International Journal of Engineering & Technology, 10 (2) (2021) 139-147

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Dynamic-IoTrust: A Dynamic Access Control for IoT Based on Smart Contracts

Eman J. Samkri[1*] and Norah S. Farooqi[2]

[1,2]*College of Computer and Information System, Umm Al-Qura University, 21955, Saudi Arabia*
[*]*Corresponding author E-mail: s43980132@st.uqu.edu.sa*

Abstract

The Internet of things (IoT) is an active, real-world area in need of more investigation. One of the top weaknesses in security challenges that IoTs face, the centralized access control server, which can be a single point of failure. In this paper, Dynamic-IoTrust, a decentralized access control smart contract based aims to overcome distrusted, dynamic, trust and authentication issues for access control in IoT. It also integrates dynamic trust value to evaluate users based on behavior. In particular, the Dynamic-IoTrust contains multiple Main Smart Contract, one Register Contract, and one Judging Contract to achieve efficient distributed access control management. Dynamic-IoTrust provides both static access rights by allowing predefined access control policies and also provides dynamic access rights by checking the trust value and the behavior of the user. The system also provides to detected user misbehavior and make a decision for user trust value and penalty. There are several levels of trusted users to access the IoTs device. Finally, the case study demonstrates the feasibility of the Dynamic-IoTrust model to offer a dynamic decentralized access control system with trust value attribute to evaluate the internal user used IoTs devices.

Keywords: *access control, blockchain, Internet of Things (IoT), smart contract, trust value.*

1. Introduction

with With the growing number of communication and networking technologies, more physical devices are connected to the Internet, leading to the 'Internet of things' (IoT) [1]. The goal of IoT technology integrates the physical world with the internet. The IoT makes objects hear, talk, wash and even think. The IoT spread quickly across the globe, constantly expanding and becoming embedded applications in our world and daily life such as smart homes, smart healthcare, security systems, intelligent transportation, etc [2]. However, the interconnection of the device brings crucial security issues to the network. Some of the security issues are: single point of failure through centralization, unauthorized access, and many others [3]. The major concerns about protected resources, data, IoT objects, and services will inevitably increase. The main security concern of the IoT is access control [4],[5], which ensures only authorized users can access certain resources. Access control restricts who can request access (user) to perform an operation (access right) on which resource (resource) [6]. The request is evaluated based on access control policies that rely on several models to decide if the requester is granted or denied access. There are several access control models known as traditional access control methods. These include role-based access control (RBAC), discretionary access control (DAC), mandatory access control (MAC), and attribute-based access control (ABAC). These methods are all centralized methods and thus may not be suitable for the IoT environment, which has a single-point failure, low reliability, untrustworthiness, unauthorized access, is difficult to scale, and requires trusting foreign entities and low throughput. IoT devices may belong to different organizations or users, and the IoT environment usually consists of a large number of constrained devices with high mobility and limited power and performance, which makes traditional access control difficult. However, may lie in emerging blockchain technology has many strengths that make it distributed and immutable. Applying blockchain technology may help address some limitations of the IoT. The integrated IoT-Blockchain system has several advantages [7]:

- Blockchain can hand over a secure, distributed, and scalable framework for the IoT.
- Blockchain guarantees IoT data integrity without a third party so sensitive data can be delivered securely without a central server, saving power for IoT devices.
- Blockchain equipment has authentication functions so it can transfer data between IoT peers without a centralized server.
- Blockchain provides another kind of emerging data to ensure the reliability of the data through distributed storage.
- Blockchain promising platform for developing distributed and trustworthy applications.

Most researches in IoT-Blockchain access controls aim to meet the distributed security and control granularity requirements of access control to authorized external users and obtain considerable achievements [8],[9],[10], [11],[12], whereas the Dynamic-IoTrust aims to overcome issues of centralization access controls issues by using blockchain to authorized external user to access authorized IoTs and also supports the trust value that evaluates the internal user by detecting malicious user behaviors to putting in suitable trust level. This system provides both static and dynamic validation processes, including credible distribution of attributes and policies defined by the user (static access control), detecting malicious user behaviors, and calculating trust value (dynamic access control). The Dynamic-IoTrust uses multiple smart contracts to achieve security and reliability and also defines alternative methods for a user to gain permissions. The rest of the paper is organised as follows: Section 2 presents details of previous work in the IoT-Blockchain field; Section 3 introduces and describes the Dynamic-IoTrust model in detail; the results and performance evaluation of Dynamic-IoTrust are detailed in Section 4; finally, we conclude our study in Section 5.

