DBS: Blockchain-Based Privacy-Preserving RBAC in IoT

Xianxian Li[1,2], Junhao Yang[1,2], Shiqi Gao[1,2], Zhenkui Shi[1,2(✉)], Jie Li[1,2], and Xuemei Fu[1,2]

[1] College of Computer Science, Guangxi Normal University, Guilin 541000, GX, China
lixx@mailbox.gxnu.edu.cn, shizhenkui@gxnu.edu.cn
[2] Guangxi Key Lab of Multi-source Information Mining and Security, Guilin, China

Abstract. In this paper, we propose a new privacy-preserving scheme for access control in IoT based on blockchain technology and role-based access control (RBAC). The decentralized property and reliability of the blockchain platform make the proposed solution fit the geographically distributed scenario for IoT better. We extend the traditional RBAC with a new device domain to realize more flexible and manageable access control for the diverse IoT devices. Besides, the scheme takes advantage of zero-knowledge proof and the trusted execution environment (TEE) to ensure the transaction information is confidential, to protect the privacy of the details of access control including information of roles, devices, and policies. To demonstrate the feasibility and effectiveness of the architecture, we implemented our scheme and evaluated on the Ethereum private chain to achieve privacy-preserving access control for IoT. The results show that our scheme is feasible and the cost is acceptable.

Keywords: Internet of Things · Privacy-preserving · Access control · Smart contract · Blockchain

1 Introduction

According to the prediction of the American Computer Industry Association (CompTIA), the number of IoT devices will increase to 50.1 billion by 2025. The ubiquitous IoT is affecting all aspects of our lives, such as intelligent transportation, smart home, environmental monitoring, street lighting system, food traceability, medical health, and so on.

The work is partially supported by the National Natural Science Foundation of China (No. 61672176), the Guangxi "Bagui Scholar" Teams for Innovation and Research Project, the Guangxi Talent Highland Project of Big Data Intelligence and Application, the Guangxi Science and Technology Plan Projects No. AD20159039, the Guangxi Young and Middle-aged Ability Improvement Project No. 2020KY02032, the Innovation Project of Guangxi Graduate Education No. YCSW2020110.

X. Yuan et al. (Eds.): QShine 2021, LNICST 402, pp. 94–110, 2021.
https://doi.org/10.1007/978-3-030-91424-0_6

Due to the IoT devices are globally distributed and the scale of managed devices may be from hundreds to millions for different organizations, companies, or institutions, how to access control these devices efficiently, conveniently, and securely become an important problem. In IoT, access control is used to grant or revoke the permission to a specified user to have access to the specified IoT devices. It consists of a set of concepts such as access control policies, authorization, granting, revoking, etc. The traditional centralized access control systems are for the human-machine oriented Internet scenarios where devices are within the same trust domain and can not meet the needs of access control of the IoT. In addition, IoT devices may be dynamic and abundant, there may be a single point of failure, and other issues such as the constrained CPU, memory, power.

The emerging technology of blockchain [1] may provide viable schemes for IoT access control. The blockchain is a distributed digital ledger. The decentralized architecture of the blockchain reduces the pressure of the old central computing of the IoT. The data is no longer controlled by the center alone, but also provides more possibilities for the innovation of the organizational structure of the IoT. The accuracy and non-tamper ability of blockchain records make the data available and more secure. And some architectures for IoT access control have been proposed [2–4]. But the current schemes of IoT access control still face the following challenges.

- Management. In most proposed schemes, the solutions to manage these access policies don't consider the scale of devices and are not flexible or efficient enough. The number of IoT devices may be from hundreds to millions for an organization. For example, considering the scenario when a device manager quits from an organization, they should execute the function of removing manager from device n times where n is the number of devices the manager controls. Similarly, when a new manager checks in, they should execute the function of adding a manager to device n times. For an access policy always indicates a relation between a device manager and a device in most current schemes [2]. Such a solution is not flexible or efficient. Besides, massive IoT devices are being deployed every day. The access policies should be updated efficiently and adaptively. Existing solutions maybe not applicable.
- Security and privacy. Some schemes are based on public blockchains. The information is public. This means all access policies are public and may face severe privacy issues for the organizations.
- Cost. Some schemes may be based on private chains. These chains need to invest a lot of nodes to satisfy the blockchain and fit the distribution of the IoT.

To address the above issues, we propose DBS, which is a privacy-preserving RBAC architecture based on blockchain in IoT. Comparing with previous work where an access policy indicates a manager's access authority to a single device, we extend the traditional RBAC with a new domain and make DBS fit the access control of IoT. Through this methodology, we ca manage the access policy in a batch manner. And it can deal with the scenarios by calling about two

functions when many access policies should be changed. DBS leverages zero-knowledge proof to protect the privacy of transactions and combines with the TEE to ensure the confidentiality and integrity of the smart contract. DBS consists of double access control contracts, one authorization contract, and one authentication contract. Authorization contract is used to assign a role to the requester and provides functions for adding, updating, and deleting access control policies. An authentication contract is used to check the access right of the requester and prevent unauthorized access. Finally, we implemented and evaluated our scheme. And the results show that by introducing the RBAC model, it can simplify management, preserve the privacy of access policies, and achieve convenience and flexibility to access and manage the IoT devices.

