightweight trust and reputation system for smart contract-based access control. If some nodes violate the predefined access control rules, this violation will trigger the blockchain events to alarm users. As a consequence, this trust and reputation system effectively standardizes user behaviors and resists some potential attacks. In Ref. [26], Cruz et al. implemented a RBAC mechanism on smart contracts. Initially, the third-party institutions authorize the user. Due to the identity information recorded in the contract, the organization can judge whether the services should be provided for the requesting users. Unfortunately, a comprehensive threat model and security analyses were not given in this research. Maesa et al. also found similar problems in their previous research [20]. In the latest research [27], Ali et al. converted the codified ABAC policies into smart contracts. The independent execution of smart contracts makes their systems more decentralized.

Although the existing architectures focus on solving various centralization problems in traditional access control, some access control systems requirements have not been well investigated yet, such as privacy preservation, storage space, and system response time. In contrast with the related studies, we improve the access control model from multiple respects. For users’ privacy preservation, we implement the post-quantum NTRU algorithm to encrypt registration information. For storage space, we fully utilize the event feature of smart contracts, which can store data in the log. Since the log is saved in different blocks, the response rate of the system would be speeded up. Furthermore, the malicious registration attack is also considered in our scheme. The smart contract would check the frequency and interval of visitors to prevent any potential attack. Finally, as the schemes in Refs. [22–25] are similar to our proposal, thus, we summarize a detailed comparison in Table 1.

### 3. Preliminaries

In this section, we present some relevant fundamental concepts that are significant to elaborate on our proposal. First, we give a brief introduction to the emerging blockchain technology. Then we talk about the core composition of our system—smart contracts and tokens. At last, the privacy-preserving NTRU encryption is presented.

#### 3.1. Blockchain technology

In 2008, Satoshi Nakamoto proposed the conception of blockchain technology [28], which is a distributed, decentralized, and public ledger. In the blockchain, a sequence of blocks constitutes the whole chain. The link method of each block is the cryptographic hash of the previous one. Fig. 2 illustrates the fundamental composition of the blockchain. As can be seen from this picture, there is a noticeable part—Tx-Root (the root of Merkle trees) in the block head, because the structure of transactions in the block is organized in the form of Merkle trees. Sometimes separate blocks can be produced concurrently, which leads to a temporary fork problem. Fortunately, the mechanism of the blockchain provides a solution to end forking by driving the nodes to choose the longest chain. When it refers to the mechanism of block generations in the blockchain, there are two most famous consensuses—Proof of Work (PoW) [29,30] and Proof of Stake (PoS) [31,32]. In the PoW, miners collect and confirm the transactions to build a candidate block, and then compete against each other to calculate the hash value of the block. If someone figures out a value that is smaller than a system-defined target, she/he will be the winner and get rewards from the coinbase transaction and transaction fees. For the sake of restricting the rate of newly generated blocks, the difficulty of this work is dynamically adjusted every 10 min. On the contrary to PoW, the PoS algorithm uses a pseudo-random election process to empower a node to be the validator of the next block. The election method depends on the coin age, which is the product of the time and number of coins. Finally, the older one can get the authority to generate the next block and receive returns.

#### 3.2. Ethereum platform

Ethereum is the second generation blockchain. In Ethereum, a series of decentralized applications named smart contracts [33] can also be developed. And in the contract, people can code to customize their own digital value—token, which runs exactly as programmed, and is accessible anywhere in the world.

![Fig. 2. The basic structure of blockchain.](image-url)
3.3.2. Encryption

For issuing tokens in the contract, Ethereum proposed many innovative standard protocols, such as ERC20 [35] and ERC721 [10]. ERC20 has another direct name that is called the Fungible Token (FT), which means a token can be exchanged for any other token with the same value. On the contrary, the ERC721 token is a NFT that cannot be replaced with other tokens, even if two tokens are in the same form. Every ERC721 token can be assigned to the special attributes. Recently, the well-known Cryptokitties [36] on the Ethereum is built on ERC721 NFTs.

