Blockchain and IOT Security

MSc Research Project
Programme: MSc in CyberSecurity

Jake Murphy
Student ID: x16442804

School of Computing
National College of Ireland

Supervisor: Imran Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Jake Murphy………………………………………………………………………………………		
Student ID:	X16642804…………………………………………………………………………………..……		
Programme:	MSc in CyberSecurity – Full Time………	**Year:**	1……………..
Module:	MSc Research Project……………………………………………………….………		
Supervisor:	Imran Khan…………………………………………………………….………		
Submission Due Date:	16/08/2021…………………………………………………………….………		
Project Title:	Blockchain and IOT Security – Can an Ethereum Blockchain be the solution for securing IOT devices.		
Word Count:	5,929………………………	**Page Count:**	14 + 3(Appendix) : 17 total

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Jake Murphy………………………………………………………………………………

Date: 16/08/2021………………………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□ yes
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□ yes
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□ yes

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Contents

Abstract

From purposing the idea of delving deep into the world of Internet of things (IOT) and blockchain (BC) security and trying to determine if blockchain could be a solution for millions of IOT devices currently in circulation that have little to no security implementation that may contain sensitive user data. IOT devices in 2021, are now imbedded into daily life from Smart bulbs to commercial scale operations. The following research has established using a blockchain that enables the use of smart contracts (SC) such as Ethereum (ETH) would allow developers to implement security rules or policies that were never intended to be implemented into mass produced IOT devices.

1 Introduction

Have carried out intensive research over the past few months to establish if blockchain (BC) can secure Internet of things (IOT) devices. Primarily focusing on the Ethereum (ETH) blockchain which to the general public is seen as a traded currency or digital asset, its actual use case is providing a network/platform for building decentralized applications. From using the native ether token to developers building out custom tokens for particular use cases such as trading digital art (NFT - Non fungible tokens) protecting its intellectual property to full scale Decentralized Web Applications (DAPP).

1.1 Blockchain

BC Resolves problems with the internet that cant be fixed with its current structure such as preventing illegal activities and achieving complete privacy. Developers in the BC space are now developing methods of interconnecting multiple types of BC and networks possibly creating a new era of the internet. Polkadot [1] a BC designed and created by one of the founders of Ethereum created a BC protocol that allows developers to scale and securely interconnect multiple BC, promising to be on the forefront of web 3.0 [2]. Web 3.0 is the next generation of internet and is composed of several technologies BC being the main transaction mechanism. It proposes an internet that is readable by machines not only humans Apples Siri is an example of a Web 3.0 application that scans the internet looking for information but is limited by the architecture of the current web. There are multiple types of BC that depending on the use case will determine the type needed and will also have to choose a consensus algorithm which is a method in which the BC transaction is validated. Public being the most renowned such as being completely transparent and public Bitcoin being the most famous public BC. Private which was the BC of choice as goal was to create a secure and private interactions between an IOT device and BC. Then Permissioned style BC is essentially a semi private BC where the source code of the BC is not public knowledge and the inner works of the BC mechanism is not open to the public Ripple XRP being a famous semi-public BC that is backed by banks and is infamous for being keeping there BC technology private. Most important factors of a BC is its decentralization and immutable records. The distributed databases on the network also known as

Nodes store the entire history transaction of the network. The distributed ledger network (DLT) cannot be taken down as there is no central authority.

1.2 Ethereum

Ethereum (ETH) has become a gold standard for building out decentralised finance (De-Fi). Its ability to enforce SCs on the BC essentially allows developers to create a decentralized backend of an application. This had become the main focus for research in securing an IOT device with BC, proving that a device that was never intended to have any malware detection or security policies due to cheap quality and poor architecture can with ETH. ETH 1.0 which is still in place as of mid-2021 which uses the traditional PoW consensus algorithm to validate the blockchain. Vitalik Buterin is plaining to transform the entire public mainnet to PoS (ETH 2.0) [3] to improve scalability, efficiency and security. PoS in ETH is still in experimental stages and is still very young in development leading to high risk of vulnerabilities. When fully deployed could be an option for securing IOT devices due to the minimal CPU resources required and enhanced security. PoW although inefficient, ensures a secure decentralized environment would not be the most effective option for connecting IOT and BC together for security. There is a new concept amongst ETH configurations, Proof of Authority (PoA) consensus which was the choice for my implementation as only communicates with authenticated nodes within a custom genesis block (Every BCs starting point). This consensus is highly effective in private blockchains as only authorised nodes can track and generate new blocks while not requiring a particular amount of ether to carry out a transaction.

1.3 BC & IOT Security Issues Found

Not only BC and IOT, everything in tech has an achilles heel that could be exploited unless addressed in the initial development and construction of a device. BC can provide security and is immune to most common attacks due to the drastically different approach to applications and networks but is still prone to 51% attack which is a node taking over 51% of the BCs computational power therefore taken control of the chain. The ETH PoA implementation only allows approved nodes into the network also nullifying the PoW approach of mining transactions mitigating this attack. An eclipse attack even harder to successfully to carry out but is a foreign node on the network which alters transactions which in a PoA again is nearly impossible as to insert a ‘Bad’ node into the network would have to insert the malicious nodes enode address into the Authority node (Seals/Mines) each transactions which would require the attacker access to the Authority node and password as well as hiding altered transactions from the bootnode

1.4 Merging IOT and blockchain

IOT is becoming more imbedded into daily life as time goes on but while it becomes more adopted in all aspects of life manufacturers focus on profit rather than a quality product. As a result of this security often takes a back seat leading to sensitive user data that could potentially be a target in a cyber-attack. BC in theory resolves a lot of common IOT security issues such as immutable admin login credentials, no encryption or no security policies in place. An ETH BC for IOT would be able to apply security logic to an IOT device through a SC which a BC validates when how an IOT device should operate. One step further is build a DAPP that provides a user Graphical User Interface (GUI) for an IOT device that does not carry out a function on the application until the BC approves its request. Providing usability and a layer of security to an IOT device that was once vulnerable. The ETH BC implemented ensures a raspberry pi camera only operates when the given smart contract in the BC is fulfilled providing security to a camera that has no stock security implementation like most cheap mass produced IOT devices.

2 Research Question

Can an Ethereum blockchain be the solution for securing IOT devices from cyber-attacks: Throughout the implementation have developed a deep understanding of the platform and even with the flexibility of ETH being able to select different types architectures such as public and private BCs with a choice of consensus algorithms depending on the use case still ran into connectivity issues such as interconnecting nodes within the private network as ETHs developers documentation provides a brief outline of what is needed which led to a series trial and error attempts to refine the BC to the implantation spec designed.

2.1 Was it worth investigating in terms of security

As a single researcher carrying out this investigation have only came across even more possibilities in terms of security. It makes sense the huge financial institutions making the leap into BC and De-Fi in recent times as gives the ability of promise a level of security and transparency that businesses could not promise the end user before. The current implementation makes use of ETHs native token ether but it is serious potential for developing custom layer 2 tokens on top of ETHs network focused around security enforcing ERC-20 around a business model. ERC-1400 [4] which is a security standard which custom layer 2 tokens are built from emphasizing secure transactions.

2.2 Has the research done contributed to current research

The research done has contributed to the world of cybersecurity and BC as prior to this investigation most research into BC in general is from the perspective of finance or software development such as real world use case and adoption seeing if BC can be successful in a real world setting. There is a small subset BC research touching on security of smart homes and cities which was enough to create my own architecture and prove the purposed hypothesis. Securing mass produced IOT devices from a cyber attack mitigating an attack and guaranteeing data integrity.

3 Literature Review

3.1 Ethereum Research

3.1.1 Designing a secure ETH IOT network/platform

IOT is playing a vital role in all domains such as tourism, agriculture, healthcare, transportation and education. It is becoming apart of how tasks are carried out in the modern age yet security in these devices are obsolete and are open to be exploited some research has been carried out into in securing a Smart Home Systems [5] with ETH which would handle access control policies, data storage and data flow management. There is multiple centralized approaches to a Smart Home System such as Samsung Smart Things, Google Brillo, Amazon Alexa, Apple HomeKit but all are vulnerable to conventional cyberattacks. Vulnerable to phishing campaigns tricking users to handover there admin details, installing malicious malware leaving a backdoor into the network or causing damage and carrying out a Distributed Denial of Service Attack. Implementing an ETH BC would remove the central authority in the network giving an attack no obvious target as they would in a conventional architecture. The Consensus algorithm chosen was a PoS model due to the efficient resource management compared to the PoW model which for low powered devices will not work or be reliable due to the lack of computational power. Similar predicament during the research for designing a secure BC as PoW may be more secure setup on a large scale but is not realistic in a private BC due to minimal nodes and resources required. There also has been purposed designs for IOT connected cities [6] that make use of an ETH BC to create the foundation for a distributed trustless system that would be fault tolerant and resistant to distributed denial of service attacks. It implements SWARM (decentralized data storage platform) to further the usability of the BC and provide high availability of data to the network.

3.1.2 Ethereum Security

There are lots of benefits to an ETH BC when implemented correctly but has vulnerabilities that must be addressed when designing an ETH BC for IOT devices. This was taken into consideration and could leave the BC wide open to an attack and being exploited [7]. **Reentrancy** could be used to take advantage of an ETH BC by disrupting the interactions between smart contracts on the network. Example would be SC(A) handing over control to SC(B) in theory call back SC(A) before the handover transaction is completed possibly retrieving multiple refunds and empty the balance on SC(A). Through secure programming practices can make use of Check-Effects-Interactions to prevent call backs during a smart contract hand over. **Transaction origin** (tx.origin) is the BC identifying who carried out a transaction on a SC. The use of tx.origin for authorization can be used to spoof a value in a contract and advised to avoid and be used to leverage privileges of a contract. **External Calls** depending on the BC use case may call for calls outside of the BC e.g. API call but is advised not to as could be a point of entry for an attacker. There has been examples of setting of a secure IOT data management system with ETH [8]. Makes use Raspberry pi's for gathering sensor data and used SWARM and or IPFS to encrypt and securely store the sensor data on the private ETH BC. Instead of PoA [9] concept on configuring authorized nodes within the network through the genesis block uses hardware such as Trusted Platform Module (TPM) which when connected to the Pi appoints an ID to the device which will only communicate with the other Pi's with a valid TPM which during this implementation was achieved at a software level rather than hardware. Choosing a TPM method does not limit this use case to PoA but now can use all transaction algorithms which in this research fulfilled the researchers hypothesis.