2 Related Work

Access Control Schemes Based on Blockchain

Due to the excellent features of blockchain such as decentralization, tamper-proof, traceability, etc., at present, many researchers have leveraged blockchain to solve the access control of IoT. The decentralized peer-to-peer network can well conform the distribution of IoT. Because of the three problems that must be solved in access control under the IoT (the lightweight terminal devices, the massive terminal nodes, and the dynamic nature of the IoT), the survey [5,6] summarized how to solve these problems and the advantages of using blockchain.

For the access model, we will take the most common models like Attribute-based access control (ABAC), role-based access control (RBAC) [7], etc. as our study cases. [8] proposed a scheme based on the ABAC. It can create, manage, and implement access control policies by using blockchain technology, which allows to access resources in a distributed manner. But this method is not completely decentralized. The scheme needs authorization centers for policy implementation points, policy management points, and policy decision points. And the policies and the rights exchange are publicly visible. These may breach the privacy of related parties. [9] also proposed an ABAC model based on a decentralized blockchain, but it is in a big data scenario that is similar to the Internet of things.

Role-based access control (RBAC) is another common access, management model. [10] uses smart contract and role-based access control to realize cross-organization role verification. [4] proposed a framework of arbitration roles and permission. Users can manage IoT devices through a single smart contract.

In terms of blockchain technology, the current schemes can fall into two categories: transaction-based access control and smart contract-based access control [11]. FairAccess in [3] uses new types of transactions to grant, get, delegate, and revoke access. [12] proposed a novel decentralized record management system to handle electronic medical records using blockchain technology. The system uses three smart contracts: register contract, patient-provider relationship contract, and summary contract to control access to electronic medical records. And [13] realized access control of IoT devices through multiple different smart contracts including register contracts, access control contracts, and judge contracts

to accomplish the functions of registration, access verification, and audit in the access control process. [14] provides a specific case to realize the application of access control to smart homes based on the smart contract.

There are few of the currently proposed schemes which considered the policies management of massive IoT devices and users. And there are a large number of new IoT devices deployed or upgraded every day. New schemes should be essential to manage access policies conveniently and efficiently.

Privacy-Preserving Schemes for IoT

Although, the anonymity of blockchain may protect a user's privacy to a certain extent, and the distributed nodes of blockchain can prevent a single point of failure. In the era of big data, attackers can still identify some user's blockchain transaction information and IP [15], leaving users' critical information exposed to the shadow of privacy disclosure.

For the privacy and security of blockchain, Microsoft [16] proposed a framework to protect the privacy of blockchain by using a trusted execution environment (TEE). The Confidential Consortium framework includes a set of key and permission management mechanisms, which can ensure that only encrypted transactions can be processed in a trusted execution environment, and only users with corresponding permissions can view the relevant status. TEE can not only prove the correctness of the code, but also ensure that the internal data is invisible to the outside and not tampered with when running, and then ensure the confidentiality and integrity of the key code and data of the blockchain protocol, so that the application of the blockchain can run efficiently on the fully trusted member nodes. And [17] also uses a hardware enclave to ensure the confidentiality of smart contracts. After the verification on the blockchain, the smart contract is put into the offline distributed TEE platform for execution and storage to maintain the integrity and confidentiality of the smart contract. For the privacy problem of smart contracts, [18] proposed a distributed smart contract system with privacy protection. Through zero-knowledge proof [19,20], the scheme can ensure that the transactions will not be disclosed. Similar privacy schemes include zerocoin and zerocash [21,22].

However, all the above schemes are not dedicated to IoT and cannot be applied directly in access control for IoT. The access policies may include information about IoT devices and users. These may raise critical privacy concerns. And new privacy-preserving access control schemes should be proposed to protect the privacy of related users and devices.

3 System Architecture

We propose a new architecture in this paper which is a distributed privacy-preserving access control system. In the scheme, access control information is stored and authenticated via blockchain and smart contract.

