REPUBLIC OF TURKEY

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

BLOCKCHAIN IN SMART CITIES

Israa sameer ABDULRAHEEM
Master of Science

Supervisor

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Istanbul, 2021

BLOCKCHAIN IN SMART CITIES

Israa Sameer ABDULRAHEEM

Electrical and Computer Engineering

Master of Science

ALTINBAŞ UNIVERSITY

2021

The thesis titled "BLOCKCHAIN IN SMART CITIES" prepared by ISRAA SAMEER ABDULALRAHEEM and submitted 18/ 08/2021 has been accepted **unanimously of votes** for the degree of Master of Science in Electrical and Computer Engineering

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Supervisor

Thesis Defense Jury Members:

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM	School of Engineering and Architecture , Altinbas University	
Asst. Prof. Dr.Muhammed ILYAS	School of Engineering and Architecture , Altinbas University	
Asst. Prof. Dr Sewale MUSADAQ TAHA	Faculty of Communication Digital Game Design Department Beykent University	

I hereby declare that this thesis/dissertation meets all format and submission requirements of a Master's Thesis.

Submission date of the thesis to the Graduate Education Institute: ___/___/___

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

ISRAA SAMEER ABDULALRAHEEM

DEDICATION

This study is wholeheartedly dedicated to our beloved parents, who have been our source of inspiration and gave us strength when we thought of giving up, who continually provide the immoral, spiritual, emotional, and financial support. To our brothers, sisters, mentor, friends, and classmates who shared their words of advice and encouragement to finish this study. And lastly, we dedicated this thesis to the Almighty God, thank you for the guidance, strength, power of mind, protection, and skills and for giving us a healthy life. All of these, we offer to you.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor Asst. Prof. Dr. Abdullahi Abdu IBRAHIM for guiding and helping me along the way in writing this dissertation. Discussing my progress, problems, and ideas with my supervisor Asst. Prof. Dr. Abdullahi Abdu IBRAHIM a couple of times every week helped me tremendously in understanding the logic behind the research. It made me better realize the technical need for this research work.

ABSTRACT

BLOCKCHAIN IN SMART CITIES

Israa Sameer ABDULRAHEEM,

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

Supervisor: Asst. Prof. Dr, Abdullahi Abdu IBRAHIM

Date: August /2021

Pages: 68

Blockchain technology has been linked with Internet of Things for a long time now. There are many issues that are hinder the implementation of IoT applications at a large scale. Surveys and studies from multiple sources reveal that security threats and data privacy are still the primary concerns. These problems are well known, and solutions exist for these problems in the IT industry. However, traditional IT security solutions cannot be applied to IoT for various reasons spanning from type of devices to sheer volume of devices. Unfortunately, like in any other industry, security is often disregarded in the IoT domain as well, and most of the resources are allocated to application development and device hardware. So, the search for a silver bullet to overcome these inhibitors has been going on for a while. After Bitcoin became prominent, people started to realize the potential of the underlying distributed ledger (blockchain) technology and considered it as a true innovation. Rather than facilitating a peer-to-peer digital payment system involving a cryptocurrency, the blockchain technology is viewed as a mechanism that provides device identity, secure data transfer, and immutable data storage. All these features can be implemented without any centralized authority and a completely transparent system with auditable cryptographic proofs. Our aim through this research thesis is to get a deep level understanding of the blockchain technology and study some of the widely used blockchain frameworks including Ethereum, Bitcoin, and Litecoin. We will further examine the exclusive features offered by each of these frameworks and define their target use cases. While researching about each framework, we plan to deploy a blockchain in the local network i.e., private blockchain and operate on it from different devices running on various operating systems. In each deployment, we will observe the functional issues and benchmark system requirements for running different types of nodes. Also, we will

study different algorithms involved in each framework, compare them with each other, and derive their suitability for IoT. Ultimately, our aim is to determine the most suitable blockchain architecture for the IoT ecosystem. A high-level comparison of the researched architectures will be provided so that managers and developers can quickly decide on a suitable framework for their application or use case depending upon the requirements. For each architecture, a set of sample use cases and on-going research will be discussed to get an idea of the usage of that architecture in the real world.

Keywords: Transaction, Internet of Things, Blockchain, Bitcoin, Ethereum.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

ABBREVIATIONS

IoT	:	Internet of Things
RFID	:	Radio Frequency Identification
IaaS	:	infrastructure-as-a service

1. INTRODUCTION

1.1 OVERVIEW

The Internet-of-Things is the global network of intelligent and connected devices utilizing Wi-Fi capabilities and sensors. Embedding electronics into physical objects builds some intelligence into them making them "smart" thereby enabling these devices to communicate with each other as mentioned in [1]. These devices can range from industrial control systems, automobiles, wearable devices, medical equipment, smart home devices, and smart energy management devices. In this manner, the predictable channels of an information system are rapidly changing and the physical world is becoming increasingly smart. The steep rise in the number of connected devices can be attributed to the ubiquitous broadband internet, low connection costs and availability of a large number of smart devices as mentioned in [2]. The following are a few applications of IoT which throw light on the advancements in wireless network technologies and the extent to which they can be leveraged for the purpose of automation and control, information analysis. Rental cars with embedded sensors provide the customers with the flexibility to pick up cars from pick up spots and return them rather than having rental centers as mentioned in [3]. The embedded sensors help in tracking the car, the distance travelled, and the rental duration. The sensors in retail membership cards retrieve the shopper's data and apply the appropriate discounts at the point of sale as mentioned in [4]. Airplane manufacturers are considering building the airframes with embedded sensors which send important data regarding the health of the critical components to monitoring systems. This can help mitigate unprecedented downtime and also provides scope for proactive maintenance as mentioned in [5]. For instance, in many manufacturing industries, the paper industry demands manual adjustments of temperature for lime kilns which can be automated through the use of sensors and also they are networked to alert the operators in case of the temperature rising beyond the threshold.

Undoubtedly IoT solutions have provided great benefits and their incorporation into various sectors like financial technology, healthcare, manufacturing, retail, transportation and supply chain can herald radical transformation as mentioned in [6]. These solutions offer enhanced tracking methodologies which can be useful in the healthcare industry where patients can be closely monitored by the doctors using monitoring devices, sensors and also alerts can be sent during an emergency situation. Given the low costs of infrastructure and the rapid growth in the number of

connected devices, ubiquitous networks are emerging. IoT solutions also help optimize the processes thereby reducing the costs and enhancing the production rates as mentioned in [7]. Although there are several benefits of these solutions, there are several factors which thwart us from tapping their full potential. The lack of sufficient knowledge on interoperable technologies and the associated standards also pose a setback for several organizations. Managing the huge volume of data generated by the IoT devices is also an arduous task demanding advanced algorithms and protocols to operate on the data as mentioned in [8]. With IoT, the conventional routes of information exchange are expanding and this is raising concerns in the security space since there are a lot of avenues for exchanging information and all of the transmitted data needs to be secured.