3.1.3 Overview

The ETH BC has potential to could be used as a security framework in the future for securing a mass connection of IOT devices providing immutable data, complete history of a devices transaction and a security layer through SC's that was never intended to be there. Multiple types, brands and quality sensors once connected to BC can be secured the universal security policies applied through a single or multiple SC's. Evident and proven that an ETH BC can be used to solve security issues with IOT but again, still in the early stages in terms of the overall maturity of the ETH platform.

3.2 IOT Research

3.2.1 IOT Design and Threats

Few interesting perspectives on the topic of securing IOT Device mostly securing the network around the IOT and not the device itself e.g. Strict firewall policies. Still leaves a possible weak point in the any networks structure. The universal structure for an IOT device is Application layer, Network Layer and Physical Layer [10]. The simplistic layer design is effective but no security takes at any stage. **Physical Layer:** is the bottom layer of the device and is responsible for interconnecting devices and device identification and a discovery service to look for new devices. For the Raspberry pi 4 would use its onboard WIFI connection Ethernet port and Bluetooth would be considered as part of the physical layer of the device. The Physical layer also requires a unique tag to improve interactions incase connection is lost and can connect back automatically. No precaution is taken before connection such as device type, connecting from an external network or malware detection. **Network layer:** is the layer where the device establishes network interfaces, communication channels, network management and intelligent processing, mainly responsible for the connectivity of all the devices in the IOT system through standard communication protocols. Most common probably being MQTT widely used transmission protocol for IOT devices. **Application Layer:** Service oriented layer that determines the devices connected and has the ability to store data to database on the IOT device. Facilitates the function of being able to communicate outside the IOT device and use of different applications e.g. Smart Home.

3.2.2 IOT and SC

BC and IOT in theory goes hand in hand, from research have came across working examples [11] of ETH BCs and IOT that where successful. One of the main reasons for success is SCs giving developers and engineers the ability to apply security through a BC. The following research [12] shows a possible hierarchy through SC's on the chain such as establishing different types of users from owner - owner of BC/SC, administrators – same rights as the owner but cannot add or remove other admins, Users – basic usability privileges. Also provides a secure way of adding authorized devices to the BC on the fly. Through another SC creating a profile of a 'Gateway' which when adding a new device must fulfill the following requirements of having a valid mac address, status, type and universal unique identifier. SC allow developers to create a decentralized hierarchy to IOT devices adding multiple layers of security from potential cyber-attacks. Also discusses the idea of whitelisting and blacklisting user IP's through a SC similar to how network engineers can block IP's on firewalls with rules and polices.

3.2.3 Overview

IOT devices naturally have simplistic architecture as it was easy for manufacturers to mass produce and turn a high profit. Decades later with no change to design, didn't realize the magnitude of adoption in all domains [13] which has resulted in millions of devices currently in circulation vulnerable to be exploited.

3.3 Blockchain

3.3.1 Implementing BC and challenges

There are many reasons why an organization may hold off on implementing a BC into their internal network or as a service as there is many factors when creating and executing a BC. Lack of adoption and understanding could deter clients or a businesses entity from using BC as there is a steep learning curve to fully understand the BC mechanism and its potential compared to other methods. Currently a huge shortage in BC skills and knowledge which may result being expensive for a businesses to develop a DAPP or private BC. Motivating financial institutes to migrate legacy systems to a modern technology stack so that BC could be implemented into the network [14]. Finance institutes avoid updating legacy systems to avoid high costs and downtime which would affect there end users experience. There is also issues to consider [15] at a BC level and if a BC is put into production to keep up maintenance of the chain to avoid dependency and framework vulnerabilities to prevent attackers taken advantage of outdated libraries.

3.3.2 Integrity of BC

One of the main selling points of BC is integrity and that all transactions are on record and immutable. Which could be a selling point for industries which have been in the dark for decades on their methods and businesses procedures which can result in the consumer not wanting there products. The electronic industry [16] could use BC to revolutionize there transparency to the world showing how products developed from start to finish giving the customer a better understanding and the amount of labor that has gone into creating the product. Also in the electronics industry adds a level of trust [17] as manufacturers may develop products using multiple factories before getting the final product so will benefit manufacturers as they can not be exploited on price and location of parts as the blockchain will record each stage in the development process therefore benefiting the not only the end user but the manufacturers themselves. Logistics could benefit greatly from BC as will give the ability to track goods from when they are created all the way to the end consumer.

3.3.3 Overview

The use of a BC can solve transparency issues while simultaneously improving security but clearly has its limitations in terms rapid application development that you may be able to do a conventional JavaScript web stack as there is a clear shortage in BC developers and engineers. In time as technologies advances and BC becomes more refined with the move to a PoS modal will encourage organizations embracing BC.

3.4 CyberSecurity for Blockchain and IOT

3.4.1 Cybersecurity in BC IOT

Most research in terms of BC and IOT is minimal and most research comes from other perspectives such as software development or briefly touching on the topic of cyber. So far, have in terms of attacking a BC would have to access the BC type first than from there could evaluate an attack. The PoA BC type is the most effective for an private network of IOT devices as requires minimal resources and only interacts with authorized nodes declaring it very awkward for an attacker to attack. There still is human errors [18] which can lead to a cyber attack on a BC such a mismanagement of cryptographic keys e.g. wallet address, enode addresses or bootnode keys which could be used to gain remote access to a node with enough recon of BC prior to an attack. There is still the potential for bug in the code [19] that hasn't been found with the BC or SC. Then in the future would have to consider the implications of Quantum computing when super these super computers will be able to comprise most if not all cryptographic algorithms in a short period of time. There's the standard BC attacks such as the 51% attack which is a malicious node taken up most of the computational power of the chain resulting in altering and intercepting transactions.

3.4.2 Overview

As for IOT and BC security cybersecurity is still a new concept all the research thus far only touches on security which all seem to have very similar results for finding weaknesses, vulnerabilities and possible solutions. From all the research gathered has throughout the investigation of this topic has provided me a set of guidelines to go froward to design and develop a secure ETH BC for IOT devices.

3.5 Reference summary

Ref	Characteristics
5	Similar to the implementation but focus's on PoS and PoW
6	Connecting a Smart City together using an ETH BC – Software focused
7	Focuses solely on secure SC design avoiding design values in ETH
8	Securing an IOT device using PoW and TPM module
9	Implementing private ETH BC in a Logistic tracking system
10	IOT vulnerabilities and common device Architecture that results in poor security
11	Discusses all the industries that BC and IOT could have huge impact in
12	Using smart contracts to create security rules and polices in a private network
13	Possible future where everything will be connected through IOT devices
14	Dives deep into limitations of using an ETH BC communications for DAPP's
15	Possible cyberattacks that could carried out on BC if not addressed in development
16	BC providing transparency in the Electronic manufacturing industry.
17	BC providing trust in a network infrastructure
18	BC although has improved securities will be a target in the future for black hats.
19	Offensive and Defensive strategies for attacking a BC

4 Research Methodology

The potential of combining IOT and BC could play significant part in society in the near future securing vulnerable devices that may contain sensitive data [5] [6]. These examples where able to design and successfully implement a working BC that secured IOT devices on both a small and large scale. Both research projects took advantage of Ethereum private network. Which is a private BC that is contained to an environment e.g. Smart Home to carry out safe transaction on the network.

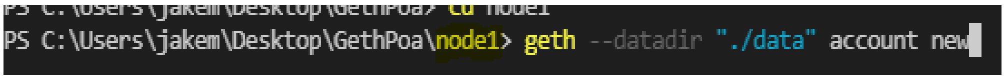
Although possible to connect IOT devices to the ETH mainnet would be completely transparent to the network and only works in some particular uses cases e.g. Decentralized Finance.

5 Design specifications and Implementation

To successfully implement an IOT BC the initial design is crucial for security as ETH can be configured in many formats such as public, private combined with either a PoW, PoS or PoA consensus algorithm. ETH also supports many programming languages to promote diversity and development on the platform but for this implementation will use Go-Ethereum (GETH) for creating the private BC as developed with Golang which is a modern programming language with built in concurrency aiding in the architectures overall integrity. It prevents Memory allocation type attacks e.g. Buffer Overflow which languages like C and C++ are susceptible to if a developer does not properly allocate memory in the development stages.

