T.C.

ALTINBAŞ UNIVERSITY

Electrical and Computer Engineering

BLOCKCHAIN AS A SOLUTION TO ACCESS CONTROL IN IOT SYSTEMS

Montdher Abed Almahdi Abedali Alabadi

Master Thesis

Supervisor

Oguz Ata

Istanbul (2019)

BLOCKCHAIN AS A SOLUTION TO ACCESS CONTROL IN IOT NETWORKS

by

Montdher Abed Almahdi Abedali Alabadi

Electrical and Computer Engineering

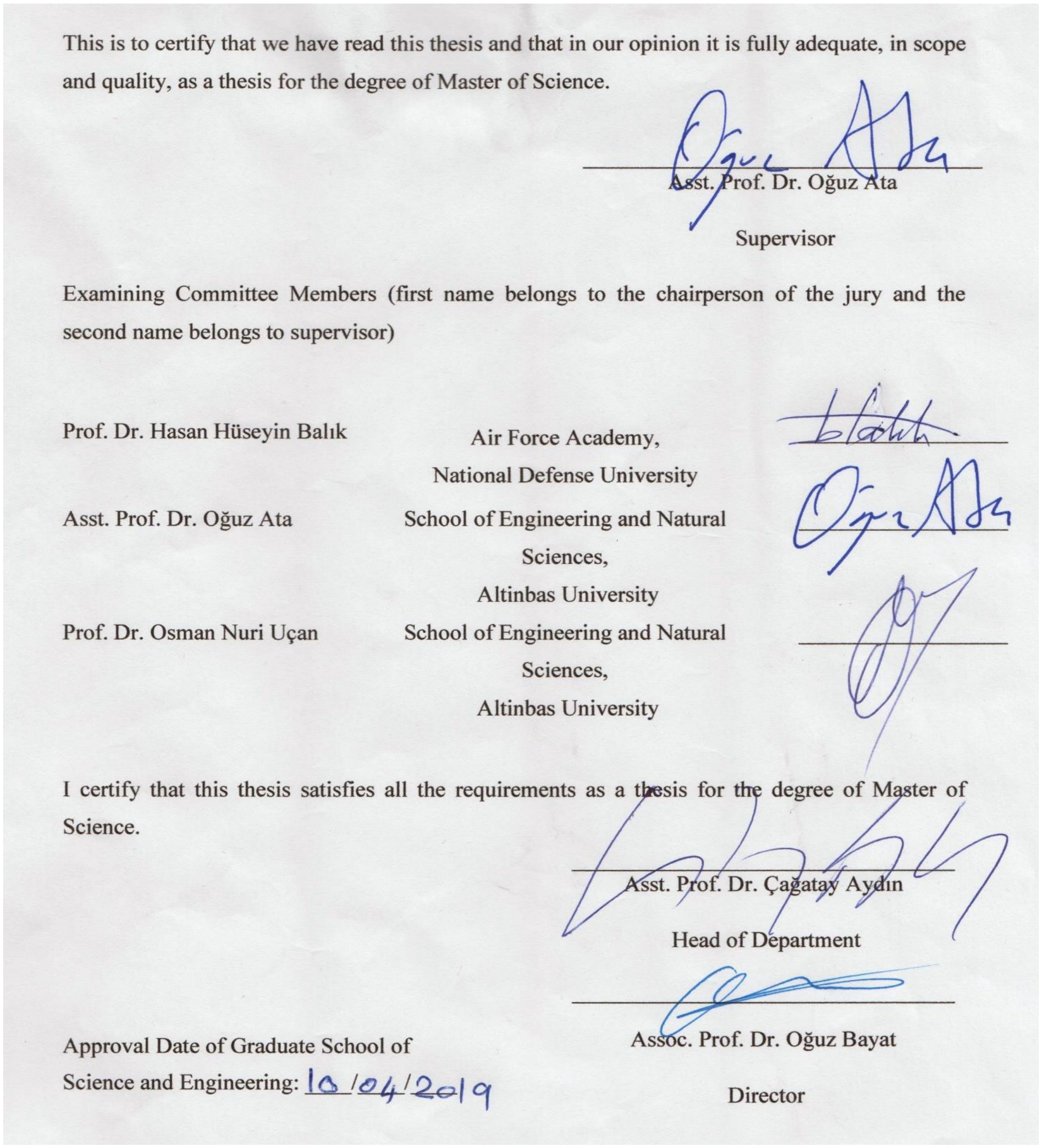
Submitted to the Graduate School of Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2019



This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

[illegible]

Asst. Prof. Dr. Oğuz Ata

Supervisor

Examining Committee Members (first name belongs to the chairperson of the jury and the second name belongs to supervisor)

Prof. Dr. Hasan Hüseyin Balık	Air Force Academy, National Defense University	[illegible]
Asst. Prof. Dr. Oğuz Ata	School of Engineering and Natural Sciences, Altinbas University	[illegible]
Prof. Dr. Osman Nuri Uçan	School of Engineering and Natural Sciences, Altinbas University	[illegible]

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

[illegible]

Asst. Prof. Dr. Çağatay Aydın

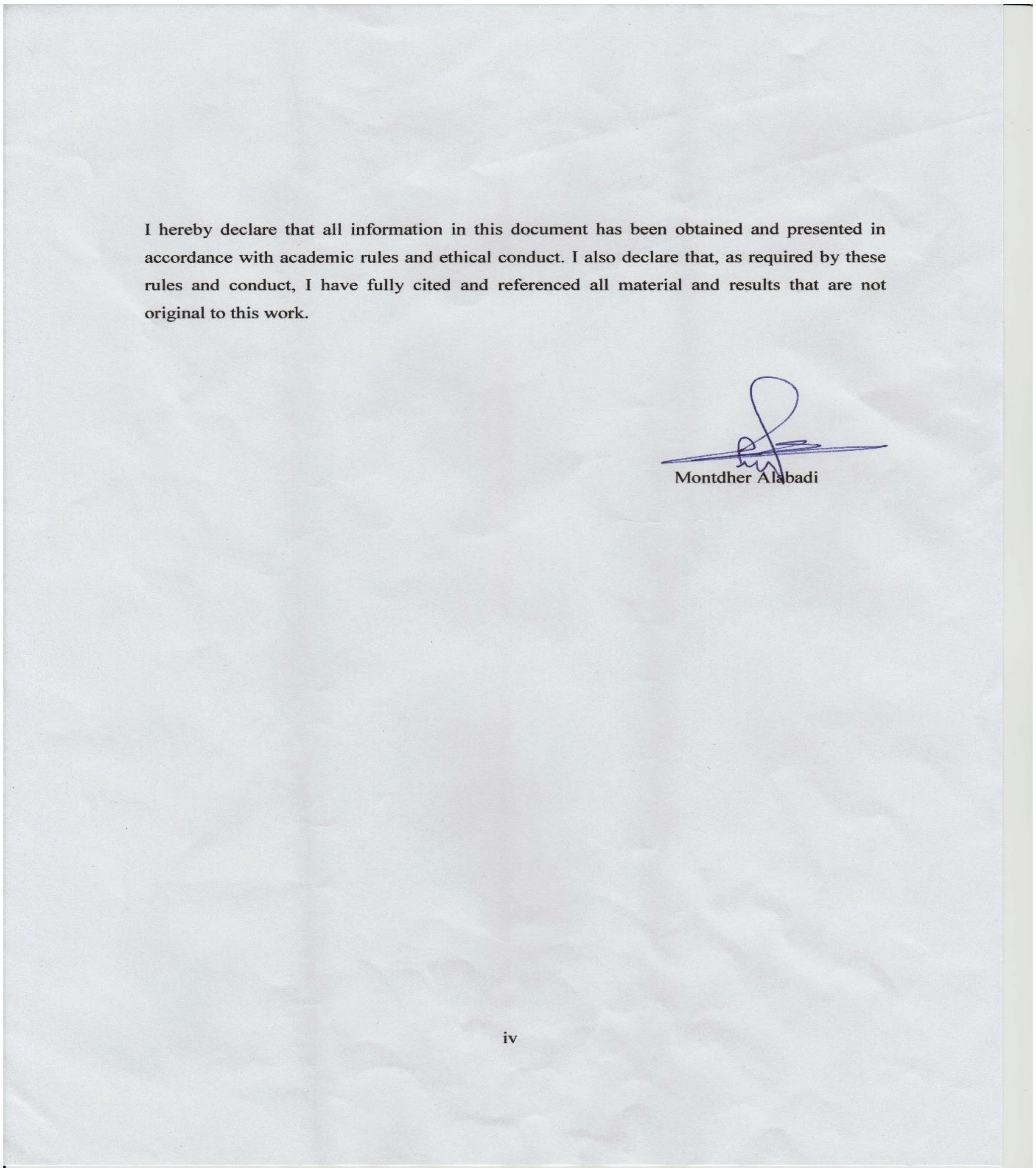
Head of Department

[illegible]

Assoc. Prof. Dr. Oğuz Bayat

Director

Approval Date of Graduate School of Science and Engineering: 10/04/2019



I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

[illegible]

Montdher Alabadi

iv

DEDICATION

To my parent ,my wife, my brothers and all my dear friends without whom this achievement could not be completed.

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Asst.Prof . Dr Oğuz Ata for his suggestion to work in the area of Blockchain which was very interesting in additional to his intention during all stages of this thesis.

ABSTRACT

BLOCKCHAIN AS A SOLUTION TO ACCESS CONTROL IN IOT NETWORKS

Alabadi, Montdher Abed Almahdi Abedali

M.S, Electrical and Computer Engineering, Altınbaş University,

Supervisor: Dr. Oguz Ata

Date: 04/2019

Pages: 71

One of the most interesting technologies in our current and future era is Internet of things (IOT) systems. It has a great impact on our lives. Day by day, the dependence on these systems is becoming larger, and this dependence may be related directly to human lives, such as health care system. However, every technology has its security and privacy concerns, and these concerns become more dangerous when it’s related to human lives. One of these concerns is the access management that controls who can access the system and how. This concern becomes more dangerous when taking the limitations of the IOT device into consideration. In this thesis, we propose a new system architecture for Access Control in IOT systems using a blockchain system which is domain-based. This system is lightweight by using cache and a block list concept that is fully controlled by leveraging both the centralized and decentralized management approaches. Our suggested system is supported by a multi-scenario implementation backed with evaluation and analysis that show the power of the blockchain technology to manage access control in IOT systems.

Keywords: Blockchain, Sensor Network Security, Access Control

TABLE OF CONTENTS

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

LIST OF ABBREVIATIONS

IOT	: Internet of things
P2P	: Peer-To-Peer
HTTP	: The Hypertext Transfer Protocol
JSON	: JavaScript Object Notation
MQTT	: Message Queuing Telemetry Transport
SSI	: Simple Sensor Interface
Rest	: REpresentational State Transfer
AMQP	: Advanced Message Queuing Protocol
CoAP	: Constrained Application Protocol
XMPP	: Extensible Messaging and Presence Protocol
LLAP	: Lightweight Local Automation Protocol
IPV6	: Internet Protocol Version 6
6LoWPAN	: Low-Power Wireless Personal Area Network
MTU	: Maximum transmission unit
RPL	: The Routing Protocol for Low-Power and Lossy Networks
DTLS	: Datagram Transport Layer
NanoIP	: Nano Internet Protocol
BLE	: Bluetooth Low-Energy
LTE	: Long term evolution protocol
LoRaWAN	: Low Power, Wide Area (LPWA) networking protocol
NB-IoT	: Narrow-Band IoT
ANT	: Ultra-Low-Power (ULP) Wireless Networking Protocol
LwM2M	: Lightweight Machine to Machine (LWM2M) protocol
DDOS	: Distributed Denial of Service
POs	: Proof-Of-Stack Algorithm

PBFT	: The Practical Byzantine Fault Tolerance Algorithm
DPoS	: Delegated Proof-Of-Stack Algorithm
EOA	: Externally Owned Account
EVM	: Ethereum Virtual Machine
Wei	: Minimum Unit That Used on The Ethereum
UTC	: Unix time- temp
API	: Application Programming Interface
ABI	: The Contract Application Binary Interface

1. INTRODUCTION

Today, there is an unprecedented revolution in digital communication which has created a huge impact on our society. This has shifted our world into becoming a digital universe. Internet of things (IOT) is one of most technologies that have participated in accomplishing the impact mentioned above. Simply, IOT devices are the devices which have remote sensing and/or actuating capabilities and the ability to exchange data with other connected devices and applications (directly or indirectly). Because of the limited processing capabilities of IoT devices, it usually leverages externally controlled third party service providers to do the additional data processing. This will create a huge number of problems and challenges, one of these concerns is related to access control. The doubt is how to offer a safe and trusted environment using devices that cause a big limitation in processing, storage, and ability. Most access control management in the current systems depends on “Centralized Access Control Systems”. This approach may run into the need of a specific trusted environment, but when IOT devices need to be fluid and demand management by various managers, it is unknown what will take place. Even so, doing away with the central management is not the right procedure because it we lose the power and advantages of central management [1]. The blockchain in the other hand is a platform has robust security features, A blockchain is fundamentally decentralized, distributed, shared, and an immutable database ledger that stores registry of assets and transactions across a peer-to-peer (P2P) network. It has chained blocks of data that have been timestamped and validated by miners. A domain-based system is our solution to handle most of the weaknesses in IOT access control challenges. The solution is presented as an architecture where IOT devices are located in separate virtual domains and the interaction between devices will be governed by rules. These connections and rules are implemented by utilizing the famous decentralized technology “blockchain”. The integrating between blockchain with IOT allow us to also provide a trustworthy environment that helps access management in IOT system, that’s because the blockchain guarantee that the data remains immutable.

1.1 CONTRIBUTIONS

Growth of the IoT must be backed by standard suitable protocols, and operations in order to reduce the existing limitation [2]. This may lead to huge side effects and concerns that could trim down the adoption of the IoT. The trustworthiness of IOT data is likewise an important topic to be wielded. Untrusted nodes or devices can alter information according to their own interests, so the information they provide might not be completely satisfied. This led us to search about what it needs to ensure that the information not altered in the stage of its motion. Recently, Blockchain has been issued a robust technique [2], that can be applied to treat the problem of privacy, trust, authorization and reliability, because of its ‘distributed ledger, that can provide trustworthy services to a group of nodes without central government agency. Even so, doing away with the central management is not the right procedure because it makes us lose power and advantages of central management [1]. The heavy use of cryptography, a key characteristic of blockchain networks, provide the security and privacy behind all the interactions in the network. Smart contracts that reside on the blockchain allow managing connection in the network with, distributed, heavily automated workflows. And this our motivation by integrating blockchain with the Internet of Things (IoT). The means to provide this trustworthiness in IoT network can be accomplished by using distributed management where the information and its movement is trusted and affirmed by all its participants that guarantees that the data remains immutable [3]. This means data verification can make the data secured, beside that trustworthiness can be accomplished. So, our main contribution in this work is as follows:

1. Using the blockchain technology to design a system architecture, allowing that system to leverage the advantages of both centralized and decentralized management approaches.The blockchain is used to provide decentralized management on executing peer (subject or Access requester) -to-peer (object or access receiver device) connection. On the other hand, the central server’s function is establishing the role and to give permission for access between system entities.

2. Using concepts of cache and block lists in smart contract design to reduce the processing cost “which is gas in Ethereum” [4]. This will make our system lightweight, which is very suitable for the IOT environment. This is demonstrated in the implementation part of this dissertation.

3. Use of two types of contract in our system. The first one is the local contract, which is specific to one domain, second one is the global contract. However, most access requests are handled by executing local contracts only, other requests will require at most executing global contracts in addition to the local contract. This approach will efficiently reduce overall operating cost and complexity in the arrangement.

4. Use of domains in our domain-based architecture makes it suitable for application in organizations with multiple geographical locations because devices with high trust level can be together in the same domain and access control will apply by using local contract only.

1.2 LITERATURE REVIEW

There are many previous works about integrating IOT with blockchain in terms of access control. In reference [5], the authors suggest a framework that consists of three types of smart contracts that communicate among each other to provide access management. However, this approach can lead to some unnecessary complexity because one contract for each pair of devices (subject-object) as a structure requires high computation capabilities as deploying many contracts and the communication between many contracts could consume a lot of gas. In reference [6], access management is achieved by using a single smart contract, and some managing nodes are responsible for managing several IOT devices. However, this approach eliminates the advantage of centralized management. Apart from that, the author does not provide an evaluation of the processes inside the blockchain. In reference [7], the author suggests using a smart contract into the voting system. As clear from the results, most of the system functions consume a huge amount of gas, which can be considered as an unreliable aspect. In reference [8], the authors provide such a good access control “Fair Access,” which uses a special type of transaction to demonstrate the system function, but the work depends on Bitcoin, which is more expensive and less reliable.

1.3 METHODOLOGY

Our methodology to achieve thesis goals or contributions will require working in five stages , first finding the problem, suggestion or solution, development or implementation ,result evaluation and finally discussion .

Outline the problem can be keyed out from many sources such as from a reference discipline, our thesis problem is about access control in IOT system. The solution to solve the thesis problem is by integrating blockchain with IOT, because it's the way that we can provide trustworthiness in IOT data through a distributed service trusted by all its participants which guarantee that the data remains immutable. Development will handle How to use of blockchain features and collaborate it with current IOT that has limitation in terms of memory, processing and power system. In our thesis, we will utilize Ethereum blockchain because its ability to implement smart contracts which is basically a computerized transaction protocol that executes the functions of the contract [9]. This solution will be by implementing as an architecture that provide a high level of reliability, privacy, authentication, and management depending on decentralized mechanism provided by blockchain platform, this architecture will manage and secure communication between system entities by enforcing and adopting an access management algorithm that implement specific policies and access functions. The proposed architecture will also be examined and valued in the actual environment with actual devices and network entities, this evaluation will be given by utilizing different scenarios which simulate the real scenario, all results collected from run time will compare to an expected result. More specifically A domain-based system is our solution to handle most of the weaknesses addressed previously. The solution is presented as an architecture where IOT devices are in separate virtual domains and the interaction between devices will be governed by rules. These connections and rules are implemented by utilizing the famous decentralized technology "blockchain. Access management rules will provide secure, authenticate fully controlled, peer (device) - to- peer (device) connection. When we talk about control and management we talk about key feature of decentralized systems, because of “the control and management will be done using IOT device itself so no server neither "third party" will involve in this process. Creation of the above architecture will be involve using some blockchain ecosystem, these ecosystems can allow us to initiate blockchain environment and provide the interfaces need to communicate with IOT devices. The thesis structure summary is explained in Table 1.1**.**

Table 1.1: Thesis Summary

Chapter	**Name**	**Aims**
1	Introduction	Provide summary about thesis objective and methodology to handle these objectives.
2	Internet of Things	Provide theoretical background about IOT technology.
3	Blockchain Technology	Provide theoretical background about Blockchain platform Explain briefly suggested.
4	Domain-Based System	Define and Explain suggested System Architecture and included operations.
5	Implementation	Experiment explanation to implement suggested system operation under different scenarios
6	Evaluation	Collect the implementation results and discuss it in term of security and performance and finalize the thesis with a conclusion and future work.

2. INTERNET OF THINGS

2.1 OVERVIEW

‘Internet of Things (IoT)" is a modern communication platform that envisions a near future, in which the things that we used it daily will be equipped with microcontrollers, transceivers for digital communication, and suitable protocol stacks that will make them able to be interconnected with each other and with the users [10]. It can be seen as a distributed network of physical objects that can act on their environment and can communicate with other machines or computers. According to the European Commission [11], ‘Internet of Things (IoT) represents the shifting towards the digitization of our world, where things and people are interconnected through communication networks and report about their status and/or the surrounding environment. Next digital world will contain a huge thing that provides information and services to the final users through standard communication protocols and unique addressing schemes [12].The IoT visualizes a totally connected world, where things are able to communicate measured data and interact with each other.This makes possible a digital representation of the real world,through which many smart applications in a variety of industries can be developed. These include: Smart homes, Weareables, Smartcities, Healthcare, Automotive, Environment, Smart water, Smartgrid, etc. IoT solutions are being deployed in many areas, optimizing production and digitizing industries. IoT applications have very specific characteristics, they generate large volumes of data and require connectivity and power for long periods

The Internet of Things connects the virtual world with the physical world. In the virtual world, virtualization technique, software defined networks, cloud resource and big data schemes are developing fast and need to be addressed as enabling technologies for the Internet of Things. In the physical world, the new wireless technologies for personal, home area networks, metropolitan and regional area networks, all promise to deliver better economies of scale in terms of cost, energy and number of connections. Make “Internet of Things” as enabler to our life requires a specified systems approach, intelligent processing and sensing development, good connectivity, software and services, along with an assistant system to interface with the smart of environment applications [13].

2.2 IOT ARCHITECTURE

Different requirement of different IOT networks makes building a general architecture for it a hard work, because of its complexity and novelty [14]. This hardness has led to a huge amount of structures for the practical implementation of IoT systems. Moreover, Structuring of the IoT Architecture is affected by the weakness in widely accepted Architecture models that can attract investments to promote the deployment of these technologies. The big step towards creating a unified architecture to the internet of things system is done In March 2015, the European Commission creates the Alliance for Internet of Things Innovation (AIOTI) [15]. This alliance allows the intention of the European Commission to work closely with all different users in different level of the Internet of Things.The devices in IoT are uniquely identifiable and are mostly characterized by low power, small memory and limited processing capability. The gateways are deployed to connect IoT devices to the outside world for remote provision of data and services to IoT users.

From all the above, we can define IoT is a complex structure of hardware, sensors, applications and devices that need to be able to communicate with each other in different ways. This requires shared standards to exchange data across different organizations.

Modern IOT architecture is mainly consists of 4 parts that work together to create "Internet of Things "environment" as clear in the Figure 2.1 and these parts are:

1. Physical Devices
2. Gateways
3. Pre-processing Services
4. Storage Systems

2.2.1 Physical Devices

The devices which have remote sensing and/or actuating capabilities and the ability to exchange data with other connected devices and applications (directly or indirectly) [10]. Sensors are the devices that have the ability to transform the information obtained in the outer world into useful data that can understand by other system component While actuators, they are the devices are able to change the status of the physical reality [16]. For example, turn on light or pump.

2.2.2 Gateways

Gateways act as connector between physical devices and the storage part of the system. Devices will connect to a gateway and the gateway then will move all that information to storage part of the system. The gateways makes communicating paths much shorter in distance which could positively affect charging or power source life.