1.2 PROBLEM STATEMENT

The blockchain technology is primarily used to facilitate the secure transaction of financial assets between two parties. Instead of trusting a third party, the blockchain technology leverages the use of hash functions and distributed systems to provide a cryptographic proof for these electronic transactions. Blockchain is a large database (or ledger) consisting of linearly attached blocks and this ledger is shared by all the distributed devices in the system, which are otherwise called as nodes. Here, each block corresponds to a set of transactions that are being processed at a certain point of time. This block contains two pieces of information – the details of the transactions, and the hash value, that is calculated from the data available from the previous block. The hash value of the previous block is calculated from the data provided by the block before it and this whole process begins from the genesis block, the hard coded first block of a blockchain. This ensures that the data in the overall blockchain is unaltered and final. This blockchain which is shared by all the nodes contains every transaction that has ever happened right from the first ever transaction to the latest one. In the recent past, there have been several distributed denial of service (DDOS) attacks which were launched by compromising millions of IoT devices. One such attack with a strength of 1.2 Tbps was conducted against the DNS service provider disrupting the access to several websites. The botnet comprising of IoT devices include digital video recorders, routers and compromised web cameras infected by the Mirai malware as mentioned in [9]. The DDOS attack on Brian Krebs' security blog was also a powerful one with a strength of 665 Gbps and involved millions of compromised IoT devices. Security Researchers believe that the attack groups targeting IoT devices are proliferating by the day. With the steady rise in IoT devices, they form easy targets

for attackers since several users do not change the default passwords of the devices and also do not update the firmware as the patches are rolled. The Open Web Application Security Project (OWASP) has recently released a list of top 10 vulnerabilities in IoT devices. "Insecure web interface, insufficient authentication/authorization, insecure network services, lack of transport encryption, privacy concerns, insecure cloud interface, insecure mobile interface, insufficient security configurability, insecure software/firmware and poor physical security" have been identified as the top 10 vulnerabilities in IoT devices as mentioned in [10].

1.3 RESEARCH CONTRIBUTION

Several organizations are conducting research on leveraging the blockchain technology, primarily used in electronic financial transactions, as the core principle to exchange information between these internet-of-things. Companies from various industries like healthcare, electronics, telecommunications, networks, transportation, and banking are coming forward to make this idea a reality. It is believed that the blockchain technology can be very efficient in exchanging information between devices and companies with minimal cost and complexity while maintaining trust, accountability, and transparency. However, the blockchain technology used in Bitcoins cannot be directly applied for internet-of-things. Depending on the type of company and the information being exchanged, there are many issues that need to be resolved. Following are some of them:

- **Unique Identification for Smart Cities:** Each device connected to the internet should have a unique identifier. This Id is used to track the device in the ledger. Usually, this is achieved by assigning a pair of keys (public and private) to each device. However, devices like lights and CCTV cameras might not have built-in capacity to manage these cryptography keys. IP address cannot be used for this purpose due to their dynamic nature and the presence of private IP addresses.
- **Double Spending issue:** Mining and proof-of-work functions are used eliminate the transactions involving double spending. This is only useful while transferring assets with monetary value. In other cases, involving the transfer of data, the same information from a device can be sent to different devices at any time. But, there should be some protocol to build a blockchain and append blocks to it in a systematic manner.
- **Cryptographic Computation for Smart Cities:** The devices in IoT may not have the computational capacity to perform complex cryptographic calculations involved in the

blockchain model. Since it is estimated that 50 billion devices will be connected to the internet by 2022, there should be enough devices online at any point of time, to perform these calculations.

- **Information Exchange for Smart Cities:** While it may be relatively easy to transfer the information and maintain a ledger for devices of the same type, IoT will have different types of devices and each device might store/process different kinds of information. Maintaining all this information in a single ledger might get complex and difficult to manage. Alternatives like sidechain can be utilized to solve this problem.
- **Trusting the devices:** More devices means more risk. Each device might have different vulnerabilities and if they are exploited successfully by an attacker, it might lead to a complete takeover of the device. Blindly trusting all the devices in the blockchain might be a problem.

Although there are many other issues and weaknesses surrounding this technology, it is believed that its advantages outweigh these issues by a large margin.

1.4 BLOCKCHAIN CONTRACTS

A block, before being appended to the blockchain, should be verified for validity by the nodes in the distributed system as mentioned in [11]. This process of verification and confirming the transactions in a block, before its addition to the ledger, is called Mining, and the nodes of the distributed system which involve in this verification process are called miners. All the unconfirmed latest transactions are broadcasted to all the miners in the network. The miners collect all these transactions and generate a hash value using a Hashcash Proof-of-work function as mentioned in [12]. There are two important characteristics of a hash function that are to be noted here – the hash value cannot be reverse engineered to generate the original content, and a unique hash value is generated each time the order of transactions is changed. While it is relatively easy for a computer to generate a hash value from the given information, "Proof-of-work" makes it difficult for the miners by demanding that the final hash value should be of a certain format. The hash value should start with a certain number of zeroes as mentioned in [13]. All the miners try to arrive at this output by adding a different nonce value each time they perform the calculation. When a miner generates this expected hash, the total block (containing the transactions and the hash value) is broadcasted to other nodes in the network. If all the transactions in that block are valid, the nodes accept the block and it'll be attached to the existing blockchain. This block is final and cannot be altered as

mentioned in [14]. This process eliminates duplicate or double spending. This process also eliminates the danger of attackers trying to pass a wrong block as legitimate by determining majority representation based on CPU effort rather than IP address. If the decision is based on the votes by a majority of the IP addresses, a hacker who has control over 51% of these IP addresses can control the blockchain as mentioned in [15]. To prevent this, the "majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it." Therefore, if the majority of the CPU power is controlled by genuine nodes, the chain of these nodes will grow the fastest as mentioned in [16].

Bitcoin is the first ever electronic peer-to-peer payment system that was built based on the blockchain technology. Although Bitcoin has started a revolution in the financial sector with its digital cryptocurrency, the distributed ledger (blockchain) technology that underlies Bitcoin is considered as the true innovation as mentioned in [17]. Though nascent and controversial, this technology is rapidly developing. Blockchains offer a framework to provide better coordination and transaction processing capabilities between the connected devices thus giving rise to a decentralized Internet of Things. Let us take the Bitcoin blockchain as an example and look at some of the unique features offered by this technology.

2. LITERATURE REVIEW

Bitcoin is a virtual currency that has been created by a person using an alias Satoshi Nakamoto. It was first introduced to a cryptography mailing list on October 31st, 2008, and was released as an open-source software in 2009 as mentioned in [18]. The idea behind building this digital or virtual currency is to build a decentralized system which would allow the currency to be electronically transferable with negligible or no transaction fees. This is the first peer-peer network which powers its users with no central authority or banks as mentioned in [19]. These bitcoins are built out of a bitcoin protocol. According to this protocol, there would be a definite number of bitcoins which can be produced (mined) which is 21 million. However, each bitcoin can be further divided into smaller parts which can go up to as small as one-hundredth million of a bitcoin. This smallest division of the bitcoin is termed as "Satoshi".

Bitcoin has users that are widespread across the world and it is not controlled by any single person. From a user's viewpoint, bitcoin is an application which provides a wallet and users the ability to make transactions among each other as mentioned in [20]. To perform transactions, the users are free to use any software they want, as long as the software is compatible to, and complies with the rules of the bitcoin protocol. This electronic payment system is built on a cryptographic proof rather than trust, which is why the need of a third party is eliminated as mentioned in [21]. The transactions made are practically impossible to reverse or destroy. The bitcoin network shares a public ledger called the blockchain.