2. Literature survey

The access control taxonomy of the IoT environment can be centralized and decentralized [13]. For all centralized access control (traditional access control methods) in IoT such as RBAC, DAC, MAC, and ABAC lead to a single point of failure problem and privacy issues, etc. Researchers have suggested ways of adapting traditional access control models to meet new IoT security requirements. Similarly, Ding et al. [8] integrated blockchain technology with the ABAC model into the IoT. For instance, Zhang and Tian [9] extended the RBAC model to include context-based access control in the IoT by changing the permission access between subject and object according to the characteristics and contextual information collected from the physical object environment. But still failed approach to manage rapidly growing IoT devices with the expanding system. the decentralized access controls can help to resolve drawbacks. The work in [13] classification the decentralized approach into non-blockchain-based access control and blockchain-based access control. The IoT and blockchain have recently garnered attention from both academic and industry researchers. The literature reveals several schemes that integrate blockchain with the IoT based on the transaction methods of the blockchain. there are many types of blockchain transactions such as tokens and smart contracts. For example for the tokens approach, Ouaddah [14] utilized blockchain and token to introduce a distributed authorization management framework called FairAccess, which uses access tokens containing the access right of a user's device and stores the token in the user's wallet. FairAccess uses a new type of transaction using smart contracts to trade access control policies for access tokens. However, FairAccess uses tokens generated by the owner so, if a token expires, the subject needs to generate a new token. This system requires creating a strategy for each device, which makes account management difficult due to contract redundancy. Also, FairAccess makes it costly to access the object because it requires mining at least two blocks to get the request. However, only a few researchers have studied the possibility to exploit the potential of the blockchain smart contract in the IoT. The main feature of used a smart contract in access control in an IoT environment is to enforce the access control policies between two parties automatically by the smart contract, also offering numerous access control management methods. Novo [10] proposed a fully decentralized access control for IoT using blockchain to store and distribute information. This solution designed to overcame issues of network overheads. Moreover, the Novo [10] contain component as follow manager, agent node and management hub. Also, contains a single smart contract that covers all processes and defines all allowed access control policies by using management hub nodes to request access control alternatives including IoT devices in the blockchain. managers responsible for adding new policies in the system and interact with the smart contract. when received access requests from IoT devices, the management hubs send to the blockchain network request. Then, the smart contract automatically checks the request to approval permission to the resource. The Novo [10] system may face a security problem if the manager is malicious. Zhang et al. [15] also utilized smart contracts consisting of multiple Access Control Contracts (ACCs), one Judge Contract (JC), and one Register Contract (RC). Their system uses an access control system that provides distributed and validated access to the data. However, the authors in [] consume high cost in terms of gas comprised with Dynamic-IoTrust. And also lacked the dynamic trust value feature provide by Dynamic-IoTrust. to address the limitations of the above works, this paper proposes a Dynamic-IoTrust, which consists of multiple smart contracts and trust value permission control to achieve dynamic distributed and trustworthy access control for IoT systems. The Dynamic-IoTrust, has multiple main smart contracts (MSCs), one register contract (RC), and one judge contract (JC).

3. Proposed system model

A dynamic-IoTrust system is proposed in this work, which is designed after getting motivation from work in [15] and [16]. The proposed system model is illustrated in Figure 4. The IoT system consists of peers which are connected together through a peer-to-peer (P2P) network. the peers may server, IoT devices such as sensors, storage devices, and user devices. For each peer in IoT applications may have some resources such as services and data needed by the other peers. Thus, access control must be implemented by all peers in IoT to restrict access and prevent illegal access requests from unauthorized peers. The goal of this paper is to propose Dynamic-IoTrust that adopt the security model of access control matrix and implement via access control list approach as in [23]. In particular, Dynamic-IoTrust exploits smart contracts to implement an access control list. Dynamic-IoTrust adopts trust value to evaluate user behavior.

3.1. System architecture

The architecture of the dynamic decentralized access control for IoT is based on the blockchain mechanism presented in Figure 2. In Dynamic-IoTrust, access control is done between two network peers called user and resource nodes. The user-node is actually a peer that requests the services such as the device or data of other peers. Whereas, the resource-node is a peer that holds services. For each resource-node has a set of services is associated with a set of access rights such as (read, write, and executed). In Dynamic-IoTrust we defined mapping user-node to resource-node to specify the access rights on the user-resource pair. Based on the security model used in Dynamic-IoTrust [13], we adopt the access control list. Furthermore, used three kinds of smart contracts to manage the access control among user and resource nodes. Also, the trust value is used in Dynamic-IoTrust to evaluate the internal user access. The system consists of two components, the detailed discussion of components below:

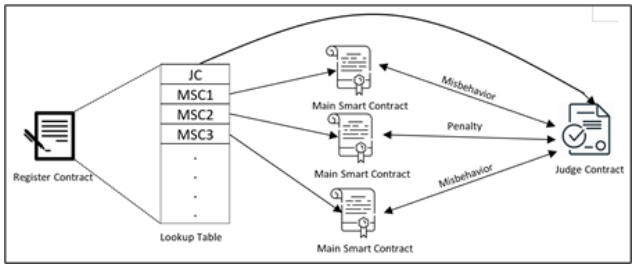


Figure 1: Smart contracts structure in Dynamic-IoTrust.