3.2.2. Fungible/non-fungible token

The birth of Ethereum allows us to establish complex distributed applications behind cryptocurrencies. A smart contract is a self-executing contract with the terms of the agreement between users being directly written into lines of code. Once deployed, unless destroyed by its owner, nobody can modify the logic of the code. Contracts are typically coded in higher-level languages, like Solidity [34], then compiled to EVM (Ethereum Virtual Machine) bytecode.

3.2.3. Decryption

For issuing tokens in the contract, Ethereum proposed many innovative standard protocols, such as ERC20 [35] and ERC721 [10]. ERC20 has another direct name that is called the Fungible Token (FT), which means a token can be exchanged for any other token with the same value. On the contrary, the ERC721 token is a NFT that cannot be replaced with other tokens, even if two tokens are in the same form. Every ERC721 token can be assigned to the special attributes. Recently, the well-known Cryptokitties [36] on the Ethereum is built on ERC721 NFTs.

3.3. NTRU Public Key Cryptosystem

The NTRU public key cryptosystem is one of the fastest known public key cryptosystems [37]. Simultaneously, as a promising candidate for the future post-quantum cryptography standard, lattice-based cryptography enjoys the advantages of strong security guarantees and high efficiency, which makes it extremely suitable for IIoT applications. Due to the constrained resource of IIoT, the lightweight NTRU is used for data encryption in our system. Just like other public key systems, there are three phases in the NTRU:

3.3.1. Key generation

Suppose we have two users—Alice (sender) and Bob (receiver), the public key is known by both Alice and Bob, and the private key is only known by Bob.

Step 1 Bob requires two polynomials $f$ and $g$ with coefficients in $\{-1,0,1\}$ and the degree up to $N-1$ to generate the key pair. Note that they can be considered as representations of the residue classes of polynomials modulo $X^N-1$ in $\mathbb{Z}$.

Step 2 Since the polynomial $f \in \mathbb{L}_p$ must own the inverse modulo $p$ and $q$, which asks Bob to compute $f_p = f^{-1}(\mod p)$ and $f_q = f^{-1}(\mod q)$.

Step 3 If the chosen $f$ is not invertible, Bob needs to return to step 1 and try another $f$. Otherwise, the public key $h = p \times f_q \times g(\mod q)$ is computed by Bob.

3.3.2. Encryption

Suppose the public key of Bob is transmitted to Alice via a secret channel.

Step 1 To send a secret message to Bob, Alice should put her message in the form of a polynomial $m$ with coefficients $\{-1,0,1\}$.

Step 2 After creating the message polynomial, Alice chooses randomly a polynomial $r$ with small coefficients in $\{-1,0,1\}$.

Step 3 Alice computes encrypted message $e = r \times h + m(\mod q)$ using Bob’s public key.

Step 4 Alice sends $e$ to Bob.

3.3.3. Decryption

Suppose Bob successfully receives the transmitted content.

Step 1 Because Bob knows his private key, he can get his private key $a$ by multiplying the encrypted message $e$ by $f$: $a = f \times e(\mod q)$.

Step 2 Bob retrieves $m = f_p \times a(\mod q)$ from $a$.

The variables used above have been collected into Table II for easy reference.

4. Our proposed token-based access control system

4.1. System architecture

The architecture of our system is depicted in Fig. 3, where an enormous number of nodes connect each other to establish an extensive peer-to-peer network. Each local server that takes control of a group of IIoT devices (e.g., actuators and sensors) can constitute the full node. Also in this network, the user can register information and request an identity token. The full node bears the major work for blockchain maintenance, e.g., transaction verifications and block generations. To decrease the network load for the full node, the users are unable to directly communicate with the full node. Given this situation, our system provides an alternative light node here, which turns the interaction process into a more convenient and efficient one. Smart contracts record the information of users and IIoT devices, and transform access control into identity tokens for future requests. The main components of this architecture are described in the following:

**Client/User**: the client is an application, which can be downloaded and installed from the light node. It provides all the interfaces of smart contracts. When the users would like to call some functions, this application can remind them of necessary inputs. After data encryption, the client automatically transmits the parameters to the light node.

**NTRU Encryption**: NTRU is a lightweight cryptography mechanism, which enables fast encryption and decryption for IIoT devices. In our architecture, it is attached to the client application, which aims to protect the privacy of users’ registration.