Let us consider a simple scenario to understand why a person would use bitcoin against another form of traditional currency. Suppose there are two parties Alice and Bob. Alice wants to pay Bob a small amount of money for a transaction they previously had. They both live in different countries and in order to send the money to another country or wire it through a bank, Alice and Bob would have to incur transaction fees which would probably be more than the amount itself. This is not a viable option hence they look to use digital currency for making that transaction as mentioned in [22]. A feasible option would be to choose a method that charges only a negligible amount of transaction fees and is decentralized so Bob and Alice do not have to depend on any banks or do not need to worry about high transaction fees. For a scenario like this, a digital decentralized currency like Bitcoin is the solution as mentioned in [23]. The only two things Alice would need to pay Bob is her own private key and Bob's public key. Bitcoin supports having an

application on a user's phone so they can make transactions easily just by opening the app and performing few simple steps. If a sender has the receiver's public key, he can make a transaction to a Bitcoin address. Alice would get Bob's public key and would send the equivalent bitcoin amount to Bob. After Alice sends the money to Bob using his public key, there would be a signature generated by the private key and this would release the money from Alice to Bob. Once this transaction is made, the bitcoin app alerts all the miners about this transaction. Miners who are on the network collate the data from the pending transactions, the last block of transactions that are recorded in the publicly available ledger, and a nonce as mentioned in [24]. The miner uses a hash function which ultimately gives out a cryptographic fingerprint which is unique. This unique fingerprint makes the transaction verifiable. The hashed block that is produced must meet certain complexity requirements as mentioned in [25]. The complexity requirements are not constant and they keep changing with the number of bitcoins mined. They must have a certain number of zeroes at the beginning of the block. To attain this, the miner has to try using different values of the random nonce. This is what would consume a lot of computational power and capacity. At a point where the miner successfully finds a nonce with the desired complexity requirements, it is announced to the network as mentioned in [26]. The other miners who are on the network validate this block and try finding a new block with the block which was just released on the network as their last component. When a miner successfully verifies or validates transactions and hence finds new blocks, he is rewarded with a certain number of bitcoins as mentioned in [27]. This hashed block is made public and would be available on the shared ledger which the entire network relies on.

The basic working of bitcoin can be illustrated in the graphic below:

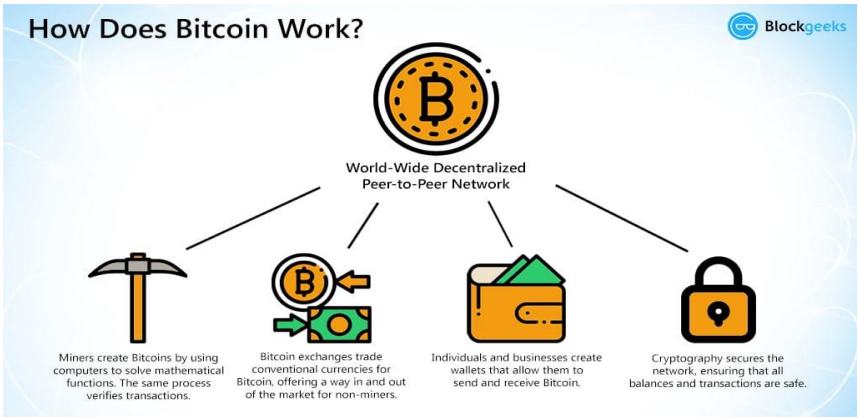

Figure 2.1: Basic Working of Bitcoin [28]

2.1. IDENTITY MANAGEMENT IN BLOCKCHAIN

In the Bitcoin blockchain, every node has a wallet (bitcoin client) which manages all the accounts belonging to that node. Accounts are nothing but a combination of a public key and a private key. So, whenever a new account is created, a unique cryptographic key pair is generated as mentioned in [29]. The hash of the public key acts as the account address of a node and all the addresses are public to the bitcoin network.

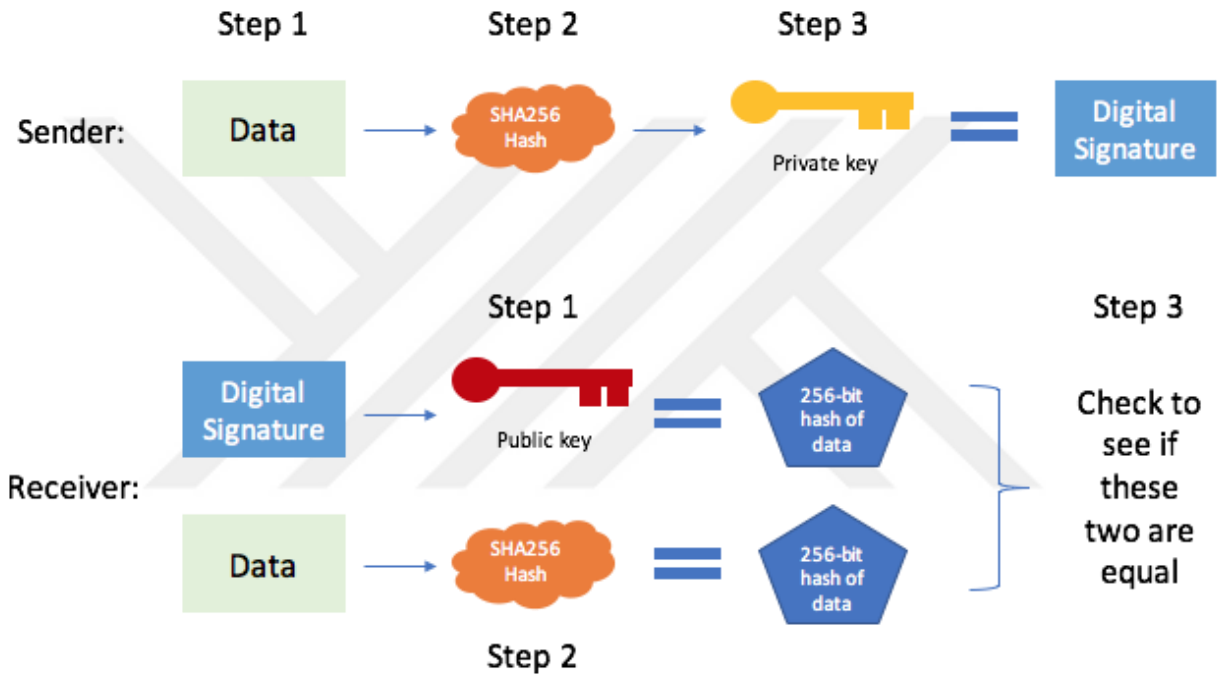

Figure 2.2: Bitcoin Wallet Encryption [30]

So, when a node (N1) wants to send some bitcoins to another node (N2), a transaction is initiated from N1:

Sender: A1, Receiver: A2, Value: 5 BTC

Where A1, A2 are accounts of N1, N2 respectively.