3.1.1. Smart contracts

The system uses the smart contract to provide distributed access control for IoT, also to eliminate third-party involvement, restrict data access to desired parties, and manage data records and utilize the data within a self-executing system. As illustrated in Figure 1 the system contains multiple Main Smart Contract (MSC)s that implements the access control for a pair of peers (user and resource), one Judging Contract (JC) to receives the user's misbehavior from an MSC and determines the penalty, one Register Contract (RC) to stores and manage the information of the JC and MSCs. The details of the contracts introduced as follows:

- **Main smart contract (MSC):** In this framework is deployed by a peer request to receive service from resource peer, in the proposed system assume the user-resource pair associated with multiple MSCs, but one MSC can be associated with only one user-resource pair. The MSC provides static access rights by allowing predefined access control policies and also provides dynamic access rights by checking the trust value and the behavior of the user. The MSC forwards the request to the resource corresponding to the JC decision.
- **Judge contract (JC):** The responsibility of JC is to check the user misbehavior, trust value validation, and also predefined policies. The JC determines penalty corresponding to the decision of all factors that check and returns the result for authorized or penalty to the MSC.
- **Register contract (RC):** The main role of RC in a system is to collect static predefined access right policies and register misbehavior history for the user-node.

3.1.2. Trust value permission control

The Dynamic-IoTrust is intended to provide dynamic access control for IoT depending on trust value attribute reflects their available levels of trust and rating among other IoT devices. The trust value levels are defined from 0 to 5 for a user-node to access services from a resource-node, which level zero means the user-node brings misbehavior to the system. Whereas, level five means user-node is trusted by the system and brings good behavior.

- **Default trust value:** At the start the default trust value for
  - the resource node is $Trust_{resource}$ determine statically according to the sensitivity level of data,
  - the user node $Trust_{user}$ initially start from 2.
- **Evaluation user-node trust value:** User-node trust value changes over time according to the factors below collected by methods as follows:
  - **Misbehaviour judging method:** In the system, misbehavior is done by the users and determined by a misbehavior judging method. JC records of misbehavior of user-node so the user gradually earns more trust value with good behavior brings benefit to the system after access permission and loses it with bad behavior brings to the system. When a user sends too frequent service access requests to the resources at a particular time, it is known as misbehavior. Furthermore, the system highly affects the trust of the user. As the result of misbehavior, the JC determine the penalty for user-node based on the following function

    $$penalty = base^{*}x/interval$$

    where x is the number of misbehavior done by user-node and the other parameter base and interval are determined during the period time of penalty. the equation used on [15]
  - **Static attribute policy method:** this system supports the static predefine access right policy that deals with attributes of the user, the resource, the action, and the environment to determine the access request.
- **Check the validity of trust value for user-node:** The trust value check successfully pass if and only if the trust value of user-node is equal to or larger than resource-node ($Trust_{user} \geq Trust_{resource}$). whereas failed if the trust value of user-node less than trust value of resource-node ($Trust_{user} < Trust_{resource}$).

3.2. System model workflow

The access request is authorized if and only if it succeeds in both static and dynamic validation processes. The same user in the same session automatically granted different access permissions. The access controls alert messages generated as in Table 1:
The logical flow of access control data in Dynamic-IoTrust proceeds as illustrated in Figure 2, below.

- At first, a user-node sends an access request for any service to resource-node (IoT device),
- After that, user-resource pair agree on registering the new MSC in the lookup table of the RC by creator of the method (i.e., resource-node),

Table 1: Alert messages for access requests.

Access alert messages	State of access process
Access Request Authorized!!	Enable access request to resource-node, Dynamic-IoTrust authorized both static and dynamic validation process.
Requests Access are Blocked!!	The user-node still in a penalty state and the access request is blocked.
Policy Predefined Check Failed!!	Block user request to access the IoT device because predefine permission for that user is not allowed.
Misbehavior Detected and User in Penalty State!!	JC determent penalty and for alert the user but result access request is authorized.
User Trust Check Failed!!	Block access request to access the resource device because the $Trust_{user} < Trust_{resource}$.