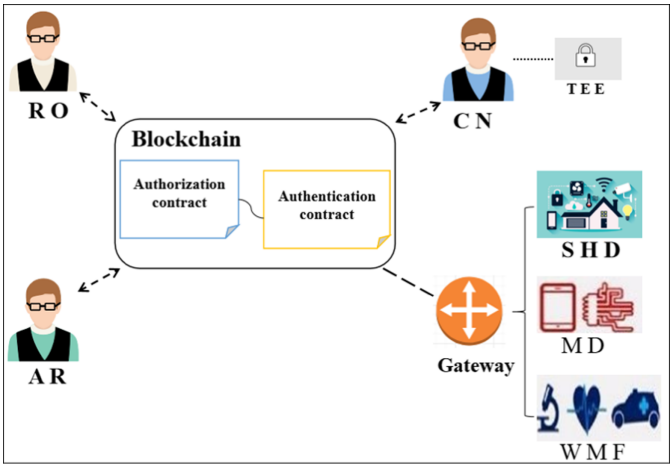


Fig. 1. Access control system overview of DBS.

3.1 Architecture Component

The architecture of our system is illustrated in Fig. 1. The architecture includes six different components: Resource owner, access requesters, consensus nodes, and smart contracts, which connect through P2P networks. At the same time, there are a large number of IoT devices (e.g. mobile devices (MD), smart home devices (SHD), wearable medical facilities (WMF), etc.), which are connected to the P2P network via the gateways. Such mechanism is common in many work [3,13]. The different components of the architecture can be explained in detail as follows.

1) *Resource owner (RO)*: RO is the owner of the IoT devices, responsible for managing the access control permission of a set of IoT devices, and has the highest management rights. Any entity can become a resource owner by registering its own IoT device on the deployed smart contract.
   RO defines access control policies and various operations in the authorization contract and deploys the authorization contract on the blockchain to realize the function of the authorization contract. By invoking the authorization contract, RO can assign different roles to different access requesters to achieve permission management. At the same time, RO deploys an authentication contract to check the role and operation of the access requester to realize access management.
2) *Access requester (AR)*: AR refers to the people who want to access the IoT devices to obtain relevant data, operate and manage IoT devices. There are two types of access requesters, the access requester in the domain, and the cross-domain access requester, such as family members and non-family members, or company employees and non-company employees.

There are different authorization methods for different access requesters. For the access requester in the domain, RO can directly assign the corresponding roles to the members and add user-role-permission-devices information to authorization contract by invoking the contract. However, the cross-domain access requester needs to apply for a role from the authorization contract. When the access control policies defined by the RO are met, the authorization contract will assign corresponding roles to the cross-domain access requesters.

3) *Consensus node (CN)*: CN is a specific blockchain node in our architecture. It is not only responsible for verifying and processing transactions but also for deploying and executing smart contracts.
   To guarantee the confidentiality of the code and data of smart contracts, these consensus nodes need trusted execution environments (TEE). TEE can provide a secure enclaved space for data and code to ensure their confidentiality and integrity. The contract will be encrypted before transferring and can only be decrypted inside the corresponding enclave. TEE will generate a key-pair for this contract and publish the public key. The invocation arguments are encrypted with the contracts public key which can only be decrypted within the enclave. The return value will be encrypted by a user-provided key which is delivered along with calling arguments. During the whole process, anyone even the CN cannot leak the internal executing state.
4) *Smart contract (SC)*: We define management operations in smart contracts to manage the access control. All the operations allowed in the access management system are triggered by blockchain transactions. We utilize two smart contracts, an authorization contract and an authentication contract to assign roles and check access rights respectively.
   - Authorization contract: RO defines access control policies and management operations in the authorization contract. The management operations include *Addrole()*, *Deleterole()*, *Grantrole()*, *PolicyAdd()*, *PolicyUpdate()*, and *PolicyDelete()*, etc. For the access requester in the domain, RO calls the *Addrole()* operation to assign the corresponding role to the access requester directly. For cross domain access requesters, who needs sends a transaction application role to the authorization contract, and the authorization contract will judge whether it meets the requirements of role assignment according to the access control policy. If the policies are satisfied, the access requester's information is added to the authorization contract with operation *Grantrole()*.
   - Authentication contract: is responsible for preventing unauthorized access and checking the access permission of the access requester. The operation of authentication contract includes *FindRole()*, *FindDevice()* and *Judge()*, etc. When the access requester sends an access request to the IoT device, the authentication contract will invoke the authorization contract by executing the *FindRole()* operation to determine whether the access requester is a legitimate user. Then, the authentication contract will execute the *FindDevice()* operation to determine whether the access requester can access the corresponding IoT devices. Finally, the authorization contract will execute

Judge() operation to match role-permission-device to determine whether the access request meets the access control policy.