Since account addresses (A1, A2) are public to the network, any node can pretend to be N1 and use A1 as the *sender* account. This problem is solved using digital signatures. When a transaction is initiated, the sender digitally signs that transaction with his/her corresponding private key. So, when the transaction is broadcasted to the network, every node which has the public key verifies the digital signature. If the signature matches, the transaction is considered valid, in the case of a mismatch, that transaction is discarded as mentioned in [31]. Since this signature is verified by

many nodes in the network, there is no possibility of a faulty transaction getting processed as a valid transaction. In short, there is no way for a node to pretend as another node. Following is an illustration of the digital signature verification process in the Bitcoin network:

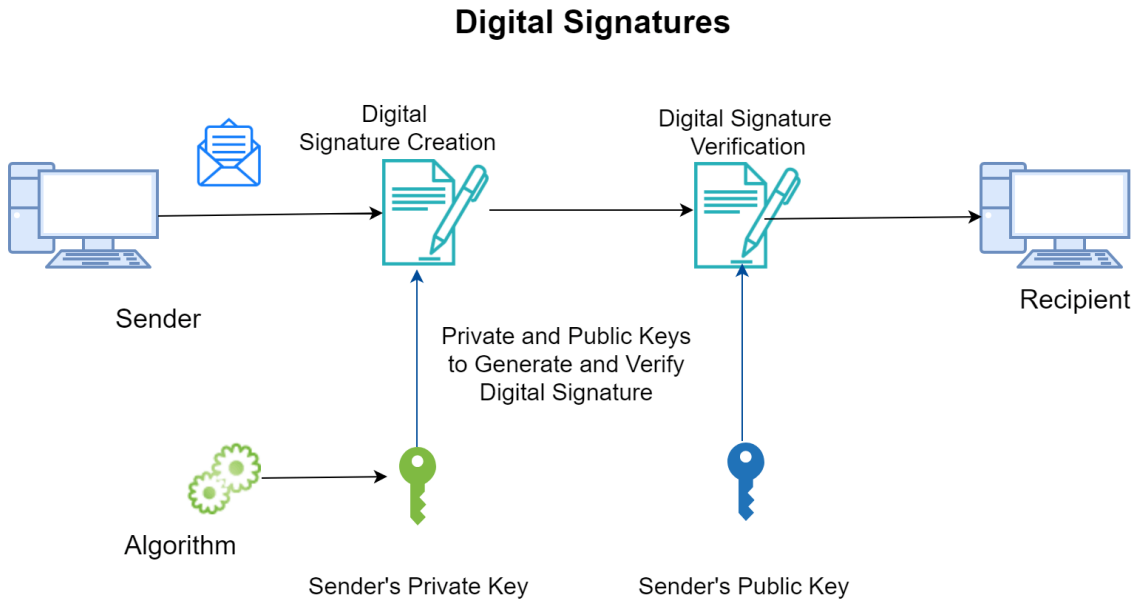

Figure 2.3: Digital Signature Verification Process in Bitcoin [32]

2.2 SECURITY, TRANSPARENCY, AND TAMPER-PROOF DATA STORAGE

In the Bitcoin network, there exists a record of every bitcoin that has ever been produced/mined. Mining is the only process through which new bitcoins are generated so when every node in the network maintains a copy of the ledger, it is easy to keep track of all the bitcoins. At any point in time, there'll be many transactions that are being broadcasted into the network and every node gets a copy of these transactions as mentioned in [33]. As depicted below, all these transactions are arranged in the form of a binary Merkle tree where the Merkle root is the transaction root.

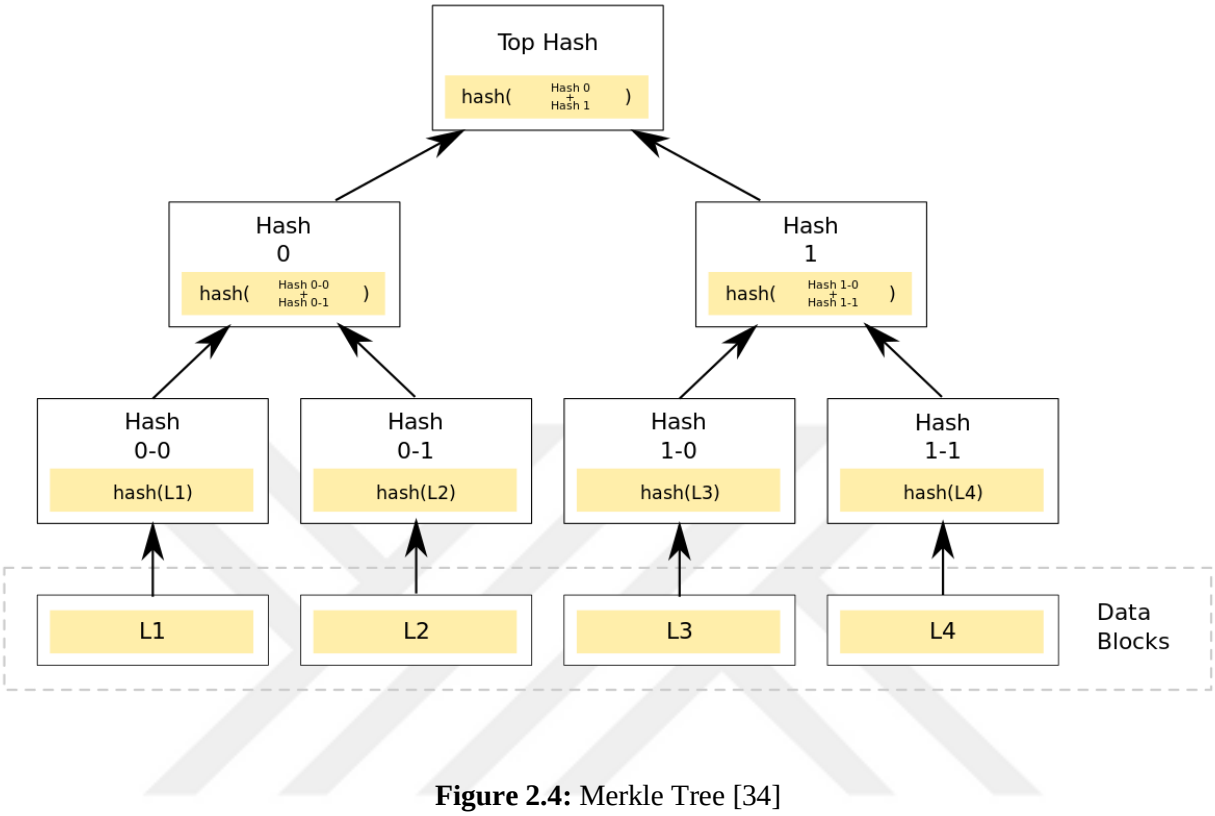

Figure 2.4: Merkle Tree [34]

Arranging the transactions in such a fashion provides a secure way of verifying any transaction as needed without having to download the complete blockchain. The integrity of a transaction can be verified through the block header, transaction, and the branch of the Merkle tree in which the transaction is present as mentioned in [35]. So, after checking for identity and double spending, the valid transactions are arranged like this and placed into a block. A block can be divided into two parts: header and data. The data part contains all the transactions and the header part contains the transaction root, the hash of the previous block, timestamp, and a nonce value.