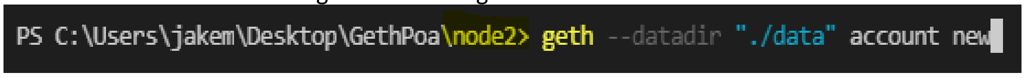
6 Ethereum – GETH

When the GETH client is installed on the operating system provides a command line interface (CLI) for creating and running a full/light ETH node simultaneously create a network, transfer funds between accounts and mine ETH's native ether token. GETH client provides a tool called Puppeth which simplifies the creation of the genesis block and getting stats of the BC performance. To run GETH on a machine must have installed Golang and GETH from the official sites to run ETH locally without problems [Appendix 1]. The ETH nodes running in this implementation will be running GETH 1.10.6 to avoid different versions of nodes often causing ETH nodes not to synchronize properly on the chain causing inconsistency's on the transaction history also slowing down the network. Need to create an account on each node also known as creating a wallet. These accounts are assigned a private-public key that are required to connect or interact with any ETH BC. Created a txt file within each node that contains the private key(password – pw.txt) which is used to sign transactions and verify itself on the network.
Node1 Public Address: 0x2Ca6b5E088B114ce79d0Bb362A5905C47344e30d
Node2 Public Address: 0xBBE0c65700C752ab72811784119E9F91C206C7C5



```
PS C:\Users\jakem\Desktop\GethPoa> cd node1
PS C:\Users\jakem\Desktop\GethPoa\node1> geth --datadir "./data" account new
```

Figure 1: Creating an Account on node1



```
PS C:\Users\jakem\Desktop\GethPoa\node2> geth --datadir "./data" account new
```

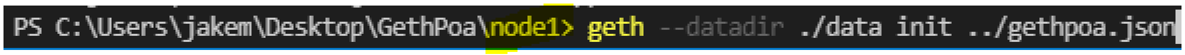
Figure 2: Creating an Account on node2

6.1 Proof of Authority (PoA)

To create an ETH PoA network, I used Puppeths CLI to create a genesis block within the 'GethPoa" directory which contains all the ETH nodes during this implementation [Appendix 2-9]. The Genesis block is the first block in any blockchain not just ETH and is often referred to as block zero. The network name is 'gethpoa' which in the project is gethpoa.json. Puppeth also allowed to select the clique consensus engine also known as PoA. The default time for a block to be mined is 15 seconds but gethpoa will be 10 as it is a private network validating the BC more often promising better security. Node1 has been appointed to seal transactions in the PoA network(Mine blocks). Both Nodes will be prefunded as this BC structure will not rely on ether to carry out functions or transactions. Then the Genesis block was assigned a network ID which will used by nodes to connect also to distinguish between other private BC that may be running.

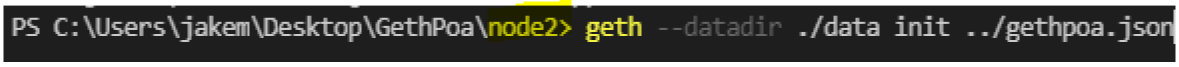
6.1.1 Ininitalize Nodes to Genesis Block

Now that the a genesis block is created needed to initialize all nodes in the PoA with the same genesis block to make sure each node is set up with the same rules and making it near impossible for an attacker to add malicious node to the network e.g. Eclipse Attack.



```
PS C:\Users\jakem\Desktop\GethPoa\node1> geth --datadir ./data init ../gethpoa.json
```

Figure 3: Initalizing node1 to the gethpoa.json (Genesis Block)

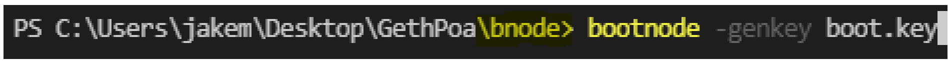


```
PS C:\Users\jakem\Desktop\GethPoa\node2> geth --datadir ./data init ../gethpoa.json
```

Figure 4: Initalizing node2 to the gethpoa.json (Genesis Block)

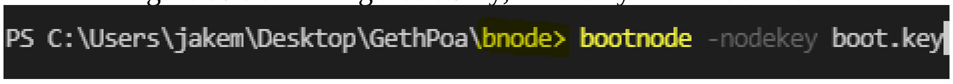
6.1.2 Creating the Bootnode

The Bootnode is for helping nodes discover each other on the network – ETH: peer to peer network. For this implementation will use localhost -127.0.0.1:30301 as meeting point for all the nodes to discover and connect to each other. When we start the bootnode with the newly generated boot.Key will get an enode address which will be needed for nodes in the network to identify the bootnode.
Bootnode enode:
enode://4c52e5ab6b8a82e4908b16f2aecdb11ab6028382ba5f0bdb9313c502507dae673bf86f124d133a aabc60276e2648b2b6da59dcc63a7a733c9c82d689e76c392b@127.0.0.1:0?discport=30301



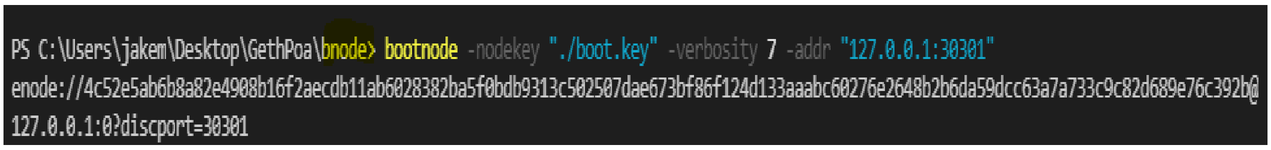
```
PS C:\Users\jakem\Desktop\GethPoa\bnode> bootnode -genkey boot.key
```

Figure 5: Generating a boot.key, necessary for the bootnode



```
PS C:\Users\jakem\Desktop\GethPoa\bnode> bootnode -nodekey boot.key
```

Figure 6: Starting a PoA network with the bootkey created



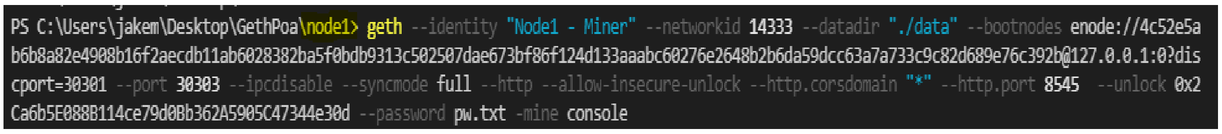
```
PS C:\Users\jakem\Desktop\GethPoa\bnode> bootnode -nodekey "./boot.key" -verbosity 7 -addr "127.0.0.1:30301"
enode://4c52e5ab6b8a82e4908b16f2aecdb11ab6028382ba5f0bdb9313c502507dae673bf86f124d133aaabc60276e2648b2b6da59dcc63a7a733c9c82d689e76c392b@
127.0.0.1:0?discport=30301
```

Figure 7: Starting the bootnode generating the enode address

6.2 Connecting the Nodes Together

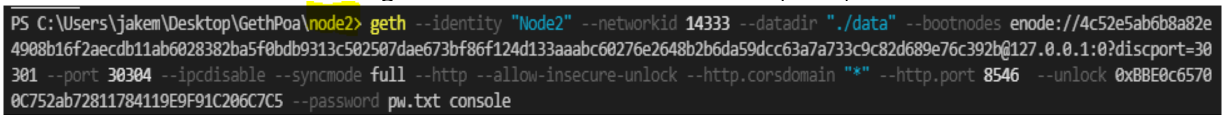
In each node directory will create two large command which will carry out the following protocols which I choose from the GETH documentation. GETH provides hundreds of variations of a node depending on the use case.

--identity	: Adding a custom name for the node – Used to Label the miner node in transactions.
--networkid	: Contecting to Network ID set in the gethpoa.json genesis block – 14333.
--datadir	: Data directory for blockchain data and keystore file.
--bootnodes	: Tells the node where to locate the Bootnode – then from there sell all the other nodes.
--port	: The port the node operates from each node including the bootnode will use different nodes
--ipcdisable	: Disable the IPC-RPC server – not nessessary but could be a point of remote access.
--syncmode full	: Helps preventing the error discarded bad propagated blocks.
--http	: Enable the IPC-RPC server – connect to node – could be avoided for secuirty.
--allow-insecure-unlock	: allowing unsecure unlocking when a account is exposed over http.
--http.corsdomain	: Domains that are allowed to accept cross orgins request (ETH in AWS).
--http.port	: The port HTTP – RPC server is listening on.
--unlock	: The account address from the node you want to use.
--password	: The password of that account – private key.
-mine	: Settomg a node to mine transactions.
Console	: Logging all events in the terminal in real time.



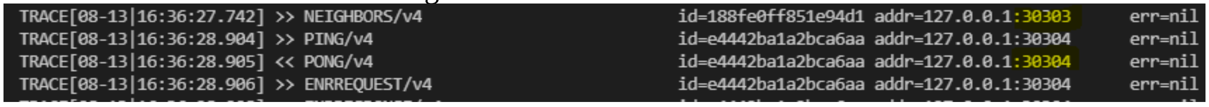
```
PS C:\Users\jakem\Desktop\GethPoa\node1> geth --identity "Node1 - Miner" --networkid 14333 --datadir "./data" --bootnodes enode://4c52e5a
b6b8a82e4908b16f2aecdb11ab6028382ba5f0bdb9313c502507dae673bf86f124d133aaabc60276e2648b2b6da59dcc63a7a733c9c82d689e76c392b@127.0.0.1:0?dis
cport=30301 --port 30303 --ipcdisable --syncmode full --http --allow-insecure-unlock --http.corsdomain "*" --http.port 8545  --unlock 0x2
Ca6b5E088B114ce79d0Bb362A5905C47344e30d --password pw.txt -mine console
```

Figure 8: Node1 Server Command (mine)



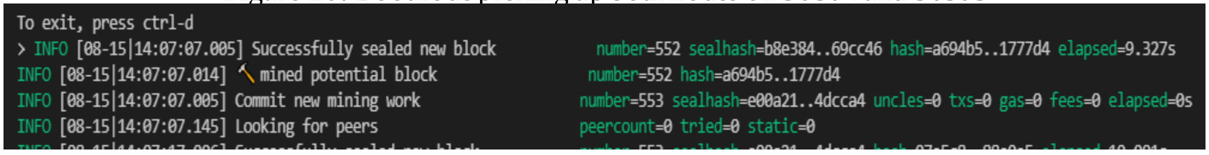
```
PS C:\Users\jakem\Desktop\GethPoa\node2> geth --identity "Node2" --networkid 14333 --datadir "./data" --bootnodes enode://4c52e5ab6b8a82e
4908b16f2aecdb11ab6028382ba5f0bdb9313c502507dae673bf86f124d133aaabc60276e2648b2b6da59dcc63a7a733c9c82d689e76c392b@127.0.0.1:0?discport=30
301 --port 30304 --ipcdisable --syncmode full --http --allow-insecure-unlock --http.corsdomain "*" --http.port 8546  --unlock 0xBBE0c6570
0C752ab72811784119E9F91C206C7C5 --password pw.txt console
```

