Secure Access Control Authorization of IoT Resources based on Blockchain Technology

Dana Haj Hussein, PhD. Systems and Computer Engineering, Ragunath Anbarasu, M.Eng. Electrical and Computer, Carleton University, Ottawa, ON, K1S 5B6, Canada.

Abstract— **In this paper, we propose an access control authorization system based on blockchain technology for the Internet of Things (IoT) systems. The security system is implemented on a private Ethereum blockchain network. We illustrate the use of a Modifier function in the smart contract of the Ethereum blockchain network and study the performance cost associated with its deployment. In addition, we analyze the security performance of the proposed system by investigating a threat model that considers a Distributed Denial of Service (DDoS) attack. The proposed defend mechanism is by keeping a real time record of legitimate users and utilizing a Modifier function in the smart contract to restrict function calls. The results show that the system that utilizes a Modifier function is immune to DDoS attacks. Moreover, the cost of deploying the Modifier function is an average of 1.7 seconds increase is the response time.**

Index terms – Distributed Denial of Services (DDoS) attack, Internet of Things (IoT), Blockchain.

I. INTRODUCTION

The Internet of Thing (IoT) systems are being increasingly implemented to enable a wide range of applications and services. Smart systems such as smart homes and smart hospitals deploy constrained IoT devices that collect data and enable remote control of the IoT infrastructure. System owners can utilize their IoT infrastructures to provide attractive services and applications. For example, a smart home owner can control the functionality of his/her home appliances based on sensor readings. Despite their noticeable advantages, smart systems impose major security risks that can prevent their fast deployment. Particularly, a standard process for authentication and access control is still in the developing phase.

The Authentication and Authorization for Constrained Environments (ACE) framework is an extension of the OAuth 2.0 framework developed and documented by the Internet Engineering Task force (IETF) [2]. It uses Datagram Transport Layer Security (DTLS) [RFC6347] and Constrained Application Protocol (CoAP) [RFC7252] for providing secure authentication and authorization. The ACE protocol is mainly designed for constrained environments with limited resources like an IoT ecosystem. The four main components of the ACE protocol are the resource server (RS), resource owner (RO), authentication server (AS) and clients. The protocol defines how third-party clients can be authenticated and granted with secure access to resources on a remote server indirectly by the RO. The RO defines the security policy which is shared with a trusted AS. The authentication flow requires clients to post their credentials to the AS to request an access token. In response, the AS verifies the credentials provided and returns an access token that includes the predefined privileges stated in the security policy. Finally, the RS uses the token information to allow clients access to authorized resources.

The ACE protocol depends heavily on the security of the AS which presents the weakest link in an ACE-based security system. To overcome the problem of a single point of failure which is presented by the AS, blockchain technology can be utilized to implement a decentralized security system based on the ACE protocol.

Blockchain technology is a distributed ledger that can record transactions in a verifiable and permanent way. It is managed using a peer to peer network making it decentralized and is considered secure by design. Therefore, a decentralized consensus is achieved using blockchains. After the success of bitcoins, many organizations and corporations have started to adopt blockchain technology [3]. Ethereum blockchain platform is an example of a popular open platform that enables the deployment of blockchain-based applications.

The objective of this paper, is to analyze the security performance of an access control authorization system using blockchain technology for IoT utilities. The authentication and authorization procedure follow the ACE protocol. However, the AS in the ACE framework is replaced by a blockchain access control authorization network. The contributions of the presented work are as follows:

- We propose an access control authorization system implemented in Ethereum platform and uses mutual username/password authentication. In addition, we illustrate the use of a Modifier function in the smart contract of the Ethereum blockchain network and study the performance cost associated with its deployment.
- We define an authorization sensitivity factor (AuthSf) which provides three classes of access privileges to users, namely: IoT-control authorization class, Real-Time IoT data visualization class and Restricted-Real-

Time IoT data visualization class. Moreover, we discuss the use of the AuthSf in real IoT systems, e.g. smart health utility.

- We analyze the security performance of the proposed system by investigating a threat model that considers a Distributed Denial of Service (DDoS) attack. The proposed defend mechanism is by keeping a real time record of legitimate users and utilizing a Modifier function in the smart contract to restrict function calls.