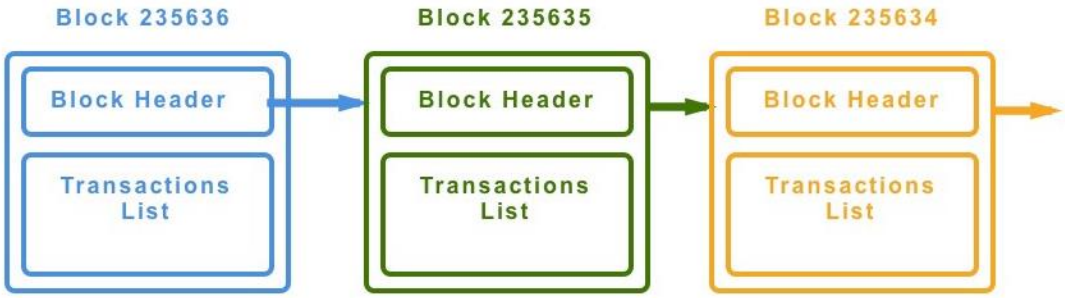

Figure 2.5: Block Header [36]

The previous hash (hash of the previous block) is present to ensure a link between the blocks. Each block can be uniquely identified by its block hash which is generated from the four parameters in the block header. Now, since this is a peer-to-peer network, every peer may not receive the transactions in the same order. So, based on their arrangement of transactions, the block hash might be different as mentioned in [37]. Only one of all these blocks can be added to the blockchain because only a single record of a transaction can ever exist. Also, every peer cannot have their own version of a block as it puts the whole system at stake. So, there should be a way for all the peers to agree upon a single block as valid so that only that block gets added to the blockchain. To achieve this consensus, the bitcoin has a rule that the block hash should have a certain no. of leading zeros as mentioned in [38]. So, all the nodes have three inputs with them – previous hash, timestamp, and transaction root. They also know that the output should be in a certain format. Now, they keep adding a random nonce value to the inputs until they arrive at a target hash. Due to the nature of the hash function, it is impossible to guess this nonce value. The only process to find this out is through trial and error method as mentioned in [39]. The difficulty is exponentially proportional to the no. of leading zeros in the block hash. This nonce value is very difficult to find and often requires performing billions of hashes. This process is not energy efficient and requires special hardware with huge computational capacity as mentioned in [40]. Once this nonce value is found out, the node broadcasts the entire block to the network and it is very easy for other nodes to verify if this is the correct nonce. This nonce value proves that a node has done a significant amount of work before generating a block and hence the name Proof-of-Work.

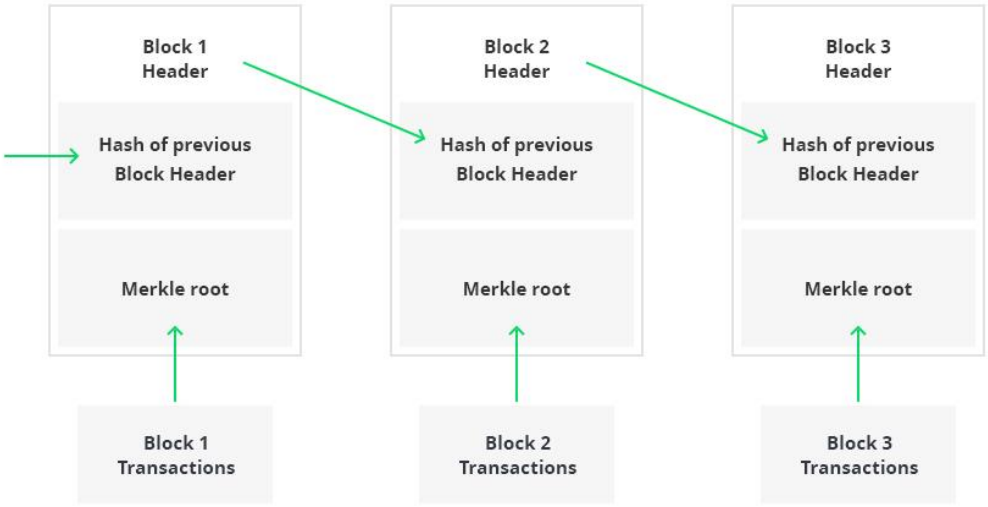

Figure 2.6: Constituents of a Block Hash [41]

After verification, this block is added to the blockchain and all the nodes start working on the next set of transactions.

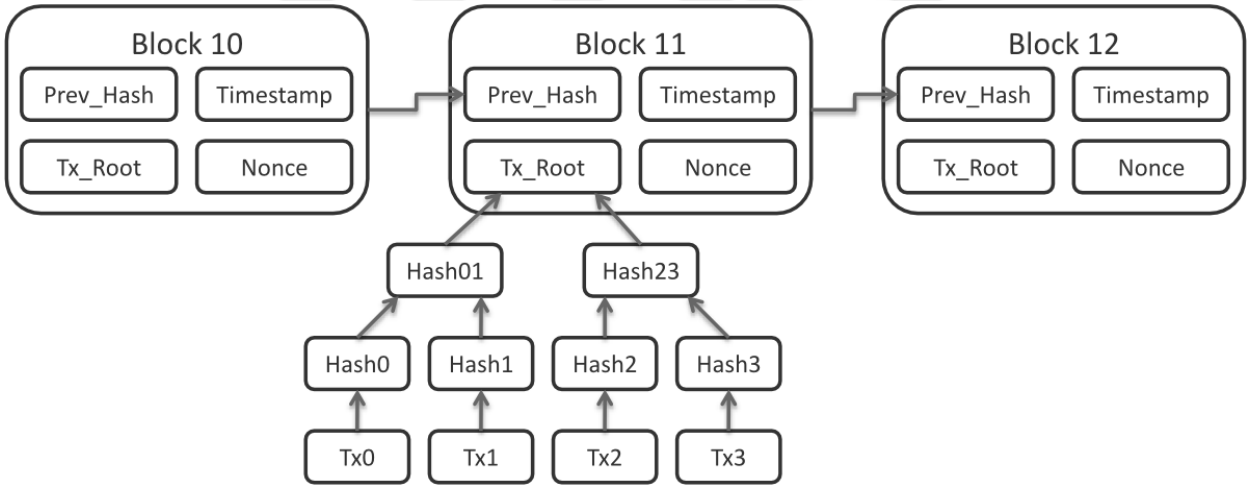

Figure 2.7: Proof of Work in Blockchain [42]

This Proof-of-Work mechanism is what makes the blockchain immutable. Imagine a scenario where a hacker wants to delete a transaction (Tx1) from Block 11 as shown in the above diagram as mentioned in [43]. Here, all the other nodes are working on generating Block 13. So, if the hacker wishes to delete Tx1, the Tx_Root of Block 11 changes and the previous nonce value is no more valid. The hacker needs to perform those billions of calculations again and find out a new nonce value. Since the hash of Block 11 (Prev_Hash) is also used while calculating the hash of Block 12, the whole process should be repeated again to find a new nonce value for Block 12.

It doesn't stop here as the hacker should also create Block 13 before any other node does because all the nodes always follow the longest chain as this is the chain in which the majority of the CPU power is invested as mentioned in [44]. In short, a single node should modify 2 blocks and create a new block while all the other nodes are trying to generate that new block only. Thus, it is impossible for a hacker to modify the data present on the blockchain. This whole Proof-of-Work and block generation mechanism keeps the blockchain secure, transparent, and immutable as mentioned in [45].

2.3 LIMITATIONS OF BITCOIN BLOCKCHAIN

To unleash the true potential of the blockchain, the technologies and protocols used in the Bitcoin blockchain had to be tweaked. Following are some of the limitations which might hinder the application of blockchain to a wider range of industries:

Block mining time: In Bitcoin, the time required to mine each block is approximately 10 minutes. Bitcoin maintains this by setting a hash difficulty for each block. While this may be acceptable for financial transactions, it is a major drawback for digital interactions between IoT devices as mentioned in [46]. There might be a need to perform an immediate action, within seconds, resulting from a transaction of information between two devices. So, there is a need to decrease the block interval by manipulating the hash difficulty.

The Size of the distributed Ledger: Since every record, right from the genesis block, is stored in the distributed ledger, it occupies a lot of space. The current size of this distributed ledger is approximately 90GB. It is impossible for all the IoT devices to have such storage capacity. Pruning is already being implemented and this can drastically reduce the disk usage as mentioned in [47].