5) *Gateway*: As mentioned before, IoT devices do not belong to the blockchain network. The majority of IoT devices are very constrained in terms of CPU, memory, and battery. Those limitations restrict IoT devices to be part of the blockchain network. Like previous work, we use a gateway to connect the blockchain network and IoT devices. Gateway is an interface that transforms the information of IoT devices into messages that can be understood by blockchain nodes. Multiple IoT devices can be connected to a gateway and multiple gateways can also be connected to multiple blockchain nodes. The gateway cannot be a constrained device. Such devices need high-performance features to be able to serve as many simultaneous requests as possible from IoT devices.
6) *IoT devices*: can be smart home devices, mobile devices, wearable medical devices, industrial sensor devices, etc. IoT devices connects to the gateway via short-range communication technologies like Bluetooth, Wi-Fi, and Zigbee, and etc. Then interact with the blockchain network via the gateway to execute access control operations.
IoT devices do not belong to the blockchain network. Consequently, one of the requirements of our architecture is that all the devices will have to be uniquely identified globally in the blockchain network. Public key generators can provide a feasible solution for the problem by producing acceptable large and unique random numbers. Typically, we can use the existing IoT cryptographic technologies to create a public key for every device.

3.2 Improved Role Based Access Control Model

Due to the constrained nature of IoT devices can not achieve accurate access management, such as sensors. We have improved the RBAC model. There are three sets: U, R and P in traditional RBAC models. We add a device set D on this basis. The improved RBAC relationship is shown in Table 1. Through this methodology, we can manage the access policy in a batch manner. And it can deal with the massive IoT device policy managements by calling about two functions.

Table 1. Relationship of user-role-permission-devices

User	Role	Permission	Device
Address 1	Role 1	execute	d_1, d_2, d_3, ...d_n
Address 1	Role 1	write	d_1, d_2, d_3, ...d_n
Address 1	Role 1	store	d_1, d_2, d_3, ...d_n
Address 1	Role 2	write	d_1, d_2, d_3, ...d_{n-1}
Address 1	Role 2	store	d_1, d_2, d_3
Address 2	Role 3	write	d_1, d_2, d_3, d_4, d_5,
Address 2	Role 4	read	d_1, d_2, d_3, ...d_{n-2}
Address 3	Role 5	read	d_1, d_2, d_3, ...d_{n-3}

4 Privacy-Preserving

In this part, we introduce how our scheme preserve the privacy. We describe three main parts: encryption and decryption system, zero-knowledge proof process, trusted execution environment (TEE) protocol.

4.1 Encryption and Decryption System

The key system of DBS is shown in Fig. 3. Where A represents a sender, which can be the resource owner or access requester in the system. B stands for a receiver, which can be a smart contract in the system.

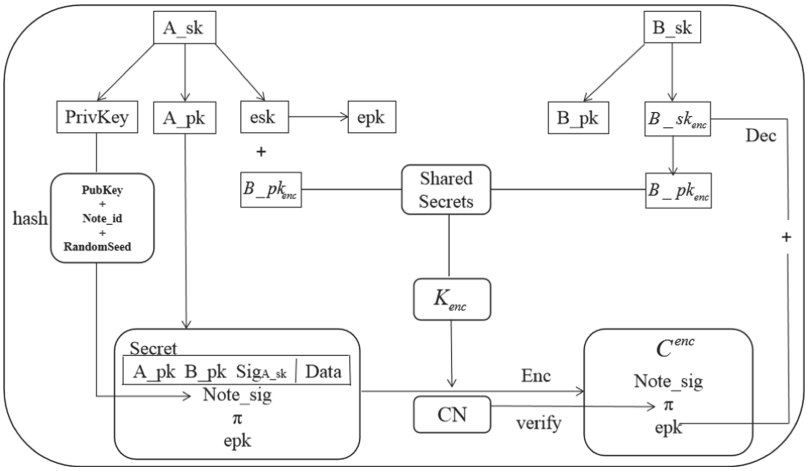


Fig. 2. Encryption and decryption system.

1) *Encryption* is used to protect the privacy of both parties and the confidentiality of the transaction. First, Sender A will obtain public key B_{pk} and encrypted public key $B_{pk_{enc}}$ of receiver B. Then the sender will generate an asymmetric key pair esk and epk temporarily. esk and $B_{pk_{enc}}$ are used to generate a sharedSecret, and then a symmetric key K_{enc} is generated. The sender can encrypt the transaction with symmetric secret key K_{enc} to protect the privacy of both parties and the confidentiality of the transaction.
At the same time, to ensure the integrity of the transaction and prevent Man-in-the-middle Attack, the sender will temporarily generate another asymmetric key pair PrivKey and PubKey to sign the transaction before encrypting the transaction. The signature Tx_{sig} is generated by hashing PubKey, RandomSeed and $Note_{id}$.
Before sending the transaction, the sender will generate a zero-knowledge proof π about the transaction amount to hide the transaction amount. To ensure the correctness of the transaction amount and prevent double flower

transactions, the proof π and random public key epk are write to the transaction.

2) *Decryption*: After the transaction is verified, it will be stored on the blockchain in an encrypted form. The receiver can leverage the encrypted private key $B_{sk_{enc}}$ and random public key *epk* to generate the symmetric secret key K_{enc} which can be used to decrypt the transaction.