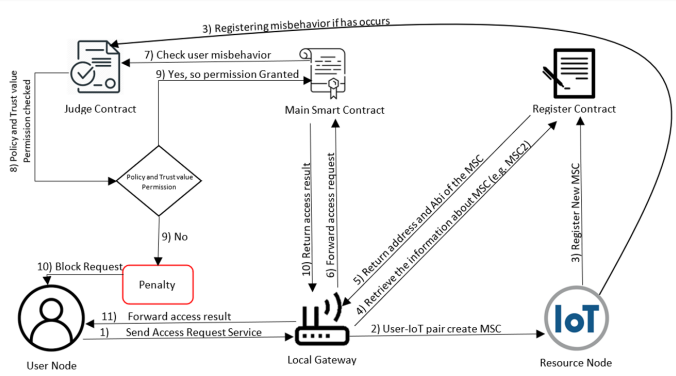


Figure 2: Dynamic-IoTrust system.

- Then, the MSC is called to authorize its current request via check both static and dynamic access request to the system,
- Afterward, JC is called for verification user misbehavior,
- If detecting misbehaviors conducted by the user, then the JC determines penalty and lose trust value,
- If the user does not occur misbehavior, then, trust value permission checked if the user passes successfully,
- Also checked the Policy permission if a user passes successfully,
- After that, MSC returns the access result of a user to corresponding resources,
- Then, the user request is fulfilled by the resource-node,
- Lastly, the transaction store in the blockchain network.

4. Case study

In this section, demonstrate the Dynamic-IoTrust case study and simulation result are discussed. Hardware and software for experiment Dynamic-IoTrust are described. Further section presents some experimental results and the cost consumption of each smart contract.

4.1. Experimental environment

In order to design an access control for the Internet of Things environment, needs in case study constructed the blockchain network using Ethereum [17], Quorum [18], and the Istanbul-BFT consensus algorithm [19]. they considered the access control test between the single-board computers with 4 nodes which one serves as the user and the other nodes serve as the resource IoT device. We implemented it on a laptop hp INTEl Core i3-4005U, CPU 1.70GHz×4, Ubuntu 18.04 LTS, memory 5.6GiB, and hard disk 491.2 GB. The network configuration of the prototype is composed of three peer nodes and one order node the details in Figure 4. The specifications of these nodes are listed in Table 2.
On each node, a Geth client [20] was installed to transform the file into an Ethereum node. The use of Geth clients can enable the creation of an Ethereum account for each node, and these nodes can be configured to create a private blockchain network. The smart contacts are

Table 2: Nodes structure.

Implementation Platform	Role	Node number	IP address and Port number
Laptop (Consortium blockchain network) HP Intel Core i3- Ubuntu 18.04 LTS	User node	Node0	192.168.191.2:30300
	IoT device1	Node1	192.168.191.2:30301
	IoT device1	Node2	192.168.191.2:30302
	IoT device1	Node3	192.168.191.2:30303

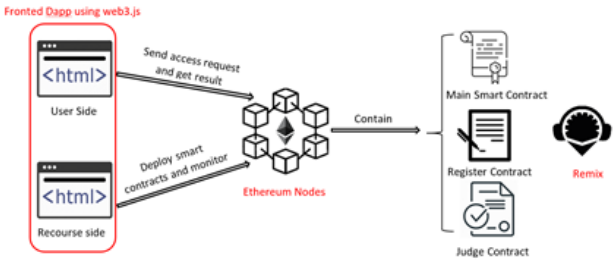


Figure 3: Software used in the Dynamic-IoTrust.

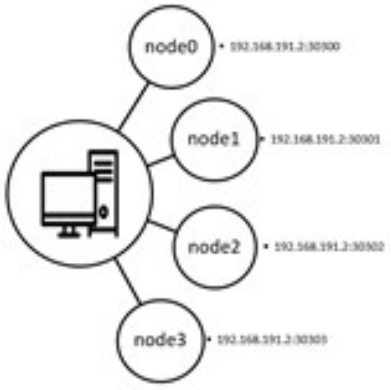


Figure 4: Nodes structure.

written in solidity language [21]. Furthermore, a Remix environment [22] is used for smart contract development, facilitating compilation, deployment, and management of the smart contracts. In addition, web3.js [23] was adapted at the user-node and resource-node sides to interact with the corresponding Geth client via HTTP connections to simulate the sending of access requests to the MSC and the receipt of the access control result from the MSC (as illustrated in Figure 3). Moreover, the cost consumption of smart contracts are taken in the plugin used in Remix known as GAS PROFLER

4.2. Experiment results

4.2.1. Dynamic-IoTrust framework feasibility

To develop a decentralised application in Ethereum (Dapp), it was first necessary to initiate a local private Ethereum blockchain network, then to create smart contracts in Remix, and then to deploy these on the blockchain network. Finally, once the Dapp front-end and the smart contracts were created, it was necessary to connect them via web3 to test the entire framework, as illustrated in Figure 3.