**Light Node (LN)**: It is given the limit of the CPU and memory in user devices, which are not proper to be a full node. Therefore, the client can use the LN provided by the system to communicate with the blockchain network. Owning to the NTRU adopted by the system, information leaks resulted from insecure encryption algorithms can be overcome. On the other side, if the users do not trust the intermediary, they can choose to build their own LNs. The application binary interface of the smart contract is public for reference.

**Full Node**: any computer that fully enforces all the consensus rules of Ethereum is called a full node. Moreover, the operation of the Ethereum system is maintained by the full nodes, which keep the entire network honest.

**Local Server (LS)**: the local server itself in our system is also a full node, which manages a cluster of IIoT devices via Wi-Fi.

**Token**: with the help of the smart contract, IIoT devices can incorporate their data into a token. Then, the users can request tokens. Note that the LS owns the absolute control over its tokens and thereby can decide to issue or destroy their tokens.

**Manage Contract (MC)**: in order to protect the code for rule
definitions, we design the MC—the entry of the whole system. Hence, anyone who would like to invoke the functions in the TIC and URC must call the corresponding interfaces in the MC first. The logical relation between MC, TIC, and URC is illustrated in Fig. 4.

**Token Issue Contract (TIC):** The TIC is mainly for LSs to manage tokens, which represent the identity of IIoT nodes in the blockchain. The LS can issue, transfer, or burn tokens. We present the detailed IIoT token issue process in Fig. 5.

### 4.1. User register contract

The users can utilize URC to register their information. After registration, the URC emits the defined event to store records in the form of logs. The structures of the registration and time tables are shown in Fig. 6.

#### 4.2. Execution details

This subsection presents the detailed execution flow between different roles in our architecture. As presented in Fig. 7, the main part has been divided into four different stages.

##### 4.2.1. IIoT token issue

In our system, every LS takes control of a cluster of IIoT devices, so the IIoT devices should firstly send their own identity attributes $a_i$ to the LS. The content of $a_i$ is denoted as

$$a_i = <\text{name, address, opcodes, pk}>$$

(1)

where opcodes is the permitted operation codes of IIoT devices and pk is the public key to the connected LS. For generating the key pair, two polynomials $f$ and $g$, with coefficients $c \in \{\pm 1, 0, 1\}$ and the degree up to $N – 1$ are required by the LS. They can be considered as representations of the residue classes of polynomials modulo $X^N – 1 \in R$. Since polynomial $f \in L_f$ must own the inverse modulo $p$ and $q$, which requires the LS to compute

$$f_p \equiv f^{-1} \pmod{p}$$

(2)

$$f_q \equiv f^{-1} \pmod{q}$$

(3)

If the chosen $f$ is not invertible, the LS repeats (2) (3) again and tries another $f$. Otherwise, the pk and secret key $sk$ is computed by the LS as

$$pk \leftarrow Gen(L_f) = f$$

(5)

Each LS manages a set of $n$ IIoT devices, thus attributes $A_i$ in the LS are

$$A_i = <a_1, a_2, \ldots, a_n>$$

(6)

Then the LS constructs a token issue transaction $TX_i$ with its signature $\text{Sig}_{\text{LS}}$ and $A_i$ to trade:

$$TX_i = [A_i|\text{Sig}_{\text{LS}}|\text{Timestamp}_{\text{user}}]$$

(7)

where Timestamp$_{\text{user}}$ records the generation time. When $TX_i$ is included in the next generated block, $Token_i$ is issued:

$$Token_i = <\text{owner, a_i}>$$

(8)

where owner matches the address of the message sender.