Above connection is established by using varying protocols, which translate that data into a standard protocol that can be sent to the storage part of the system [16]. Gateways can help with pre-process services minimize overall system requirements. Gateways sometimes can perform some computation on the gateway itself instead of storage part of the system which consequently affect system latency Gateways can implement a number of physical devices connected to the internet which make them the first line of defense. For that the security must be a high priority for any gateway.

2.2.3 Pre-processing Services

They are services that used to filter, pre-process, aggregate or score IoT data. It uses the power and flexibility that can provided by the storage part of the system to run complex analytics on those data beside support decisions and actions [16]. Beside that these services could apply additional processing before moving the data to storage part of the system [17]. These services could install near to the physical device or in other places depending on the prior plan.

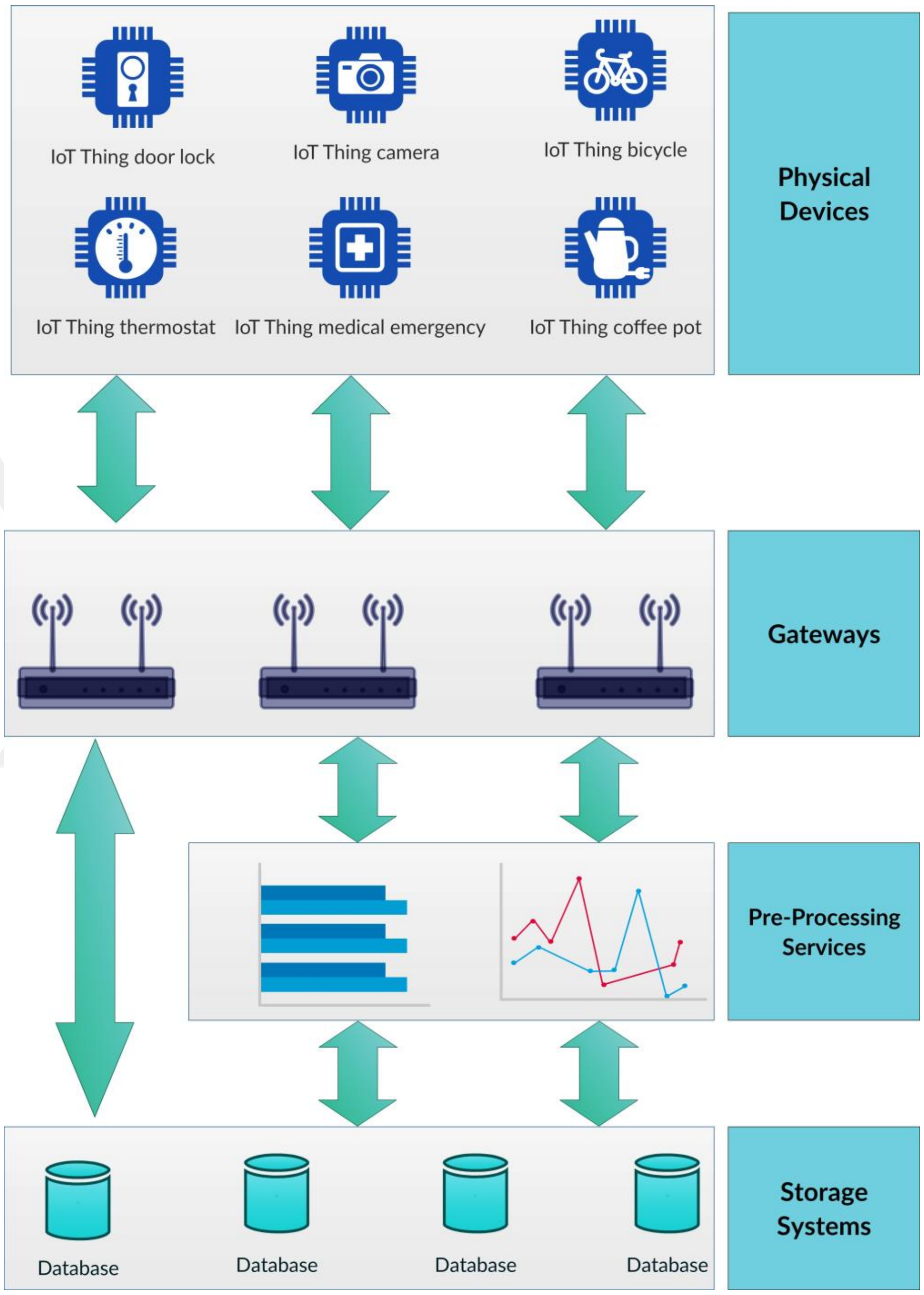


Figure 2.1: IOT Architecture Diagram

2.2.4 Storage Systems

IoT data usually coming from physical devices. Most of the IoT data (structured or unstructured) will prepare to perform analytics functions to generate insights [16]. This process requires high capacity storage capabilities, high-speed storage, and memory processing technologies. IoT data are generally small, but due to this fact, it can quickly add petabytes of data. An object storage is recommended for the storage of IoT as it can accommodate an increasing number of data files into storage devices. Cloud service providers are suitable as a storage due to this feature.

2.3 IOT CONNECTION TYPES

According to previous section we can define connection types that occur inside IOT environment depend on the source and destination of data as shown in Figure 2.2 and These types are:

1. Device-to-Device: here both of source and destination are IOT device, this connection occurs directly between devices without involving third parties. This connection is limited in processing and usually unsecured because of nature of these devices [18].

2. Device- Gateway**:** this connection is the most frequently occurred in IOT network, this connection type is the way for move IOT data between networks using the internet. Nature of gateways and its suitable capabilities allow establishing required security mechanisms in this connection type [18].

3. Gateway - Other Systems**:** third types of IOT connections is occurring between gateways and pre-processing systems or directly between storage system and gateways. This connection requires more attention and processing capabilities because it could include using specific protocols, handling some security issues or mange synchronization between source and destination [18].

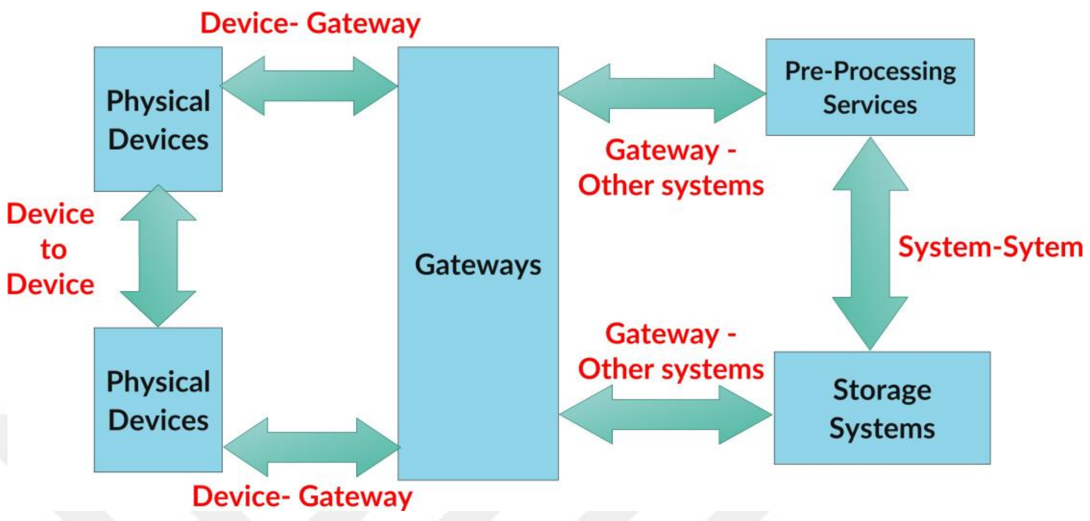


Figure 2.2: IOT Connection Types

4. System-System**:** this connection occurs where the source is storage system and destination are pre-processing system or vice versa. This connection required high attention in term of data recovery and availability because it could deal with different application and services inside each system [18].

2.4 IOT PROTOCOLS

The limited capabilities of the IOT in term of processing, memory and power made use of existing protocols inappropriate, and this reason why there is modified protocols and new protocols to work within an IOT environment [2]. IOT depend on in a layered architecture like what is existed in computer networks, which allow creating protocols to work in the specified layer. A protocol in layered approach is shown in Figure 2.3. This layer structure gives IOT the reliability in the management and control of moving data between system component. So, we will explain each protocol in term of the layer that it works within it.

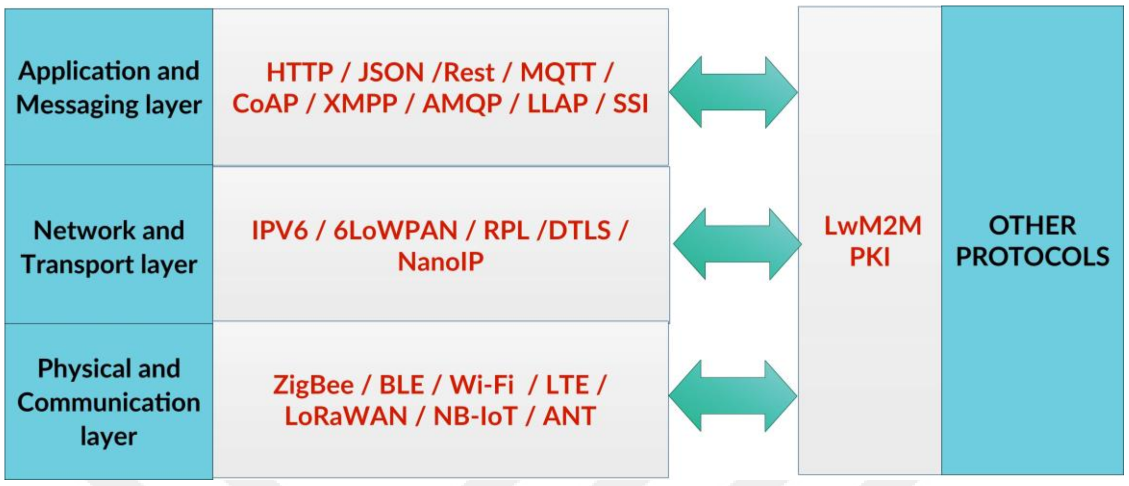

Application and Messaging layer
HTTP / JSON /Rest / MQTT / CoAP / XMPP / AMQP / LLAP / SSI
Network and Transport layer
IPV6 / 6LoWPAN / RPL /DTLS / NanoIP
Physical and Communication layer
ZigBee / BLE / Wi-Fi / LTE / LoRaWAN / NB-IoT / ANT
LwM2M PKI
OTHER PROTOCOLS

Figure 2.3: IOT Protocols

2.4.1 Application and Messaging layer Protocols

1. HTTP: protocol that is common to use for distributed information systems. HTTP is structured as a text that depend on logical links among nodes containing text [13]. HTTP is the way to exchange logical links (hypertext). HTTP can consider as a request-response protocol in the central management, computing model (client -server model). The client starts the work by sending an HTTP request message to the server. The server, where the files and data are stored, returns a response message to the client contains information that requested by the client.

2. JSON: it is the way that IOT data can be exchanged between a variety of computer languages. It is not just a protocol, but also can consider as an encoding format. JSON enables structured data to be implemented as a text format that can be sent to the destination [19]. JSON is usually used as complementary to work with other IoT protocols that do their process of data structure serialization such as HTTP/Rest, MQTT. For serializing and de-serializing structured data sent on their network, JSON encoder/decoder must be used.

3. SSI: (Simple Sensor Interface) this protocol makes is very easy to execute data transfer between computers and IOT devices [13].

4. Rest: (REpresentational State Transfer) is an architectural style for developing web services. REST is good for many applications because of its simplicity and because it can be built upon existing systems and HTTP in order to achieve its advantages. Beside all this REST is a reliable platform because of the fact the REST-based applications can be developed using any language [13]. Finally, REST can be considered as diffusion platform this because there are a variety of REST- based frameworks for helping developers in both server and client side.

5. AMQP: (Advanced Message Queuing Protocol) this protocol provides many features to the sent message such as orientation, queuing, routing, reliability and security [13, 18].

6. MQTT: (Message Queue Telemetry Transport) is a lightweight protocol for exchanging IOT data between IOT devices and other IOT network systems. The protocol includes three components: subscriber, publisher and broker. The publisher job as a pool of data that's will send a letter to subscribers [13, 18]. The broker checks publishers and subscribers, in term of security mainly.

7. CoAP:(Constrained Application Protocol)"CoAP is a protocol that is common to use in limited capabilities' internet devices. Cap is designed to easily translate to HTTP to be understandable in the web. CoAP has many features such as minimizing the complexity of mapping with HTTP, reduce header overhead, Support for the identifying of resources provided by known CoAP services [18, 20].

8. XMPP: (Extensible Messaging and Presence Protocol) a protocol designed for connecting devices to other system components .XMPP uses the XML as a text format. XMPP offers an easy way to identifying a device. XMPP cannot be considered as a fast processing protocol. Also, XMPP very optimal in addressing, security, and scalability which is very suitable consumer- oriented IoT applications [21].

9. LLAP**:** (lightweight local automation protocol) it is a simply a short message that is sent between specific devices (smart devices) using normal text [13].

2.4.2 Network and Transport Layer Protocols

1. IPV6: it is a network Layer protocol; its function is to provide end-to-end datagram transmission across multiple IP networks and provide packet-switched inter-networking [13].

2. 6LoWPAN: (low-power wireless personal area network) its modified version IPv6 protocol, this is because the fact that the size of the maximum transmission unit (MTU) used by the IEEE 802.15.4 standard is small, it allows each IoT is uniquely identified by an IPv6 network address [22].

3. RPL: The Routing Protocol for Low-Power and Lossy Networks (RPL) is used to support 6LoWPAN environments. It operates connection with type point-to- point beside communication between multi-points and single point [23].

4. DTLS: (Datagram Transport Layer) It is the protocol that is used to secure datagram protocols. The protocol allows client/server applications to communicate in a way that is designed to prevent fake, tampering of messages [2].

5. NanoIP: (nano Internet Protocol): this is the protocol that provides networking services into embedded and sensor devices with minimal overheads [13].

2.4.3 Physical and Communication Layer Protocols

1. ZigBee: is a low-power, low data-rate wireless network protocol. ZigBee is mostly used in industrial part. ZigBee make, it is possible for IOT devices to work securely on any network [18].

2. BLE: (Bluetooth Low-Energy) it’s also called Bluetooth Smart. This protocol is invented to deal with the IoT, because it is scalable and flexible to all needs in this area. Beside its ability to reduce power consumption [18].

3. Wi-Fi: this protocol allows the network to establish radio wireless networking of devices. It has many features such as fast data transfer large amounts of data processing.

4. LTE: (Long term evolution): it is the protocol that is integrated into the existing cellular infrastructure for 2G and 3G. LTE nature makes it ideal for IoT applications. Besides that, the LTE guarantee high level of security in addition to scalable traffic management capabilities [24] [18].

5. LoRaWAN: it a communication protocol between gateways and the end devices in networks of battery-operated things beside support varying data rates [18].

6. NB-IoT: (Narrow-Band IoT) it is the protocol that standardized by the 3GPP standards body. This protocol supports indoor coverage while using LTE spectrum [2].

7. ANT: it is a wireless protocol that enables device operating in the 2.4 GHz to communicate between each other by control the connection according to specific standard rules [13].

2.4.4 Other Protocols

1. LwM2M: (LightweightM2M): it is a device management protocol designed for IOT devices and the requirement of machine-to-machine (M2M) environment. With LwM2M, OMA. The LwM2M protocol, used mainly in remote management of M2M devices. It has allowed operating by Constrained Application Protocol (CoAP) [2].

2. PKI: (public key infrastructure): it is security procedure which is used to implement a high level of authentication, data encryption and digital signatures. Current PKI technologies are difficult to use in resource-constrained devices, instead its run only in the gateways. This protocol depends on in another technique to do the functions related to IOT devices which called "X.509" [25].

2.5 SECURITY CHALLENGES IN IOT

Most researchers use a layered-based approach to describe the connection process between IOT devices, to describe this process, they made the whole connection process fall into three main layers which are application layer, network and transport layer and physical layer [2].

Above layered approach makes it easy to analyze security and privacy issue in each layer and trying to find a proper solution. Before we forward in our thesis, we must understand these challenges.

2.5.1 Physical Layer

1. Insecure Initialization: it’s necessary to take care of configuration of whole network device to ensure that the system will work without violating privacy and disruption of network services [26].
2. Low-level Sybil and Spoofing Attacks: The Sybil attacks which usually occur in a wireless communication network is some node with unreal identity trying to affect the network dynamic [27].
3. Insecure Physical Interface: there are many factors can collect together, so they can enforce big problem to proper functioning of devices in IOT [28].

2.5.2 Network and Transport Layer

1. Replay or Duplication Attacks: A reconstruction of the packet fragment fields in the 6LOWPAN layer may result in depletion of resources, buffer overflows and rebooting of the devices [22].
2. Privacy Violation on Cloud-Based IOT: cloud-based systems have its security and privacy challenges, this thing also affects cloud-based- IOT systems[29].
3. Session Establishment and Resumption: denial of service can use normal network session functions to enforce its attacks and occupy system resources [30].

4. Insecure Neighbor Discovery: The usage of neighbor discovery packets without proper verification may severe implications along with denial-of service [31].

5. RPL Routing Attack: The IPV6 Routing Protocol for Low-Power and Lossy Networks (RPL) is vulnerable to several attacks, these attacks will directly affect dynamic and processing and routing decision on the network. This attack has many details and types such as a Hello flood attack, Version Attack, Rank Attack and many other [32].

6. Sinkhole and Wormhole Attacks: its special type of RPL attacks, in sinkhole attacks, the attacker node responds to the routing requests which made this node a main routing point, using that it can do the malicious activity on the network [23].

7. Authentication and Secure Communication: Any loophole in security at the network layer or large overhead of securing communication may expose the network too many vulnerabilities [33].

8. Transport Level End-to-End Security: any threat in the end-to-end data delivery can cause some alteration of data exchanged between nodes [20].

9. The Selective Forwarding Attack: specially designed to disturb the routing path on the network, the DOS attack also could be launched where the malicious node forward the packets [23].

2.5.3 Application Layer

1. Middleware Security: The IOT middleware designed to render communication among heterogeneous, so if there is a threat in provision or mapping process this will make full system at risk [21].

2. Insecure Interfaces**:** For accessing IOT services, the interfaces used through web, mobile, and cloud are vulnerable to different attacks which may severely affect the data privacy [28].

3. Insecure Software/Firmware**:** insecure software/firmware can damage system entities, so software such as JSON, XML and other must code securely [2, 28].

4. DDOS (Distributed Denial of Service): It some types of attacks that target internet-based services. This attack usually uses a simple service discovery protocol (SSDP) to attack IOT devices which use this protocol, this simply aims to overload resource of targeted device resources which made it at high risk of fault, also this attack is difficult to discover before run time and to discover the source of these attacks [34].

5. CoAP Security with Internet: application layer messaging protocols supported by CoAP is highly subjected to attacks because of different formatting and protocols involved in this layer, this attack can use weakness in privacy and authentication [35].

2.6 IOT AND BLOCKCHAIN

The fact that blockchain is robust and operates in a trustless environment, makes it very difficult to alternate its data could be a magnificent solution in IOT security in general. The common mechanism to use IoT on blockchains, is by considering each IoT device as a blockchain node. Since the use of consensus algorithms in blockchains enable it to operate in a trustless environment, this will eliminate completely need to trust existence between devices.

This fact faces another fact which is that IoT devices are limited in storage to be able store a copy of blockchain, depending on their capacity, it is possible to either use them as lightweight clients, or have a higher capacity device as their blockchain representative node.

.

3. BLOCKCHAIN TECNLOGOY

3.1 OVERVIEW

Sensitivity of data in the economic field was a big headache for most financial organizations with questions such as how can they secure transactions and how can they provide trust in the financial environment?

In 2008, Nakamato presented a concept called the blockchain to the world and its application “bitcoin” [36]. A bitcoin is just a virtual currency that can be transferred and maintained without third parties such as banks. In general, a bitcoin blockchain has been the underlying platform and technology of many of today’s most popular cryptocurrencies. But what is the blockchain? A blockchain is fundamentally a de-centralized, distributed, shared, and an immutable database ledger that stores a registry of assets and transactions across a peer-to-peer (P2P) network [37]. It has chained blocks of data that have been time- stamped and validated by miners. The blockchain database is fully available to all network nodes, so each node has a copy of the whole database since the first transaction’s block contain several successful transactions, and these blocks are connected to form chains.

Generally, blockchain technology is the best implementation of decentralization Systems that provide additional features such as accountability and security. This platform can improve efficiency and reduce the costs significantly. Blockchain has many magnificent characteristics [38] such as:

1. Cryptography**:** blockchain is robust because of that all transactions are validated using complex computations and cryptographic proof among involved parties.

2. Immutability**:** blockchain data cannot be changed or deleted

3. Provenance**:** in blockchain we can track the origin of every transaction that have been registered.

4. Decentralization**:** each node in the blockchain network has access to the whole distributed database, this done by using a consensus algorithm.

5. Anonymity**:** each node in the blockchain network has a generated address, not user identity. This keeps users' anonymity, especially in a public blockchain structure

6. Transparency**:** the blockchain system cannot be faulted, because it requires high computing power to alter the blockchain network completely.

3.2 ARCHITECTURE

Blockchain architecture can be divided into three categories according to the nature of operations that internally occurred, permissions and how the network is structured [38], these categories are as follows:

1. Public Blockchain**:** All records available to the public and anyone could involve in the agreement process, so the efficiency is low, but it is completely decentralized.

2. Private Blockchain: This type is controlled by a group of users. Has feature of high privacy beside increase in the efficiency, but it is considered as central system.

3. Consortium Blockchain: This type of system is controlled by preliminary assigned users, in this type the efficiency and privacy are high, but it could be considered as semi-central system.