The rest of the paper is organized as follow: Section II provides a comprehensive background on Ethereum blockchain describing the blockchain parameters and the smart contract. Section III discusses the recent work in literature on access control applications based on blockchain technology. Section IV presents the proposed system model and section V describes the threat model. Finally, the implementation and the results are discussed in section VI and VII, respectively.

II. Ethereum Blockchain

The Ethereum protocol facilitates the easy development of decentralized applications (DApps)[4]. It allows the users to write smart contracts and develop DApps tailored to their usage. Geth is a light version of Ethereum written in Go language, a language similar to Java. It is more useful for DApps development as it also provides a safe and easy testnet for testing the developed smart contracts. The Ethereum blockchain network can be classified as public and private blockchains. While any node can participate in a public blockchain, a private blockchain is fully controlled by its administrator who controls the parameters of the blockchain and the participants of the network. This makes it feasible for private organizations to develop applications on top of private Ethereum blockchain. Each blockchain network is identified using a unique Network ID. Any number of accounts can be created in a blockchain which can be accessed by interfacing with the Ethereum platform over the internet. Each account is identified by a unique 20-byte address.

The transactions inside the Ethereum blockchain network are carried out through smart contracts written using solidity language. Solidity is an IFTTT (IF-THIS-THEN-THAT) logic-based programming language supported by Ethereum. Solidity comes with many inbuilt functions which make the development very fast and easy. A Modifier is one such function that is used for limiting access controls to function calls. For instance, using a Modifier function the owner of the smart contract can specify a set of whitelisted account addresses that can call a specific function. The smart contracts are managed using supported smart contract managers, one majorly used software is Truffle. Truffle is a console-based manager that helps in compiling, migrating and operating the smart contracts. The transactions are verified and sealed by miners which work on solving mathematical problems for ethers, a cryptocurrency for Ethereum, as a reward.

One main part of Ethereum is the Ethereum gas, all operations inside Ethereum networks require some amount of gas to execute. The gas price is the cost of using the system, e.g. the cost of executing the smart contract, and it reflects the amount of computational power required from an initiator to have his/her transaction executed. The gas price is set based on the opcodes used for each transaction; it also applies to the number of bytes used in the transaction. The owner can set the gas limit for each smart contract, which helps in restricting the smart contract from running in a continuous loop.

III. Related Work

The use of blockchain technology in IoT networks is gaining more interest among researchers because of its highly distributed manner which can be utilized to rectify IoT related issues such as access control management, client authentication and authorization. The problem of managing access control of a fast growing IoT networks is discussed in [5]. The work proposes the use of specific nodes named management hub nodes which interact through the blockchain to provide access control for clients. The experimental results show that the system scales well as numerous IoT devices can be connected simultaneously to the blockchain network.

Similarly, the work in [6] has shown promising results in using the blockchain technology for managing access control of IoT clients. The authors in [6] proposes the use of multiple smart contracts, namely the judge contract (JC) and the register contract (RC). The role of the smart contracts is to provide a subject-object pair with a static access based on the security policies and a dynamic access based on the behavior history of the subject. They demonstrated through a case study using Ethereum platform the feasibility of their framework in providing a decentralized access control for IoT networks.

A variation of the aforementioned proposals is illustrated in [7] which uses fog nodes to interface with the Ethereum blockchain network and creates smart contracts. The proposed architecture reduces the computational overhead of the blockchain technology in IoT devices by the use of fog nodes.

The authors in [8] propose the use of blockchain in smart home-based IoT application to ensure data privacy. The proposed approach allows users to enforce the privacy policies in an access control contract implemented using Ethereum blockchain network. In addition, a judgement contract is proposed to judge a misbehavior action and determine the corresponding penalty. The experimental results show advantages of using the blockchain to provide data privacy and access control in IoT networks.

In [9], a defend mechanism, namely an out-of-band two-factor authentication scheme, is proposed in response to an attacker who succeeds in stealing the access token. The secondary authentication is proposed to be carried over light or sound signals. The work focuses on investigating the overheads of using the blockchain and smart services on emulated devices.

The focus of most of the research done on using blockchain technology in IoT networks study the performance of the proposed system in terms of the computational overhead added by the use of blockchain [9], system resource utilization such as throughput and delay [5] and/or management concerns such as

adding/removing clients [6][7][8]. However, the security aspect of the proposed works has gained lower attention. Although some work discussed possible security threats and mechanisms [7], they did not investigate a threat model or implement it in real settings. On the other hand, our work aims to analyze the security of an access control authorization system based on Ethereum blockchain by investigating a threat model and verifying the system's behavior under attack in real implementation.