- **Private blockchain build test** The pre-generation of each node and related files such as the node json file, the first block in blockchain known as the genesis file, and the node keys was carried out on the laptop using istanbul-tools. Istanbul-tools was used for configuring an Istanbul BFT network and testing the IBFT. Geth and Quorum were used to build the private network (as illustrated in Figure 5).
- **Smart contract function test** In this phase the smart contract function was tested. The Remix editor was used to write, compile, deploy and test the smart contract code written in solidity language.
- **Front-end test for the dynamic access control result** The access control results are shown in Figure 6. When the user makes a request for the first time, access is authorised because they successfully pass both the policy and trust value checks. A second access request is still authorised, but misbehaviour is detected, so that warning messages appear and loss of trust value and a user penalty are determined. The user still misbehaves by making frequent access requests in a specified time, as shown in Figure 6(b), and the trust value is lowered. The user's fourth access request fails because the trust value of the user node is less than the trust value of the resource node.

Figure 5: Private blockchain build test

```
Send access request?(y/n)y
Contract Address: 0x94e4fa4289047F7e8968F27b30D79c8674eeaFA5
Message: Access Request Authorized!!
Trust value for subject: 4
Trust value for object: 3


Send access request?(y/n)y
Contract Address: 0x94e4fa4289047F7e8968F27b30D79c8674eeaFA5
Message: Misbehavior Detected and User in Penalty State!!
Trust value for subject: 3
Trust value for object: 3
Requests are blocked for 1seconds!
```

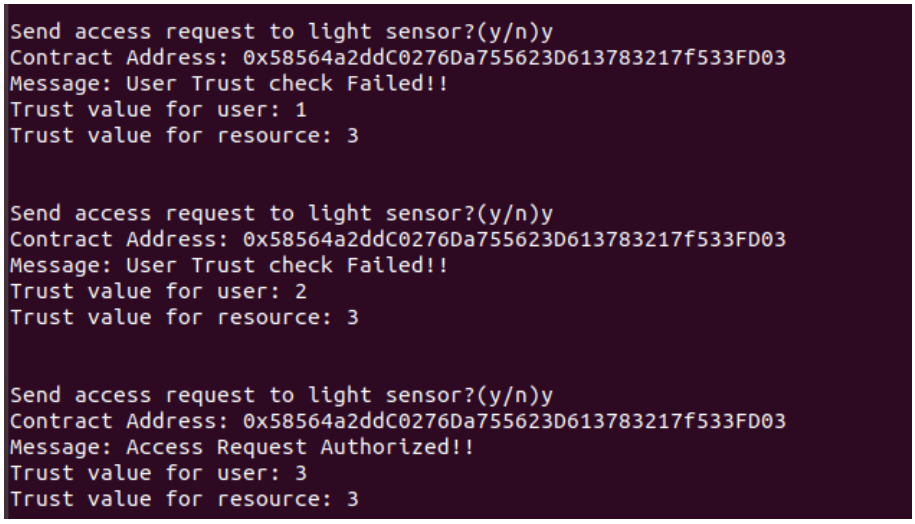
```
Send access request?(y/n)y
Contract Address: 0x94e4fa4289047F7e8968F27b30D79c8674eeaFA5
Message: Requests Access are Blocked!
Trust value for subject: 2
Trust value for object: 3


Send access request?(y/n)y
Contract Address: 0x94e4fa4289047F7e8968F27b30D79c8674eeaFA5
Message: Requests Access are Blocked!
Trust value for subject: 1
Trust value for object: 3


Send access request?(y/n)y
Contract Address: 0x94e4fa4289047F7e8968F27b30D79c8674eeaFA5
Message: User Trust check Failed!!
Trust value for subject: 2
Trust value for object: 3
```

(a)