Figure 9: Node2 Server Command



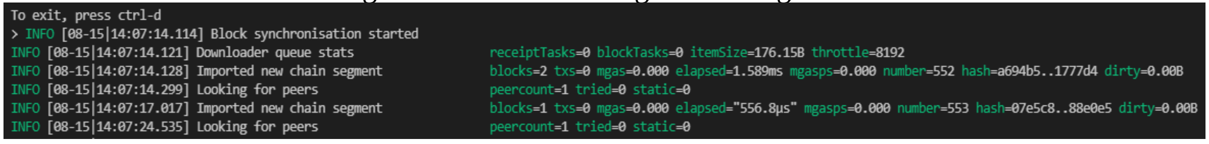
```
TRACE[08-13|16:36:27.742] >> NEIGHBORS/v4                         id=188fe0ff851e94d1 addr=127.0.0.1:30303      err=nil
TRACE[08-13|16:36:28.904] >> PING/v4                              id=e4442ba1a2bca6aa addr=127.0.0.1:30304      err=nil
TRACE[08-13|16:36:28.905] << PONG/v4                              id=e4442ba1a2bca6aa addr=127.0.0.1:30304      err=nil
TRACE[08-13|16:36:28.906] >> ENRREQUEST/v4                        id=e4442ba1a2bca6aa addr=127.0.0.1:30304      err=nil
```

Figure 10: Bootnode picking up both nodes on 30304 and 30303



```
To exit, press ctrl-d
> INFO [08-15|14:07:07.005] Successfully sealed new block          number=552 sealhash=b8e384..69cc46 hash=a694b5..1777d4 elapsed=9.327s
INFO [08-15|14:07:07.014] 🔨 mined potential block               number=552 hash=a694b5..1777d4
INFO [08-15|14:07:07.005] Commit new mining work               number=553 sealhash=e00a21..4dcca4 uncles=0 txs=0 gas=0 fees=0 elapsed=0s
INFO [08-15|14:07:07.145] Looking for peers                    peercount=0 tried=0 static=0
```

Figure 11: Node1 running and mining blocks

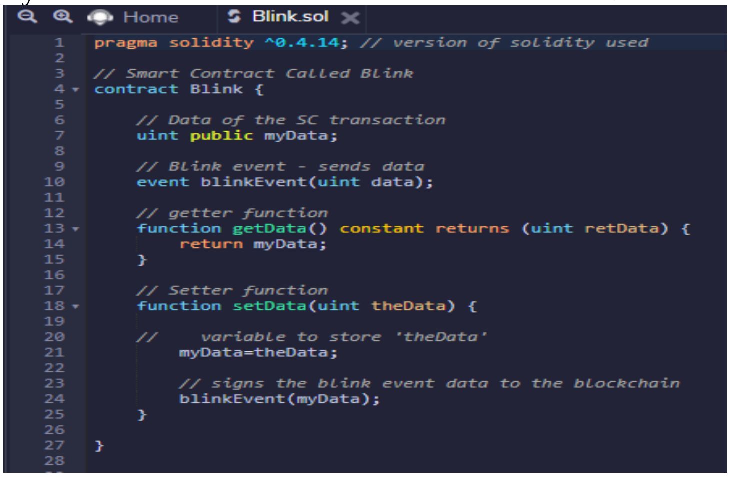


```
To exit, press ctrl-d
> INFO [08-15|14:07:14.114] Block synchronisation started
INFO [08-15|14:07:14.121] Downloader queue stats                   receiptTasks=0 blockTasks=0 itemSize=176.15B throttle=8192
INFO [08-15|14:07:14.128] Imported new chain segment               blocks=2 txs=0 mgas=0.000 elapsed=1.589ms mgasps=0.000 number=552 hash=a694b5..1777d4 dirty=0.00B
INFO [08-15|14:07:14.299] Looking for peers                        peercount=1 tried=0 static=0
INFO [08-15|14:07:17.017] Imported new chain segment               blocks=1 txs=0 mgas=0.000 elapsed="556.8µs" mgasps=0.000 number=553 hash=07e5c8..88e0e5 dirty=0.00B
INFO [08-15|14:07:24.535] Looking for peers                        peercount=1 tried=0 static=0
```

Figure 12: Node2 running with node1 keeping track of transactions

6.3 Smart Contract

Using the REMIX ETH editor, allowed myself to develop SC locally or on the online editor which then could be combined with Meta Mask to carry out transactions on the BC. Meta Mask is a cryptocurrency software that stores crypto wallets and allows users to interact a ETH BC through a browser. As it is a PoA BC no ether is needed but is used to submit a SC onto the BC. The contract address is hard coded into the web application improving security and preventing an attacker if injecting a malicious SC. The SC in this implementation writes blink event data to the chain ensuring the status and integrity of the IOT device and the correct SC was mined.

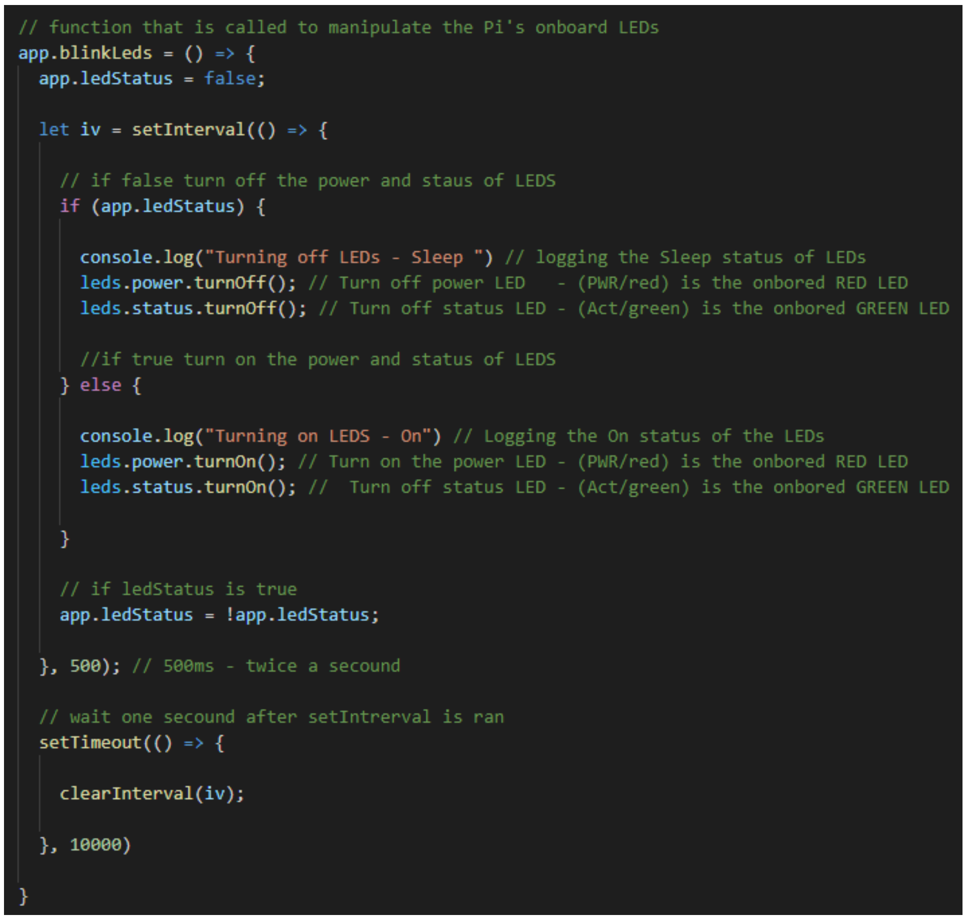


```
pragma solidity ^0.4.14; // version of solidity used

// Smart Contract Called Blink
contract Blink {

    // Data of the SC transaction
    uint public myData;

    // Blink event - sends data
    event blinkEvent(uint data);

    // getter function
    function getData() constant returns (uint retData) {
        return myData;
    }

    // Setter function
    function setData(uint theData) {

    //    variable to store 'theData'
        myData=theData;

        // signs the blink event data to the blockchain
        blinkEvent(myData);
    }

}
```

Figure 12: Solidty Contract

6.4 IOT Device

The IOT device used in this implementation initially was raspberry pi camera v2. It is a 8 megapixel camera that can also record a 1080p30 video making it a perfect IOT device for a security camera setup for a home or building. The pi camera has a kit of features in terms of camera quality and video recording but has no security features like most IOT devices in circulation around the world. Ran into server configuration issues between a node app and pi camera. On the raspberry pi itself there is onboard LEDs signifying the status of the Pi. The ETH BC was able to control and flash the onboard the LEDS when a SC was triggered on the BC.



```
// function that is called to manipulate the Pi's onboard LEDs
app.blinkLeds = () => {
  app.ledStatus = false;

  let iv = setInterval(() => {

    // if false turn off the power and staus of LEDS
    if (app.ledStatus) {

      console.log("Turning off LEDs - Sleep ") // logging the Sleep status of LEDs
      leds.power.turnOff(); // Turn off power LED   - (PWR/red) is the onbored RED LED
      leds.status.turnOff(); // Turn off status LED - (Act/green) is the onbored GREEN LED

      //if true turn on the power and status of LEDS
    } else {

      console.log("Turning on LEDS - On") // Logging the On status of the LEDs
      leds.power.turnOn(); // Turn on the power LED - (PWR/red) is the onbored RED LED
      leds.status.turnOn(); //  Turn off status LED - (Act/green) is the onbored GREEN LED

    }

    // if ledStatus is true
    app.ledStatus = !app.ledStatus;

  }, 500); // 500ms - twice a secound

  // wait one secound after setIntrerval is ran
  setTimeout(() => {

    clearInterval(iv);

  }, 10000)

}
```