3.3 COMPONENTS OF BLOCKCHAIN

1. Nodes :It is the clients connected to the Blockchain system, and they are an essential part of the system. They are the key in processing of internal blockchain operation such as routing, mining, storing the blockchain data and serving as a wallet. All nodes are involved in verification and spreading of transactions and are responsible for discovering and control the connection with their peers. The node must store a copy of the blockchain, which is full data about all the transactions that have been happening since network starts and this feature is reason for make blockchain as decentralized systems because of eliminating the need of having a centralized server to store the data above [39]. Usually Nodes work as Miners, which verify and validate all the transactions made by all the network clients. All miners are nodes, but all nodes are not necessarily miners.
2. Blocks are one or more transaction that's created after mining processes in additional

to some other information. Blockchain is simply an ordered and timestamped linked list of blocks, that store all information about the transactions that have been occurring since the start of the system. Each of these blocks is linked to the previous one, i.e. Its parent, through a unique hash. These hashes are provided using the SHA256 hashing algorithm [38]. That's mean that the header of each block has a reference to its parent's hash. This linking continues all the way up to the first block in the blockchain, also known as the genesis block. The genesis block is the start of the blockchain. It is like the "settings" for your blockchain [40]. Details of block fields are shown in Figure 3.1.

3. Mining **:**It is the process of validation and verification of transactions. Mining is the way that the miner can get rewarded if he solves the next block in the blockchain. The process is starting after calculating the difficulty level of the blockchain. All the full nodes connected to the blockchain network recalculate this difficulty level after a certain amount of time [38]. After that miner downloads full blockchain copy and constructs a Merkle tree out of it, Merkle tree is used to connect transactions to blocks. Finally, the process of forming a block, a miner decides the number of transactions that will be included in the block, this could depend on in another factor depending on in blockchain implementation type, for Example in Bitcoin the factor is the maximum block size, while in Ethereum it is a maximum gas limit.

4. Consensus Algorithm: A consensus algorithm is a process in computer science used to achieve agreement on a single data value among distributed processing or systems [41]. Thus, the importance of consensus mechanisms in blockchain is to simplify, secure updating of a process or a state, in accordance with certain state transition rules, where a distributed set has the right to perform the state transitions. There are many consensus algorithms used by the Blockchain network [41]. the most common is the proof of work, Other common consensus algorithms include the practical Byzantine fault tolerance Algorithm (PBFT), the proof-of-stack algorithm (POs), and the delegated proof-of-stack algorithm (DPoS), the difference between them is shown in table 3.1.

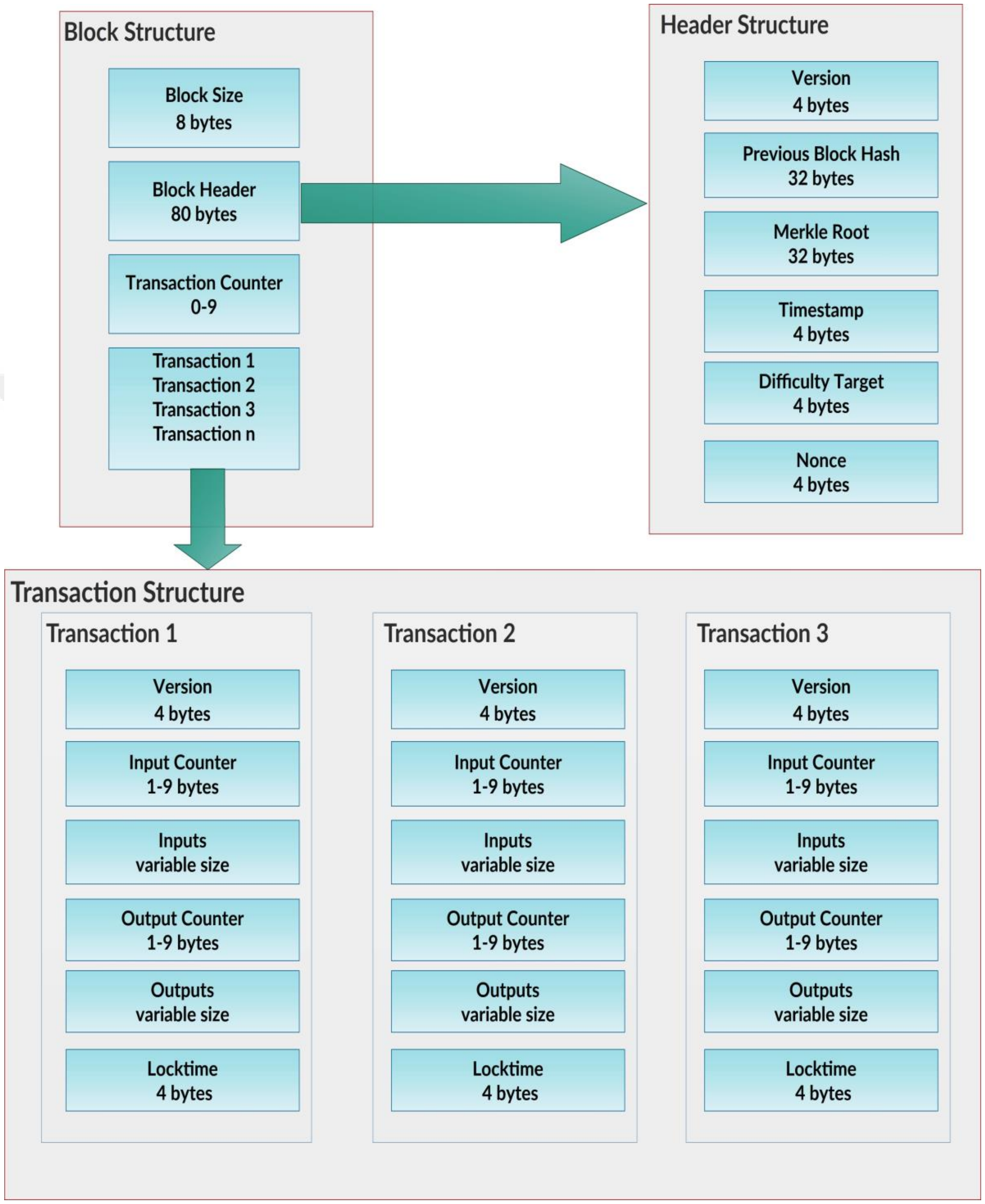


Figure 3.1: Block Structure

Table 3.1: Consensus Algorithm Summary

Algorithm	**Abstract**	**Pros**	**Cons**
Proof of work[14]	Ensure that the mining block is one and only version	1. Popular	1. Require high computation resources 2. Slow
Practical Byzantine fault tolerance algorithm[14]	Two nodes can communicate safely across a network	1. High throughput 2. Cost efficient	1. Unreliable in achieving a consensus across a distributed network of devices
Proof-of-stack[14]	Blocks are created by validator staking their tokens compete on which blocks are valid	1. Energy efficient	No cons noticed
Delegated proof-of-stack[14]	Maintaining irrefutable agreement on the truth across the network, validating transactions and acting as a form of digital democracy	1. No need to wait until a certain number of interested nodes has verified a transaction before it can be confirmed	1. Less secure

3.4 BLOCKCHAIN WORKFLOW

To understanding of how a blockchain works, it must know how a blockchain is working step by step Figure 3.2. Every client in blockchain can act as an entry point for several different blockchain users into the network [42]. In this section let us assume that each user transacts on the network via their own node. This workflow steps are as follows:

1. First the client interacts with the blockchain via a pair of private/public keys. They use their private key to sign their own transactions, and they are addressable on the network via their public key. This signed transaction is sent by the given client to its one-hop peers.

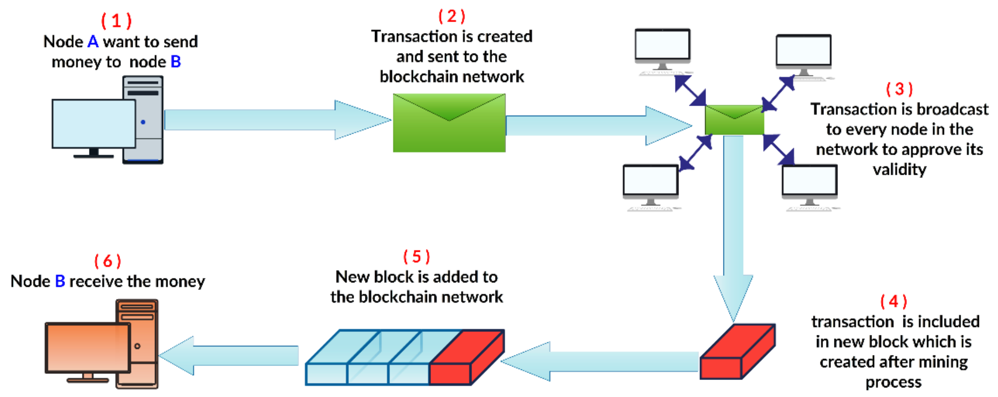

(1)
Node A want to send money to node B
(2)
Transaction is created and sent to the blockchain network
(3)
Transaction is broadcast to every node in the network to approve its validity
(4)
transaction is included in new block which is created after mining process
(5)
New block is added to the blockchain network
(6)
Node B receive the money

Figure 3.2: Blockchain Workflow

2. Second The neighboring peers check this incoming transaction validity before publishing it, invalid's transactions are discarded, and valid transaction is published to the entire network.

3. Valid transactions are ordered and packaged into a timestamped block, this done by using the mining process. Winner miner broadcasts this block back to the network. The winner miner here is chosen to depend on the consensus mechanism that the network implement.

4. All blockchain nodes check the validity of creating block first by checking if it contains valid transactions and the references via hash previous block on their chain is correct.

The above mechanism contains some functions that must be processed and implemented by a network node to enforce the concept of "blockchain" and these functions are:

1. Routing**:** controls the propagation of block and transaction.

2. Storage: each node must keep copies of blockchain (except for light node).

3. Wallet service: manages security mechanism that allows nodes to operate with digital currency such as "bitcoin".

4. Mining: it is th process of creating new blocks after solving complex mathematical functions or what is called “consensus algorithms”.

3.5 BLOCKCHAIN IMPLEMENTATION

A Blockchain great technology that can strongly affect many industries. The number of platforms big and in constant change which make it impossible to explain all of them, in this part define the most popular and most suitable for IoT domains.

Bitcoin is the first blockchain platform [36]. It provides a mechanism to manage transactions in a fast, cheap and reliable way, which can be integrated into the applications that need secured platform for paying. In IoT domain, autonomous devices can use Bitcoins to perform micro-payments. Bitcoin have a scripting language that allows to establish the rules, but it's weak in comparison with other smart contract platforms. The big revolution was in the invention of a new platform called Ethereum. Ethereum made the potential user space endless for the blockchain. The Ethereum blockchain was launched and opened for use to the public in July 2015 [43]. Ethereum uses a concept called the smart contract [9]. The smart contract is a program coded with a specific programming language called solidity [44]. This program controls how the transaction operates between nodes, so it would be executed whenever there is a transaction. It has a specific address that can be initiated by calling it.

Another platform is Hyperledger [45], which is an open-source platform on which various projects related to block chain have been developed. Hyperledger provide different objects and services for consensus and membership. One of the best features of Hyperledger is the application developer where Distributed application can be coded using general purpose languages. IoT devices can supply data to the blockchain through the IBM Watson IoT Platform, which have many services and features that provide data analysis and filtering [39].

The Multichain platform that is usually used in the creation of private blockchain architecture. Multichain depend on an API that extends the core of the original Bitcoin API with new functionality, allowing the management of all blockchain processes [45]. Litecoin is technically identical to Bitcoin, but with faster transaction validation times and improved storage efficiency This feature made the requirements of Litecoin processing in clients are lower, so it is more suitable for IoT [39]. Comparison between different types of blockchain implementation is summarized in Table 3.2.

Table 3.2: Comparison Between Blockchain Platforms

Platform	**Pros**	**Cons**
Bitcoin	1.decentralized	1. Mining needs high computation resources
	2. Open network	2. The transaction is confirmed in minutes, which is too much
	3. Alternative to regular money	3. No smart contract
		4. Bitcoin price is high
Ethereum	1.use smart contract	1. Ether price is high
	2. Creation of DApps	2. Mining time is high
	3. Supported languages- python, go, c++	
	4. Overhead is less than bitcoin	
	5. Alternative to regular money	
Hyperledger	1. Open-source enterprise-scale blockchain framework	1. Not a public block chain
	2. Has smart contracts	2.complete immutability is not possible
	3. Has database services	
	4. High scalability and fast transactions	
Multichain	1. Native multi-currency support	1. No scripting languages, so no smart contract
	2. Fast, quick in deployments	2. Source code for multichain is public
	3. Supported languages- python, c#, JavaScript, PHP, Ruby	
Litecoin	1. Efficient in storage management	1. Block-release timing is very high
	2. High transaction rate	

3.6 ETHEREUM

In this thesis, we utilize Ethereum blockchain because of its ability to implement smart contracts, which is a computerized transaction protocol that executes the functions of the contract [9].
It Is an open and free blockchain platform. Using this platform, the user can simply deploy his own smart contract, which is used to build what is called decentralized application. Ethereum is a network of connected nodes most of which have a copy of the blockchain full node and others called light nodes which only store headers of blocks of the blockchain. The Ethereum blockchain was launched and opened for use to the public in July 2015.

3.6.1 Ethereum Architecture and Core Component

3.6.1.1 Ethereum accounts

Ethereum usually has two types of accounts. The first one is called the externally owned account EOA, which is owned and controlled by the user. Each EOA has its own ether balance. Each account has its own address, so it can send and receive transactions to and from other accounts [9]. The second one is the contract account, which is controlled by the code of the smart contract, and has its own address, too.

3.6.1.2 Ethereum virtual machine

It is an environment running on the Ethereum node. It provides an isolated space to run the smart contract code [9].

3.6.1.3 Smart contract

It is a specific code located in the blockchain, which has its own address. Once the smart contract is written, compiled and deployed in the network, it gets its own address. Any transaction will not be done until the execution of the related smart contract.

The execution of smart contract is done by calling it by its address. Execution of smart contracts can be considered as a part of the verification of a successful transaction [5].

3.6.1.4 Keys

The Ethereum account has two types of key. The first one is the private key, which is used to send transactions from your account to another account. This key is dedicated to the user to control his account and must not be shared with others. The public key is used to generate an actual cryptocurrency address. This address serves as a user account identifier to which funds can be paid into [4].

3.6.1.5 Transaction and messages

Transactions are messages that are sent from an EOA to another EOA or to a contact address. It is already included in the block when created, while messages are sent just between the contracts and no EOA is involved; the messages are not included in the blocks [9].

3.6.1.6 Ether

It Is the currency used in the Ethereum platform, like bitcoin. Ether is the prize or rewards of winner miners. There is another unit that used to be as a base of the Ethereum called “Wei”, (1 ether equal to 10^18) [4].

3.6.1.7 Gas

It Is a term used to define digital fuel for performing work or operating on the Ethereum blockchain network. All transactions spend a specific amount of gas and gas is charged from the transaction sender to the miners. The gas is the concept used to define work difficulties, so it is proportional to work that is needed to execute a transaction and proof of work [4].

3.6.1.8 Solidity

Solidity is the most popular programming language used to write smart contract. It is a high- level language which, when compiled, gets converted to EVM (Ethereum Virtual Machine) byte code [44].

3.6.2 Ethereum Workflow

Every Ethereum client is connected to other clients or nodes to form an Ethereum blockchain network. A full node publishes transactions and blocks to the network and receives other transactions and blocks from it as explained in above in section of blockchain workflow.

The Ethereum blockchain differ from Bitcoin because of existing of smart contract [46]. A smart contract is a code that compiled from a high-level language like Solidity, this smart contract code is stored in the form of bytecode on the blockchain. This smart contract is executed by calling its address which it was gotten after deploying, the execution environment here is a VM or Virtual Machine. Virtual Machine allows transitions from one state to another, this transition mechanism may include accessing transaction-related accounts, computing operations, and updating/writing the state of the virtual machine. Whatever is executed on the virtual machine will alter its state. After completion and execution of transactions that will include in the block, the current state will be stored in what will become the next block.

The above mechanism or workflow is including some internal operation, the most important one is Merkle tree structure and mining [47]. In each Ethereum’s block, three Merkle roots are actually stored first one is a Merkle root of the Merkle tree of all transactions, second one is a Merkle root for receipts, last one is a Merkle root in the state of the virtual machine. Mining in Ethereum is little different from other platforms, it is starting by package all transactions from the transaction MemPool, executes smart contract on the EVM if it's required, creates Merkle tree structures and finally runs ETHash proof-of-work algorithm. The summary of Ethereum workflow is shown in figure 3.3.

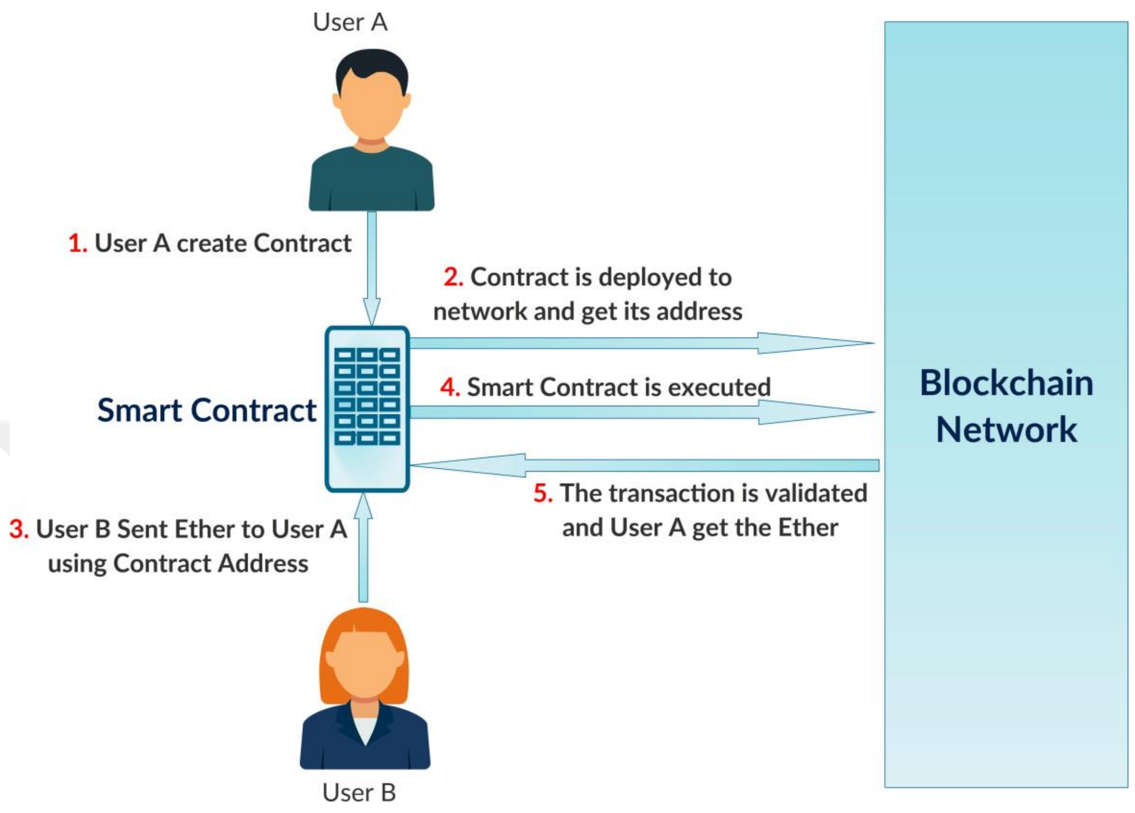

User A
1. User A create Contract
2. Contract is deployed to network and get its address
Smart Contract
4. Smart Contract is executed
Blockchain Network
5. The transaction is validated and User A get the Ether
3. User B Sent Ether to User A using Contract Address
User B

Figure 3.3: Ethereum Workflow

3.7 BLOCKCHAIN LIMITATIONS

1. Platform Size: network day by day become bigger and bigger, however, this network Must be resistant to bad actors such as attacks and must still grow stronger .

2. Scalability: All transactions have to be stored for validating the transaction, this generates a big problem for example, with the Bitcoin block chain produce nearly 7 transactions per second, which not suitable to fulfil the requirement of processing millions of transactions in a real-time fashion.

3. Privacy: block chain cannot guarantee the transaction privacy since the values of all transactions and balances for each public key are publicly visible .

4. 51% Attack : it means that nodes with over 51% computing power could control the blockchain and all the transactions.

4. DOMAIN-BASED SYSTEM

A domain-based system is our solution to handle most of the issues and concerns that exist in current IOT access management system. The solution is presented as an architecture where IOT devices are in separate virtual domains and the interaction between devices will be governed by rules .Domain-based system is constructed based on two platform IOT network and blockchain technology ,this integrating require some middleware systems to handle the interfaces and data type conversion between two platforms as shown in Figure 4.1. The rules are implemented by utilizing the famous decentralized technology “blockchain.”

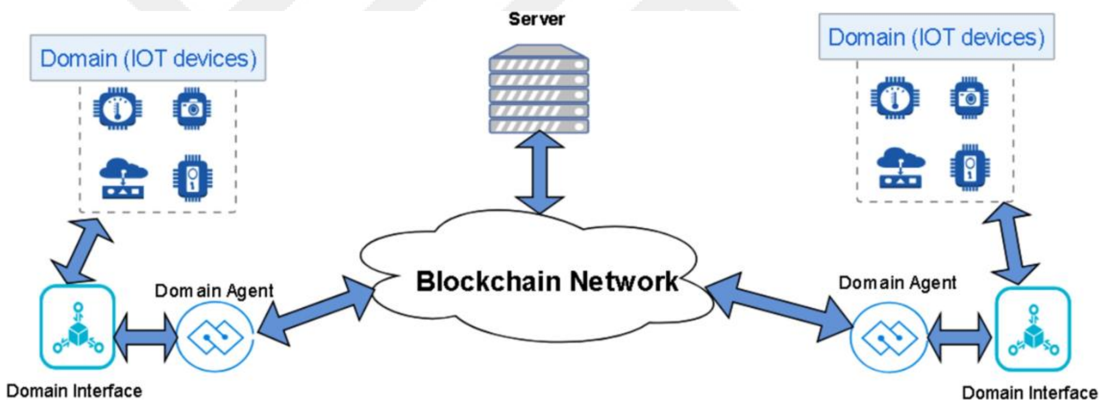


Figure 4.1 : Domain-Based System Architecture

4.1 SYSTEM COMPONENTS

4.1.1 IOT Devices

They are the devices that can perform the function of sensing and/or actuating and exchange data with other connected devices and applications. IOT devices can have one or more resources depending on the technology and functions of that device. For this reason, we build our system to deal with resources, not with the devices. Each resource in our system has a unique identifier called “Resource ID”. This is the most important requirement in our system. This ID can be generated using many technologies such as IOT cryptography solutions.