##### 4.2.2. User registration

In this process, the user should firstly get a specified application from the LN, and then search for target device $ad$ among all device records $ad$ in the blockchain network.

$$\text{Search}(\forall ad \in LN, \exists ad = td)$$

(9)

If the user successfully finds the devices, she/he needs to register for authorizations. However, to protect the privacy of users, before the registration, input data $m$ must be encrypted by the embedded NTRU. Suppose the user has got the pk of the target LS. To send encrypted $m$ to the LS, the user should put $m$ in the form of a polynomial $M$ with coefficients $c \in \{\pm 1, 0, 1\}$. After creating $M$, the user chooses randomly a polynomial $r$ with small coefficients $c \in \{\pm 1, 0, 1\}$. The ciphertext $e$ is computed as

$$e = \text{Enc}_{\text{pk}}(M) = r \times h + M \pmod{q}$$

(10)

When $e$ is produced, the user builds a transaction $TX_{\text{reg}}$ with input $m$ and its signature $\text{Sig}_{\text{user}}$ to activate the register function $\text{Reg}$ defined in the smart contract:

$$TX_{\text{reg}} = [e|\text{Sig}_{\text{user}}|\text{Timestamp}_{\text{reg}}]$$

(11)
where $\text{Timestamp}_{\text{reg}}$ records the generation time.

$$\text{Reg(e)} \leftarrow \text{TX}_{\text{reg}}$$

(12)

It is worth mentioning that there is a time interval function $\text{TIF}$ during the life cycle of the smart contract, which aims to check the interval of the users from the same address. If the interval time is shorter than a defined time $\text{dtime}$, this system will decline the $n$-th access and accumulate a frequency $\text{Feq}$. When $\text{Feq}$ is over 5, this address will be prohibited in the next 24 h. This function can be effective to defend against the malicious registration attack. The detail of the user register function can be seen in the algorithm 1.

Algorithm 1. User Register Contract

Require: userAddr, userName, iotAddr
Ensure: userInfo
1: Create a URC instance.
2: Modify userRegisterCheck(userAddr, iotAddr)
3: if msg.sender == MC.addr then
4: if now - user[userAddress].time >= 1day then
5: user[userAddress].frequency $\leftarrow$ 0
6: end if
7: if iotAddr != address(0) && userAddr != iotAddr then
8: if user[userAddress].frequency $\leq$ 5 then
9: if user[userAddress].time == 0 || now - user[userAddress].time $\leq$ 3minutes then
10: Function
11: $\text{URC Register(userAddr, userName, iotAddr)}$
12: user[userAddress].time $\leftarrow$ now
13: Emit registrationReminder(userAddr, userName, iotAddr, time) $\rightarrow$ userInfo
14: else
15: user[userAddress].frequency $+$ +
16: return "FREQUENT ACCESS!!"
17: end if
18: else
19: return "ACCESS RESTRICTION!!"
20: end if
21: else
22: return "ADDRESS CANNOT BE 0 OR ITSELF!!"
23: end if
24: else
25: return "UNAUTHORITY!!"
26: end if

4.2.3. Authorization

In the token authorization process, when the registration transaction is contained in the next block, the LS can retrieve the registration table and use its sk to restore the original information of registration. $M$ is multiplied by $e$ and $a$ (i.e., part of sk):

$$a \equiv f \cdot e \pmod{q}$$

(13)

where $a$ is polynomial, coefficients $\in \mathbb{F}_q$, then the LS retrieves $m$ from $a$:

$$m \equiv f^{-1} \cdot (mod p) \times f \cdot e (mod q) (mod p)$$

(14)

Because of the different pk used, only some of the encrypted data can be decrypted successfully. If the LS verifies the identity and decides to authorize a token transfer transaction $\text{TX}_t$ with the $i$-th token info $\text{Info}_i$ and signature $\text{Sig}_{\text{LS}}$, all the nodes in the network will execute the smart contract with relevant parameters of $\text{TX}_t$ to transfer token $\text{x}$ to the user’s address $\text{addrusr}$:

$$\text{Transfer(token, addrusr)} \leftarrow \text{TX}_t = \left[ \text{Info}_i | \text{Sig}_{\text{LS}} \right] / \text{Timestamp}_t$$

(15)

where $\text{Timestamp}_t$ records the generation time of $\text{TX}_t$.