Figure 13: Fuction to manipualte the onbored LEDS of the pi

6.5 Raspberry Pi 4

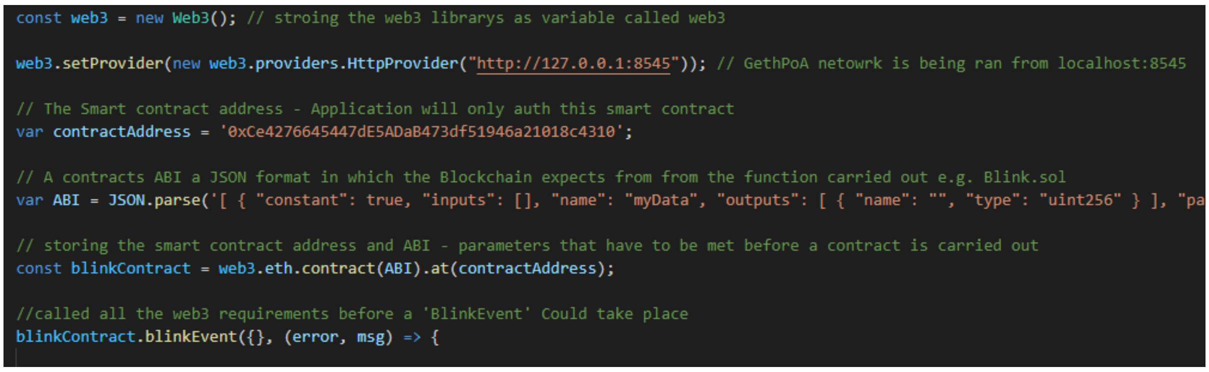
Used the raspberry pi 4 [20] model as other models of the pi such as the raspberry pi 3 and down due not have adequate resources to work or be part of a blockchain. The pi 4 has a modern 1.5GHz quad core processor (ARM v8) which supports 64 bit operating systems which helps in the overall performance of the pi. Raspbian 32 bit was used as the operating system of choice due to problems with the 64 bit compiling issues with some libraries e.g. Node.js. The 64 bit capabilities although improve usability of the pi but there still is improvement to be made. The pi will host a node.js app which be validated and secured by the PoA blockchain. The Node app will not carry out a function unless the smart contract is fulfilled. For the BC to work with the Pie had to
Install v7 CPU compatible libraries to get node and npm working which at times was very experimental some dependencies libraries did not support often showing comping errors. In this implantation for the web3 library.

6.6 Laptop

My laptop will be the Host machine for the ETH nodes and network. To clearly display the internal workings of how ETH secures an IOT device. In theory could set up these ETH nodes in cloud instances (AWS) by simply changing the IP's of each node to the IP of that cloud server, which would be realistic setup for an organisation. This PoA setup could use in any local environment where it might be needed e.g. Smart heating system.

6.7 Meta Mask

Usually used to carry out fund related transactions on the mainnet but can be used for carrying out SC transactions in DAPPS and private networks. Using the web3 library was able to connect to the BC through meta mask by hard coding smart contract credentials into the node app.



```
const web3 = new Web3(); // stroing the web3 librarys as variable called web3

web3.setProvider(new web3.providers.HttpProvider("http://127.0.0.1:8545")); // GethPoA netowrk is being ran from localhost:8545

// The Smart contract address - Application will only auth this smart contract
var contractAddress = '0xCe4276645447dE5ADaB473df51946a21018c4310';

// A contracts ABI a JSON format in which the Blockchain expects from from the function carried out e.g. Blink.sol
var ABI = JSON.parse('[ { "constant": true, "inputs": [], "name": "myData", "outputs": [ { "name": "", "type": "uint256" } ], "pa

// storing the smart contract address and ABI - parameters that have to be met before a contract is carried out
const blinkContract = web3.eth.contract(ABI).at(contractAddress);

//called all the web3 requirements before a 'BlinkEvent' Could take place
blinkContract.blinkEvent({}, (error, msg) => {
```

Figure 14: Web3 authenication will only use GethPoS and contract address before a transaction takes place

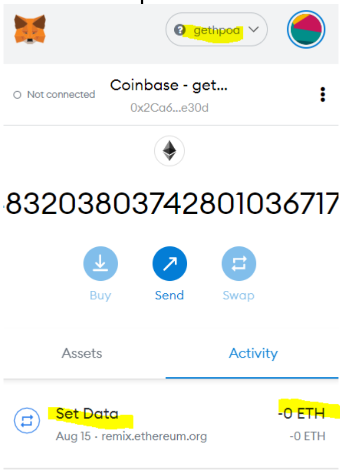
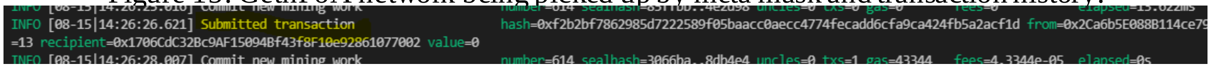


Figure 15: GethPoA network being picked up by meta mask and transaction history.



```
INFO [08-15|14:26:25.616] Commit new mining work                   number=614 sealhash=85f1bb..4e2d98 uncles=0 txs=0 gas=0        fees=0           elapsed=13.622ms
INFO [08-15|14:26:26.621] Submitted transaction                    hash=0xf2b2bf7862985d7222589f05baacc0aecc4774fecadd6cfa9ca424fb5a2acf1d from=0x2Ca6b5E088B114ce79
=13 recipient=0x1706CdC32Bc9AF15094Bf43f8F10e92861077002 value=0
INFO [08-15|14:26:28.007] Commit new mining work                   number=614 sealhash=3066ba..8db4e4 uncles=0 txs=1 gas=43344    fees=4.3344e-05  elapsed=0s
```

Figure 16: SC being submitted onto the GethPoA BC

7 Diagrams

7.1 Sequence Diagram

Breaks down how the implemented BC interacts with all components on the network. The Bootnode helps the nodes find each other in the BC creating the peer to peer network. The Miner node ensures integrity in the network, mines and validates SC's to protect the IOT device connected to the pi, in this case the LEDs bult into the pi.

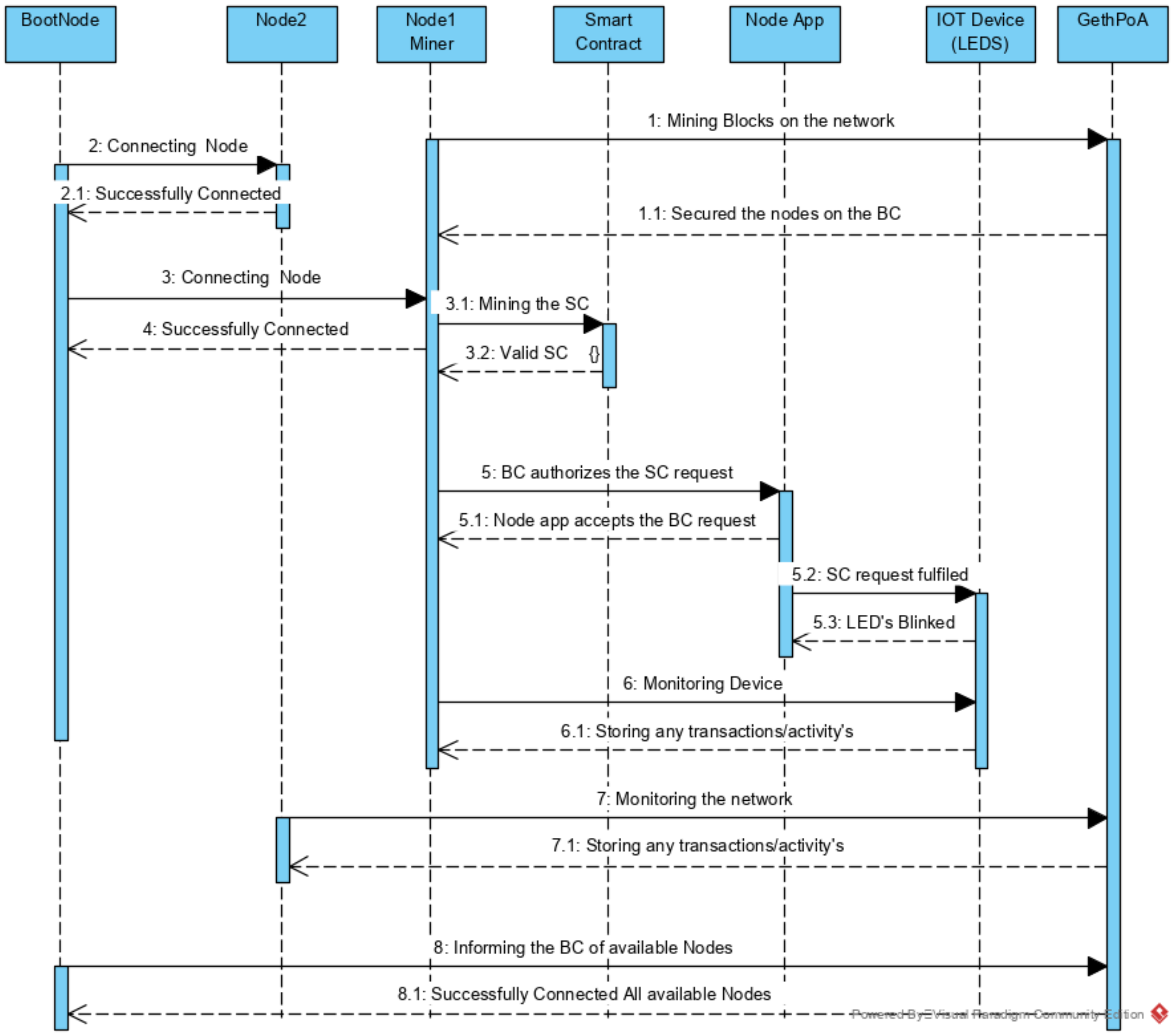

BootNode
Node2
Node1 Miner
Smart Contract
Node App
IOT Device (LEDS)
GethPoA
1: Mining Blocks on the network
2: Connecting Node
2.1: Successfully Connected
1.1: Secured the nodes on the BC
3: Connecting Node
3.1: Mining the SC
4: Successfully Connected
3.2: Valid SC
5: BC authorizes the SC request
5.1: Node app accepts the BC request
5.2: SC request fulfiled
5.3: LED's Blinked
6: Monitoring Device
6.1: Storing any transactions/activity's
7: Monitoring the network
7.1: Storing any transactions/activity's
8: Informing the BC of available Nodes
8.1: Successfully Connected All available Nodes
Powered By Visual Paradigm Community Edition

Figure 17: Sequence Diagram of the GethPoa BC

7.2 Network Diagram

The network overview shows how the ETH BC interacts with the PI. Unlike the sequence diagram displays how meta mask is implemented into the network. Meta Mask funds the transaction, but in a PoA style BC is not required. It initiates the transaction instead, if I had more time would have created a GUI using meta mask in the background making it more user friendly.