IV. Proposed System Model

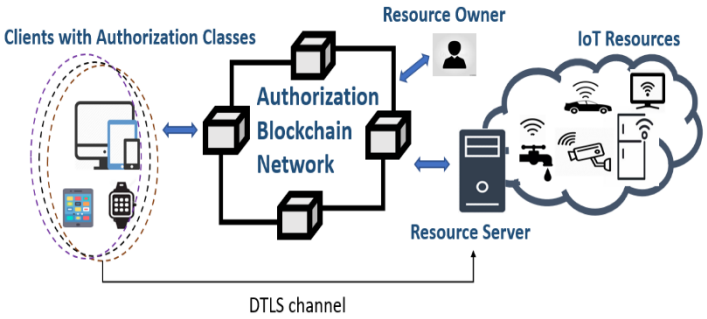

Fig. 1: The Blockchain Access Control Authorization Architecture for IoT Resources

A. System Architecture

The system follows the terminology specified by the ACE protocol. As previously discussed, four main entities, the RS, RO, AS and clients interact to enable a secure authentication and authorization of clients requesting access to the IoT resources. The AS in our system is replaced by an authentication and authorization blockchain network. The RS acts as an intermediate gateway between the protected IoT resource (e.g. smart sensors, IP cameras, etc.) and the external network. The RO is the legal owner of the IoT system who decides the security polices of the system. Finally, the clients are third party users requesting access to the IoT system. Clients have different classes that specify their access and control privileges.

Fig. 1 shows the network architecture consisting of the main interacting entities. The authentication and authorization interactions between the RO, RS and the clients are done over the blockchain network. In addition, a DTLS channel is required between the legitimate clients and the RS for secure transmission and/or control of IoT resources.

B. Access Control Authorization Blockchain for IoT Systems

The Ethereum blockchain network has a very attractive security related characteristics that makes it a feasible solution for access control management in IoT utilities. Besides its decentralized nature, the Ethereum blockchain supports a unique address assignment using a collision resistant hash function. Each node in the network can create its own Ethereum account which has a unique Ethereum address and a pair of private/public keys. Thus, allowing for a cryptographic key distribution without the need for a key distribution algorithm. The interactions between the different nodes in a blockchain network is done through interfacing with a smart contract. A node can initiate a transaction to a smart contract's address that will be broadcasted to the blockchain network. A transaction can call a specific function in a smart contract and can include one or more function input arguments. The miners of a blockchain network has a complete blockchain ledger and are responsible for verifying network transactions. However, an Ethereum node can either be a miner e.g. a full node or it can interact over a blockchain network without performing network mining. This flexibility allows devices with low computational power such as IoT devices to take part in a blockchain network.

In the following we discuss the authentication and access control problem in the context of a smart health care system. A smart health care utility deploys IoT devices for health monitoring, data collection and remote control of the infrastructure. The smart IoT infrastructure allows the introduction of new smart services and applications. For instance, the nursing department could be able to remotely monitor the health conditions of their patients on real time. A smart service would be to allow a relative of a patient to have an online visualization access of the health data of the patient. Moreover, a physician could be able to remotely control an IoT device to take an action in a timely fashion, if an emergency situation is observed. The utility health provider should be able to define specific access and control privileges and enforce them in the IoT system. For instance, smart health utility provider should be able to restrict nurses and patients' relatives from controlling the IoT system. In addition, system administrator should be able to be securely authenticated and granted access to control the IoT infrastructure. Thus, a key enabler for IoT systems is the secure authentication of third parties (e.g. doctors, relatives, etc.) and the access control management of the IoT utility.