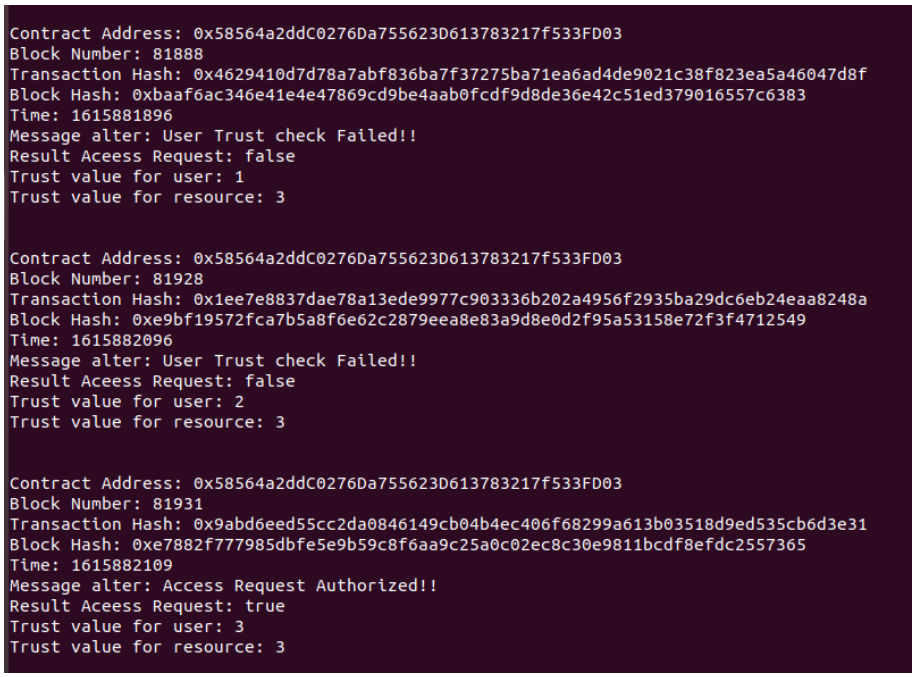
```
Contract Address: 0x58564a2ddC0276Da755623D613783217f533FD03
Block Number: 81684
Transaction Hash: 0x23311d12cea4f61dd03d5cf9d41c8ae68812e9fad1bed8576fc3c415eb82736f
Block Hash: 0x667f346730563e39c05c6f887c73f79e8088e573d537a509bd1f0d023ba5fb91
Time: 1615880878
Message alter: Access Request Authorized!!
Result Aceess Request: true
Trust value for user: 3
Trust value for resource: 3


Contract Address: 0x58564a2ddC0276Da755623D613783217f533FD03
Block Number: 81686
Transaction Hash: 0xa98274eed91242ed3e7e3e1c8d67cc3e8c35aa73a907ece6cbbccc202642fa3f
Block Hash: 0x98a75f73fea08d689e8f5b57837ac6d6c3538661d44e8d9587f63f844c308404
Time: 1615880889
Message alter: Misbehavior Detected and User in Penalty State!!
Result Aceess Request: true
Trust value for user: 2
Trust value for resource: 3
Requests are blocked for 2seconds!


Contract Address: 0x58564a2ddC0276Da755623D613783217f533FD03
Block Number: 81691
Transaction Hash: 0x33941c0981b29bcd5daa5109e509be5f5b5c838998e8bdab24c0576584f99a05
Block Hash: 0xa2d81c9c790293838e6cfb8087d73e18b65c1f6f5c3378fdd73dcf1d39b851a0
Time: 1615880913
Message alter: Requests Access are Blocked!
Result Aceess Request: false
Trust value for user: 1
Trust value for resource: 3

Contract Address: 0x58564a2ddC0276Da755623D613783217f533FD03
Block Number: 81737
Transaction Hash: 0x6cead22a9551d3b97e98940c846df00ed1f62eb598c3b9dec970e7377eccd51a
Block Hash: 0x24bcb6f0c2bc7ea3c27869975c7b9aeb120619e257d7cc5a4339d0e7ad4d6a22
Time: 1615881141
Message alter: User Trust check Failed!!
Result Aceess Request: false
Trust value for user: 2
Trust value for resource: 3
```

(b)

Figure 6: Results of an access request on (a) the monitor side and (b) the requester side when misbehaviour is detected.