4.2.4. Access control

In the final process of access control, to perform permitted operations defined in the IoIoT tokens, the user can call the access control function in the MC. According to the LN, the user invokes token inquire function $\text{Inq}$ to check the ownership of tokens:

$$\text{Token}_{\text{owner}} = \text{Inq(\text{Token})}$$

(16)

If a user finds the owner address of the token authority matches his address $\text{addr}_{\text{auth}}$, she/he will be able to carry out access control. On the other side, the LS receives the requests from the blockchain network continually, it can quickly verify the identity of requesters. If a user identity is confirmed, the LS orders the inquired IoT device to execute $\text{Exe}$ defined operations of opcodes:

$$\text{Exe(opcodes)} = \text{Inq(\text{Token})} = \text{Token}_{\text{user}} < \text{address} >$$

(17)

4.3. Security analysis

In this subsection, we analyze several security features of our proposed system.

4.3.1. Decentralization

To prevent any party from taking full control of our system, we
construct a decentralized system based on the Ethereum platform. If an adversary attempts to intrude into the blockchain system, the hash power of the adversary is $H_{\text{adv}}$. According to the famous phenomenon—51% attack in the blockchain, if the adversary tries to take control of all the system, this requires seizing at least 51% of the computation power.

$$H_{\text{adv}} / H_t \geq 51\% \quad (18)$$

where $H_t$ means the total computation power in the system. However, the current hash power in the Ethereum mainnet has reached 174,867.04 GH/s. One of the fastest supercomputers in the entire world—Sequoia is only able to process 16.32 petaflops per second, a value that corresponds to only 2% of the estimated power of the Ethereum mainnet. Therefore, from the analysis in Ref. [38] and the current situation, it is very hard for an adversary to own such enormous computation power to crash our system.

4.3.2. Partial anonymity

Neither users nor IIoT devices need to disclose private information in our proposed system. On the one hand, users have the choice to input some random identity information rather than real identities. On the other hand, the IIoT device just presents its address in the blockchain. Since no real data could be leaked, partial anonymity can be ensured by our proposed scheme [39].

4.3.3. Malicious registration attack prevention

There is a time interval function defined in our contract. Whenever the user requests the functions in the contract, this function will execute in advance to check the interval of the users from the same address. If the interval time recorded in the contract is shorter than a defined time, this system will decline this access undoubtedly and accumulate the frequency. Once the frequency of a given user is over 5, this address will be prohibited in the next 24 h. Due to this mechanism, our system can prevent most of the malicious registration attacks efficiently.

4.3.4. Replay attack resistance

Owing to the unique feature of the private key in the blockchain, the adversary cannot forge the same private key and public key to prove the ownership of the target address. Besides, every transaction should be attached to a timestamp, so it is hard for the adversary to conduct replay attacks against our system.

4.3.5. Unbreakable encryption

We assume a polynomial-time adversary aims to conduct a Chosen-Ciphertext Attack (CCA) against our system, which requires two polynomials $T_1$ and $T_2$ to have a collision with the same coefficient. The intersection polynomial $k$ of $(T_1, T_2)$ is

$$k = \sum k_i x^i \quad (19)$$

If coefficient $T_1$ and $T_2$ are both equal to 1, $k_i$ will be 1. Otherwise, $k_i$ equals to 0. For a convenient explanation, we substitute $Zpk + c$ for $c$ in this case, where $Z$ is an integer. Then, $a$ is described as

$$a = f \times Zpk + Zf \pmod{q}$$
$$= Zg + Zf \pmod{q}$$
$$= Z + Zpk - qk \quad (20)$$

Subsequently, $M$ can be obtained as

$$M \equiv ZF_p \times f + ZF_p \times g - qF_p \times k \pmod{p}$$
$$\equiv Z + Zpk - qF_p \times k \pmod{p} \quad (21)$$

If $Z$ is chosen:

$$Z = 0 \pmod{p}$$

$$M \equiv -qF_p \times k \pmod{p} \quad (22)$$

Finally, $sk$ can be calculated from

$$f = -qk \times M^{-1} \pmod{p} \quad (23)$$

From the definitions of the probability computation for $D_1$ and $D_2$ in Ref. [40], it is an enormous challenge to find collisions of $f$ and $g$. Therefore, our encryption algorithm is resistant to CCAs.