The proposed access control authorization system follows the ACE protocol in authenticating clients and issuing an access token that defines the clients' access privileges. Ethereum offers an open source platform in which an IoT utility can use to implement a private blockchain network that enforces security policies. Typically, an IoT utility consists of edge servers which connects to a number of IoT devices. Edge servers are the RSs in our security system; they are the miners of the private Ethereum blockchain network. Each edge server and user/client of the IoT utility will have a unique Ethereum account which enables it to interact over the blockchain network. Clients are authenticated by mutual username/password agreement that takes place in an out of bound communication. The Ethereum account address of a client presents its username whereas the password is a secret string that a client should have to be able to access the IoT resources. The RO, e.g. the smart IoT utility owner, governs the security policies of the system. The RO classifies legitimate clients into different classes in which each class has specific access privileges. The Ath_{sf} will be used in the

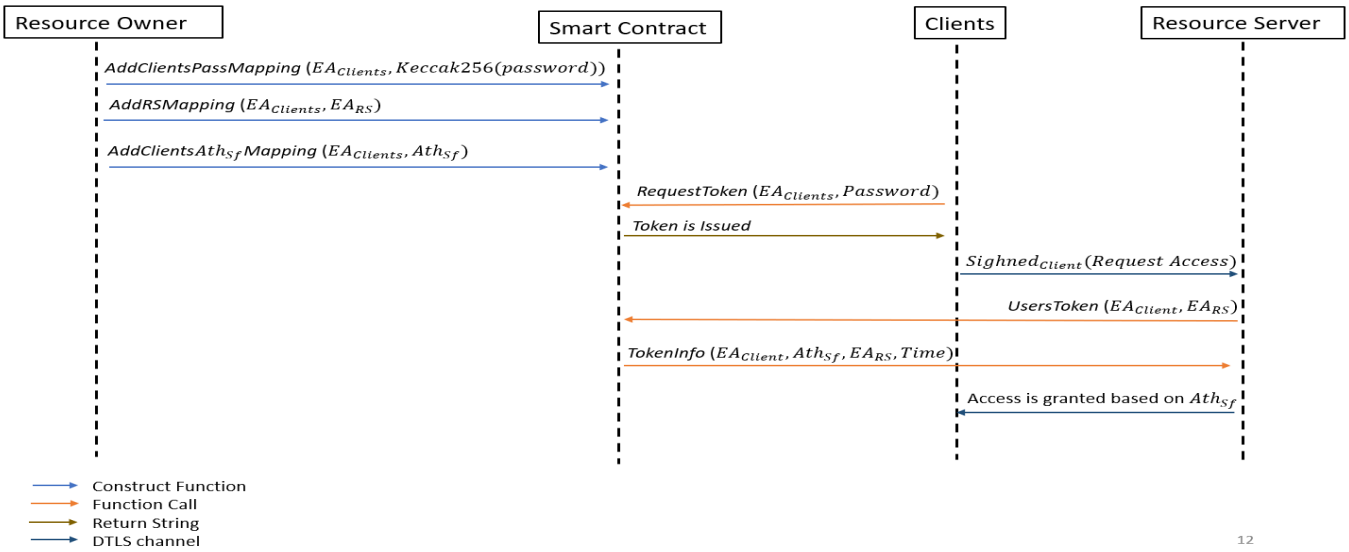

Fig. 3: Access Control Authorization Mechanism

blockchain network to distinguish the access privilege class of a client requesting access to the system, as described in Fig. 2. A client can be of Class 1 which has the most access privileges of the IoT system. A class 1 client is authorized to control the functionalities of the IoT system, e.g. switch ON/OFF an IoT devices. Class 2 clients are allowed to have a real-time IoT data visualization. In the context of a smart health utility, nurses' departments can be assigned to Class 2 in which they cannot control the IoT system. Finally, Class 3 clients are authorized to have a restricted IoT data visualization, for example a relative of a patient is allowed to view the health information of his/her related patient only. The Ath_{Sf} information of a client will be included in the access token and used by the RS to enforce access control policies.

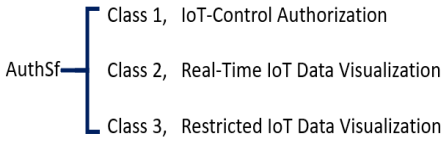

Fig. 2: Ath_{Sf} definition of Access Control Classes

C. Smart Contract Description

The RO deploys the security policies in a smart contract and publishes it in the network. The smart contract functionalities can be divided into three main tasks. The first task is establishing a distributed ledger that contains the mapping of legitimate client Ethereum accounts (usernames) to their respective hashed passwords, and Ath_{Sf}. In addition, the RO specifies the Ethereum addresses of the resource servers which each client can access. This is needed to provide an additional access control restriction. For instance, in a smart health provider utility a nurse department that have class 2 access privileges should be able to visualize IoT data of a specific medical section, e.g. not be able to visualize all the IoT data of the health utility. This can be performed by specifying for each client the Ethereum address of the RS which is responsible for the IoT devices that a client is authorized to access. The second task of the smart contract is to provide means of function calls which enables clients to request access tokens and enables RSs to verify the access token information of a specific client.