Block Size Limit: The block size in Bitcoin is limited to 1MB. All the blocks greater than this size are considered to be invalid and cannot be mined. This block size limits the no. of transactions per second. Currently, there are 3tps (transactions per second) and Bitcoin network supports a maximum of 7tps. There are a lot of debates going on regarding this block size. In an industrial IoT setting, the architecture should support many transactions per second. So, there cannot be any limitations on the block size as mentioned in [48].

2.4 RISKS OF ADOPTING BLOCKCHAIN

Technology Change: While blockchain presents an environment of non-tangible trusted third parties to its customers, it is a time consuming process for the technology to be unanimously welcomed. Customers must get comfortable with trusting the blockchain for electronic payments and transactions as mentioned in [49].

Bootstrapping: It would be a time consuming and cost intensive process to migrate existing contracts and business documents or framework to a blockchain methodology. For instance, documents related to liens which are smart companies must be converted to equivalent blockchain form which could be a challenging process as mentioned in [50].

Government regulations: It would be difficult for blockchain technology to be adopted by the existing financial institutions if the government regulatory status is unsettled as mentioned in [51].

Quantum computing: The primary concept behind the strength of blockchain is that it would be practically impossible for a single party to break the system due to the lack of computer power that is needed to break it as mentioned in [52]. But by using quantum computing, it could be very easy to brute force the cryptographic keys in less time. However, there is also a possibility that the keys can get even stronger which would make such attacks less likely.

Capital costs: Blockchain is very useful for cutting the transaction costs but the capital costs are inevitably high as mentioned in [53].

2.5 ETHEREUM FOR CONTRACTING

Ethereum is an open source project built by developers around the world similar to the Bitcoin protocol but much more adaptable and flexible since it allows users to build and utilize the decentralized applications running on the underlying blockchain technology as mentioned in [54]. Generally, there are two approaches for building a blockchain application: Starting an independent network or establishing a protocol on top of Bitcoin. Ethereum introduced a blockchain with a built-in Turing-complete programming language that allows anyone to write smart cities and decentralized applications with customized rules and parameters as mentioned in [55].

Contrary to the states in Bitcoin, Ethereum has accounts. The two types of accounts are: (1) Externally controlled accounts (or) User accounts, and (2) Contracts, i.e., Snippets of code. Transactions can be initiated from both type of accounts but the contracts can start a transaction only as a result of other transactions they have received. Contracts are written in a high-level programming language (e.g. Solidity) which is further converted into Ethereum Virtual Machine

(EVM) bytecode as mentioned in [56]. The built-in currency for the Ethereum network, Ether, can be used to exchange digital assets and it also provides a mechanism to pay the transaction fees. The smallest denomination of Ether is Wei (10^{18} Wei = Ether). In Ethereum, there is no block size limit like in Bitcoin, but there is a concept called "Gas". In Ethereum, all programmable computations including creating contacts, executing operations, and making message calls have universally agreed cost measured in terms of gas as mentioned in [57]. Instead of a block size limit, there is a gas limit (set by the sender of the transaction) for every transaction which means that the validation of that transaction should not use more gas than the mentioned limit. The remaining gas that is unused at the end of the transaction is refunded to the sender account as mentioned in [58]. Also, the block mining time in Ethereum is significantly reduced to an average of 15 seconds when compared to Bitcoin's 10 minutes. This is done by the implementation of GHOST protocol, which is a policy for selection of the main chain in the block tree as mentioned in [59]. However, this time can be further reduced depending on the size of transactions and the computational difficulty for validating a block.

2.6 GHOST PROTOCOL

Well-formed blocks which are verified by some nodes but have the same block – height (number of blocks away from the genesis block) since they are verified by the nodes at around the same time, cause a fork in the blockchain and only one of the blocks is added to the blockchain, casting off the other block which eventually becomes a stale block as mentioned in [60]. In Ethereum, these stale blocks are termed 'uncles'. "Greedy Heaviest Observed Subtree" GHOST protocol was proposed to combat the problem of reduced security and having a high number of stale blocks in blockchains with fast block time. In a situation where we desire a short block time and a minimized incentive for pooled mining, GHOST can be helpful since it involves the uncle blocks too in the calculation of the longest chain with highest cumulative difficulty. In order to solve the issue of centralization bias, block rewards are extended to the uncle blocks and nephew blocks (child of uncle block) too with the uncles and nephews receiving 87.5% and 12.5% of the base reward respectively.

The GHOST protocol implemented by Ethereum involves seven levels. Each block states its parent and the number of uncles. The uncles of a block have the following features. The uncle block must be a direct child of the ancestor of the block B. The generation of the ancestor could lie between 2 and 7 but the uncle cannot be an ancestor of B. The uncle block does not necessarily have to be

verified previously but has to be a valid block header. Each uncle block needs to be different from the uncles in the same block and those of the previous blocks. The uncle block's miner receives 93.75% of the block reward and an additional 3.125% reward for each uncle it mines as mentioned in [61].

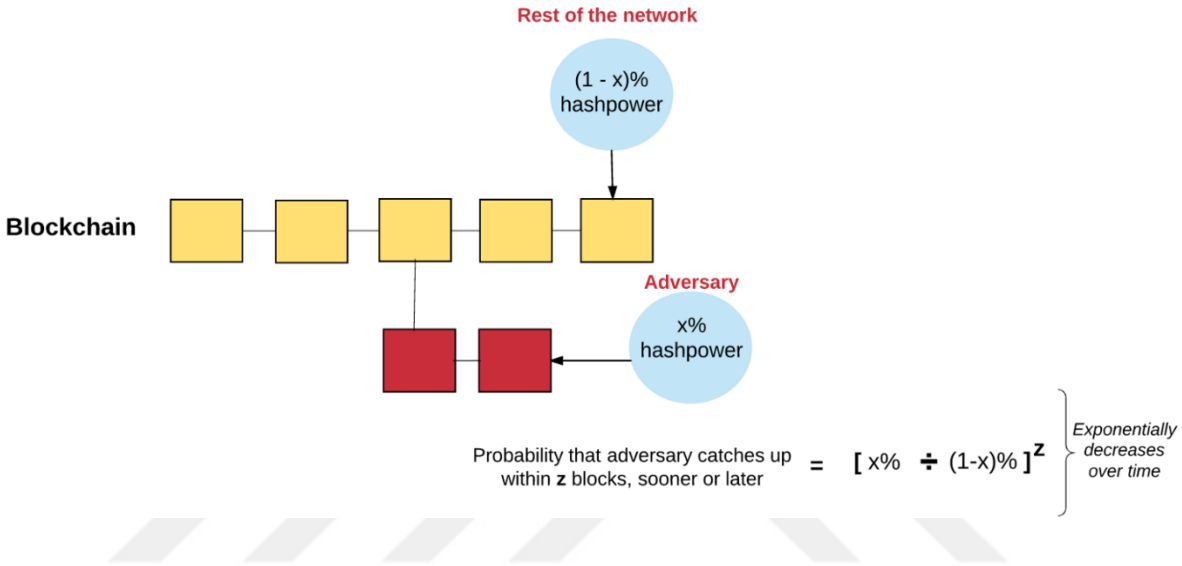

Figure 2.8: The Ethereum Blockchain using the Ghost Protocol Architecture [62]

3. METHODOLOGY

The idea behind "smart cities" is to change the behavior of something, based on the input (data), when certain conditions are fulfilled. In the context of a distributed ledger system, a smart city is a piece of code which is stored at a particular address on the blockchain. The address for storage is decided when creating a smart city. When a transaction is sent to this address, the code in the smart cities gets executed, which might result in reading/writing data to the distributed ledger on which it is present. A smart city can make decisions, read data, and execute other contracts.