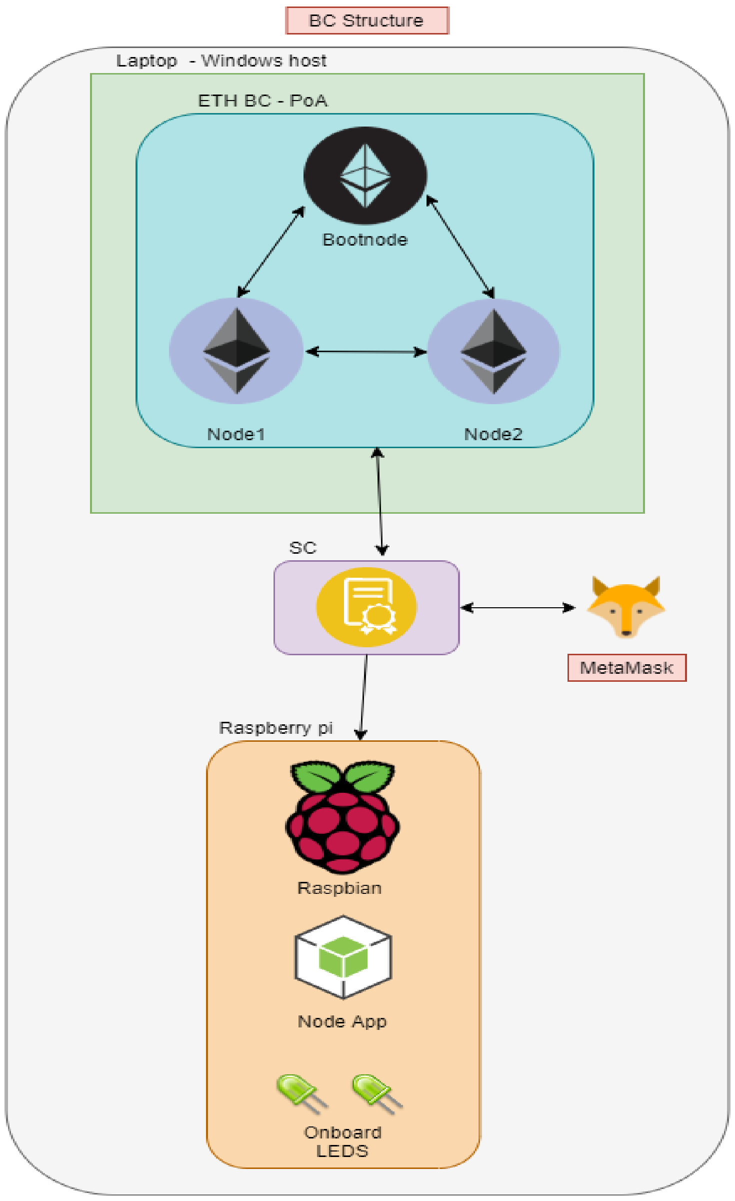
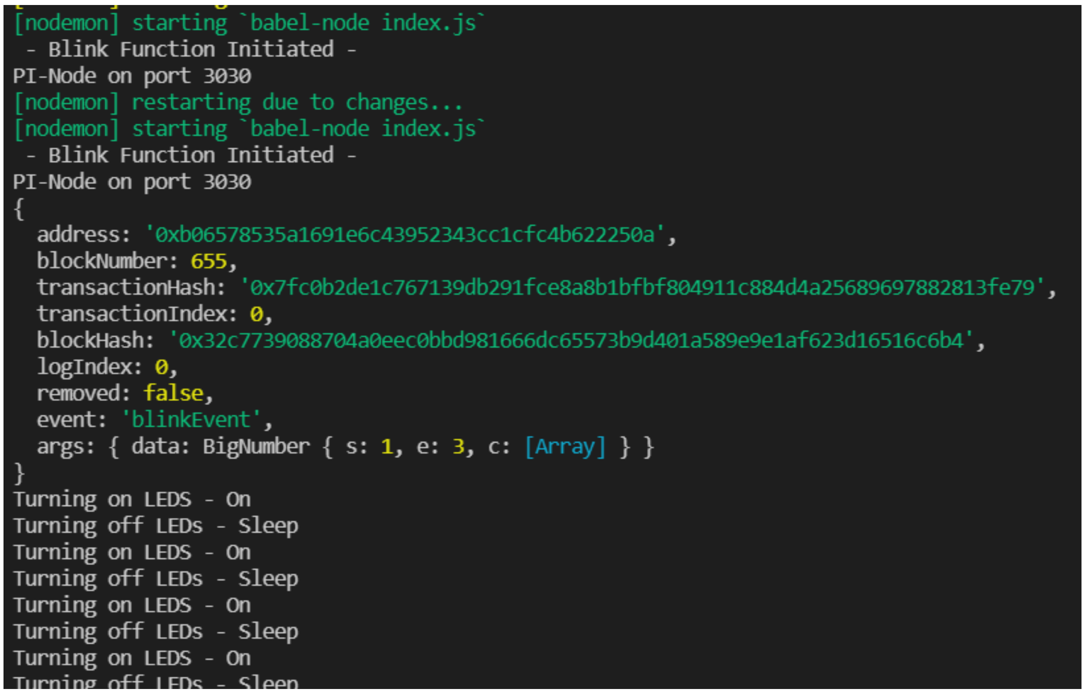


Figure 18: Network Diagram of entire BC network

8 Evaluation

Was able to transact a SC that was validated by the BC which then processed a secure transaction on node application which resulted in activating the onboard LEDS of the raspberry pi, had no security in place. The initial goal was to get the smart contract to interact with a raspberry pi camera but due to technology stack of choice (JavaScript) for the Web app made connecting the BC to the camera which is fundamentally built for Python very difficult and the libraries found clashed with the web3 library. If I had more time would have restructured the web app for flask (Python Web Servers) then would have been easier to interconnect the BC, web app and IOT device. The BC still was able to make a secure request across the BC to trigger the Pi's onboard LEDs to turn on/off when a transaction is made on the BC.



```
[nodemon] starting `babel-node index.js`
 - Blink Function Initiated -
PI-Node on port 3030
[nodemon] restarting due to changes...
[nodemon] starting `babel-node index.js`
 - Blink Function Initiated -
PI-Node on port 3030
{
  address: '0xb06578535a1691e6c43952343cc1cfc4b622250a',
  blockNumber: 655,
  transactionHash: '0x7fc0b2de1c767139db291fce8a8b1bfbf804911c884d4a25689697882813fe79',
  transactionIndex: 0,
  blockHash: '0x32c7739088704a0eec0bbd981666dc65573b9d401a589e9e1af623d16516c6b4',
  logIndex: 0,
  removed: false,
  event: 'blinkEvent',
  args: { data: BigNumber { s: 1, e: 3, c: [Array] } }
}
Turning on LEDS - On
Turning off LEDs - Sleep
Turning on LEDS - On
Turning off LEDs - Sleep
Turning on LEDS - On
Turning off LEDs - Sleep
Turning on LEDS - On
Turning off LEDs - Sleep
```

Figure 19: Node App successfully receiving the SC request.

9 Conclusion and Future

Overall has been a great learning experience, researching and testing the forefront of technologies as this implementation contained aspects from all fields not just cyber security including software development, computer architecture, IOT, cryptocurrencies and a great understanding of BC. In the beginning wanted to build a full scale DAPP with a custom private BC but realistically is a mammoth task and in terms of security research not most practical. Managed to understand ETH's approach to private networks and endless potential could provide for security e.g. Securing IOT. Still as of today BC in general is still a relatively new technologies when it comes to implementing web applications. Still has a long way to go in terms of developers adopting BC as still brittle and requires an alternative approach to the tried and tested central authority method. Although with the need for security increasing by the day BC is a feasible solution for all fields not just IOT. This implementation was intended to use a raspberry pi camera which has to postponed as Node does not like working on a pi from my experience and if I could go back would use python instead avoiding JavaScript due to it relying on so much external libraries for functionality. ETH BC for IOT security could be a potential BC stack for securing devices as the PoA structure uses public-private keys to securely sign transactions, unique wallet addresses, encrypted enode addresses, SC addresses also the ability to assign the nodes required in the development stages so that an attacker cant simply inject a bad node into the network. Going forward from the research gathered BC will be used not only for finance but all domains in the future as the advantages of BC when set up correctly outweigh the cons.

References

[1] "Getting Started · Polkadot Wiki." Polkadot, Aug. 10, 2021. Accessed: Aug. 11, 2021. [Online]. Available: https://wiki.polkadot.network/docs/getting-started

[2] F. A. Alabdulwahhab, "Web 3.0: The Decentralized Web Blockchain networks and Protocol Innovation," in *2018 1st International Conference on Computer Applications Information Security (ICCAIS)*, Apr. 2018, pp. 1–4. doi: 10.1109/CAIS.2018.8441990.