Finally, the smart contract includes the function *ADDToken* which is responsible for generating token access to authenticated clients. The *ADDToken* function is modified by the use of a Modifier function to restrict function calls to specific client addresses. Those addresses are preserved in a whitelist record which is used by the Modifier function to allow/prevent function calls. The Modifier function is essential to defend against DDoS attacks as will be discussed in section VII.

D. Authentication and Authorization flow

Fig. 3 illustrates the access control authorization procedure, which can be described as follows:

In the first stage, the RO publishes a smart contract that includes the security policies needed for the access control authorization. The smart contract maps the legitimate clients' Ethereum addresses ($EA_{Clients}$) to their hashed passwords, to the ethereum addresses of the RSs they are allowed to access, and to a specific Ath_{Sf} value.

Secondly, a client who wants to access the IoT resources needs to request an access token. This is done by activating the smart contract through sending a transaction to the smart contract's address that includes the client's password. The transaction is published to the blockchain network. The RSs will validate the transaction and execute the smart contract. An access token is generated only if the client includes the correct password. Contract transaction is added to the block chain and the token is added to the contract internal storage.

In the third stage, the client sends an off-chain access request to the resource server. The access request includes the token information and is signed using the client's private key.

When a RS receives the access request message, it calls the *UsersToken* function in the smart contract and includes the

EA_{Client} and the EA_{RS}. The smart contract sends the user's token information that specifies the EA_{Client} , the Ath_{Sf} and the EA_{RS}. Upon recieving the access token information, the RS will verify the client's signiture, and use the Ath_{Sf} information to enforce access control policies.

V. Threat Model

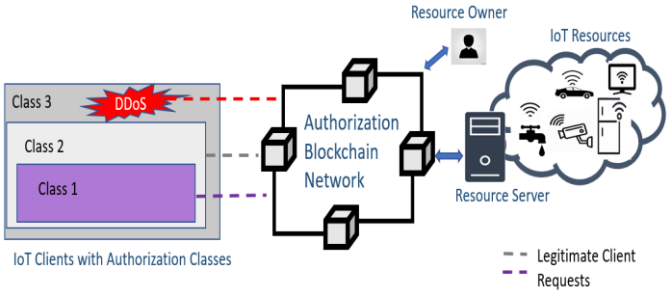

Fig. 4: Threat Model of the Access Control Authorization Blockchain

The threat model illustrated by Fig. 4 studies the system vulnerability against a DDoS attack that targets the application layer of the authentication and authorization system. We assume that an insider is able to compromise a number of class 3 Ethereum accounts creating a bot network that continuously triggers the smart contract requesting an access token. While the same attack can occur by compromising any class, class 1 and class 2 accounts should have strict security policies that makes them more secure, eg. can be only accessed from the utility's local area network. We assume that the attacker is not able to have the passwords of the compromised accounts. As such, the stolen accounts cannot access the IoT resources. However, the attacker will be able to launch a DDoS attack which will result in performance degradation of the access control authorization system.

The defend mechanism to the aforementioned attack is by introducing a modifier function in the smart contract which restricts the function calls to authorized client addresses. The smart contract will keep a record of whitelisted Ethereum account addresses in its internal storage. A client of class 3 whose address is not whitelisted will not be able to call the smart contract thus defending against the DDoS attack.

VI. Implementation

To simulate a blockchain network we have implemented a private Ethereum blockchain network (version: Geth 1.9.5-stable-a1c09b93). Specifically, three machines are interacting over a private Ethereum network as shown in Fig. 5. The first machine is a 64-bit Ubuntu 18.04.3 LTS virtual machine running on the Oracle VM VirtualBox Version 6.0.8 r130520 (Qt5.6.2) equipped with 4GB of RAM and 50GB of HDD. The second machine is a Windows 10 running Geth 1.9.5. Finally, a light version of Geth runs on the raspberryPi 3.

In this setup, the Ubuntu machine is used to implement a bot network that will launch a DDoS attack. The RO and the clients access the blockchain network from the windows machine. The RS is the raspberryPi which connects to other IoT devices, e.g. sensors. Accounts are created in all three nodes and the miners are activated to do mining. The smart contract is written using solidity 0.5.0 and Truffle v5.1.1 is used for compiling, migrating and accessing the smart contract functions.