4.2 Zero Knowledge Proof Process

This part introduces the application of zero-knowledge proof in DBS. Let s be a safety parameter, n be a large prime number, p,q be two random large prime numbers. n, λ and k are calculated by formulas (1), (2), and (3) respectively. g is an element with larger order in the Z_n^*, and g_1, g_2, h_1 and h_2 are the elements in the cyclic group generated by g.

$$n = pq \tag{1}$$

$$\lambda = lcm(p-1, q-1) \tag{2}$$

$$k = g^{\lambda} \mod n^2 \tag{3}$$

1) Proof of equal transaction amount: Formula (4) takes parameters r_1, g_1, h_1 and n to generate commitment C_1 for transaction amount t. Formula (5) takes parameters r_2, g_2, $h2$ and n to generate commitment C_2 for transaction amount t. And formula (6) takes parameters r_1, r_2, g_1, g_2, h_1, h_2 and n to generate evidence $eproof$ for the equivalence of transaction amount t. The consensus node performs the verification process by using formula (7) through commitment C_1, C_2 and evidence $eProof$.

$$C_1 = g_1^x h_1^x \mod n \tag{4}$$

$$C_2 = g_2^x h_2^x \mod n \tag{5}$$

$$eProofGen = (t, r_1, r_2, g_1, g_2, h_1, h_2, n) \tag{6}$$

$$eProofVer = (eProof, C_1, C_2, g_1, g_2, h_1, h_2, n) \tag{7}$$

2) Proof of transaction amount greater than 0: Formula (8) takes the parameters r, g, h and n as the secret transaction amount t to generate commitment C. Formula (9) takes parameters a, r, g, h and n as secret transaction amount t to produce proof $gproof$ larger than parameter a. Formula (10) takes the proof gproof, parameter, a, r, g, h, C, n to judge whether the secret amount in commitment C is greater than parameter a.

$$C = g^x h^r mod n \tag{8}$$

$$gProofGen = (t, a, r, g, h, n) \tag{9}$$

$$gProofVer = (gProof, a, C, g, h, n) \tag{10}$$

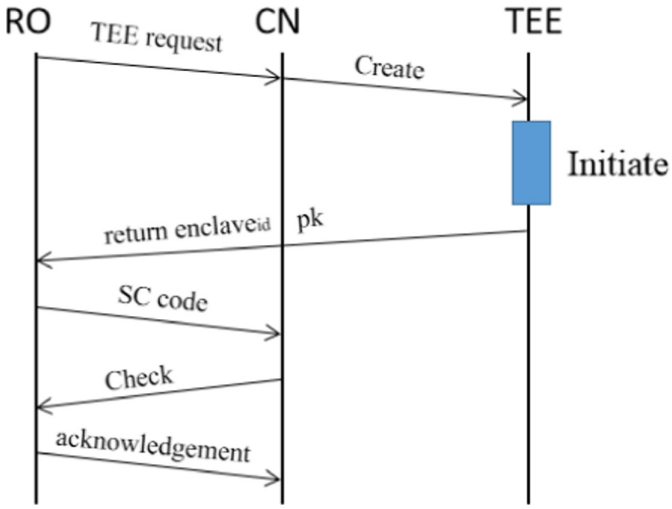


Fig. 3. Contract deployment.

4.3 Trusted Execution Environment Protocol

To protect the privacy of smart contract and access control process. We use trusted execution environment technology in DBS. In this part, we will introduce the details of the trusted execution environment (TEE) protocol. We use TEE protocol in two scenarios: contract deployment and contract invocation.

1) Contract Deployment: The RO write contract code on their clients using native languages (like C/Solidity), then compiles the code, and deploys them to TEE. Figure 3 shows the protocol process for deploying contracts.
   Firstly, the RO will request the TEE from the CN through a secure channel. The CN initializes an enclave in the local TEE. The enclave will generate an asymmetric key pair, of which the private key is only kept by the enclave, so that the subsequent invocation can be protected by the public key.
   After receiving the enclave information and public key, the RO will upload the hash of the contract binary code. After confirming the contract, the CN loads the binary code of the contract into the previously initialized enclave.
   Finally, after the contract deployment transaction is acknowledged, the contract information and enclave public key will be broadcast to all nodes of the blockchain.
2) Contract Invocation: Once the contract is deployed, the AR can send an access request to invoke the smart contract. Figure 4 shows the protocol process of invoking the contract.

First, the access requester sends the access request to the smart contract. The consensus is found in the trusted enclave.