This approach is very reliable and scalable because it does not matter how many resources there are in each device. These devices will be in different domains according to a prior plan addressed by the system owner and executed by the server using the smart contract. However, this device will not be apart from the blockchain network because of the limitations in these devices.

4.1.2 Domain Interface

It is a part of the system that works as the middleware between IOT devices' environment and the blockchain network. This part will cause the system to override the problem of limitation in IOT devices. The domain interface takes data from the IOT device as an asynchronous message format by using a request- response based model protocol, which is called "CoAP". Then, it translates it to the JSON format. After that, exchange of the data with the blockchain takes place using the RPC [48] port opened by the domain agent. The process towards IOT devices from the blockchain is vice versa.

4.1.3 Domain Agent

The domain agent works as a coordinator for a specific domain in the blockchain. It listens to all the requests from that domain and returns access results. The domain agent has a normal account in the blockchain that usually works in light mode [39] . This account has an address that implements all domain devices of the system into the blockchain, so any domain can exchange the data with any other domain using this address.

4.1.4 Server

The server implements the centralized aspect of our system. It is responsible for choosing domain agents and devices that are managed by this agent. This is done using the smart contract. A server is a normal node in the blockchain that deploys the smart contract on the network. By using this smart contract, the server will manage and control the whole system it be a device with suitable specifications to be a full node inside the blockchain. That is because the server can do the mining function besides creating smart contracts.

4.1.5 Miners

Miners are computers with good processing capabilities. Miners are the guarantee to enforce the robustness of the blockchain in our network because they verify transactions “Access requests” in our system. Those miners compete to add blocks, which contain a finite number of transactions to the blockchain. Another important thing is that the miner is providing the system with storage capabilities. All contracts of the system with their lists and functions are stored in the blockchain.

4.1.6 Smart Contract

This is the most important part in the system. It implements a decentralized aspect of the system. Smart contracts in our system manage access control according to specific rules and permissions. This access management is executed directly between subject and object resources, so the server or any third party will not be involved. The smart contract is created and deployed in the network by using a server. Once it is deployed, the contract gets its address that can be initiated by calling it. In our system, we choose to create two types of contracts, which are explained below:

4.1.6.1 Local contract

For each given domain in our system, there is one local contract. That means it is responsible for managing several devices located at that domain. The local contracts in our system must store a list of domain resources identified by their IDs, so accessing resources on the same domain requires executing the code in the local contract only. This will reduce gas usage and shift our system to be lightweight. The resource list in the local contract contains some information about each resource as shown in table 4.1.

Table 4.1: Resource List Illustration

Resource ID	Resource name	Permission	Agent Address
155666	“Temp”	“Allow”	“0xxxxxxxxxx. ”
198956	“Press”	“Deny”	“0xxxxxxxxxx. ”

The main function of the local contract is the local access control function. The local contract has other functions that manage resources related to the given domain, such as deleting or updating or adding a resource to the contract, and most of these functions can only be called by the server. These functions are:

1. Register Resource: this function is used to Add new resources to the system in local contract and can called only by the server

2. Delete Resource: this function is used to Delete resource from the system and can called only by the server.

3. Fetch Resource: this function is used to getting information on specific resource and called only by the server.

4. Update Resource Permission: this function is used to Update information on specific resource and can call by server.

5. Check Resource: this function is used to Check existence of specific resource in the system and can call by server and domain Agent.

6. Change Agent: this function is used for Change domain agent of specific local contract and can call by the Server.

7. Local Access Control: this function Accept subject and object resource IDs as parameters and apply access control algorithms either by using local contract only or using local contract and global contract together and finally return the result of the domain agent, it can call by domain agent.

8. Delete local Contract: this function is used to Delete the contract for the system and can call by server.

4.1.6.2 Global contract

This contract is responsible for managing access control between two resources located in different domains. To do that, this contract stores a list of methods that include permission and rules related to two resources, as shown in Table 4.2.

Table 4.2 : Method List Illustration

ID	Resource name	Resource permission	Object Agent Address	Subject Agent Address	Limit	Positive Count	Negative Count	Status
1	“Temp”	“Allow”	“0x. . . . .”	“0x. . . .. . . ”	5	0	0	TRUE
2	“Press”	“Deny”	“0x. . . .. “	“0x. . .. . . . ”	4	0	0	FALSE

Each method manages access between a subject agent in one domain and an object resource in another domain. All subject resources are implemented by a domain agent to reduce the number of methods in the system, which will efficiently decrease the gas needed. Besides the method list in the global contract, there are two other lists. The first one is the cache list, and the second is the block list. The cache list contains methods that are frequently executed, and their access result is positive, while the block list contains methods that frequently called, and the access result is negative. We add a limit numbers as a field for each method in the method list. These fields are used to control whether the method should be in the cache or the block list, as we will see in the system operation section.

A global contract is executed after the failure of a subject local contract to verify access request, so the domain agent would not need to know anything about the global contract. The main function in the global contract is the global access control function, which checks permission and manages the methods in cache and block lists according to a global access control algorithm. There are many functions in the global control to manage the method list, the block list and the cache list, these functions are:

1. Register Method: this function is used to Add access method between specific domain agent and specific resource locate in another domain and save it in the global contract and can call by server.

2. Delete Method: this function is used to Delete access method from global contract and can call by server.

3. Get Method: this function is used to getting information about specific access control methods and can call by server.

4. Update Method: this function is used to Update information about specific access control methods and can call by server.

5. Activate Method: this function is used to Change the status of deactivated access control method to true and make it active and can call by server.

6. Deactivate Method: this function is used to Change the status of specific access control method to false and make it deactivate and can call by server.

7. Add to Cache: this function is used to Add specific access control method to cache list, and it's called by Global access control function in global contract.

8. Add To block: this function is used to Add specific access control method to block list and it's called by Global access control function in global contract.

9. Trigger Block List: this function is used to Refresh the status of specific access control method to check if the block time is ending and the remove it from block list and it's called by Global access control function in global contract.

10. Global Access Control: This function takes the control from local contract and it takes subject agent address and object resource ID as a parameter and apply global access control algorithms and then return the result to the caller local contract, this function is called by local contracts.

11. Delete from Cache: this function is used to Delete specific access control method from the cache, and it called by Global access control function in global contract.

12. Delete Global Contract: this function is used to Delete specific access control method from the cache, and it can call by server.

4.2 SYSTEM OPERATION

Domain-based access control system is initialized after the server deploys the unique global contract. Then, it deploys the local contract for each domain in the system. The priority of deploying the global contract before the local contract happens because each local contract must know the address of the global contract. Once the local contracts are deployed, system entities can communicate with the local contracts using domain interfaces. However, to add some transparency in our system The domain interfaces do not need to know the address of the global contract.

Our system operation related to access control is started after the domain interface receives an access request, which contain (Subject resource ID/Object resource ID), usually this request is received as a CoAP message format. Thus, the domain interface converts this message format to something accepted and understood by the blockchain which is JSON format. The JSON request format is then sent to the blockchain by using a listening port in the domain agent. The agent at that moment will send a transaction containing an access request to the local contract address. This transaction will be verified and then included in the next mining block. The access control operation of our system is built-on using the smart contract as we mentioned before. However, the mechanism of handling the request above depends on the domains of both subject and object resources. Because of that, there are two scenarios for the operation of our system, which are:

4.2.1 Same Domain Access

This type of access control is managed by the local contract in our system as shown in Figure 4.2. The local access control function in the local contract accepts (Subject resource ID, Object resource ID) as a parameter. After that, a local access algorithm which shown in Figure 4.3 will be applied to these parameters to calculate the Access result. Local Contract Algorithm Apply three conditions which in consequence provide three levels of privacy and security check.

If the above conditions are satisfied, the result will return to the domain agent, and if the first condition is not satisfied, which checks whether the function called is from the pre-established domain agent, the request will be rejected immediately without having it forwarded to another procedure, while the failure of the other two conditions will transfer the execution of the global contract. After execution of the global contract, the local contract gets the result from the global contract and returns it to the domain agent.

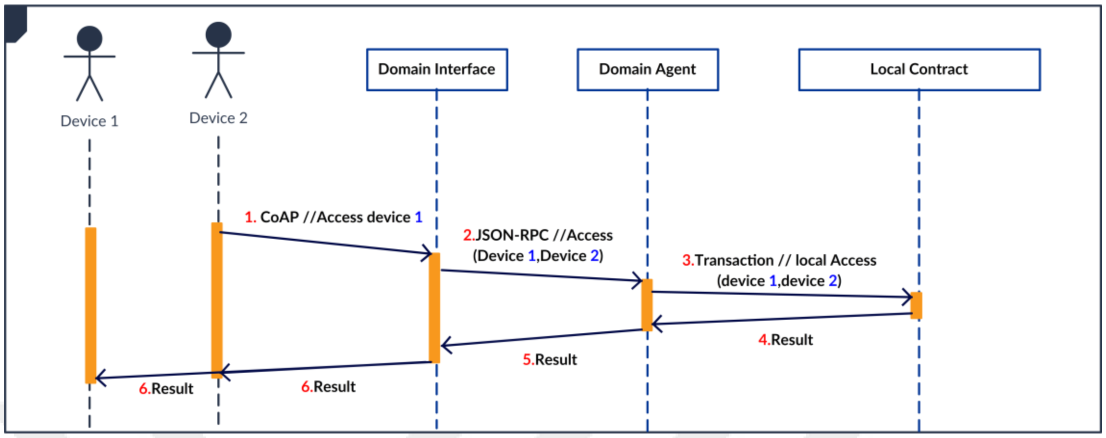

Device 1
Device 2
Domain Interface
Domain Agent
Local Contract
1. CoAP //Access device 1
2.JSON-RPC //Access (Device 1,Device 2)
3.Transaction // local Access (device 1,device 2)
4.Result
5.Result
6.Result
6.Result

Figure 4.2 : Same Domain Access Scenario

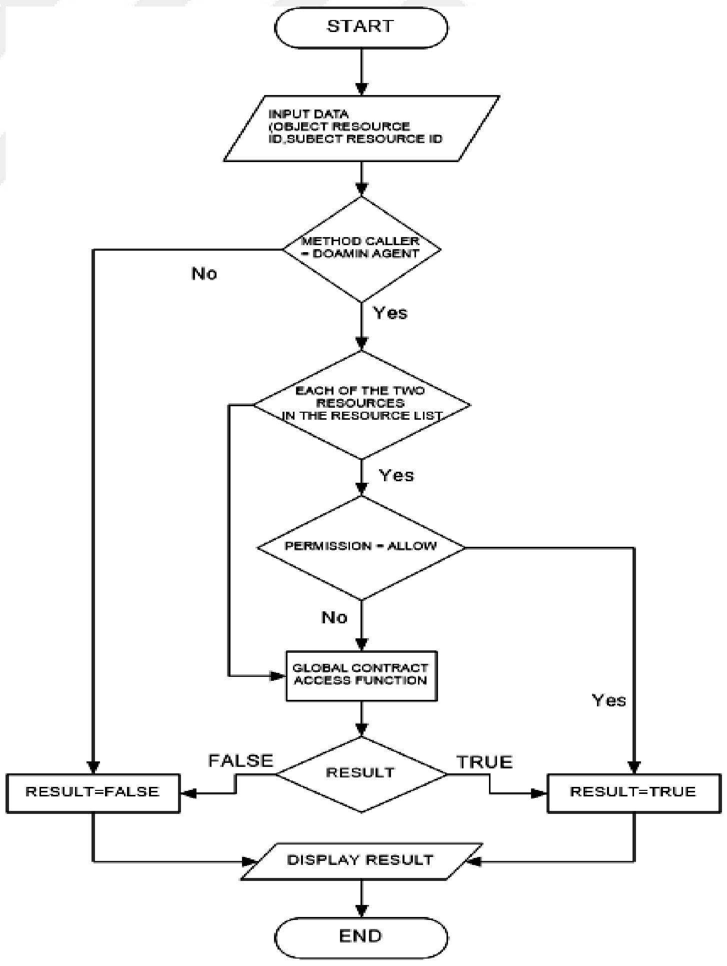

START
INPUT DATA (OBJECT RESOURCE ID,SUBECT RESOURCE ID
METHOD CALLER = DOAMIN AGENT
No
Yes
EACH OF THE TWO RESOURCES IN THE RESOURCE LIST
Yes
PERMISSION = ALLOW
No
GLOBAL CONTRACT ACCESS FUNCTION
Yes
FALSE
RESULT
TRUE
RESULT=FALSE
RESULT=TRUE
DISPLAY RESULT
END

Figure 4.3: Local Access Algorithm

4.2.2 Different Domain Access

Failure of one of the last two conditions in the local access control method causes the local contract to send a message to the global contract as shown in Figure 4.4. The message is the way that contracts can exchange data between each other. This scenario of the access control Different Domain is managed using a global access control function in the global contract. This function uses smart algorithms which are shown in Figure 4.5 to handle access control. This smartness is enforced by using the concept of cache and block lists. The cache list and the block list make our system work in a dynamic way to handle access request changes depending on the history of similar access requests. As we mentioned in our contribution, our system has the feature of being lightweight. This is satisfied by using this dynamic approach. In more detail, we design our code to check the cache list and the block list before forwarding it to the other check procedures. Reducing the number of executing line codes will affect gas consumption [4].

The global access function uses the methods list to define the rules and permissions for access of specific subject agents into specific object resources. The global contract has a function called (trigger block list) that refreshes the block list before checking the block list in search if there is any function that must be removed from the list. This check happens according to the time of blocking. We define specific mathematic functions (4.1)to calculate the block time, which is as follows:

$$T = T\_block + 100\ second \qquad (4.1)$$

Where T is the total block time and (T_block) is the time at the start of the blocking. Time in Ethereum is performed using a Unix time- temp [49], which is for example (01/22/2019 @ 11:40am (UTC)) equal to (1548157205). After completion of the execution of the global contract, the result is sent to the local contract of the Subject. The final stage in our system sends the result to the domain interface of the object and subject resources

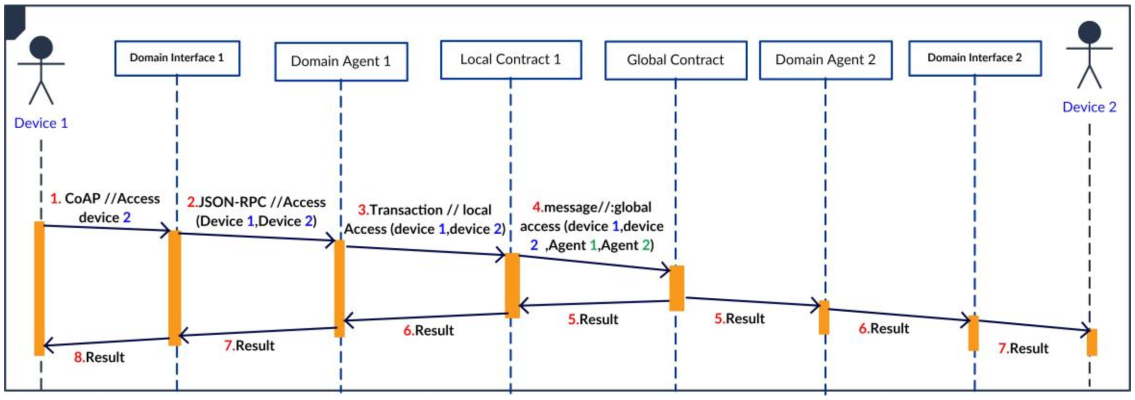

Device 1
Domain Interface 1
Domain Agent 1
Local Contract 1
Global Contract
Domain Agent 2
Domain Interface 2
Device 2
1. CoAP //Access device 2
2.JSON-RPC //Access (Device 1,Device 2)
3.Transaction // local Access (device 1,device 2)
4.message//:global access (device 1,device 2 ,Agent 1,Agent 2)
5.Result
5.Result
6.Result
6.Result
7.Result
7.Result
8.Result

Figure 4.4 : Different Domain Access Scenario

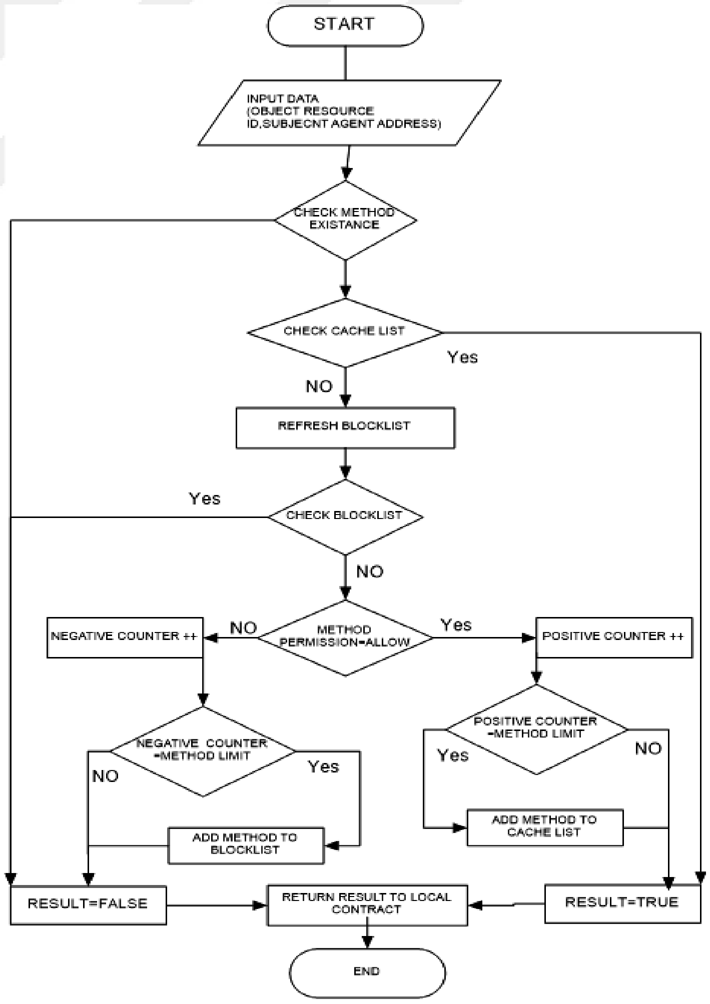


Figure 4.5: Global Access Algorithm

4.3 SYSTEM ASSUMPTION

1. Each resource in the system has a unique ID. This provides scalability and big addressing space. In addition, numbers are used instead of the resource name as a key in a smart contract code in order to reduce gas usage and to make the code much simpler.

2. The server is secured through the system for a lifetime. However, it is not recommended that the server is fault tolerant because the access control operation of the system is managed by the blockchain and smart contract, so if the server stops for a while, this will not affect the system operation.

3. Each IOT device belongs to one domain at least, so the dependence on the prior plan by the system administrator prefer to put frequently connected IOT device in the same domains. This will reduce access control processes because it may lead to the use of the local contract only.

5. IMPLEMENTATION

5.1 SYSTEM COMPONENTS IMPLEMENTATION

5.1.1 Domain Agent

We use Raspberry Pi 3 as a domain agent, this helps us to enforce the access issue explained in our system scenarios. Using Raspberry Pi and its processing capabilities allows us to run a domain interface. As we mentioned previously, the main function of the domain agent (raspberry pi) is to be a part of the blockchain network and implement domains in the blockchain. For that, we install GeTH client on it [50]. Then, we run GETH in light mode [39] for the domain agent to be able to accept requests from the domain interface and process it to the blockchain network using the GeTH client.

5.1.2 Domain Interface

The function of the domain interface is to interact with both the IOT environment and the blockchain network. Our domain interface is simply a JavaScript API [51] that accepts access Requests and then processes it to blockchain. Our JavaScript API uses web3.js library [52] to interact with the blockchain. In our system, we use HTTP and WebSocket providers [53] to get the GeTH client run on the domain agent. Each Domain runs two types of the JavaScript API. The first one is to request access and seeing the result of a request, that's means its run on the subject side, and a second one just to receive access result in the object side.

5.1.3 Server

First, the server must run the blockchain, so we would install the GeTH client on it again, but in this case, the server will run in full node to help the network in the mining process. Then, the remix will connect to the blockchain using this GeTH client. This connection allows the server to directly control the smart contract of the system.

5.1.4 Smart Contract

The smart contract in our experiment is created and deployed using the server. We assume that we have two domains in our system. This makes the implementation able to cover the system scenarios.For that, we create two local contacts, one for each domain, and create the global contract that has some methods that handle all access possibilities between the resources in these two domains. Our smart contract builds depending on the algorithms Explained in the previous section of this thesis.

5.1.5 Blockchain Network

For simplicity, we use a private Ethereum blockchain network in our experiment [9]. The main part of this network is the genesis.json file [54], which must be the same in all network clients. However, we use another laptop besides the server to work as a miner in our system. This allows us to partially simulate the mining mechanism in the real blockchain network. Our experiment starts by creating smart contracts and deploying it using remix IDE [55] run on the server. While doing that, the server and the other laptop are in mining status. After the deployment, the smart contract is included in the blockchain as a part of the last block, and they get their addresses. We use the addresses of the local contract and its ABI [56] to design a domain interface for each domain.

5.2 EXPERIMENT CONFIGURATION

5.2.1 Hardware and Software Specification

In Tables 5.1 & 5.2, we summarize most hardware and software specification of this experiment. This experiment consists of 2 laptops on of them work as a server and miners and the other laptop is just used for mining support, beside that we used two Raspberry Pi devices to implement 2 domains in this experiment. All mentioned 4 devices must run blockchain which is Ethereum in our system, to transfer these devices into Ethereum we choose GeTH client, which is "go programming language" implementation of the Ethereum [50].The goal of installing GeTH client is to create a private Ethereum network, this private network can simulate the real Ethereum network and it usually used for test and development. The most important thing in setting of the private Ethereum network is the genesis. json file [54], this file must be the same exact in all clients. Genesis. json file is the setting for the network and define many things.