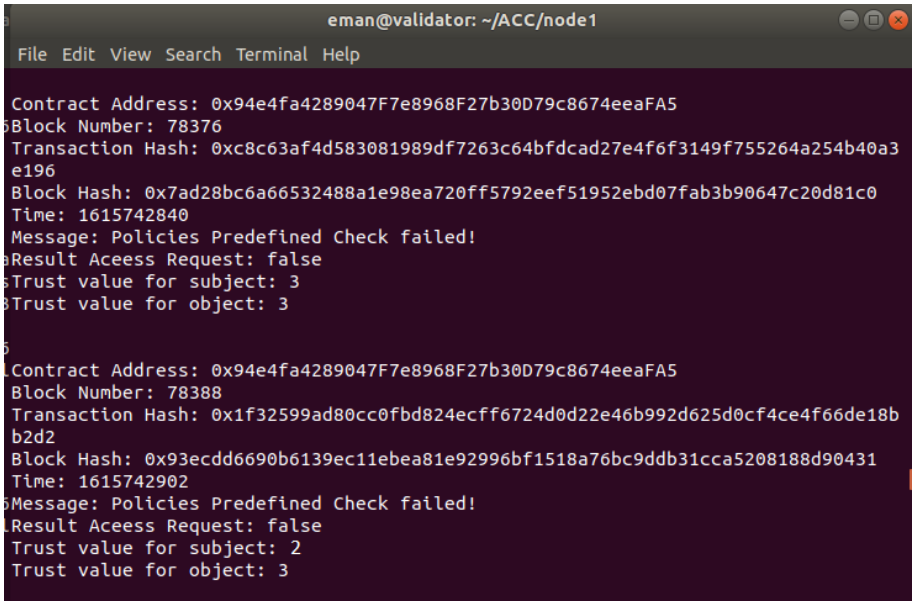


(a)

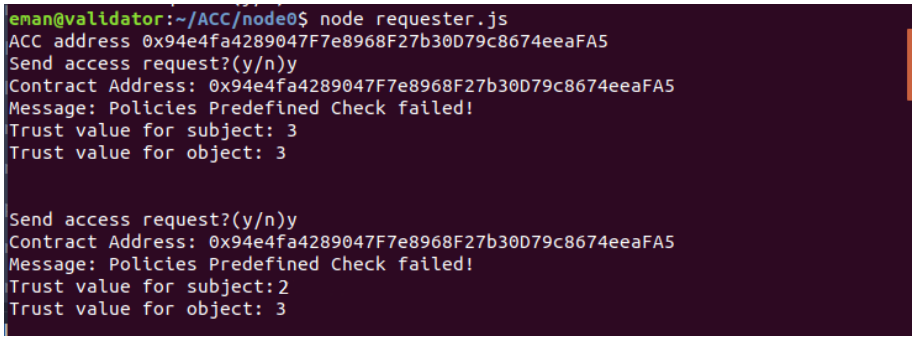


(b)

Figure 7: Results of an access request on (a) the monitor side and (b) the requester side for trust value.



(a)



(b)

Figure 8: Results of an access request on (a) the monitor side and (b) the requester side when a predefined policy does not allow access.

4.2.2. Trust value permission evaluation

The proposed system supports different access permissions for the same user. The flexibility of access requests is reflected in the trust value. In essence, the proposed system's trust value for user node is dynamic, based on the two factors mentioned in section 3.1.2.
To investigate the evaluation of trust value, a scenario is considered in which the user requests access to a light sensor. The trust value for the light sensor, $Trust_{light}$=3, is determined according to the sensitivity-level of data as explained in Section 3.1.2. As shown in Figure 7, the user's first request to access the light sensor fails because the $Trust_{light} < 3$. The $Trust_{user}$ value increases between the first and second access requests, because the user demonstrates good behaviour by not making frequent system requests (see the time in Figure 7(a)), so that at the third access request $Trust_{user} = 3$, meaning that access is authorised by Dynamic-IoTrust. However, in Figure 6 the user sends frequent requests to the light sensor, so the user is penalised for a certain time. If the user continues to send frequent requests to the light sensor, the user will still be misbehaving, and their trust value will be reduced further. Hence, in Figure 6, the fourth access request occurs when $Trust_{user} = 2$ and the access request not in a blocked state (no excessively frequent requests are observed over time). As result, the user's access request to light sensor fails because the trust value of the user node is less than the trust value of the light sensor, and the user's access remains blocked until the $Trust_{user} \geq 3$.

4.2.3. Predefined policy evaluation

The proposed system also supports static access permissions. The flexibility of the access control system is reflected both in trust value and policy. The proposed system policy is determined by pre-allocation.
To investigate the evaluation of a predefined policy, the following scenario was considered. The user requests access to a light sensor, for which the predefined policy is p=Deny in afternoon. The access request is authorised if and only if the user makes their request at any time of day except the afternoon and the trust value of the user is equal to or greater than the light sensor's trust value. On other hand, the access is blocked if the user makes the request in the afternoon or if the user's trust value is less than the light trust value. So, as shown in Figure 8, the user's first request fails because the predefined user request policy does not allow access that time, and as result the user's trust value is reduced due to the user's misbehaviour.

4.2.4. Cost consumption

the cost consumed by smart contracts in the system is calculated in terms of gas units. The Ethereum platform uses the gas units to measures the computational power to execute some tasks (transactions). In this case study, the execution costs of gas required for deploying the MSC, JC, and RC are 2373765, 975960, and 1182227 respectively.