While dealing with IoT devices, we would essentially want to perform some action based on the received data. For example, imagine a lux sensor that sends data to a system every one minute. Based on this data, that system will have to switch on/off the lights surrounding that sensor. If all the devices are connected, this might be possible in a normal setting. But, for this to work with an application built on the blockchain technology, smart cities are required because they provide the logic for the on/off operations. Fortunately, Ethereum's implementation of smart cities is completely deterministic and isolated i.e., a smart city won't have access to data which is external to the blockchain. It can only work with on-chain data. After coding a smart city in a high-level language like Solidity or Serpent, the code will be compiled into EVM (Ethereum Virtual Machine) bytecode and will be stored at an address on the blockchain. It gets invoked when a transaction is sent to that address.

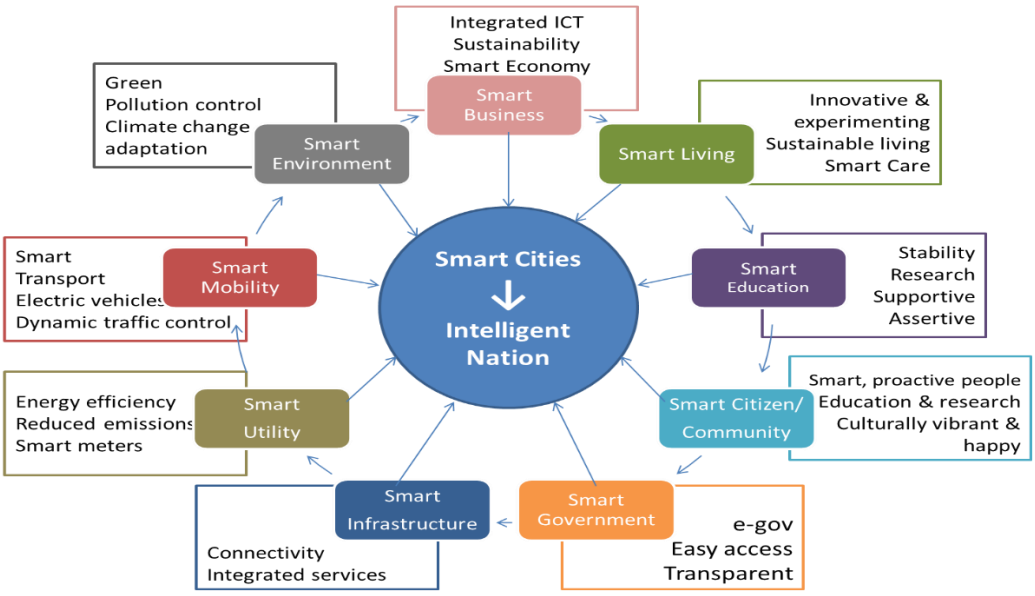

Figure 3.1: The major benefits of blockchain in smart cities.

3.1 SMART CITIES IN BLOCKCHAIN

Smart cities, in order to perform an action, might need data from external location. Cryptographic signatures submitted by outside systems called ‘Oracles’ are relied upon to interact with the real world. So, oracles are trusted systems which provide signed data to smart cities from the outside world. When triggered, smart cities are executed on all the nodes on which the ledger is present. But, when the code is based on advanced logic, the integration of smart cities in the blockchain might get complicated. Smart oracle is a way of implementing smart cities without making the consensus network complicated. It is a separate trusted entity which can gather information from the outside world and execute the code which pertains to the smart cities. So, the code for the smart cities is separated from the blockchain. Instead of being executed on all the nodes, the code gets executed on a single oracle. Although this concept is being discussed for a long time now, there are only a handful implementations on this. Currently, Contracts and Reality Keys seems to be the most advanced companies in providing these oracle services. Following are some of the data sources that are supported by the contract service.

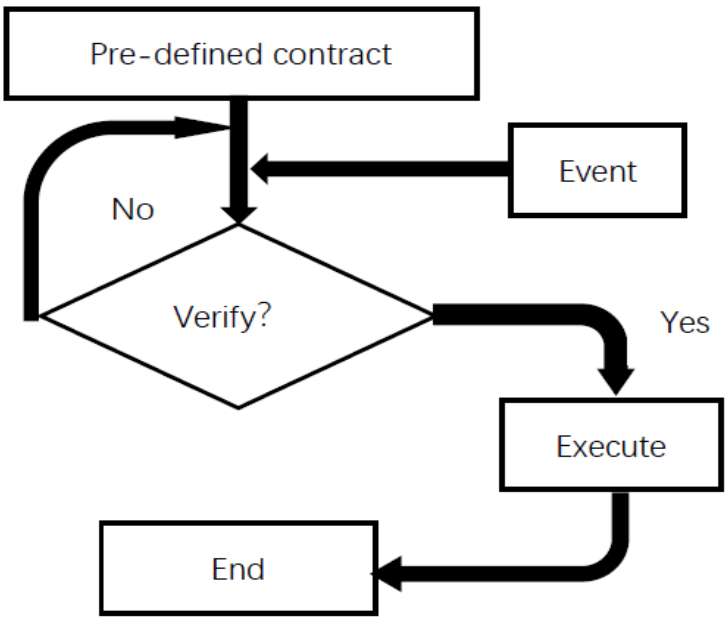

Figure 3.2: Data flow in smart cities supported by blockchain.

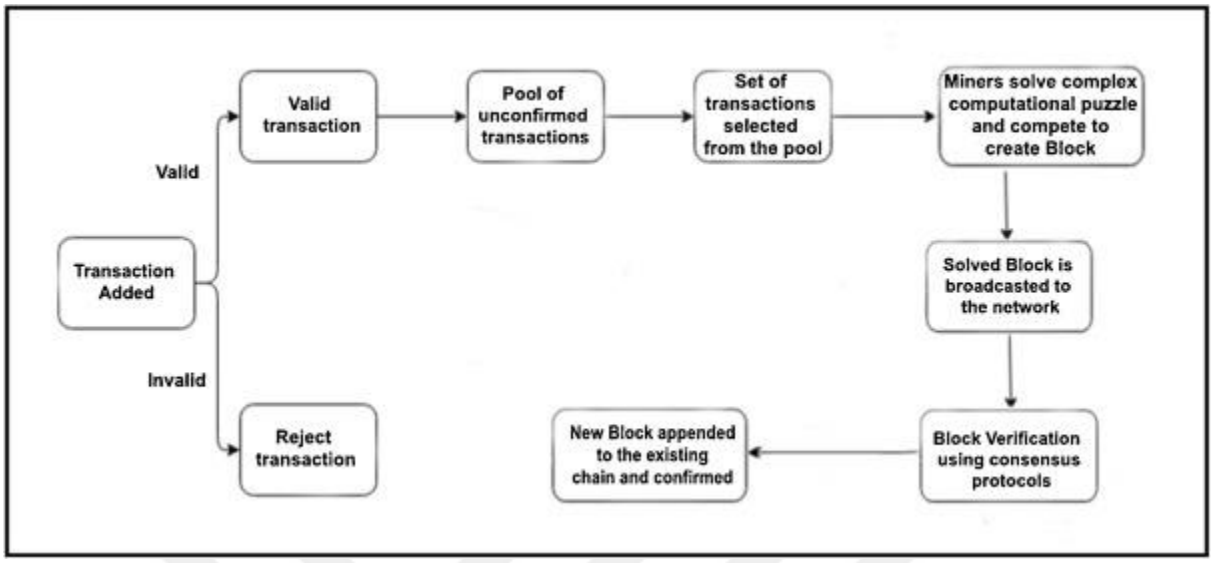

Figure 3.3: Working of Blockchain Service

URL is the most generic data source which can be used to access any API and any page on the internet. Oraclize will fetch the responses for these HTTP requests and feed it to smart cities with an attached proof. Blockchain related data which cannot be accessed by smart cities directly can also be fetched through oracles. Data like blockchain height, difficulty, and balance of a specific address can be retrieved. To get data about the weather in a location, solution to mathematical problems, and any other simple statistics and data analysis, a computational knowledge engine like Wolfram Alpha can be used as a data source. The response from the Alpha API will be sent to the smart cities. If the smart cities require a file which is stored in a distributed fashion using the Inter Planetary File System (IPFS) protocol, the multihash of that file can be used to retrieve the contents of that file. So, IPFS can also be used as a data source.