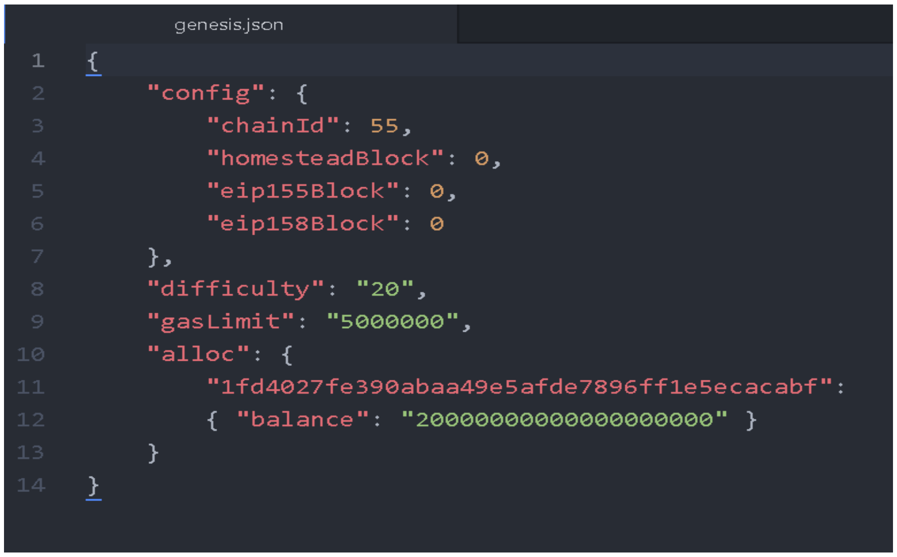
Table 5.1: Hardware Specification of The Experiment

Device	**Number of items**	**Operating system**	**Processor**
Dell laptop	1	Windows 10	Intel Cor i7
Lenovo laptop	1	Windows 10	Intel Cor I5
Switch	1	D-Link UI	-
Raspberry Pi 3 Model B+	2	Debian Linux (Raspbian Stretch with desktop)	1.4GHz, 64-bit Quad-core (Arm 7)

Table 5.2 : Software Specification of the Experiment

Ethereum implementation	GeTH client (v1.7.0-stable)
Smart contract development environment	Remix (Compiler version 0.4.25)
Smart contract interconnection	Web3 library (version 1.0.0)
Network difficulty	20
Gas limit	5000000
Provider between web3 and blockchain	WebSocket, and http
JavaScript environment	Npm 6.4.1

Our experiment genesis file is like the figure 5.1 .and its parameter is explained in the APPENDIX A .



```
{
    "config": {
        "chainId": 55,
        "homesteadBlock": 0,
        "eip155Block": 0,
        "eip158Block": 0
    },
    "difficulty": "20",
    "gasLimit": "5000000",
    "alloc": {
        "1fd4027fe390abaa49e5afde7896ff1e5ecacabf":
        { "balance": "20000000000000000000" }
    }
}
```

Figure 5.1: Genesis File

The interconnection between the software components of the Experiment is shown in Figure 5.2.

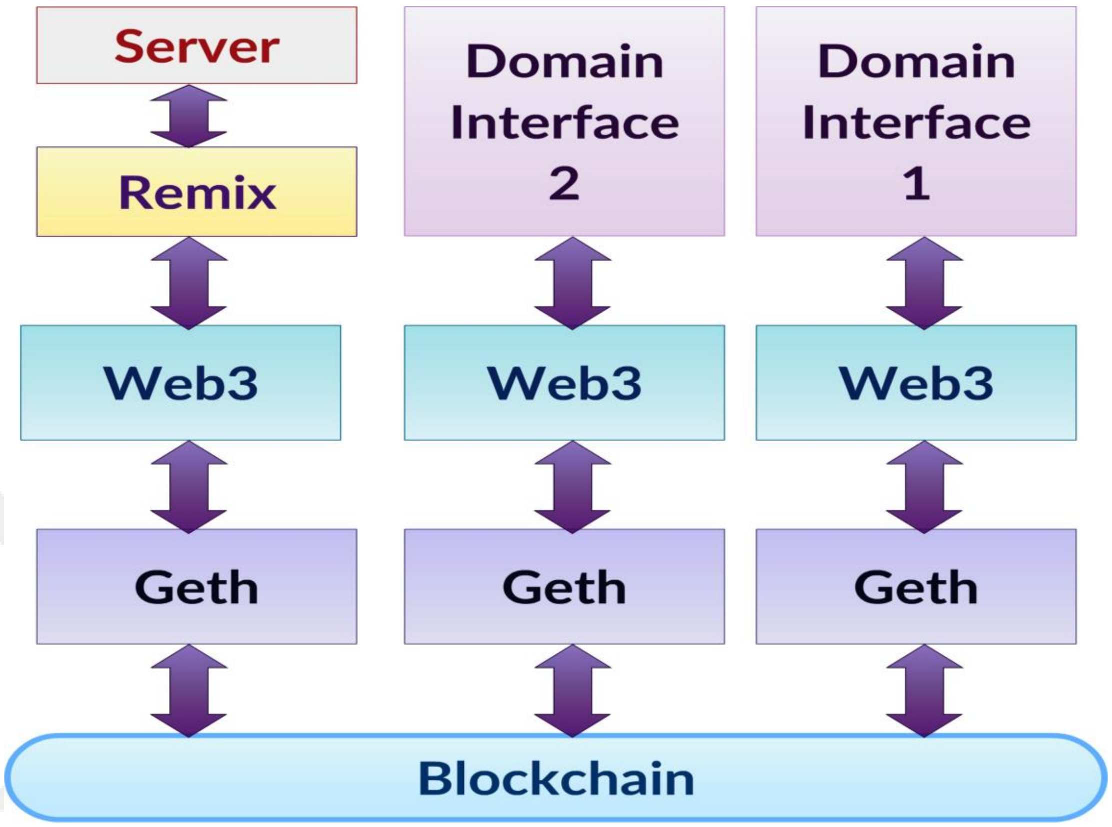


Figure 5.2: Experiment Components Interconnection

5.2.2 Ethereum clients Setup

5.2.2.1 Laptops setup

As we mentioned before both of laptops must run Ethereum client and this can be done by using GeTH. Installing Ethereum implementation was done using the following steps:

1. Download and install GeTH for windows from [57], in this experiment we use version (v1. 7.0-stable).

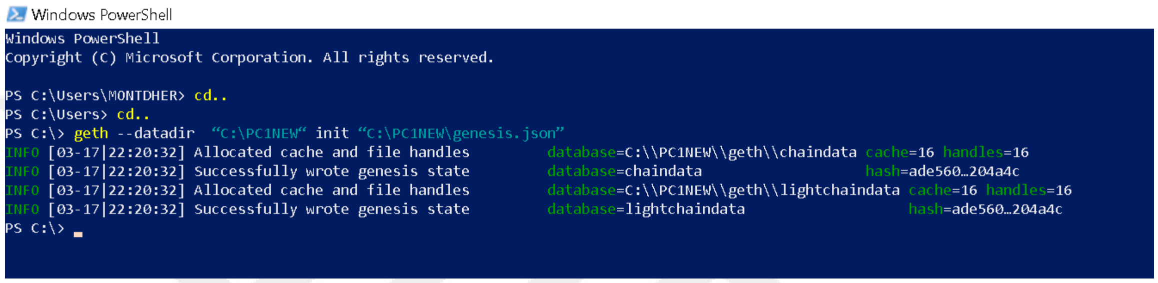
2. Create Data Directory (in this experiment, it is "C:\PC1NEW" for server Laptop and "C:\PC2NEW" for the second laptop).
3. Save the Genesis. json file in above directories and should be the same.

4. Run PowerShell with administrator privileges.

5. Run the following commands in both of laptops with care of changing directory field:

- GeTH –datadir “C:\PC1NEW“ init “C:\PC1NEW\genesis.json”

The result should be the same as in the Figure 5.3 This command will initiate Ethereum With this genesis file.



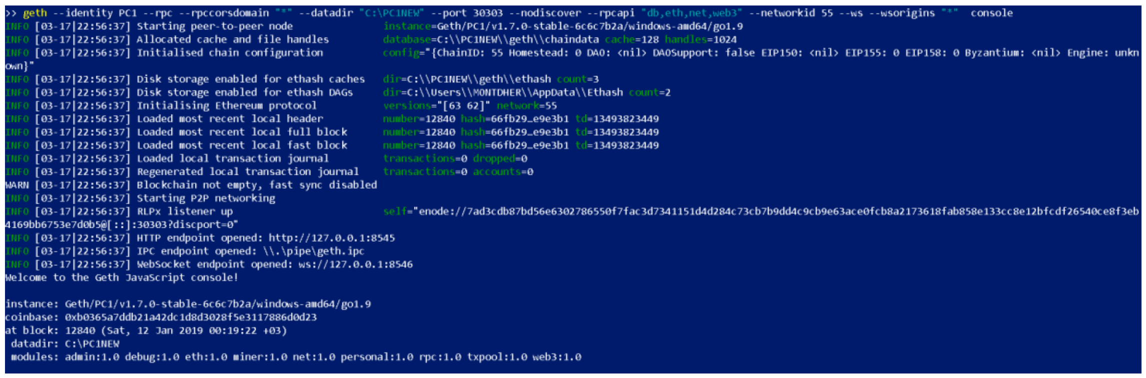
```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\MONTDHER> cd..
PS C:\Users> cd..
PS C:\> geth --datadir  "C:\PC1NEW" init "C:\PC1NEW\genesis.json"
INFO [03-17|22:20:32] Allocated cache and file handles         database=C:\\PC1NEW\\geth\\chaindata cache=16 handles=16
INFO [03-17|22:20:32] Successfully wrote genesis state         database=chaindata                      hash=ade560…204a4c
INFO [03-17|22:20:32] Allocated cache and file handles         database=C:\\PC1NEW\\geth\\lightchaindata cache=16 handles=16
INFO [03-17|22:20:32] Successfully wrote genesis state         database=lightchaindata                      hash=ade560…204a4c
PS C:\> _
```

Figure 5.3 : Genesis File Initialization

6. Run the following commands in both of laptops with again care of changing directory field and identity field to be "pc2":

- GeTH –identity PC1 –rpc –rpccorsdomain "*" –datadir "C:\PC1NEW" –port 30303 –nodiscover –rpcapi "db,eth,net,web3" –networkid 55 –ws –wsorigins "*" console

The Explanation of above command can be found in the [58]. The result should the same as in the Figure 5.4.



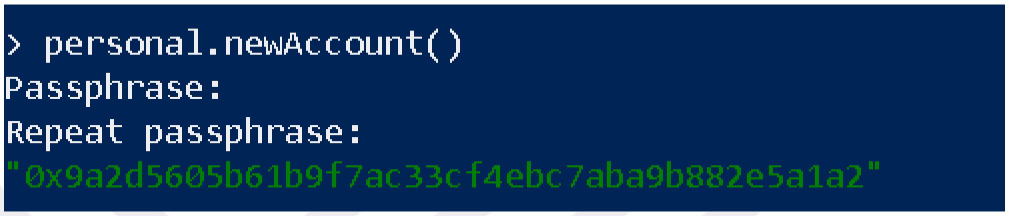
```
>> geth --identity PC1 --rpc --rpccorsdomain "*" --datadir "C:\PC1NEW" --port 30303 --nodiscover --rpcapi "db,eth,net,web3" --networkid 55 --ws --wsorigins "*"  console
INFO [03-17|22:56:37] Starting peer-to-peer node               instance=Geth/PC1/v1.7.0-stable-6c6c7b2a/windows-amd64/go1.9
INFO [03-17|22:56:37] Allocated cache and file handles         database=C:\\PC1NEW\\geth\\chaindata cache=128 handles=1024
INFO [03-17|22:56:37] Initialised chain configuration          config="{ChainID: 55 Homestead: 0 DAO: <nil> DAOSupport: false EIP150: <nil> EIP155: 0 EIP158: 0 Byzantium: <nil> Engine: unkn
own}"
INFO [03-17|22:56:37] Disk storage enabled for ethash caches   dir=C:\\PC1NEW\\geth\\ethash count=3
INFO [03-17|22:56:37] Disk storage enabled for ethash DAGs     dir=C:\\Users\\MONTDHER\\AppData\\Ethash count=2
INFO [03-17|22:56:37] Initialising Ethereum protocol           versions="[63 62]" network=55
INFO [03-17|22:56:37] Loaded most recent local header          number=12840 hash=66fb29…e9e3b1 td=13493823449
INFO [03-17|22:56:37] Loaded most recent local full block      number=12840 hash=66fb29…e9e3b1 td=13493823449
INFO [03-17|22:56:37] Loaded most recent local fast block      number=12840 hash=66fb29…e9e3b1 td=13493823449
INFO [03-17|22:56:37] Loaded local transaction journal         transactions=0 dropped=0
INFO [03-17|22:56:37] Regenerated local transaction journal    transactions=0 accounts=0
WARN [03-17|22:56:37] Blockchain not empty, fast sync disabled
INFO [03-17|22:56:37] Starting P2P networking
INFO [03-17|22:56:37] RLPx listener up                         self="enode://7ad3cdb87bd56e6302786550f7fac3d7341151d4d284c73cb7b9dd4c9cb9e63ace0fcb8a2173618fab858e133cc8e12bfcdf26540ce8f3eb
4169bb6753e7d0b5@[::]:30303?discport=0"
INFO [03-17|22:56:37] HTTP endpoint opened: http://127.0.0.1:8545
INFO [03-17|22:56:37] IPC endpoint opened: \\.\pipe\geth.ipc
INFO [03-17|22:56:37] WebSocket endpoint opened: ws://127.0.0.1:8546
Welcome to the Geth JavaScript console!

instance: Geth/PC1/v1.7.0-stable-6c6c7b2a/windows-amd64/go1.9
coinbase: 0xb0365a7ddb21a42dc1d8d3028f5e3117886d0d23
at block: 12840 (Sat, 12 Jan 2019 00:19:22 +03)
 datadir: C:\PC1NEW
 modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0
```

Figure 5.4: Running Geth (Laptop)

7. Now we can create a new account in both of laptops using the following command

- personal.newAccount()

8. After that, enter the password for this Account and press Enter. The result should be as in Figure 5.5. Now the GeTH client is installed with one account in both of laptops.



```
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0x9a2d5605b61b9f7ac33cf4ebc7aba9b882e5a1a2"
```

Figure 5.5 : Create New Account (Laptop)

5.2.2.2 Raspberry pi setup

Install GeTH in Raspberry Pi is little different because of the Raspberry Pi use Linux -based operating system, so the first step is installing stable version of the Raspbian operating system which is " Raspbian Stretch with desktop " in this Experiment. Install of both of two raspberry devices will be the same except for directories and identity, GeTH can be installed using the following steps:

1. Run Terminal and check the last update installed software to be with the latest versions using the following commands:

- sudo apt,-get update sudo apt-get dist-upgrade

2. Install the packaged dependencies using the following command:

- sudo apt-get install git golang libgmp3-dev

3. Create an Experiment's directory using the following commands:

- mkdir src

- cd src

4. Inside above directory creates another folder to store Genesis. json file:

- mkdir ETHER3NEW

- cd ETHER3NEW

For the second Raspberry Pi its "ETHER4NEW".

5. Now it's time to install GeTH using the following commands:

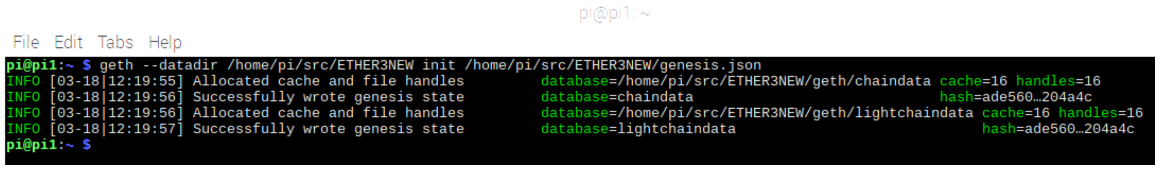
- git clone -b release/1.7 https://github.com/ethereum/go-ethereum.git
- cd go-ethereum
- make
- sudo cp build/bin/GeTH/usr/local/bin/

6. Run the following commands in both of raspberries with care of changing directory field:

- GeTH –datadir /home/pi/src/ETHER3NEW init

/home/pi/src/ETHER3NEW/genesis.json"

The result should be the same as in the Figure 5.6, this command will initiate Ethereum with this genesis file.



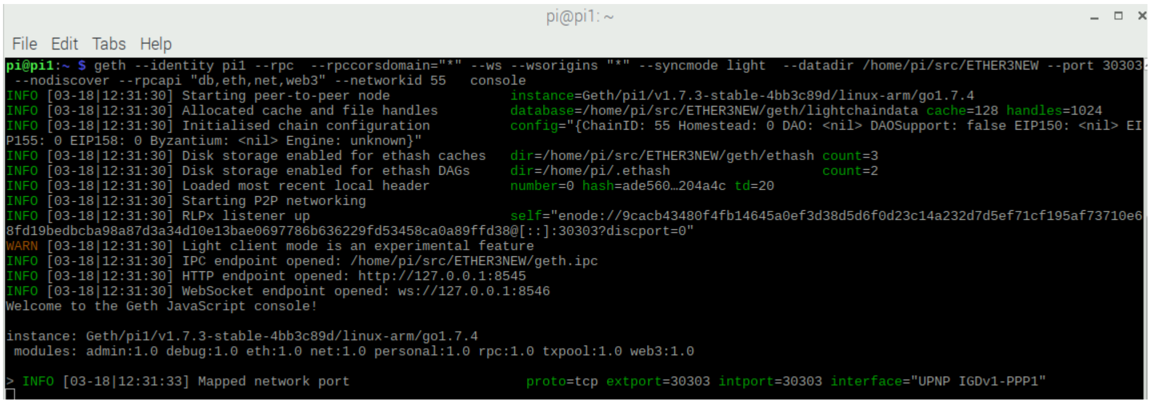
```
pi@pi1:~ $ geth --datadir /home/pi/src/ETHER3NEW init /home/pi/src/ETHER3NEW/genesis.json
INFO [03-18|12:19:55] Allocated cache and file handles         database=/home/pi/src/ETHER3NEW/geth/chaindata cache=16 handles=16
INFO [03-18|12:19:56] Successfully wrote genesis state         database=chaindata                             hash=ade560…204a4c
INFO [03-18|12:19:56] Allocated cache and file handles         database=/home/pi/src/ETHER3NEW/geth/lightchaindata cache=16 handles=16
INFO [03-18|12:19:57] Successfully wrote genesis state         database=lightchaindata                             hash=ade560…204a4c
pi@pi1:~ $
```

Figure 5.6 : Genesis File Initialization (Raspberry Pi)

7. Run the following commands in both of laptops with again care of changing directory field and identity field to be "ETHER4NEW" in the second Pi:

- GeTH –identity PC1 –rpc –rpccorsdomain "*" –datadir /home/pi/src/ETHER3NEW – port 30303 –nodiscover –rpcapi "db,eth,net,web3" –networkid 55 –ws –wsorigins "*" – syncmode light console

The result should the same as in the Figure 5.7, the important thing here that we run Raspberry Pi with light mode as we mentioned in the previous chapter.

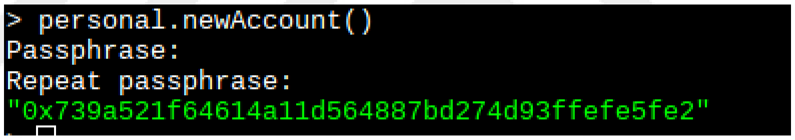


```
pi@pi1:~ $ geth --identity pi1 --rpc  --rpccorsdomain="*" --ws --wsorigins "*" --syncmode light  --datadir /home/pi/src/ETHER3NEW --port 30303
 --nodiscover --rpcapi "db,eth,net,web3" --networkid 55   console
INFO [03-18|12:31:30] Starting peer-to-peer node               instance=Geth/pi1/v1.7.3-stable-4bb3c89d/linux-arm/go1.7.4
INFO [03-18|12:31:30] Allocated cache and file handles         database=/home/pi/src/ETHER3NEW/geth/lightchaindata cache=128 handles=1024
INFO [03-18|12:31:30] Initialised chain configuration          config="{ChainID: 55 Homestead: 0 DAO: <nil> DAOSupport: false EIP150: <nil> EI
P155: 0 EIP158: 0 Byzantium: <nil> Engine: unknown}"
INFO [03-18|12:31:30] Disk storage enabled for ethash caches   dir=/home/pi/src/ETHER3NEW/geth/ethash count=3
INFO [03-18|12:31:30] Disk storage enabled for ethash DAGs     dir=/home/pi/.ethash                   count=2
INFO [03-18|12:31:30] Loaded most recent local header          number=0 hash=ade560…204a4c td=20
INFO [03-18|12:31:30] Starting P2P networking
INFO [03-18|12:31:30] RLPx listener up                         self="enode://9cacb43480f4fb14645a0ef3d38d5d6f0d23c14a232d7d5ef71cf195af73710e6
8fd19bedbcba98a87d3a34d10e13bae0697786b636229fd53458ca0a89ffd38@[::]:30303?discport=0"
WARN [03-18|12:31:30] Light client mode is an experimental feature
INFO [03-18|12:31:30] IPC endpoint opened: /home/pi/src/ETHER3NEW/geth.ipc
INFO [03-18|12:31:30] HTTP endpoint opened: http://127.0.0.1:8545
INFO [03-18|12:31:30] WebSocket endpoint opened: ws://127.0.0.1:8546
Welcome to the Geth JavaScript console!

instance: Geth/pi1/v1.7.3-stable-4bb3c89d/linux-arm/go1.7.4
 modules: admin:1.0 debug:1.0 eth:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

> INFO [03-18|12:31:33] Mapped network port                      proto=tcp extport=30303 intport=30303 interface="UPNP IGDv1-PPP1"
```

Figure 5.7: Running GeTH (Raspberry Pi)

8. Now we can create a new account in both of two devices using the following command:

- personal.newAccount()



```
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0x739a521f64614a11d564887bd274d93ffefe5fe2"
```

Figure 5.8 : Create New Account (Raspberry Pi)

9. After that, enter the password for this Account and press Enter, the result Should same as exist in Figure 5.8.

In this stage all the 4 experiment devices have been installed with GeTH client and each one has its own account.