4.3.6. Token forge detection

If an adversary forges the IIoT token similar to $a_i$, the LS can immediately perceive the forgery from recent transaction record $TX_{\text{recent}}$, when $\text{adv}_3$ communicates with the LS with the access control function in the
5. Performance evaluation and comparison

In this section, we first describe the configuration of our experiment, and then evaluate the gas cost and concurrent time consumption in our work as compared to Refs. [24,25]. Finally, we list some engineering applications for our proposed protocol.

5.1. Environment setup

The experiments are established on one desktop computer with two Raspberry Pi 4 Model B. The desktop computer takes on the roles of the LN, blockchain network and LS. The local blockchain environment is deployed by Ganache, which can allow other nodes to participate. Two Raspberry Pis simulate the user and IIoT device. Then, we use Solidity 0.6.0 to create three core smart contracts (TIC, MC, and URC) in our system. The program of NTRU for data encryption is referred to Ref. [41]. Finally, we implement the interaction between objects and smart contracts by web3.py. The detailed configuration is presented in Table 2 and Fig. 8.

5.2. Cost evaluation

In Table 3, we present all the gas costs of different functions from the experiment results. From Fig. 9(a), it is evident that the highest cost stage is the contract creation. Since the complex functions are comprised in Ref. [24], the total cost even reaches 5,484,074 gas, which is equivalent to 7.45 dollars (The exchange rate is collected on April 20, 2020, 1 gas ≈ 0.000000008 ether, and 1 ether ≈ 170 USD). Although compared with [25], our cost of the contract creation is not better (i.e., more than 1,000,000 gas). However, the design of [25] imports many other centralized objects (e.g., dedicated data storage and attribute authority) to alleviate the communication pressure of the system rather than entirely relying on the smart contract. Note that this cost is only once, so we can concern less about this metric. On the other side, the experiment data about the register function cost were not provided in Ref. [24], and [25] did not use smart contracts to conduct access control interactions (i.e., IIoT devices directly interact with the data storage). Therefore, in the process of user registration and access control, we choose one of the researches to build respective controlled experiments. As illustrated in Fig. 9(b) and (c), the imported event feature transfers the original data from the main block to the respective block, which saves the gas cost in our system. From Fig. 9(d), when a new relationship between the user and IIoT device has been established, a new access control contract should be deployed by Ref. [24], which leads to nearly ten times the cost of our contract. In our system, we use a NFT to represent the identity of IIoT devices. It simplifies the process of authorizations, only to change the owner’s address of tokens is enough. Although [25] also introduces some features of the token, the storage of token identities is not transferred. However, in our work, we make full use of the event feature more than once, which reduces a great amount of gas than expected.

5.3. Computation evaluation

In this subsection, we use multiple concurrent requests to simulate the real IIoT devices. As illustrated in Fig. 10(a), with the increment of concurrent requests, the curve of the response time climbs steeply at the first stage. This situation is caused by network latencies. However, when

Table 2
Configuration of device.

<table>
<thead>
<tr>
<th>Name</th>
<th>OS</th>
<th>CPU</th>
<th>Memory</th>
<th>Hard Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>iMac</td>
<td>macOS Catalina</td>
<td>Intel Core i5, 3.5 GHz</td>
<td>16 GB</td>
<td>1 TB</td>
</tr>
<tr>
<td>Raspberry Pi 4 Model B</td>
<td>Raspbian 10 buster</td>
<td>ARMv7 rev 3 @ 4 × 1.5 GHz</td>
<td>4 GB</td>
<td>64 GB (microSD card)</td>
</tr>
</tbody>
</table>

Table 3
Gas cost of the different functions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Gas Used</th>
<th>Ether Used</th>
<th>USD ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Contract</td>
<td>553,960</td>
<td>0.0060896</td>
<td>0.8769</td>
</tr>
<tr>
<td>userRegister</td>
<td>50720</td>
<td>0.0000558</td>
<td>0.00804</td>
</tr>
<tr>
<td>tokenIssue</td>
<td>201,672</td>
<td>0.0002218</td>
<td>0.03194</td>
</tr>
<tr>
<td>tokenBurn</td>
<td>32,179</td>
<td>0.0000354</td>
<td>0.0051</td>
</tr>
<tr>
<td>tokenTransfer</td>
<td>59,822</td>
<td>0.0000658</td>
<td>0.00948</td>
</tr>
<tr>
<td>accessControl</td>
<td>26323</td>
<td>0.000029</td>
<td>0.00418</td>
</tr>
<tr>
<td>Mcdestruct</td>
<td>21,734</td>
<td>0.0000239</td>
<td>0.00344</td>
</tr>
<tr>
<td>getURCAddress</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>getTICAddress</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>getTokenOwnerof</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>accessControl</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 8. Demonstration of the real software&hardware platform.