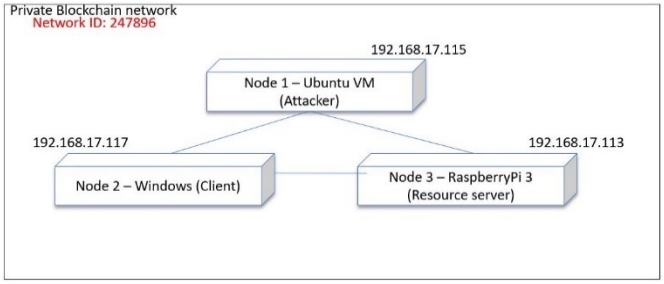

Fig. 5: Illustration of experimental setup

VII. Results and Discussion

A. System Performance Under DDoS Attack

In this section we study the system performance under a DDoS attack as described in the threat model. A total of 6 Ethereum accounts are created to represent the compromised accounts and create a bot network. The bot accounts will continuously send transactions to the smart contract requesting access tokens. Since the attacker does not have the passwords of the compromised accounts, the function calls will include false passwords resulting in unsuccessful authentication. While the bot accounts are launching the DDoS attack, a legitimate account requests an access token. The response time of the legitimate client's request is recorded to study the performance of the system. Moreover, we compare the performance of the system with and without the defend mechanism which is implementing a Modifier function in the smart contract. We ran three identical trials of each point in the below results and report their average.

Fig. 6 shows the average response time of the system for the two scenarios under study, a system that implements a Modifier function and one that do not restrict function calls. From the figure, it can be noticed that when one bot account launches a DoS attack, an average of 5 seconds increase in the response time of a legitimate request occur in the system that do not use the Modifier function compared to the system with the Modifier function. The average response time of the system that does not restrict function calls increases exponentially. This is expected because as the number of bot accounts increases, more DDoS traffic will be injected into the network which keeps the miners busy. Thus, increasing the response time of legitimate requests; denying legitimate clients from getting the access control authorization service. When six bot accounts continuously trigger the smart contract, an average of 17 seconds increase in the system response is found in the system that does not utilize the Modifier function compared to the system that restricts function calls.

The system that uses the Modifier function is immune against the DDoS attack. This is seen from the almost constant behavior of its average response time. While a maximum of six bot accounts are launching a DDoS attack on a smart contract with function call restrictions, the average response time is stable, e.g. around 5 seconds, and seems to be not affected by the DDoS attack. The results confirm that the modifier function

reduces the response time for wrong or illegitimate function calls and keeps the pipeline free for legitimate requests.

We expect that the response time of the system that uses a Modifier function should be the same regardless of the number of bot accounts. The slight increase and decrease in the response time are due to low number of identical trails for each point (3 trials). Regenerating the results with larger number of runs and reporting the average value, will eliminate the randomness in the network and the human errors that might be presented in the below results. We were not able to accomplish this additional task due to the time limitation of the course. However, it will be considered for future work.

Another main finding is that the gas price per transaction of the DDoS attack increased significantly when using the mitigation method. The gas price of illegitimate calls, calls from Ethereum addresses that are not whitelisted, is the maximum gas limit that is set for the smart contract. This observation shows that an attacker needs more computational power to launch a DDoS attack on a system that uses the modifier function which thereby reduces the feasibility of the attack.