After receiving the access request, the CN will find the enclave where the invoked contract is located, and then load the encrypted access request into the

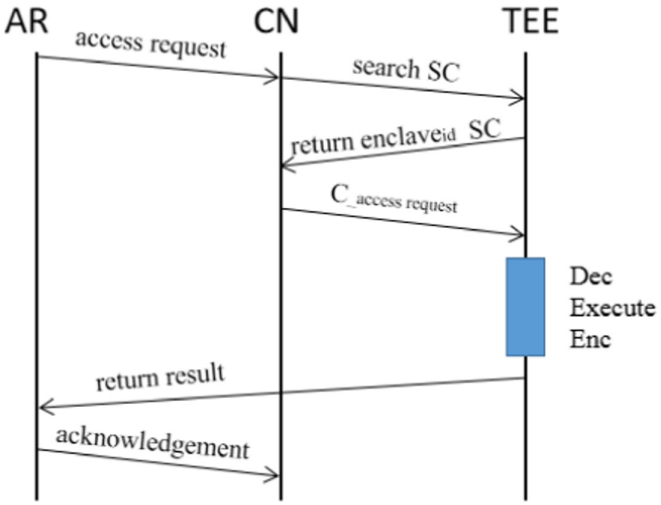


Fig. 4. Contract invocation.

target enclave. The enclave will decrypt the access request with the private key, and then take the data in the access request as the input to execute the contract. The execution result is encrypted with the public key of the AR and returned.

At last, after the access request transaction is acknowledged, the encrypted access results will be broadcast to all nodes of the blockchain.

5 Access Control Process

This section presents the smart contract-based distributed access control process. We introduce the access control flow of two different access requesters in the same domain and cross domain. The optimization of the IoT access control system can be realized with the help of an authorization contract and authentication contract.

To protect the privacy of transactions, each entity conducts a one-time setup phase that results in two public keys: a proving key pk and a verification key vk before sending the transaction. The proving key pk is used to encrypt the transaction and produce a proof π. The non-interactive proof π is a zero-knowledge proof. Then the encrypted transaction (Tc) and proof π are uploaded to the blockchain. The encrypted transaction address, transaction amount, and transaction information cannot be visible. CN uses the verification key vk to verify the proof π; in particular zk-SNARK proofs are publicly verifiable: anyone can verify π, without interacting with the prover that generated π.

5.1 Access Control Process Within the Domain

For requesters in the domain access, the RO generates an invoke transaction including user ID, role, permission, and devices, which is encrypted with the

public key of the authorization contract, then sends to the blockchain. After the invocation transaction is acknowledged, the *Addrole()* operation will add the information to the access control list of the authorization contract. The access requester sends an encrypted access request to the authentication contract, which will identify the access requester. The access control flow of the access requester in the domain is shown in Fig. 5.

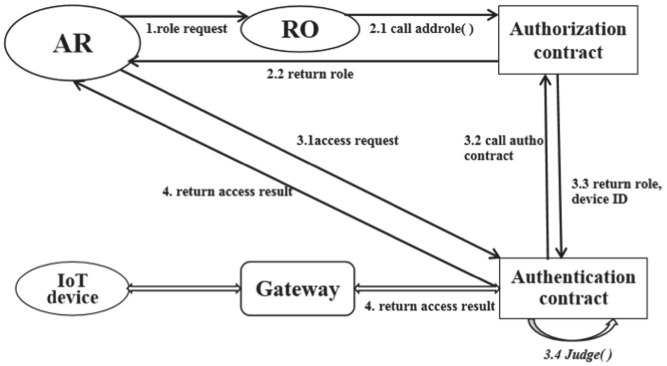


Fig. 5. Access control process within the domain.

- Step 1: The AR sends the role request to the RO.
- Step 2: The RO assigns the role to the AR after receiving the role request.
- Step 3: The AR sends an encrypted access request to the authentication contract.
- Step 4: The authentication contract returns the encrypted access result.

5.2 Access Control Process Cross Domain

For the cross-domain access requester, the RO will add the newly defined access control policy to the authorization contract after receiving the signature of the cross-domain access requester for identification. The cross-domain access requester sends an encrypted role request to the authorization contract, including signature and user ID. The authorization contract assigns roles to cross-domain access requesters according to the defined access control policies, and adds the information of cross-domain access requesters to the access control list by using the *Grantrole()* operation. The cross-domain access requester sends an encrypted access request to the authentication contract, which will identify the cross-domain access requester. The access control flow of the cross-domain access requester is shown in Fig. 6.

- Step 1: The AR sends the signature to the RO.
- Step 2: The RO adds a new access control policy to the authorization contract.

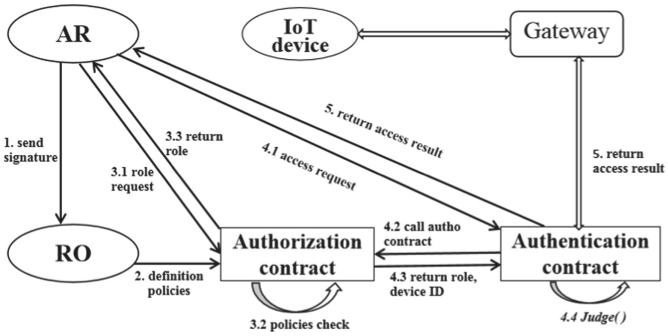


Fig. 6. Access control process cross the domain.