Fig. 9. Gas cost comparison for four key terms.

Finally, we implement the interaction between objects and smart contracts by web3.py. The detailed configuration is presented in Table 2 and Fig. 8.
which can be developed in many realistic environments. Here, we list

to implement the MC for preventing users from invoking core functions

to process increasing requests in a fixed time interval. The result in

the number of simulation requests is up to 100, the effect of network
latency tends to be hugely minimized. Finally, the response time fluctuates in the nearby of 18 s, which means that our system has the ability
to operate more functions in the runtime, such as token issue, token transfer, and
registration query. On the other side, the user only needs to register for
tokens and receive token data. Finally, the experiment results demonstrate that our system for IIoT has an elastic load balance, which is
appropriate for being deployed in a large-scale IIoT environment.

5.4. Engineering applications

Our proposed system owns strong security and scalability features,
which can be developed in many realistic environments. Here, we list
two engineering applications to illustrate the ideal functions. As shown in
Fig. 11, Electrical Vehicle (EV) charging can use our system. When an EV
is approaching a Charging Station (CS), the EV can register its informa-
tion in the blockchain network by invoking the pre-defined smart contract. Then the requested CS verifies the identity of the EV. If the
verification passes, the identity token will be issued to the EV. Shortly
after, when the EV arrives at the CS and calls the smart contract. A randomly selected master node will check whether this EV has a legiti-
mate token. At last, the CS can transmit power to the EV in time. It is
worth mentioning that our system can support heterogeneous EVs and
CSs. The other engineering application is for automatic hotel operations.
As usual, if customers want to check into a hotel, there will be a recep-
tionist who helps them register the information and issues the room keys
or cards. This process may be simplified by utilizing our proposed system.
As Fig. 12 shows, the customers can directly register their identity informa-
tion for check-in according to the smart contract in the blockchain. Then
the hotel reception confirms the validity of these customers. If the
check is successful, the smart contract will distribute the tokens to the
customers in the same as the process mentioned above. Before the cus-
tomers enter their rooms, they can request the related smart contract
again. Finally, the smart contract verifies the token owned by the cus-
tomers once again and transmits the electronic key to open the door.

6. Conclusion and future work

In this paper, a smart contract token-based access control scheme,
aiming at providing secure, nimble, and automatic services for large-
scale IIoT systems, is proposed. To protect the system security, we
implement the MC for preventing users from invoking core functions
directly. At the same time, there are also two other contracts (i.e., TIC and
URC) for IIoT token management and user registration, respectively. The
significant contribution of this work is the NFT included in the contract,
which innovatively converts traditional access control records into a
flexible identity token. Furthermore, we utilize the lightweight post-
quantum cryptography NTRU to preserve users’ sensitive information.
Finally, security analyses and simulation results demonstrate that our
decentralized access control system could satisfy the reliability and
compatibility requirements of IIoT devices. In the future, to augment the
practicability and convenience, we will migrate our proposed access
control architecture into different chains.

Declaration of competing interest

The authors state that they have no conflicts of interests to disclose.

References

[1] P. Wang, W. Hao, Y. Jin, Fine-grained traffic flow prediction of various vehicle types

contracts: architecture, applications, and future trends, IEEE Transactions on

in internet of things (iot) using cryptography and steganography techniques, IEEE

models, Computer 29 (2) (1996) 38–47.

[5] V.C. Hu, D.R. Kuhn, D.F. Ferraiolo, J. Vos, Attribute-based access control,


P. Humenn, S. Godik, S. Anderson, S. Crocker, et al., eXensible access control

citeser.ist.psu.edu/viewdoc/download?doi=10.1.1.209.8141&rep=rep1

internet of things: big challenges and new opportunities, Comput. Network. 112