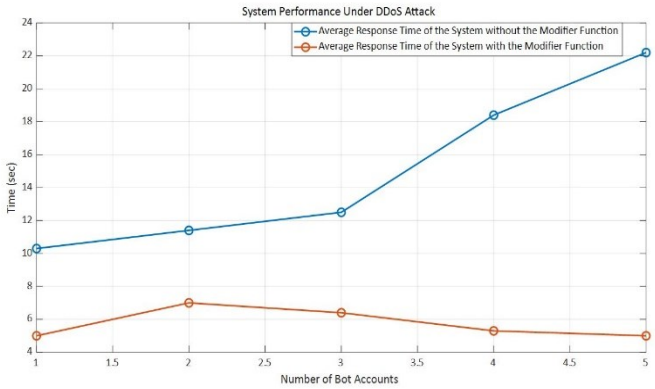

Fig. 6: Comparison of average response time

B. *Modifier Function Performance Trade-off*

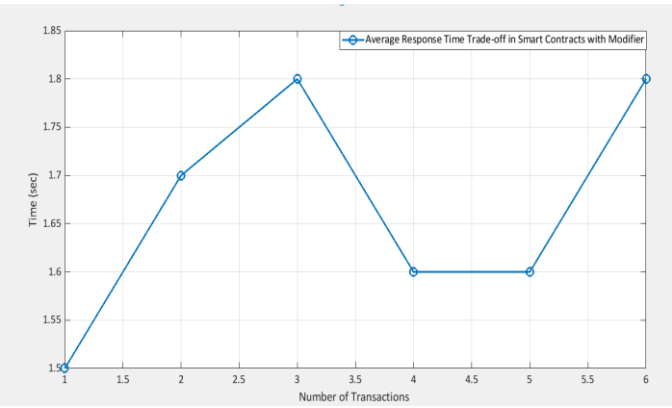

Fig. 7: Modifier function performance analysis

In this section we study the trade-off in the system's performance when the Modifier function is deployed in the smart contract. The average response time of legitimate requests is found for a system that uses the Modifier function and a system that does not restrict function calls. Fig. 7 shows the difference in the response time between the two systems. For instance, a system that utilizes a Modifier function will consume an additional 1.5 seconds on average to respond to legitimate requests compared to the system without the Modifier function. We compared the performance of the systems while increasing the number of requests to confirm whether the cost of the response time will increase. However, the average response time results do not increase but rather stays slightly above or below 1.7 seconds. This shows that deploying a Modification function in the smart contract will cost an average of 1.7 seconds increase in the average response time of legitimate requests. We expect that the variations of the average response time are due to randomness and human errors as discussed in previous section.

VIII. Conclusion

In conclusion, this paper proposes an access control authorization system based on blockchain technology for the Internet of Things (IoT) systems. The system is implemented and tested in real setting using the Ethereum platform. We analyzed the performance of the system under DDoS attack and proposed a mitigation method using the Modifier function. The results show that the system that utilizes a Modifier function outperforms the system that do not use the Modifier function in terms of the average response time. In addition, we observed that the gas price of the DDoS transactions increased significantly when the Modifier function is used which makes it more difficult for an attacker to initiate a DDoS attack. The cost of deploying the Modifier function is an average of 1.7 seconds increase is the response time. We argue that the cost of the Modifier function is acceptable when considering the high security advantages of its deployment.

References

[1] O. Alphand *et al.*, "IoTChain: A blockchain security architecture for the Internet of Things," *IEEE Wirel. Commun. Netw. Conf. WCNC*, vol. 2018-April, pp. 1–6, 2018.

[2] "Authentication and Authorization for Constrained Environments (ace) - Documents." [Online]. Available: https://datatracker.ietf.org/wg/ace/documents/. [Accessed: 05-Dec-2019].

[3] "White Paper | eth.wiki." [Online]. Available: https://eth.wiki/en/white-Paper. [Accessed: 05-Dec-2019].

[4] "Geth · ethereum/go-ethereum Wiki · GitHub." [Online]. Available: https://github.com/ethereum/go-ethereum/wiki/geth. [Accessed: 05-Dec-2019].

[5] O. Novo, "Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT," in IEEE Internet of Things Journal, vol. 5, no. 2, pp. 1184-1195, April 2018.

[6] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang and J. Wan, "Smart Contract-Based Access Control for the Internet of Things," in IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1594-1605, April 2019.

[7] R. Almadhoun, M. Kadadha, M. Alhemeiri, M. Alshehhi and K. Salah, "A User Authentication Scheme of IoT Devices using Blockchain-Enabled Fog Nodes," 2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA), Aqaba, 2018, pp. 1-8.

[8] T. L. N. Dang and M. S. Nguyen, "An Approach to Data Privacy in Smart Home using Blockchain Technology," 2018 International Conference on Advanced Computing and Applications (ACOMP), Ho Chi Minh City, 2018, pp. 58-64.

[9] L. Wu, X. Du, W. Wang and B. Lin, "An Out-of-band Authentication Scheme for Internet of Things Using Blockchain Technology," 2018 International Conference on Computing, Networking and Communications (ICNC), Maui, HI, 2018, pp. 769-773.