[3] wackerow, “Consensus mechanisms.” Ethereum, Jun. 30, 2021. Accessed: Aug. 11, 2021. [Online]. Available: https://ethereum.org

[4] adam dossa, “ERC 1400: Security Token Standard · Issue #1411 · ethereum/EIPs.” Github - Ethereum, Sep. 13, 2018. Accessed: Aug. 11, 2021. [Online]. Available: https://github.com/ethereum/EIPs/issues/1411

[5] Y. N. Aung and T. Tantidham, “Review of Ethereum: Smart home case study,” in *2017 2nd International Conference on Information Technology (INCIT)*, Nov. 2017, pp. 1–4. doi: 10.1109/INCIT.2017.8257877.

[6] K. R. Ozyilmaz and A. Yurdakul, “Designing a Blockchain-Based IoT With Ethereum, Swarm, and LoRa: The Software Solution to Create High Availability With Minimal Security Risks,” *IEEE Consum. Electron. Mag.*, vol. 8, no. 2, pp. 28–34, Mar. 2019, doi: 10.1109/MCE.2018.2880806.

[7] A. Dika and M. Nowostawski, “Security Vulnerabilities in Ethereum Smart Contracts,” in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Jul. 2018, pp. 955–962. doi: 10.1109/Cybermatics_2018.2018.00182.

[8] V. K. C. Ramesh, Y. Kim, and J.-Y. Jo, “Secure IoT Data Management in a Private Ethereum Blockchain,” in *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, Jul. 2020, pp. 369–375. doi: 10.1109/COMPSAC48688.2020.0-219.

[9] A. Cong An, P. Thi Xuan Diem, L. Thi Thu Lan, T. Van Toi, and L. Duong Quoc Binh, “Building a Product Origins Tracking System Based on Blockchain and PoA Consensus Protocol,” in *2019 International Conference on Advanced Computing and Applications (ACOMP)*, Nov. 2019, pp. 27–33. doi: 10.1109/ACOMP.2019.00012.

[10] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, “Internet of Things: Security vulnerabilities and challenges,” in *2015 IEEE Symposium on Computers and Communication (ISCC)*, Jul. 2015, pp. 180–187. doi: 10.1109/ISCC.2015.7405513.

[11] T. M. Fernández-Caramés and P. Fraga-Lamas, “A Review on the Use of Blockchain for the Internet of Things,” *IEEE Access*, vol. 6, pp. 32979–33001, 2018, doi: 10.1109/ACCESS.2018.2842685.

[12] K. M. Giannoutakis *et al.*, “A Blockchain Solution for Enhancing Cybersecurity Defence of IoT,” in *2020 IEEE International Conference on Blockchain (Blockchain)*, Nov. 2020, pp. 490–495. doi: 10.1109/Blockchain50366.2020.00071.

[13] A. Rustagi, C. Manchanda, and N. Sharma, “IoE: A Boon Threat to the Mankind,” in *2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT)*, Apr. 2020, pp. 114–119. doi: 10.1109/CSNT48778.2020.9115748.

[14] M. Pustišek, A. Umek, and A. Kos, “Approaching the Communication Constraints of Ethereum-Based Decentralized Applications,” *Sensors*, vol. 19, no. 11, 2019, doi: 10.3390/s19112647.

[15] B. Putz and G. Pernul, “Detecting Blockchain Security Threats,” in *2020 IEEE International Conference on Blockchain (Blockchain)*, Nov. 2020, pp. 313–320. doi: 10.1109/Blockchain50366.2020.00046.

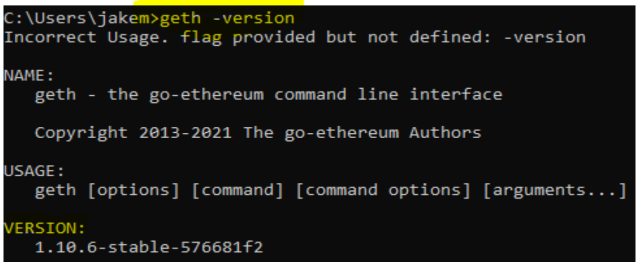
[16] J. Lee and M. Pilkington, “How the Blockchain Revolution Will Reshape the Consumer Electronics Industry [Future Directions],” *IEEE Consum. Electron. Mag.*, vol. 6, no. 3, pp. 19–23, Jul. 2017, doi: 10.1109/MCE.2017.2684916.

[17] L. D. Santis, V. Paciello, and A. Pietrosanto, “Blockchain-Based Infrastructure to enable Trust in IoT environment,” in *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, May 2020, pp. 1–6. doi: 10.1109/I2MTC43012.2020.9128817.

[18] T. R. Vance and A. Vance, "Cybersecurity in the Blockchain Era : A Survey on Examining Critical Infrastructure Protection with Blockchain-Based Technology," in *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S T)*, Oct. 2019, pp. 107–112. doi: 10.1109/PICST47496.2019.9061242.
[19] A. Rot and B. Blaicke, "Blockchain's Future Role in Cybersecurity. Analysis of Defensive and Offensive Potential Leveraging Blockchain-Based Platforms," in *2019 9th International Conference on Advanced Computer Information Technologies (ACIT)*, Jun. 2019, pp. 447–451. doi: 10.1109/ACITT.2019.8779855.
[20] "Raspberry Pi 4 Computer Model B." Raspberry Pi, 2019. Accessed: Mar. 20, 2021. [Online]. Available: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-4-Product-Brief.pdf

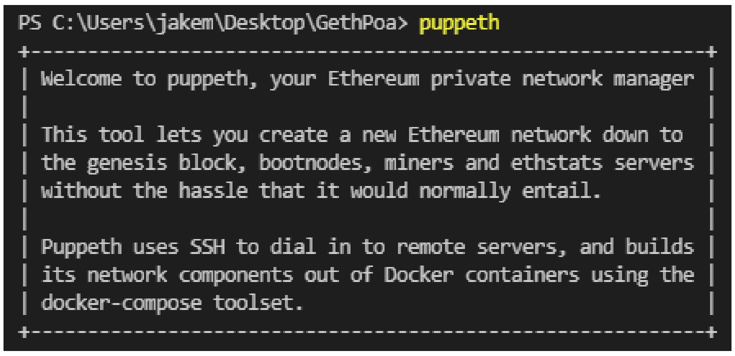
10 Appendix

10.1 Geth Version Used

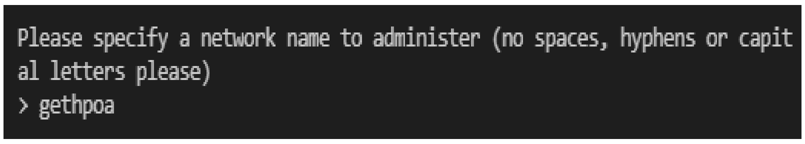

```
C:\Users\jakem>geth -version
Incorrect Usage. flag provided but not defined: -version

NAME:
   geth - the go-ethereum command line interface

   Copyright 2013-2021 The go-ethereum Authors

USAGE:
   geth [options] [command] [command options] [arguments...]

VERSION:
   1.10.6-stable-576681f2
```

10.1.1 Appendix 1: Geth version 1.10.6

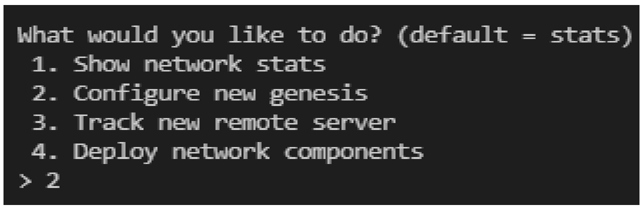
10.2 Setting up Genesis Block with Puppeth.


```
PS C:\Users\jakem\Desktop\GethPoa> puppeth
+-----------------------------------------------------------+
| Welcome to puppeth, your Ethereum private network manager |
|                                                           |
| This tool lets you create a new Ethereum network down to  |
| the genesis block, bootnodes, miners and ethstats servers |
| without the hassle that it would normally entail.         |
|                                                           |
| Puppeth uses SSH to dial in to remote servers, and builds |
| its network components out of Docker containers using the |
| docker-compose toolset.                                   |
+-----------------------------------------------------------+
```

10.2.1 Appendix 2: Puppeth CLI Terminal

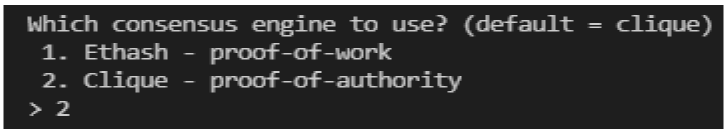

```
Please specify a network name to administer (no spaces, hyphens or capit
al letters please)
> gethpoa
```

10.2.2 Appendix 3: Setting up a genesis block – choose networks name



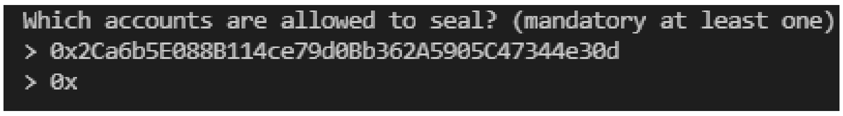
```
What would you like to do? (default = stats)
 1. Show network stats
 2. Configure new genesis
 3. Track new remote server
 4. Deploy network components
> 2
```

10.2.3 Appendix 4: Options of a BC but used to configured new genesis



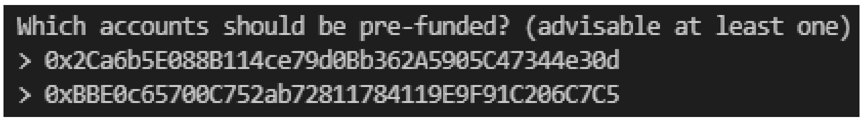
```
Which consensus engine to use? (default = clique)
 1. Ethash - proof-of-work
 2. Clique - proof-of-authority
> 2
```