5. Conclusion

This paper proposes an access control model for the Internet of Things based on smart contracts, supporting with trust value called Dynamic-IoTrust. It aims to overcome issues with centralization, provide a reliable and valid system, and achieve authorized access control and authenticated users. The system uses multiple smart contracts to achieve security and reliability and also defines alternative methods for a user to gain permission. The experiment result demonstrated the feasibility of Dynamic-IoTrust in achieving distributed dynamic trust access control for the IoT. Dynamic-IoTrust uses only validates authorized users to access the required resource. The proposed system provides and maintains the trustworthiness of IoT devices via JC. The cost is calculated in terms of gas units. As our future work, we are going to reduce the complexity and cost of the system in terms of gas to reduce also processing time of the system.

References

[1] Ibrar Yaqoob, Ejaz Ahmed, Ibrahim Abaker Targio Hashem, Abdelmuttlib Ibrahim Abdalla Ahmed, Abdullah Gani, Muhammad Imran, and Mohsen Guizani. Internet of things architecture: Recent advances, taxonomy, requirements, and open challenges. *IEEE Wireless Communications*, 24(3):10–16, 2017.
[2] Li Da Xu, Wu He, and Shancang Li. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4):2233–2243, 2014.
[3] Oscar Novo. Scalable access management in iot using blockchain: A performance evaluation. *IEEE Internet of Things Journal*, 6(3):4694–4701, 2019.
[4] Sheng Ding, Jin Cao, Chen Li, Kai Fan, and Hui Li. A novel attribute-based access control scheme using blockchain for iot. *IEEE Access*, 7:38431–38441, 2019.
[5] Di Lin and Yu Tang. Blockchain consensus based user access strategies in d2d networks for data-intensive applications. *IEEE Access*, 6:72683–72690, 2018.
[6] Sergio Gusmeroli, Salvatore Piccione, and Domenico Rotondi. A capability-based security approach to manage access control in the internet of things. *Mathematical and Computer Modelling*, 58(5-6):1189–1205, 2013.
[7] Laphou Lao, Zecheng Li, Songlin Hou, Bin Xiao, Songtao Guo, and Yuanyuan Yang. A survey of iot applications in blockchain systems: Architecture, consensus, and traffic modeling. *ACM Computing Surveys (CSUR)*, 53(1):1–32, 2020.
[8] Sheng Ding, Jin Cao, Chen Li, Kai Fan, and Hui Li. A novel attribute-based access control scheme using blockchain for iot. *IEEE Access*, 7:38431–38441, 2019.
[9] Guoping Zhang and Jiazheng Tian. An extended role based access control model for the internet of things. In *2010 International Conference on Information, Networking and Automation (ICINA)*, volume 1, pages V1–319. IEEE, 2010.
[10] Oscar Novo. Blockchain meets iot: An architecture for scalable access management in iot. *IEEE Internet of Things Journal*, 5(2):1184–1195, 2018.
[11] Han Liu, Dezhi Han, and Dun Li. Fabric-iot: A blockchain-based access control system in iot. *IEEE Access*, 8:18207–18218, 2020.
[12] Tanzeela Sultana, Ahmad Almogren, Mariam Akbar, Mansour Zuair, Ibrar Ullah, and Nadeem Javaid. Data sharing system integrating access control mechanism using blockchain-based smart contracts for iot devices. *Applied Sciences*, 10(2):488, 2020.
[13] Adam Ibrahim Abdi, Fathy Elbouraey Eassa, Kamal Jambi, Khalid Almarhabi, and Abdullah Saad AL AL-Ghamdi. Blockchain platforms and access control classification for iot systems. *Symmetry*, 12(10):1663, 2020.
[14] Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. Fairaccess: a new blockchain-based access control framework for the internet of things. *Security and communication networks*, 9(18):5943–5964, 2016.
[15] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan. Smart contract-based access control for the internet of things. *IEEE Internet of Things Journal*, 6(2):1594–1605, 2019.
[16] R.S. Sandhu and P. Samarati. Access control: principle and practice. *IEEE Communications Magazine*, 32(9):40–48, 1994.
[17] Ethereum. Ethereum homestead documentation, 2021. Last accessed 15 March 2021.
[18] Quorum. Goquorum, 2021. Last accessed 10 March 2021.
[19] istanbul. istanbul-tools, 2021. Last accessed 15 March 2021.
[20] geth. Go ethereum, 2021. Last accessed 13 March 2021.
[21] Solidity. Solidity language, 2021. Last accessed 16 March 2021.
[22] remix. Remix environment, 2021. Last accessed 15 March 2021.
[23] web3. web3.js - ethereum javascript api, 2021. Last accessed 15 March 2021.