- Step 3: The AR sends a role request transaction encrypted with the contract public key to the authorization contract.
- Step 4: The AR sends an encrypted access request to the authentication contract.
- Step 5: The authentication contract returns the encrypted access result.

6 Implementation

6.1 Hardware and Software

In our experiment, the specifications of the devices used are listed in Table 2. The station plays the role of RO, desktops plays the role of CN, laptops play the role of AR due to their relatively large computing and storage capability, and the Raspberry function plays as local gateways. On each device, a geth client is installed as an Ethereum node.

Table 2. Specification of devices

	CPU	Operating system	Memory	Hard disk
Lenovo ThinkStation P910	Intel Xeon E5-2640 v4, 2.4 GHz	Window 10 (64bit)	64 GB	2 TB
Lenovo 10N9CTO1WW	Intel Core i7-7700, 3.6 GHz	Window 7 (64bit)	8 GB	2 TB
Lenovo N50	Intel Core i5-4210, 1.7 GHz	Window 7 (64bit)	4 GB	500 GB
Raspberriy ModelB	Contex A53, 1.2 GHz	Raspbian GNU/Linux 8	1 GB	16 GB

6.2 Experiment

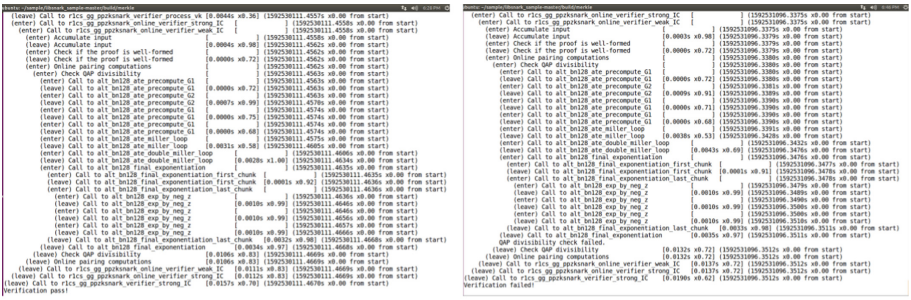
The experiments were done on an Ubuntu-16.4.3 desktop with Intel Core i7-7700, 3.6 GHz. We used Truffle to deploy and invoke smart contracts, which is a development framework based on Ethereum's solidity language. We use Ganache to develop and test local blockchain, which simulates the functions of a real Ethereum network, including multiple accounts and ether for testing.

In the experimental case, we completed the establishment of local blockchain, the compilation and deployment of authorization contract, and authentication contracts to achieve fine-grained access control between entities. To verify the feasibility of the access, the access requester sends an access operation to the IoT device, and the result of the access request is shown in Fig. 7. To verify the privacy of the transaction, the consensus node verifies the proof pi, and the verification result is shown in Fig. 8.

Fig. 7. Access result.

6.3 Performance

Because of the mass of IoT devices, the local gateway needs high-performance features to serve as many requests from IoT devices at the same time as possible. We tested the throughput and bandwidth of the local gateway, and the results are shown in Fig. 9. We also tested the response time required for the access request from the local gateway to the IoT device. As the number of access requests increases, the response time also increases, as shown in Fig. 10.



(a) Verification pass (b) Verification failed

Fig. 8. Verification result.

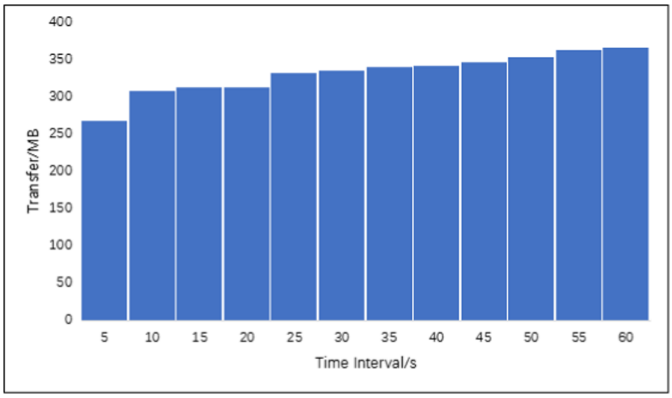


Fig. 9. Throughput of local gateway.