10.2.4 Appendix 5: Chossing a Clique consensus engine - PoA



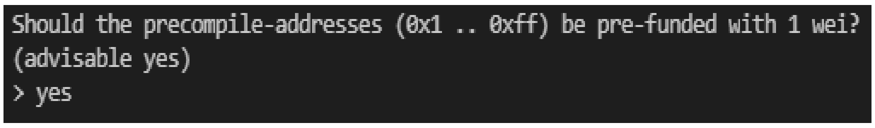
```
Which accounts are allowed to seal? (mandatory at least one)
> 0x2Ca6b5E088B114ce79d0Bb362A5905C47344e30d
> 0x
```

10.2.5 Appendix 6: Choosing node1s address to Mine/Auth blocks on the chain



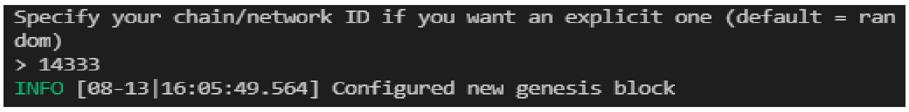
```
Which accounts should be pre-funded? (advisable at least one)
> 0x2Ca6b5E088B114ce79d0Bb362A5905C47344e30d
> 0xBBE0c65700C752ab72811784119E9F91C206C7C5
```

10.2.6 Appendix 7: Pre funding node1 and node2 addresses to avoid ether issues



```
Should the precompile-addresses (0x1 .. 0xff) be pre-funded with 1 wei?
(advisable yes)
> yes
```

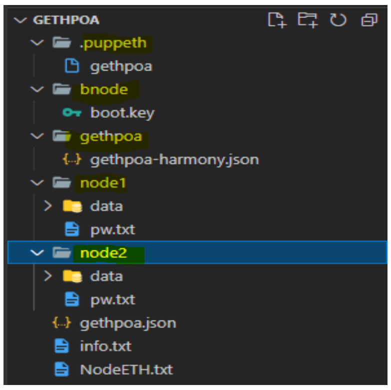
10.2.7 Appendix 8: confirm the pre funding of both nodes



```
Specify your chain/network ID if you want an explicit one (default = ran
dom)
> 14333
INFO [08-13|16:05:49.564] Configured new genesis block
```

10.2.8 Appendix 9: choosing a network ID which will be necessary for connecting nodes on the network.

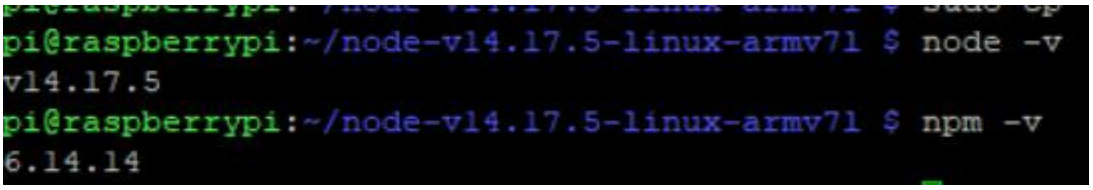
11 BC file Structure



```
GETHPOA
  .puppeth
    gethpoa
  bnode
    boot.key
  gethpoa
    gethpoa-harmony.json
  node1
    data
    pw.txt
  node2
    data
    pw.txt
  gethpoa.json
  info.txt
  NodeETH.txt
```

11.1.1 Appendix 10: BC file Strucutre

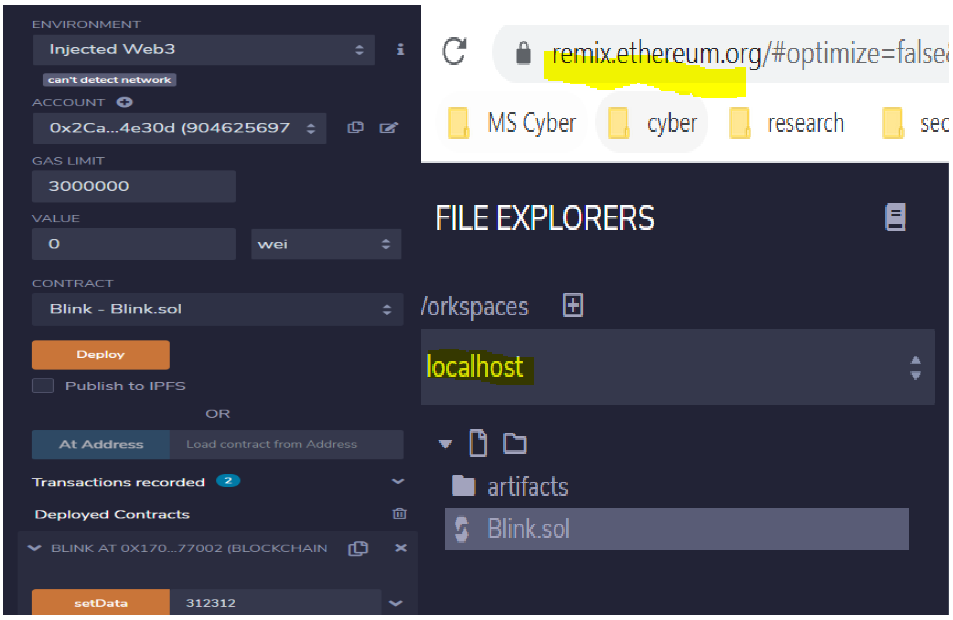
12 Raspberry Pi



```
pi@raspberrypi:~/node-v14.17.5-linux-armv71 $ node -v
v14.17.5
pi@raspberrypi:~/node-v14.17.5-linux-armv71 $ npm -v
6.14.14
```

12.1.1 Apendix 11: Node version running on Pi

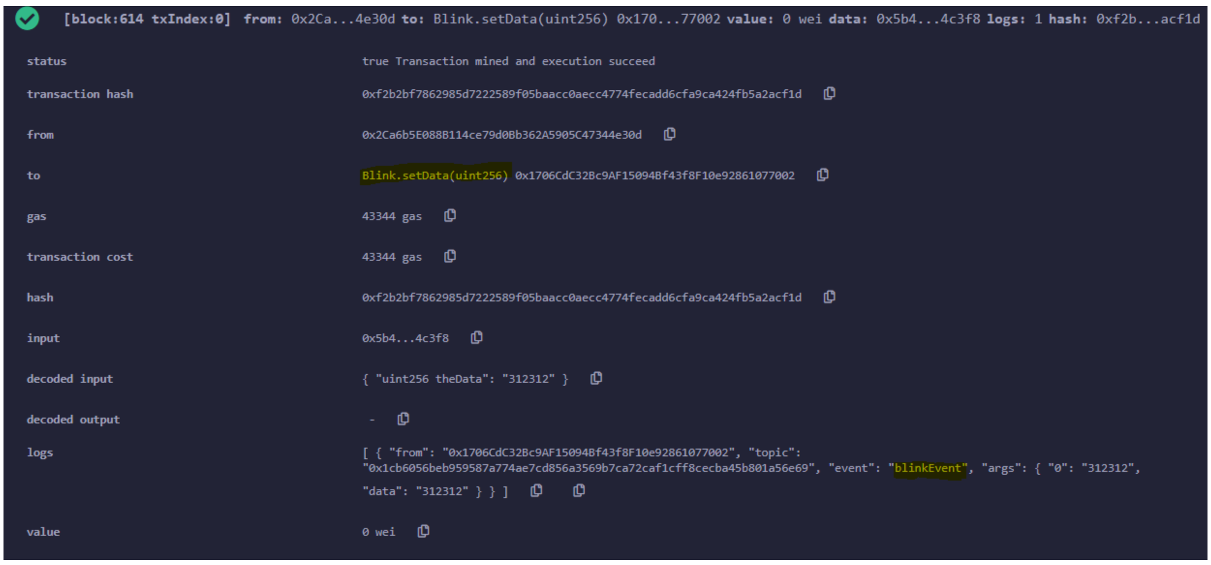
13 Remix Editor



```
ENVIRONMENT
Injected Web3
can't detect network
ACCOUNT
0x2Ca...4e30d (904625697
GAS LIMIT
3000000
VALUE
0          wei
CONTRACT
Blink - Blink.sol
Deploy
Publish to IPFS
OR
At Address    Load contract from Address
Transactions recorded 2
Deployed Contracts
BLINK AT 0X170...77002 (BLOCKCHAIN
setData    312312
```

```
remix.ethereum.org/#optimize=false
MS Cyber   cyber   research   sec
FILE EXPLORERS
/orkspaces
localhost
artifacts
Blink.sol
```

13.1.1 Apendix 12: GUI for Remix to carry out a transaction + working on SC locally



```
[block:614 txIndex:0] from: 0x2Ca...4e30d to: Blink.setData(uint256) 0x170...77002 value: 0 wei data: 0x5b4...4c3f8 logs: 1 hash: 0xf2b...acf1d

status                  true Transaction mined and execution succeed
transaction hash        0xf2b2bf7862985d7222589f05baacc0aecc4774fecadd6cfa9ca424fb5a2acf1d
from                    0x2Ca6b5E088B114ce79d0Bb362A5905C47344e30d
to                      Blink.setData(uint256) 0x1706CdC32Bc9AF15094Bf43f8F10e92861077002
gas                     43344 gas
transaction cost        43344 gas
hash                    0xf2b2bf7862985d7222589f05baacc0aecc4774fecadd6cfa9ca424fb5a2acf1d
input                   0x5b4...4c3f8
decoded input           { "uint256 theData": "312312" }
decoded output          -
logs                    [ { "from": "0x1706CdC32Bc9AF15094Bf43f8F10e92861077002", "topic":
                        "0x1cb6056beb959587a774ae7cd856a3569b7ca72caf1cff8cecba45b801a56e69", "event": "blinkEvent", "args": { "0": "312312",
                        "data": "312312" } } ]
value                   0 wei
```

Apendix 13: Remix displaying the results from the mined block(blink event)