5.2.3 Initiate Private Ethereum Network

At this point all the devices are run Ethereum, but they can’t know each other, so to connect these devices to form the private blockchain network we must do the following steps:

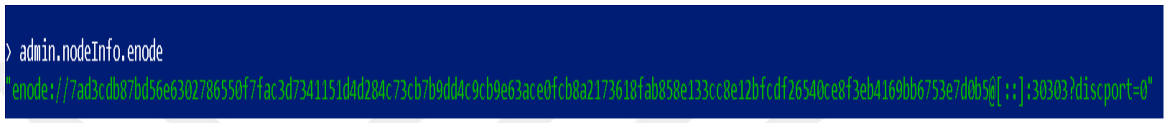
1. Ensure that the system time is the same on all 4 devices.

2. Set the IP address of each device, in this experiment we use LAN for simplicity (for laptop1 was "192.168.1.100" /laptop 2 "192.168.1.200" / pi 1 "192.168.1.110" /pi2 "192.168.1.210").

3. To establish the connection, we will use a technique called "Enode" [59]. An enode is a way to describe an Ethereum node in the form of a URI, separated from the host by an @ sign. to get the enode information of each node we must run the following command in each device:

- admin.nodeInfo.enode

the result should be the same as in the Figure 5.9.

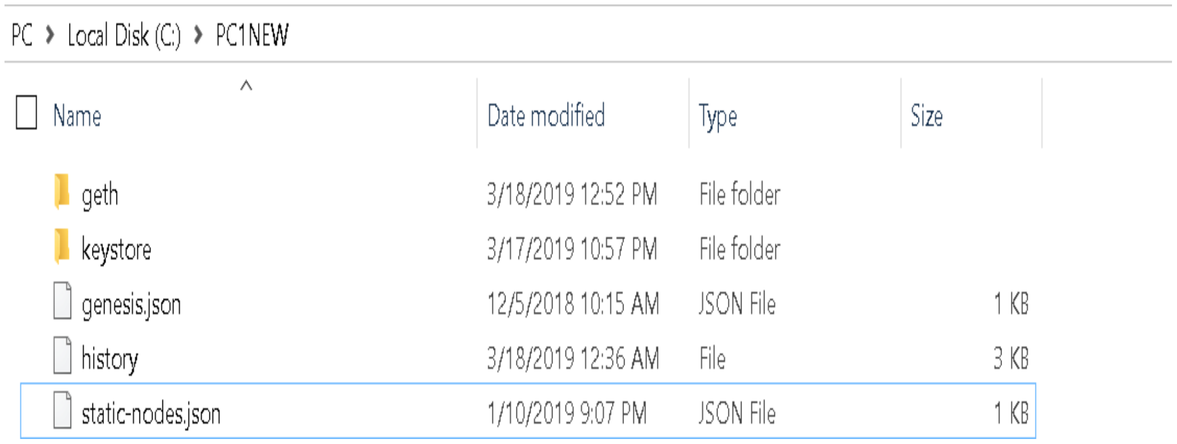


```
> admin.nodeInfo.enode
"enode://7ad3cdb87bd56e6302786550f7fac3d7341151d4d284c73cb7b9dd4c9cb9e63ace0fcb8a2173618fab858e133cc8e12bfcdf26540ce8f3eb4169bb6753e7d0b5@[::]:30303?discport=0"
```

Figure 5.9 : Enode Command Execution

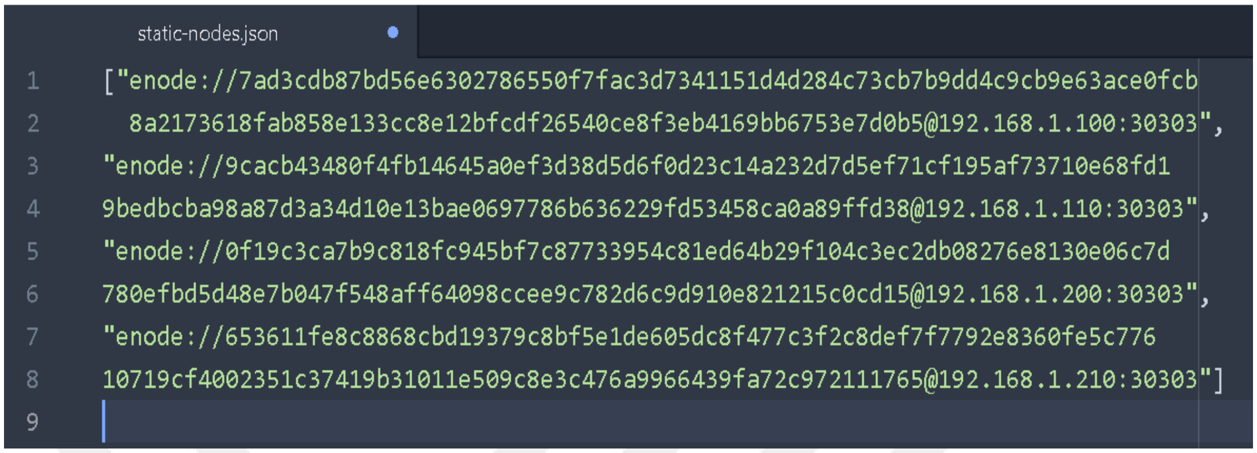
4. Now we collect enode information of 4 devices and put them in one json file and modify the IP and port fields in coordinating with information of each node, the resulted file should contain data similar the data in the blow file data

5. Rename Above file to be " static-nodes. json" and saves it in the same folder that contain Genesis. json file in each of devices same as shown in the Figure 5.10 and 5.11.



```
PC › Local Disk (C:) › PC1NEW

Name                 Date modified        Type          Size

geth                 3/18/2019 12:52 PM   File folder
keystore             3/17/2019 10:57 PM   File folder
genesis.json         12/5/2018 10:15 AM   JSON File     1 KB
history              3/18/2019 12:36 AM   File          3 KB
static-nodes.json    1/10/2019 9:07 PM    JSON File     1 KB
```

Figure 5.10 : Project Folder with Enode Configuration



```
["enode://7ad3cdb87bd56e6302786550f7fac3d7341151d4d284c73cb7b9dd4c9cb9e63ace0fcb
  8a2173618fab858e133cc8e12bfcdf26540ce8f3eb4169bb6753e7d0b5@192.168.1.100:30303",
"enode://9cacb43480f4fb14645a0ef3d38d5d6f0d23c14a232d7d5ef71cf195af73710e68fd1
9bedbcba98a87d3a34d10e13bae0697786b636229fd53458ca0a89ffd38@192.168.1.110:30303",
"enode://0f19c3ca7b9c818fc945bf7c87733954c81ed64b29f104c3ec2db08276e8130e06c7d
780efbd5d48e7b047f548aff64098ccee9c782d6c9d910e821215c0cd15@192.168.1.200:30303",
"enode://653611fe8c8868cbd19379c8bf5e1de605dc8f477c3f2c8def7f7792e8360fe5c776
10719cf4002351c37419b31011e509c8e3c476a9966439fa72c972111765@192.168.1.210:30303"]
```

Figure 5.11 : Static-nodes File

5.2.4 Write and Deploy of Smart Contract

the system part responsible for creating and deploying of smart contract in Domain-Based system is the server .in this experiment one of two laptops will play a role of server, writing and deploying smart contract was done by follow the following step:

1. Run the network be repeating steps (5 & 6 in laptops setup / 6 & 7 in raspberries setup)

2. Open the IDE of coding the smart contract which is Remix using following URL:

- http://remix.ethereum.org/#optimize=false&version=soljsonv0.4.25+commit.59dbf8 f1.js

The opened page should be like Figure 5.12.

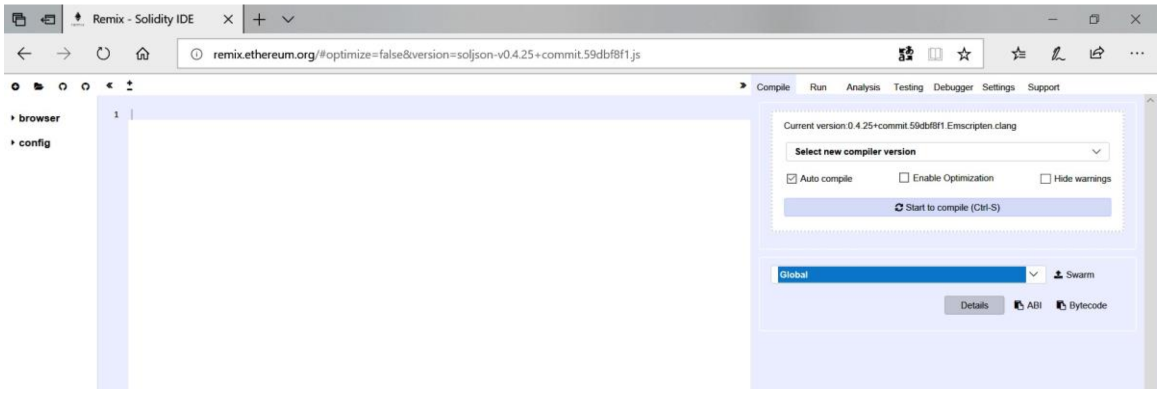
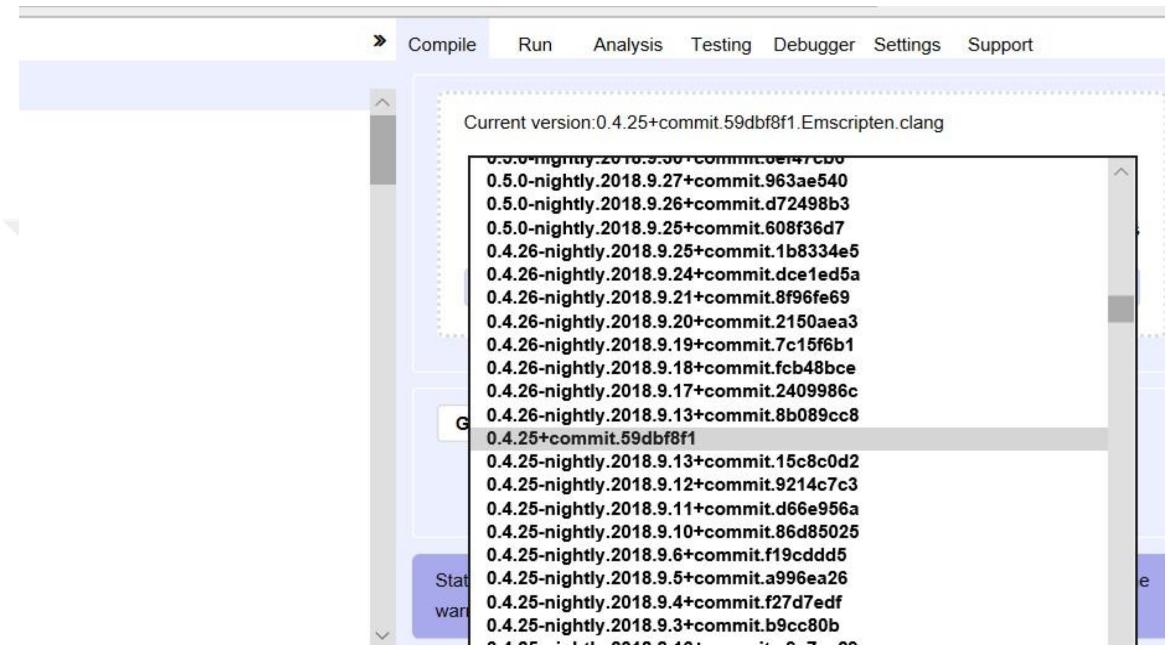

Remix - Solidity IDE
remix.ethereum.org/#optimize=false&version=soljson-v0.4.25+commit.59dbf8f1.js
browser
config
Compile
Run
Analysis
Testing
Debugger
Settings
Support
Current version:0.4.25+commit.59dbf8f1.Emscripten.clang
Select new compiler version
Auto compile
Enable Optimization
Hide warnings
Start to compile (Ctrl-S)
Global
Swarm
Details
ABI
Bytecode

Figure 5.12: Remix IDE

3. On server laptop run the following command to unlock an account that will deploy the contract by executing following command in GeTH command line opened previously:

- *personal.unlockAccount(eth.accounts[account index], "password", time)*



Current version:0.4.25+commit.59dbf8f1.Emscripten.clang

```
0.5.0-nightly.2018.9.27+commit.963ae540
0.5.0-nightly.2018.9.26+commit.d72498b3
0.5.0-nightly.2018.9.25+commit.608f36d7
0.4.26-nightly.2018.9.25+commit.1b8334e5
0.4.26-nightly.2018.9.24+commit.dce1ed5a
0.4.26-nightly.2018.9.21+commit.8f96fe69
0.4.26-nightly.2018.9.20+commit.2150aea3
0.4.26-nightly.2018.9.19+commit.7c15f6b1
0.4.26-nightly.2018.9.18+commit.fcb48bce
0.4.26-nightly.2018.9.17+commit.2409986c
0.4.26-nightly.2018.9.13+commit.8b089cc8
0.4.25+commit.59dbf8f1
0.4.25-nightly.2018.9.13+commit.15c8c0d2
0.4.25-nightly.2018.9.12+commit.9214c7c3
0.4.25-nightly.2018.9.11+commit.d66e956a
0.4.25-nightly.2018.9.10+commit.86d85025
0.4.25-nightly.2018.9.6+commit.f19cddd5
0.4.25-nightly.2018.9.5+commit.a996ea26
0.4.25-nightly.2018.9.4+commit.f27d7edf
0.4.25-nightly.2018.9.3+commit.b9cc80b
```

Figure 5.13: Remix Compiler Options

In this experiment index is [0], password "1234" which set by the time of creating accounts, time is "300000 "seconds. Back to remix IDE, In the right choose the compiler version for the solidity code (0.4.25 in this experiment) as in Figure 5.13.

4. In the left corner click in the button add new contract as in Figure 5.14.

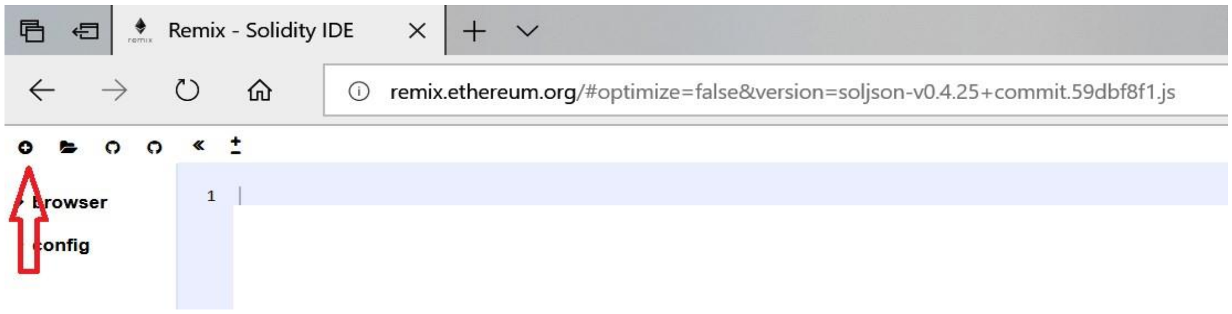

Remix - Solidity IDE
remix.ethereum.org/#optimize=false&version=soljson-v0.4.25+commit.59dbf8f1.js
browser
config

Figure 5.14 : Add New Contract

5. Connect remix with the installed ethereum by clicking on the run tab in upper right of screen then choose "Web3 provider" as in Figure 5.15.

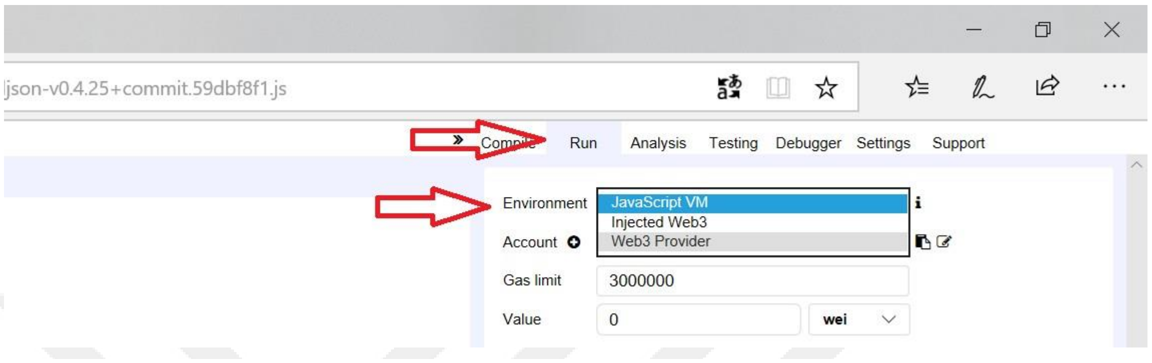
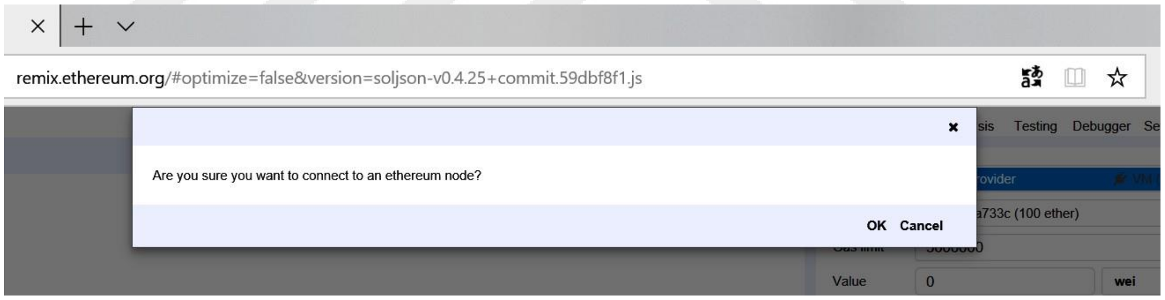


Figure 5.15 : Connect to Ethereum Network 1

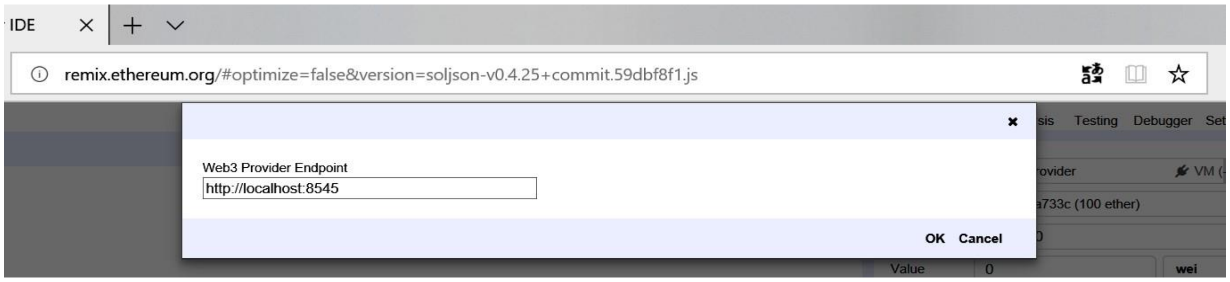
6. Pop-up will appear as in figure 5.16. press OK.


remix.ethereum.org/#optimize=false&version=soljson-v0.4.25+commit.59dbf8f1.js
OK
Cancel

Are you sure you want to connect to an ethereum node?

Figure 5.16: Connect to Ethereum Network 2

7. Another pop-up will appear request confirmation same as in Figure 5.17 (because we don’t change default setting), press OK.


IDE
remix.ethereum.org/#optimize=false&version=soljson-v0.4.25+commit.59dbf8f1.js
OK
Cancel

Web3 Provider Endpoint
http://localhost:8545

Figure 5.17 : Connect to Ethereum Network

8. Back to code space and write the contracts code, contracts used in this experiment are in [60] which are two local contract and one Global Contract.