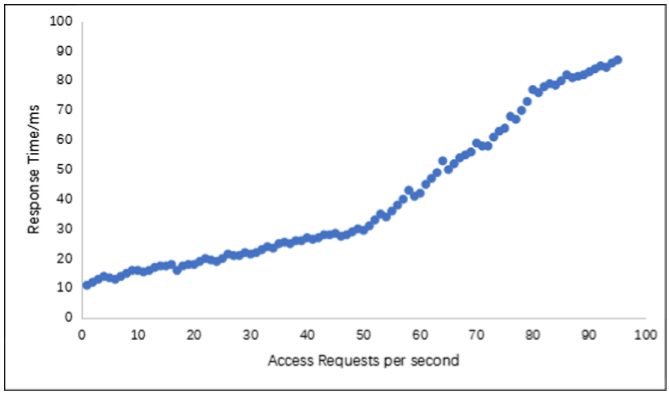


Fig. 10. Response time.

7 Conclusion

In this paper, we investigated the privacy issue of access control schemes based on blockchain in the IoT and proposed an architecture with privacy-preserving based on blockchain to achieve distributed, trustworthy, secure, and convenient access control. The architecture consists of two parts: an access control part, and a privacy protection part. The access control part includes double smart contracts to grant roles and check access rights respectively. The privacy part includes transaction privacy and smart contract privacy. The transaction information is kept secret by zero-knowledge proof algorithm, and the smart contract is protected by TEE. Also, we performed the deployment and invoke of smart contract on the local private chain Ganache and implemented zero-knowledge proof algorithm on Ubuntu. The results demonstrated the feasibility and confidentiality of the proposed architecture in achieving distributed and trustworthy access control for the IoT.

References

1. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
2. Novo, O.: Blockchain meets IoT: an architecture for scalable access management in IoT. IEEE Internet Things J. **5**(2), 1184–1195 (2018)
3. Ouaddah, A., Abou Elkalam, A.: Fairaccess: a new blockchain-based access control framework for the IoT. Security **9**, 5943–5964 (2017)
4. Du Rhuizhong, L.Y., Liu, A.D., Du, X.H.: An access control method using smart contract for Internet of Things. J. Comput. Res. Dev. **56**(10), 2287 (2019)
5. Reyna, A., Martin, C., Chen, J., Soler, E., Diaz, M.: On blockchain and its integration with IoT. Challenges and opportunities. Future Gener. Comput. Syst. **88**, 173–190 (2018)
6. Konstantinos Christidis, M.D.: Blockchains and smart contracts for the Internet of Things. IEEE Access **4**, 2292–2303 (2016)
7. Sandhu, R.S., Feinstein, H.L., Coyne, E.J.: Role-based access control models. IEEE Comput. **29**(2), 38–47 (1996)
8. Di Francesco Maesa, D., Mori, P., Ricci, L.: Blockchain based access control. In: Chen, L.Y., Reiser, H.P. (eds.) DAIS 2017. LNCS, vol. 10320, pp. 206–220. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59665-5_15
9. Ding, S., Cao, J., Li, C., Fan, K., Li, H.: A novel attribute-based access control scheme using blockchain for IoT. IEEE Access **7**, 38431–38441 (2019)
10. Cruz, J.P., Kaji, Y.: RBAC-SC: role-based access control using smart contract. IEEE Access **6**, 12240–12251 (2018)
11. Buterin, V.: A next-generation smart contract and decentralized application platform (2016)
12. Azaria, A., Ekblaw, A., Vieira, T., Lippman, A.: Medrec: using blockchain for medical data access and permission management. In: 2016 2nd International Conference on Open and Big Data (OBD), pp. 25–30, August 2016
13. Zhang, Y., Kasahara, S., Shen, Y., Jiang, X.: Smart contract-based access control for the Internet of Things. IEEE Internet Things J. **6**(2), 1594–1605 (2019)
14. Dorri, A., Kanhere, S.S., Jurdak, R., Gauravaram, P.: Blockchain for IoT security and privacy: the case study of a smart home, pp. 618–623 (2017)

15. Meiklejohn, S., Pomarole, M., Jordan, G., Levechenko, K.: A fustful of bitcoin: characterizing payments among men with no names. In: Internet Measurment Conference, pp. 127–140 (2013)
16. Russinovich, M., Ashton, E., Avaessians, C., Castro, M.: CCF: a framework for building confidential verifiable replicated services (2019)
17. Yuan, R., Xia, Y.-B., Chen, H.-B., Zang, B.-Y., Xie, J.: Shadoweth: private smart contract on public blockchain. J. Comput. Sci. Technol. **33**(3), 542–556 (2018)
18. Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: The blockchain model of cryptography and privacy-preserving smart contracts, pp. 839–858 (2016)
19. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct non-interactive zero knowledge for a von Neumann architecture, pp. 781–796 (2014)
20. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy, pp. 238–252 (2013)
21. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: anonymous distributed e-cash from bitcoin. In: 2013 IEEE Symposium on Security and Privacy, pp. 397–411 (2013)
22. Ben Sasson, E., et al.: Zerocash: decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, pp. 459–474 (2014)