9. Before deploying we must compile the code to check if there are any errors as in Figure 5.18.

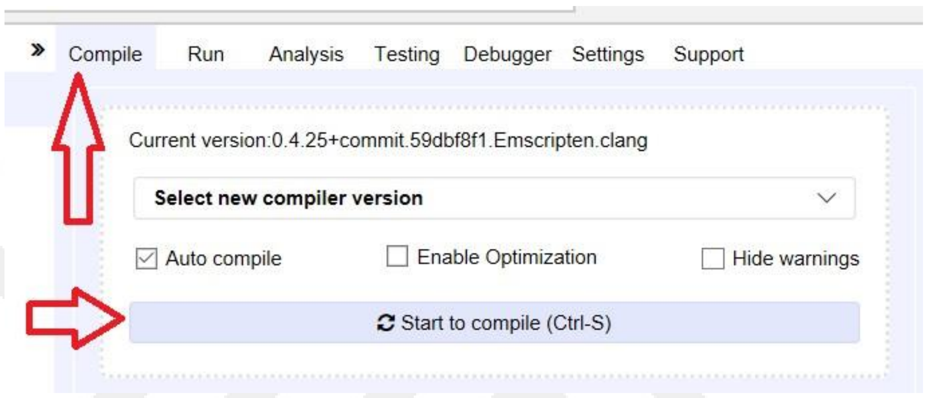
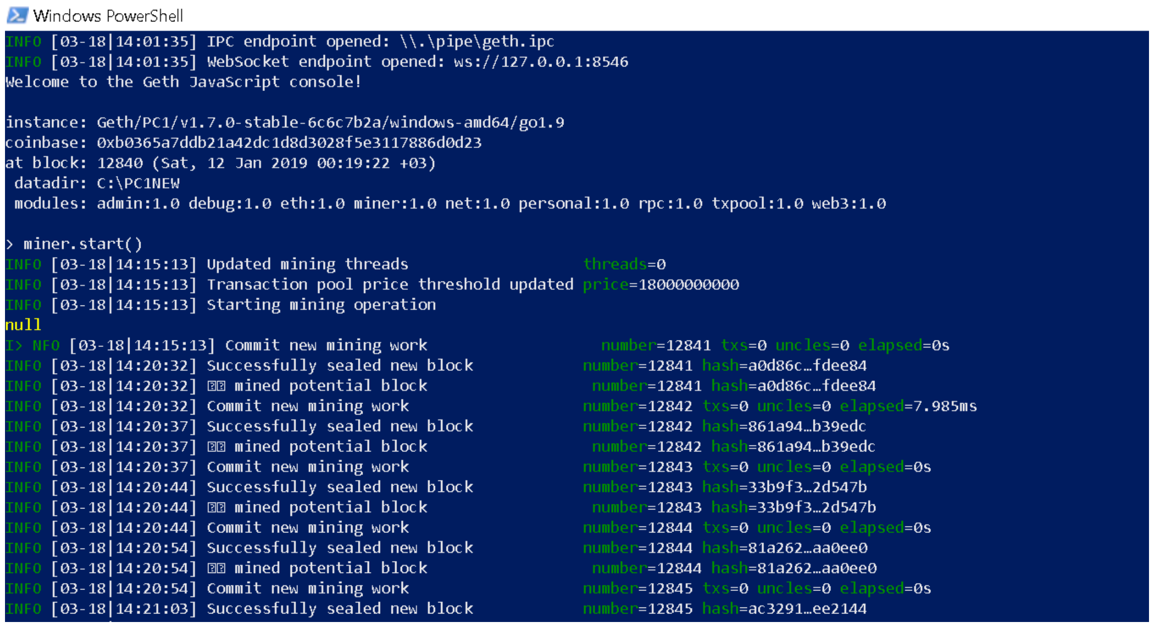

Compile
Run
Analysis
Testing
Debugger
Settings
Support
Current version:0.4.25+commit.59dbf8f1.Emscripten.clang
Select new compiler version
Auto compile
Enable Optimization
Hide warnings
Start to compile (Ctrl-S)

Figure 5.18 : Compile the Contract

10. In both of two laptops we run the following command:

- miner.start()

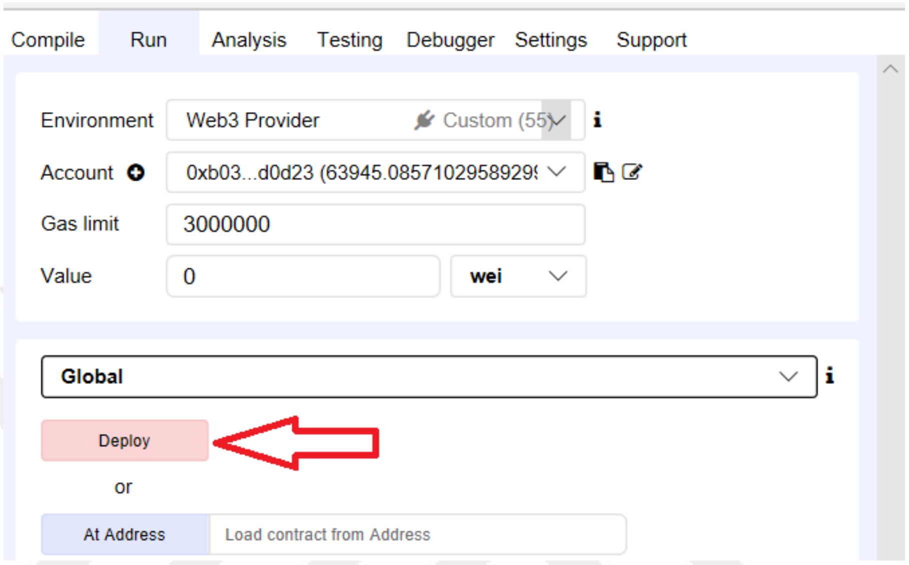
The result should the same as in the Figure 5.19.



```
Windows PowerShell
INFO [03-18|14:01:35] IPC endpoint opened: \\.\pipe\geth.ipc
INFO [03-18|14:01:35] WebSocket endpoint opened: ws://127.0.0.1:8546
Welcome to the Geth JavaScript console!

instance: Geth/PC1/v1.7.0-stable-6c6c7b2a/windows-amd64/go1.9
coinbase: 0xb0365a7ddb21a42dc1d8d3028f5e3117886d0d23
at block: 12840 (Sat, 12 Jan 2019 00:19:22 +03)
 datadir: C:\PC1NEW
 modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

> miner.start()
INFO [03-18|14:15:13] Updated mining threads                   threads=0
INFO [03-18|14:15:13] Transaction pool price threshold updated price=18000000000
INFO [03-18|14:15:13] Starting mining operation
null
I> NFO [03-18|14:15:13] Commit new mining work                   number=12841 txs=0 uncles=0 elapsed=0s
INFO [03-18|14:20:32] Successfully sealed new block            number=12841 hash=a0d86c…fdee84
INFO [03-18|14:20:32] ?? mined potential block                  number=12841 hash=a0d86c…fdee84
INFO [03-18|14:20:32] Commit new mining work                   number=12842 txs=0 uncles=0 elapsed=7.985ms
INFO [03-18|14:20:37] Successfully sealed new block            number=12842 hash=861a94…b39edc
INFO [03-18|14:20:37] ?? mined potential block                  number=12842 hash=861a94…b39edc
INFO [03-18|14:20:37] Commit new mining work                   number=12843 txs=0 uncles=0 elapsed=0s
INFO [03-18|14:20:44] Successfully sealed new block            number=12843 hash=33b9f3…2d547b
INFO [03-18|14:20:44] ?? mined potential block                  number=12843 hash=33b9f3…2d547b
INFO [03-18|14:20:44] Commit new mining work                   number=12844 txs=0 uncles=0 elapsed=0s
INFO [03-18|14:20:54] Successfully sealed new block            number=12844 hash=81a262…aa0ee0
INFO [03-18|14:20:54] ?? mined potential block                  number=12844 hash=81a262…aa0ee0
INFO [03-18|14:20:54] Commit new mining work                   number=12845 txs=0 uncles=0 elapsed=0s
INFO [03-18|14:21:03] Successfully sealed new block            number=12845 hash=ac3291…ee2144
```

Figure 5.19: Mining

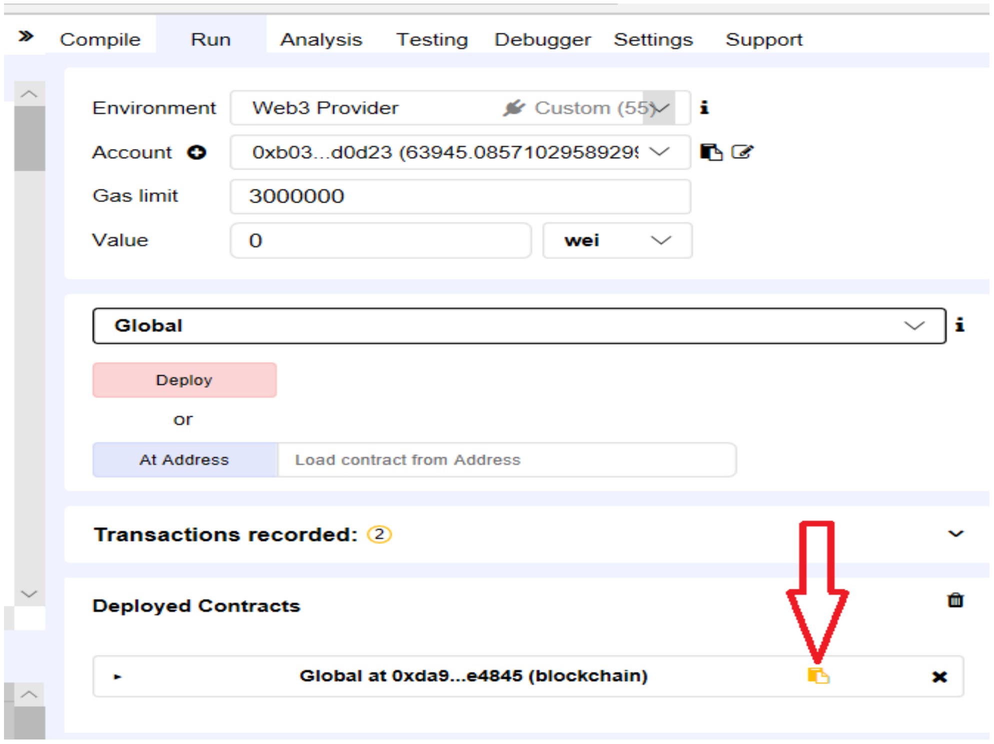
11. Deploying contract is done by clicking the deploy button in run tab as in shown in the Figure 5.20.



Environment: Web3 Provider — Custom (55)
Account: 0xb03...d0d23 (63945.0857102958929
Gas limit: 3000000
Value: 0 wei
Global

Figure 5.20: Deploy the Contract

12. After seconds the result of deploying will appear and Contract address can be gotten using the icon shown in Figure 5.21.



Environment: Web3 Provider — Custom (55)
Account: 0xb03...d0d23 (63945.0857102958929
Gas limit: 3000000
Value: 0 wei
Global

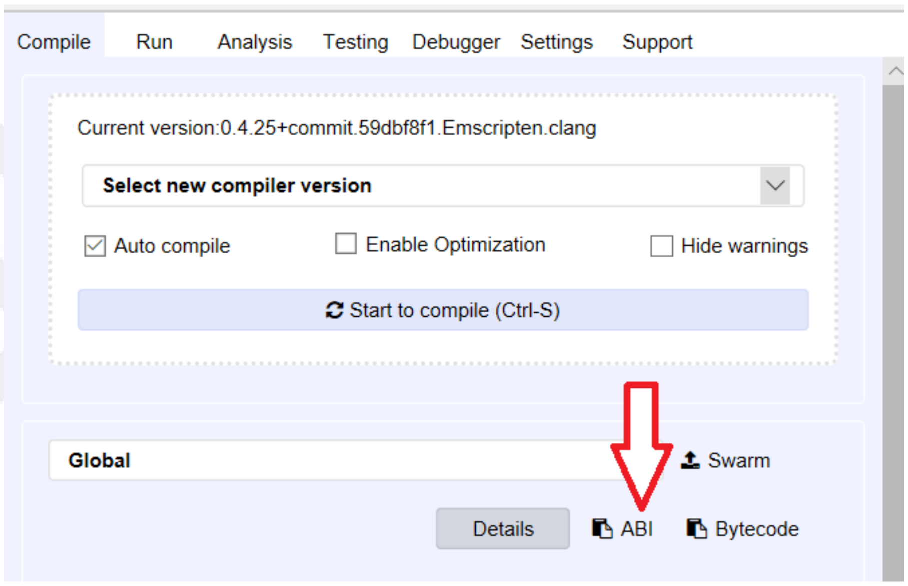
Transactions recorded: 2
Deployed Contracts
Global at 0xda9...e4845 (blockchain)

Figure 5.21 : Get Address of the Contract

13. Stop mining process by executing the following command:

- miner .stop()

Other contracts are deployed in the same way explained above. After deploying we must save contract address and get the ABI of that contract to be used in the design of each Domain Interface, this ABI can copy from a compiler tab menu in Remix page as in Figure 5.22.



Current version:0.4.25+commit.59dbf8f1.Emscripten.clang

☑ Auto compile ☐ Enable Optimization ☐ Hide warnings

Figure 5.22 : Get the ABI of the Contract

5.4.5 Create of Domain Interfaces

At this point we have private ethereum network, and we also deployed the contracts in the network in addition to that we know the address and ABI of each contract.

It's time to define the way that smart contract can handle access requests from outside, ethereum use Web3 library [52] to be as middleware between external request and ethereum smart contract. Previously in system architecture explanation domain interface is specific to one domain, and because of that each domain is implemented by one domain agent (Raspberry pi) so we install this domain interface in both of raspberries only.

In this experiment we create two types of domain interfaces first one for handling access request and the other one to monitor received access request. The design of domain interfaces of this experiment is done by doing the following steps:

1. Install node.js platform in both of raspberries [61].

2. Open a terminal and processed to project directory, then Initialize the project using the following command:

- npm init

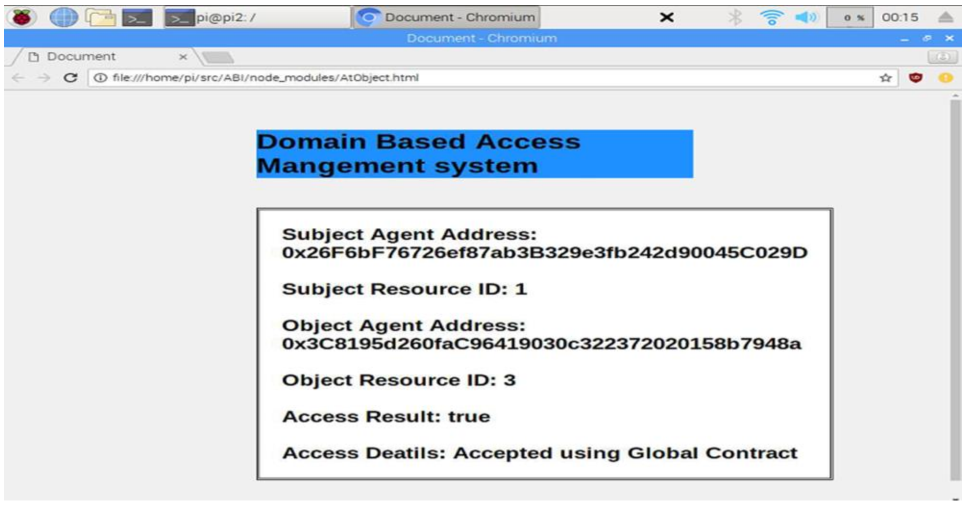
3. Install web 3 library by executing the following command:

- npm install web3@1.0.0- beta.35 –save

4. Using any text editor open a new file and start writing the code of each request and monitor interfaces, this code will contain information about how to connect with Ethereum contract and for that we use Web socket connection, for calling smart contract we used ABI that we get when deploying the contract, interfaces used in this experiment in [60].

5. Save the file as an html file, in the project directory of both of two devices.

6. Run both of interfaces, so one of the devices will be implement request side and the other as a receiver or monitor, the result should be same as exist in Figure 5.23 and 5.24 .



Domain Based Access Mangement system

Subject Agent Address: 0x26F6bF76726ef87ab3B329e3fb242d90045C029D

Subject Resource ID: 1

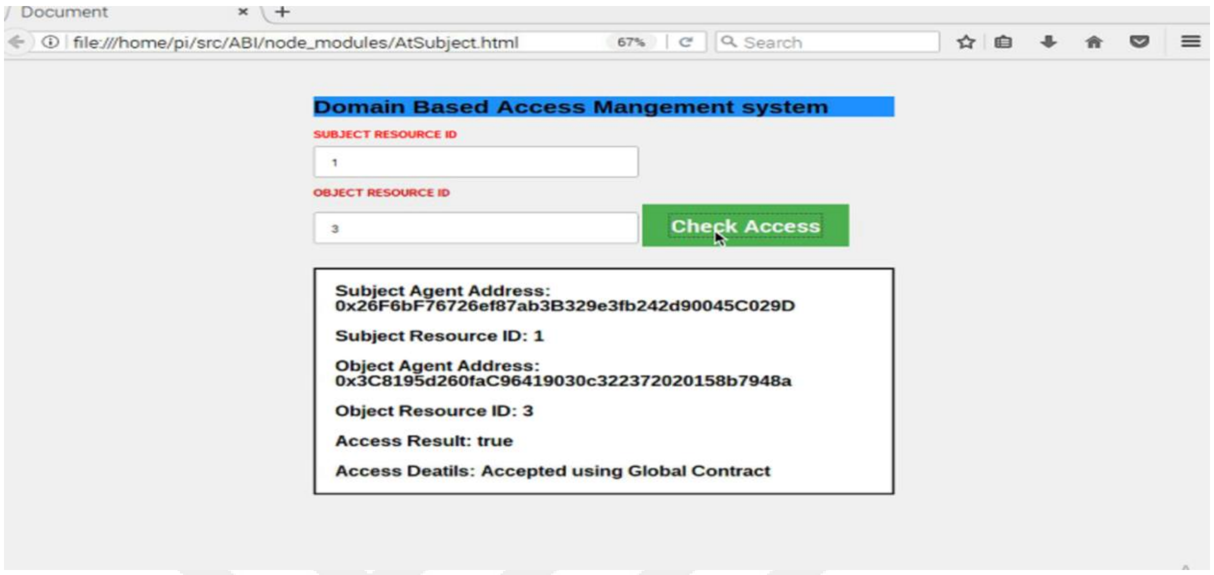
Object Agent Address: 0x3C8195d260faC96419030c322372020158b7948a

Object Resource ID: 3

Access Result: true

Access Deatils: Accepted using Global Contract

Figure 5.23: Monitor Interface


Document
file:///home/pi/src/ABI/node_modules/AtSubject.html
67%
Search
Domain Based Access Mangement system
SUBJECT RESOURCE ID
1
OBJECT RESOURCE ID
3
Check Access

```
Subject Agent Address:
0x26F6bF76726ef87ab3B329e3fb242d90045C029D

Subject Resource ID: 1

Object Agent Address:
0x3C8195d260faC96419030c322372020158b7948a

Object Resource ID: 3

Access Result: true

Access Deatils: Accepted using Global Contract
```

Figure 5.24: Request Interface

5.5 RUN THE EXPERIMENT

Running the experiment is done by following the following steps:

1. Repeating steps (5 & 6 in laptops setup / 6 & 7 in raspberries setup).
2. Executing the following command in GeTH command line opened previously

- personal.unlockAccount(eth.accounts[account index], "password", time)

in this experiment index is [0], password "1234" which set by the time of creating account, time is "300000 "seconds.

3. In both of two laptops we run the following command:

- miner.start()

4. Run both of two interfaces , send the request and watch the result in both of two interfaces.

6. RESULT AND DISCUSSION

6.1 RESULT

To test the above system implementation, we add some sample data into the system by using a server, Sample data in tables 6.1,6.2 and 6.3. These sample data allow us to evaluate our system under different scenarios. Figure 5.21 and Figure 5.22 shows the result on the subject and object by using a domain interface. The experiment proves the efficiency of our system to handle the access control. This was clear from gas usage as shown in Tables 6.4 and 6.5.

Table 6.1: Local Contract 1 Sample Data

Id	Resource Name	Resource Permission	Domain Agent Address
1	“TEMP”	“ALLOW”	0x26f6bf76726ef87ab3b329e3fb242d90045c029d
2	“GAS”	“DENY”	0x26f6bf76726ef87ab3b329e3fb242d90045c029d

Table 6.2: Local Contract 2 Sample Data

Id	Resource Name	Resource Permission	Domain Agent Address
3	“LIGHT”	“ALLOW”	0x3c8195d260fac96419030c322372020158b7948a
4	“PRES”	“DENY”	0x3c8195d260fac96419030c322372020158b7948a

Table 6.3 : Global Contract Sample Data

Method number	Id	Resource name	Resource permission	Domain Agent Address	Subject Domain Agent Address	Limit
1	3	“LIGHT”	“ALLOW”	0x3c8195d260fac96419030c322372020158b7948a	0x26f6bf76726ef87ab3b329e3fb242d90045c029d	3
2	4	“PRESS”	“DENY”	0x3c8195d260fac96419030c322372020158b7948a	0x26f6bf76726ef87ab3b329e3fb242d90045c029d	4
3	2	“GAS”	“DENY”	0x26f6bf76726ef87ab3b329e3fb242d90045c029d	0x3c8195d260fac96419030c322372020158b7948a	5
4	1	“TEMP”	“ALLOW”	0x26f6bf76726ef87ab3b329e3fb242d90045c029d	0x3c8195d260fac96419030c322372020158b7948a	3

Table 6.4: Gas Usage Using Local Contract Only

Subject Resource ID	**Object Resource ID**	**Gas usage 1**	**Gas usage 2**	**Gas usage 3**	**Gas usage 4**	**Gas usage 5**
2	1	29943	29943	29943	29943	29943
1	2	26137	26138	26138	26138	26138

Table 6.5: Gas Usage Using Global Contract

Method number	**Limit**	**Gas usage 1**	**Gas usage 2**	**Gas usage 3**	**Gas usage 4**	**Gas usage 5**	**Gas usage 6**	**Gas usage 7**
1	3	75272	60034	70197	30337	30337	30337	30337
2	4	75540	60302	60302	107322	31229	31229	31229
3	5	75540	60302	60302	60302	107322	31229	31229
4	3	75272	60034	70857	30337	30337	30337	30337

To make the evaluation process clearer, we compare our experiment results with the transaction base gas usage [4] in which the minimum amount that any transaction in Ethereum blockchain can consume is (21000). Experiment gas consumption result is shown in Figure 6.1, Figure 6.2, Figure 6.3.

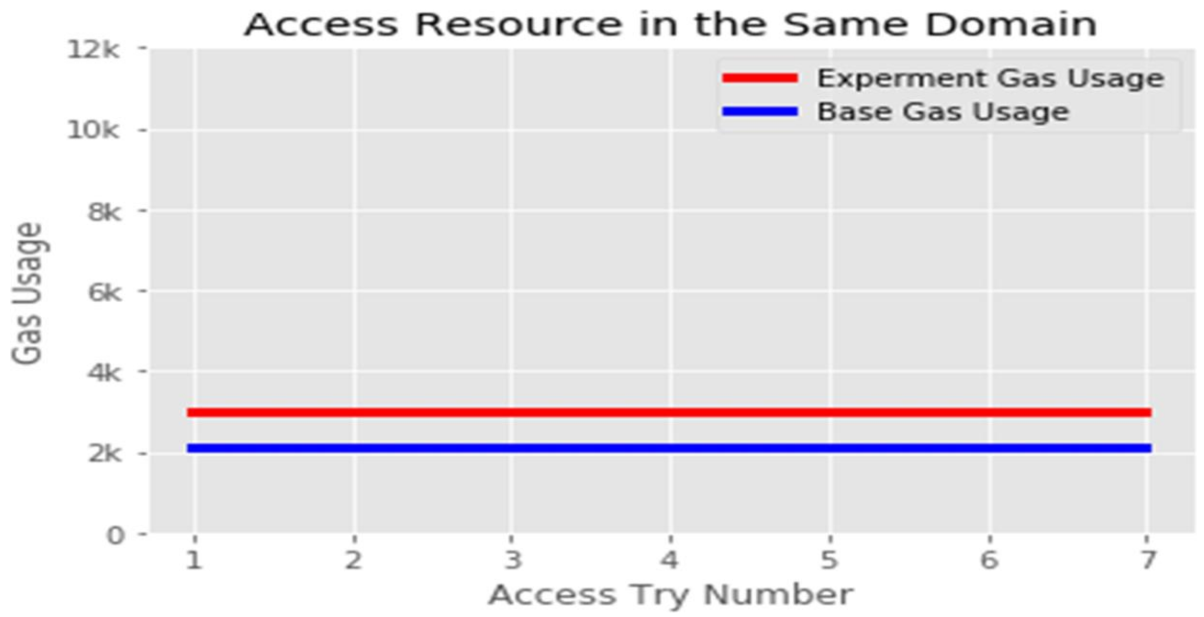


Figure 6.1: Same Domain Access Result

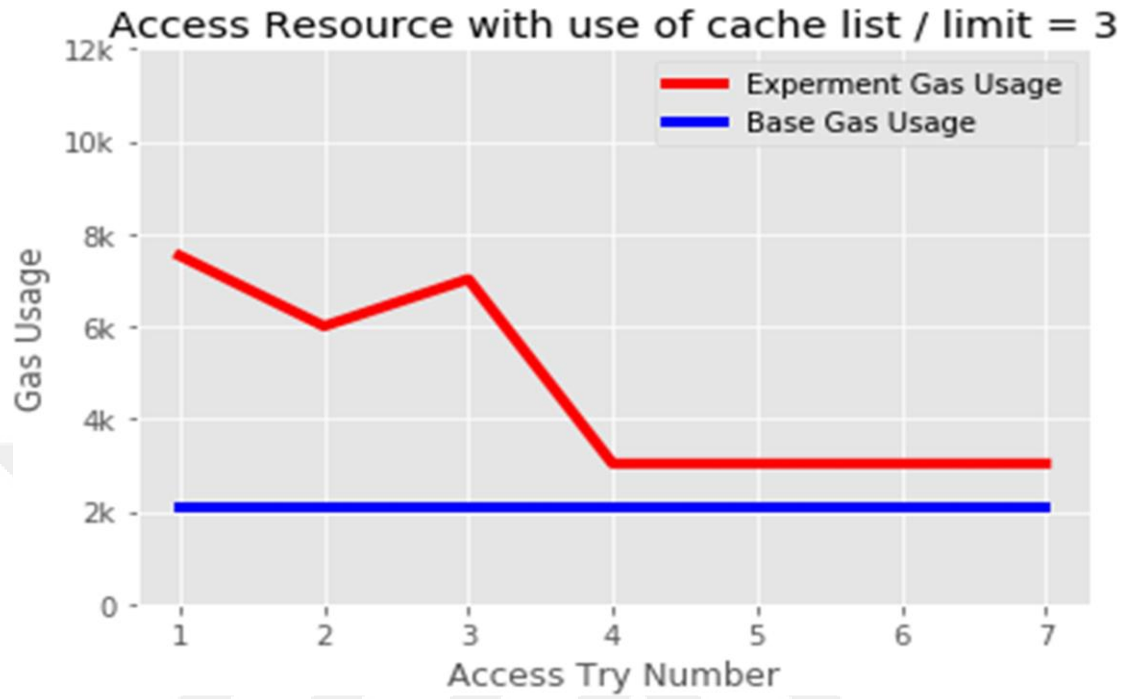


Figure 6.2: Access Control Result with Cache List

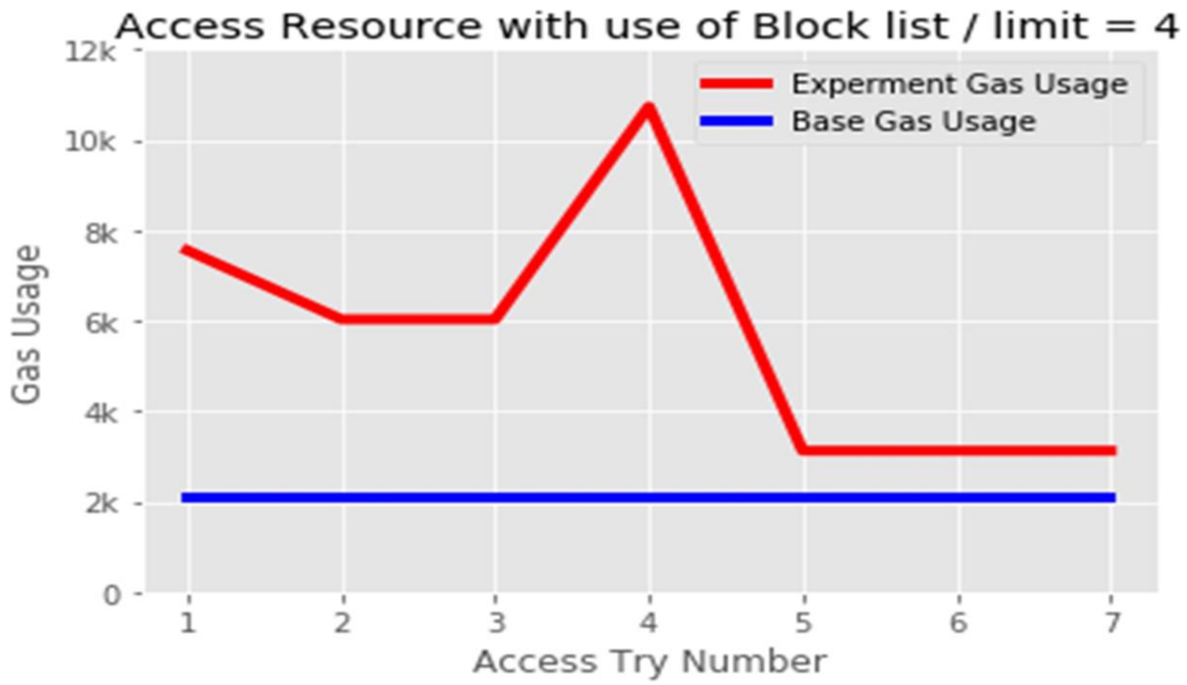


Figure 6.3: Access Control Result with Block List

6.2 DISCUSSION

6.2.1 Performance Analysis

Analysis of our domain-based system is done by checking and testing its behavior in response to different access requests. We choose these access requests to implement different situations.

First, we evaluate our system If both subject and object resources are in the same domain and the result is shown in Figure 6.1, we make this request 7 times and the gas consumed will be about (26137) which is not much greater than the base gas usage of Ethereum This gas usage is low because it is required to execute the domain local contract only. That proves our idea for creating domains for each group of devices, and it will be better for future use if the system administrator puts devices that have frequent access and data exchange between them in the same domain.

The second situation in our evaluation is to show the effectiveness of using the cache list in our system as shown in Figure 6.2. We choose the limit to be 3 in this method, and we also perform7 requests for this method. In the first request, the gas used was about (75272), and it was reduced a little in the second request. However, in the third request it rises slightly,This is because of the executions “add to cache function” in the global contract. The effect of using the cache is clear starting from the fourth request, which shows that the gas consumed is reduced about (60%) to be about (30337), so this proves our contribution in terms of lightweight.

The last situation that we check it is to verify the effect of the block list in our system as shown in Figure 6.3. We choose a method that has permission of “deny” and the limit is 4, and the request access was performed 7 times. In the first 3 requests, the gas used was about (75540–60302). In the fourth request, the gas usage increased to about 107322. That was because of the execution of “Add to block method.” Starting from the fifth request, the gas usage dropped to about 31229. That was because the method then was in the block list, and there was no need to execute other procedures in the global access control algorithm.

Integrating block chains with IOT provides many solutions, and in terms of security, the blockchain can provide the IOT environment with many facilities, such as identity, address space, integrity and reliability. Integrating the blockchain could also reduce complexity in the IOT protocol, such as simplifying the complexity of DTLS and TLS protocol needed to secure HTTP and MQTT CoAP protocols.

6.2.2 Security Analysis

Our system ensures that the data in the blockchain is immutable and can track every change that may occur to it, but the blockchain cannot handle problems related to the IOT devices themselves, such as data corruption, device failure and Sybil attacks [27].

IOT devices have a great chance to be hacked because of their limitation. However, many procedures can be adapted along with our system to override the concerns mentioned. These procedures could be firmware updates using identification techniques such as PKI and using modified hardware such as Filament [62].

Using the blockchain in our system eliminates the problem of “single point of failure " [63], so in our implementation, if a server fails, the mechanism of access control will not be affected because it is managed by the blockchain smart contract.

The denial of service problem is controlled in the blockchain by using gas limitation [39]. This provides our system with another level of security. Therefore, we made transactions for every request to handle this problem because for example, if we use a query with the blockchain, there is a great chance of encountering DOS attacks.

6.2.3 System Limitation

1. Storage Challenges: The blockchain has scalability problems because the chain is growing day by day [37] , so integration with the IOT made this problem more complex because of the IOT limitations. However, in our system, we try to override this problem by making the agent work in light mode, where storage space is miners

We also design our code to store only the necessary information about resources. For example, the cache list must not store more than one method for each Subject agent.

2. Ether Cost: At the time of writing this paper ether's price was $115.92 [64]. The cost of the system increases with the increase in the number of devices Each transaction in the system consumes gas, which is ether in another way. The lightweight feature in our system makes the gas consumption very low. This has a serious effect on the process. In addition, the communications between contracts occurs in a scenario where two resources in different domains do not charge each other. Another approach that we adapt to reduce gas consumption is the access relationship of the methods in the global contract, which we choose to be between the agent of one domain and a specific resource on the other hand, where the agent here can implement several resources and consequently one method will implement several methods.

3. Time: Mining and transaction times are one of the biggest concerns in the blockchain development [39], usually it is between 10–20 seconds, which is very high in operation related to the computer science field. This time has a great influence on the robustness and power of the blockchain. A few solutions to this problem can be applied. One of them, which we use in our implementation, is to tune the blockchain parameter to reduce the time. This solution might not work in the real-world blockchain.

6.3 CONCLUSION

In this thesis, we present a system architecture as a solution to the access control problem in the IOT environment. This system manages access control by using both centralized and decentralized management approaches. Using these approaches made our system override the limitations present in IOT devices. Domain-based system integrate blockchain with IOT to control access inside IOT networks. This thesis shows how the blockchain can be used as a decentralized aspect in any system while the server implements the centralized management aspect of this system.

Handling access control in our thesis is enforced by using an Ethereum smart contract. The smart contract can be used to store the rules and permissions that manage access requests. The domain- based system is the system where IOT devices are in multiple domains, and each domain is controlled by one local contract in the blockchain, and the connection between the devices and the blockchain is maintained by using the domain interface and the domain agent. In addition, there is a global contract that manages access requests between different domains.

To validate our work, we provide an implementation in different scenarios that our system can face in a real-life application. The analysis of the implementation result proves that the domain-based system could be magnificent and a lightweight solution for access control in IOT systems. Besides that, using the concept of cache and block lists was very effective in order to reduce the gas needed to execute access requests in the system.

Blockchain is promising technology and our thesis proves that it can be very suitable solution to access control in IOT system.

6.4 FUTURE WORK

Integrating IOT with Blockchain is a hot topic now, there is a great opportunity to go forward in this area. In the context of the thesis topic and suggested system, there are many aspects need to be searched more especially what is related to blockchain challenges related to the storage, cost and time. Domain-based system can be a base for another good system, it can develop and adopted as a magnificent solution for access control in healthcare systems or smart cities.

Develop the system algorithms and add many custom access control rules could the right step for making Domain-Based system more suitable for many applications that need good access control mechanism. Future development of suggested solution must search in IOT part of the system, because this thesis focused on the processes inside blockchain part itself. This could make this solution part of complete End-to-End platform for managing IOT networks

REFERENCES

[1] L. R.S. Schuff, D., “CENTRALIZATION VS. DECENTRALIZATION OF APPLICATION SOFTWARE,” COMMUNI- CATIONS OF THE ACM, vol. 44, 2001.

[2] M. A. Khan and K. Salah, “IoT security: Review, blockchain solutions, and open challenges,” Future Generation Computer Systems, vol. 82, pp. 395–411, 2018. [Online]. Available: 10.1016/j.future.2017.11.022

[3] A. Bahga and V. K. Madisetti, “Blockchain Platform for Industrial Internet of Things,” Journal of Software Engineering and Applications, vol. 09, no. 10, pp. 533–546, 2016. [Online]. Available: 10.4236/jsea.2016.910036

[4] “Ethereum, "Design Rationale".” [Online]. Available: https://github.com/ethereum/wiki/wiki/Design-Rationale

[5] 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, pp. 1–11, 2018. [Online]. Available: 10.1109/JIOT.2018.2847705

[6] O. Novo, “Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT,” IEEE Internet of Things Journal, vol. 5, no. 2, pp. 1184–1195, 2018. [Online]. Available: 10.1109/JIOT.2018.2812239

[7] A. Ramachandran and M. Kantarcioglu, “Using Blockchain and smart contracts for secure data provenance management,” vol. 8, 2017.

[8] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, “FairAccess: a new Blockchain-based access control framework for the Internet of Things,” Security and Communication Networks, vol. 9, no. 18, pp. 5943–5964, 2016. [Online]. Available: 10.1002/sec.1748

[9] “The Ethereum network.” [Online]. Available: http://ethdocs.org/en/latest/network

[10] A. B. M, “Internet of Things: A Hands-On Approach,” 2014.

[11] “Research & Innovation in Internet of Things.” [Online]. Available: https://ec.europa.eu/digital-single-market/en/research- innovation-IOT

[12] R. V. M. J. Rivera, “Forecast: Internet of Things — Endpoints and Associated,” 2016.

[13] “IoT Standards and Protocols.” [Online]. Available: https://www.postscapes.com/internet-of-things-protocols/.

[14] S. Lawson, “Why Internet of Things ’standards’ got more confusing in 2014.” [Online]. Available: https: //www.pcworld.com/article/2863572/iot-groups-are-like-an-orchestra-tuning-up-the-music-starts-in-2016.html

[15] “ALLIANCE FOR INTERNET OF THINGS INNOVATION Smart Mobility.” [Online]. Available: https://aioti.eu/

[16] P. Stokes, “4 Stages of IoT architecture explained in simple words.” [Online]. Available: https://medium.com/ datadriveninvestor/4-stages-of-iot-architecture-explained-in-simple-words-b2ea8b4f777f

[17] A. Reale, “A guide to Edge IoT analytics.” [Online]. Available: https://www.ibm.com/blogs/internet-of-things/edge-iot- analytics/

[18] “Internet of Things (IoT) Protocols and Connectivity Options: An Overview.” [Online]. Available: https: //www.sam-solutions.com/blog/internet-of-things-iot-protocols-and-connectivity-options-an-overview/

[19] “JSON C/C++ Library for IoT Communication.” [Online]. Available: https://realtimelogic.com/products/json/

[20] O. G.-M. M. Kirsche M. Brachmann, “Security for practical CoAP applications: Issues and solution approaches,” 0th GI/ITG KuVS Fachgespraech Sensornetze (FGSN 2011), 2011.

[21] P. B. A. L. R. T. M.A. Spirito D. Conzon, T. Bolognesi, “The VIRTUS middleware: An XMPP based architecture for secure IoT communications,” 21st International Conference on Computer Communications and Networks, ICCCN, pp. 1–6, 2012.

[22] undefined H. Kim, “Protection against packet fragmentation attacks at 6LoWPAN adaptation layer,” 2008 International Conference on Convergence and Hybrid Information Technology, pp. 796–801, 2008.

[23] K. P. K. Weekly, “Evaluating sinkhole defense techniques in RPL networks,” 20th IEEE International Conference on Network Protocols (ICNP), ICNP ’12, IEEE Computer Society, Washington, DC, USA, pp. 1–6, 2012.

[24] K. Tassin, LTE and the Internet of Things, Sequans Communications, 2014 [Online]. available :http://www.3gpp.org/news-events/3gpp-news/1607-iot).

[25] “X.509 certificate.” [Online]. Available: https://searchsecurity.techtarget.com/definition/X509-certificate.

[26] Y. Z. T. Wood W. Xu, W. Trappe, “The feasibility of launching and detecting jamming attacks in wireless networks,” Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc ’05, ACM, New York, NY, USA, pp. 46–57, 2005.

[27] W. L. X. G., “Channel-Based Detection of Sybil Attacks in Wireless Networks,” IEEE Transactions on Information Forensics and Security, vol. 4, no. 3, 2009.

[28] “OWASP, Top IoT Vulnerabilities.” [Online]. Available: https://www.owasp.org/index.php/Top_IoT_Vulnerabilitie

[29] X. D. A.V. Vasilakos J. Zhou, Z. Cao, “Security and privacy for cloud-based IoT: Challenges,” IEEE Commun. Mag, vol. 55, no. 1, pp. 26–33, 2017.

[30] N. Park and N. Kang, “Mutual authentication scheme in secure internet of things technology for comfortable lifestyle,” Sensors, vol. 6, no. 1, 2016.

[31] K.-H. K. H.F. Ahmed R. Riaz, “Security analysis survey and framework design for IP connected LoWPANs,” International Symposium on Autonomous Decentralized Systems, pp. 1–6, 2009.

[32] T. H. L. Buttyan A. Dvir, “VeRA - version number and rank authentication in RPL,” IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems, pp. 709–714, 2011.

[33] J. Granjal, E. Monteiro, and J. S. Silva, “Network-layer security for the Internet of Things using TinyOS and BLIP,” Int. J. Commun. Syst, vol. 27, no. 10, pp. 1938–1963, 2014. [Online]. Available: http://dx.doi.org/10.1002/dac.2444

[34] F. J. Ryba, M. Orlinski, M. Wählisch, C. Rossow, and T. C. Schmidt, “Amplification and DRDoS Attack Defense - A Survey and New Perspectives,” 2015.

[35] M. Brachmann, O. Garcia-Morchon, and M. Kirsche, “Security for practical CoAP applications: Issues and solution approaches,” in 10th GI/ITG KuVS Fachgespraech Sensornetze.

[36] Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System.” [Online]. Available: https://bitcoin.org/en/bitcoin-paper.

[37] Z. Xie, H. Dai, X. Chen, and H. Wang, “Blockchain challenges and opportunities: a survey,” International Journal of Web and Grid Services, vol. 14, no. 4.

[38] A. Lastovetska, “Blockchain Architecture Basics: Components, Structure, Benefits & Creation,” 2019. [Online]. Available: https://mlsdev.com/blog/156-how-to-build-your-own-blockchain-architecture

[39] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, “On blockchain and its integration with IoT. Challenges and opportunities,” Future Generation Computer Systems, vol. 88, pp. 173–190, 2018. [Online]. Available: https://doi.org/10.1016/j.future.2018.05.046

[40] B. ARVANAGHI, “Explaining the Genesis Block in Ethereum.” [Online]. Available: https://arvanaghi.com/blog/ explaining-the-genesis-block-in-ethereum/

[41] G.-T. Nguyen and K. Kim, “A Survey about Consensus Algorithms Used in Blockchain,” J Inf Process Syst, vol. 14, no. 1, pp. 101–128, 2018. [Online]. Available: https://doi.org/10.3745/JIPS.01.0024

[42] K. Christidis, M. Devetsikiotis, and K. Christidis, “SPECIAL SECTION ON THE PLETHORA OF RESEARCH IN INTERNET OF THINGS (IoT) Blockchains and Smart Contracts for the Internet of Things,” vol. 4, 2016. [Online]. Available: 10.1109/ACCESS.2016.2566339

[43] “A Next-Generation Smart Contract and Decentralized Application Platform.” [Online]. Available: :https://github.com/ ethereum/wiki/wiki/White-Paper

[44] “Solidity.” [Online]. Available: https://solidity.readthedocs.io/en/v0.5.3

[45] Z. Hintzman, “Comparing Blockchain Implementations.” A Technical Paper prepared for SCTE/ISBE [Online]. Available: https://www.nctatechnicalpapers.com/Paper/2017/2017-comparing-blockchain-implementations/download.

[46] F. Gadaleta, “This is how Ethereum works.” [Online]. Available: https://medium.com/fitchain/this-is-how-ethereum- works-60f37abd5ef5

[47] V. Buterin, “Merkling in Ethereum.” [Online]. Available: https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/

[48] “R.P.C. JSON.” [Online]. Available: https://github.com/ethereum/wiki/wiki/JSON-RPC.

[49] “What is the Unix time stamp?” [Online]. Available: https://www.unixtimestamp.com/index.php.

[50] “GeTH.” [Online]. Available: https://github.com/ethereum/go-ethereum/wiki/GeTH.

[51] “JavaScript API.” [Online]. Available: https://github.com/ethereum/wiki/wiki/JavaScript-API#web3ethfilter

[52] “web3.js - Ethereum JavaScript API.” [Online]. Available: https://web3js.readthedocs.io/en/1.0/

[53] “WebSocketClient.” [Online]. Available https://github.com/theturtle32/WebSocketNode/blob/master/docs/WebSocketClient.md#connectrequesturl-requestedprotocols-origin-headers-requestoptions

[54] B. ARVANAGHI, “Explaining the Genesis Block in Ethereum.” [Online]. Available: https://arvanaghi.com/blog/ explaining-the-genesis-block-in-ethereum/

[55] “Remix, Ethereum-IDE.” [Online]. Available: https://remix.readthedocs.io/en/latest/

[56] “Contract ABI Specification.” [Online]. Available: https://solidity.readthedocs.io/en/develop/abi-spec.html

[57] “Download GeTH.” [Online]. Available: https://GeTH.ethereum.org/downloads/

[58] “Command Line Options.” [Online]. Available: https://github.com/ethereum/go-ethereum/wiki/Command-Line-Options

[59] S. Gueiros, “The GeTH’s saga: setting up Ethereum private network on windows.” [Online]. Available: https://medium.com/ @solangegueiros/https-medium-com-solangegueiros-setting-up-ethereum-private-network-on-windows-a72ec59f2198

[60] “montdher10/Domain_based-system.” [Online]. Available: https://github.com/montdher10/Domain_based-system

[61] “Node.js.” [Online]. Available: https://nodejs.org/en/download/

[62] “Filament.” [Online]. Available: https://filament.com/ .

[63] R. . Mauri, “Three features of blockchain that help prevent fraud.” [Online]. Available: https://www.ibm.com/blogs/ blockchain/2017/09/three-features-of-blockchain .

[64] “Ethereum Price.” [Online]. Available: https://www.coindesk.com/price/ethereum

APPENDIX A

GENSIS FILE EXPLANATION

1. Config: this work as the main setting to describe the blockchain network, it must not be between (0-3) because this value used in real networks.

2. ChainId: This value is a unique value for running the private network.

3. HomesteadBlock: this value is always 0 for private networks, in real network it could be hold another value.

4. Eip155Block & Eip158Block: these values are used in hard-forking, for private network its 0.

5. Difficulty: this value is used as a scalar of the mining Target, which can be calculated from the previous block's difficulty level and the timestamp.

6. GasLimit: this value is used to limit of Gas expenditure per block.

7. Alloc & Balance: list of pre-filled Accounts.