3.2 PUBLIC VS PRIVATE BLOCKCHAINS

Public blockchains can be accessed by any user across the world, valid transactions can be sent and added to the blockchain. Also, users can contribute to the consensus process which determines the blocks which are added to the blockchain. These blockchains are fully decentralized and they are secured by a combination of rewards-based incentives and cryptographic verification schemes like proof of work/proof of stake.

Private blockchains - If an organization decides to run a blockchain application restricted to the users in that organization a private blockchain is a better choice where the write permissions are centralized for the organization and the read permissions could be public or restricted depending

upon the organization's needs. The transactions turn out to be cost effective in private blockchains since a large number of nodes are not required for the verification process thereby reducing the processing power.

3.3 ALGORITHMS AND PROTOCOLS

3.3.1. Elliptic Curve Digital Signature Algorithm (ECDSA)

Ethereum uses ECDSA for generating public, private keys and performing digital signature creation, verification. This comes under Elliptic Curve Cryptography (ECC), which is considered as the next generation of public key cryptography because of its significant advantages over current generation algorithms like RSA and DSA. Although Elliptic Curve Cryptosystems have been around for a long time, they became famous with the usage of ECDSA in Bitcoin. In ECC, the elliptic curve is generally of the form $y^2 = x^3+ax+b$. Like other group-based cryptographic algorithms, ECC also requires a finite field of elements, which usually contains integers modulo a prime k, and suitable security is achieved when k is close to 160 bits. The private key (p) is a random integer which is less than the order of the generator n and the public key (P) is calculated by multiplying the private key with a point on the elliptic curve (G) i.e., $P = pG$. In ECDSA, key pairs are generated with reference to a particular set of domain parameters such that the public key (P), the private key (p) and this set of domain parameters are mathematically related to each other. In Ethereum, a 32-byte private key and a 64-byte public key is generated for every account. This public key undergoes a Keccak-256 hash resulting in a 32-byte output hash string. The right most or the last 20-bytes of this hash string is used as an account address. This account address is usually prefixed with a '0x' to represent that it is in Hexadecimal format. The mathematical and implementation details for ECDSA are quite complex and not easily understandable, which is one of the reasons why it is not being implemented at a large scale. In short, it can be thought of as an elliptic curve analog to Digital Signature Algorithm. As far as security is concerned, even after decades of research, cryptographers couldn't come up with a computationally practicable method to solve the discrete-logarithm for elliptic curves.

One of the primary advantages of ECC over traditional asymmetric key algorithms is the key size. Currently, RSA is the most popular and considered a secure encryption algorithm but, a 256-bit ECC key offers the same level of security as a 3072-bit RSA key. The following table offers a more detailed view of key comparisons:

Table 3.1: RSA vs ECC Key Size Comparison

RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
1024	160
2048	224
3072	256
7680	384
15360	521

So, even with a shorter key length, it requires more time to break an ECC key than an RSA key. Following is a graph comparing the key size and the MIPS years to break it for RSA, DSA, and ECC.

One more interesting way to compare the difficulty levels of algorithms is that "a cryptosystem is said to offer pool security if breaking it requires as much energy as it takes to boil a single Olympic size swimming pool (i.e., 2500 cubic meters of water)." By this standard, it takes less energy to break a 228-bit RSA key than to boil a teaspoon of water. Conversely, it would take more energy to break a 228-bit ECC key than to boil all the water on earth.

Also, an ECC key requires less storage space when compared to an RSA key. The key generation time is faster in ECC and the encryption and decryption process utilizes less CPU. With all these advantages, ECDSA is suitable for high-end devices like laptops and desktops as well as low-end IoT devices like raspberry pi and other embedded systems. ECDSA offers efficient digital signature creation and verification services with less hardware and software resources while maintaining strong security.

3.3.2. Keccak-256

Keccak is a family of sponge functions which forms the basis for SHA-3 standard approved by the National Institute of Standards & Technology. Usually, in primitive hash functions, the length of the output hash string is fixed, no matter what the input size is. But, in the case of a sponge construction, the hash function maps variable-length input to variable-length output. Cryptographic hash functions are said to be secure if their behavior is close to a random oracle i.e., a truly random output string is returned for every unique query. Since all the hash functions in use operate on a finite memory, there is a chance for collisions i.e., different inputs might result in the same output. So, in practice, it is impossible to implement a truly random oracle. But, the creators

of Keccak claim that a random sponge, which can be obtained by applying the sponge construction with a random permutation, offers security that is close to a random oracle. So, random sponges can act as a substitute to random oracles in terms of providing security.

The sponge construction operates on a state of $b=r+c$ bits where b denotes the width of the permutation, r is the bitrate, and c is the capacity. It is recommended to use the largest permutation ($b=1600$) but lower permutations (as low as $b=25$) are also available for environments with constrained resources. Each permutation consists the iteration of a simple round function. The round function is a permutation-substitution network with 5-bit S-boxes. The choice of operations in round function is limited to rotations, bitwise XOR, AND, and NOT.

The arbitrary-output length makes Keccak applicable to various use cases of which tree hashing is one. This makes it suitable for building transaction trees in the Ethereum blockchain. Ethereum uses both Keccak 256-bit and Keccak-512-bit hash values. For Keccak 256, the parameters used are $r=1088$, $c=512$, $n=256$. For Keccak 512, $r=576$, $c=1024$, $n=512$. Here, n represents the no. of rounds. Complements previous standards like SHA-2 but offers better security and resistance to known attacks. All the popular hashing algorithms like MD5, SHA1, and SHA2 are based on the *Merkle-Damgard* construction which provides a method for building collision-resistant hash functions. NIST wanted the algorithm for SHA3 standard to be resistant to attacks that can be performed on SHA2 so that it can act as a substitute to SHA2 if need be. With this principle in mind, Keccak used the sponge construction instead of *Merkle-Damgard*, so that any successful attack on SHA1, SHA2 will not have a potential threat on SHA3. Keccak was subject to cryptanalysis and is provably secure against generic attacks. Also, Keccak does not have the length-extension weakness which was present in SHA1 and SHA2.

While maintaining better security, Keccak has better performance in both hardware and software. Keccak is excellent in hardware performance and outperforms SHA2 by an order of magnitude. Throughput per area is considered as a major deciding factor while evaluating hardware performance and following is a comparison of 256-bit and 512-bit variants of SHA3 finalists:

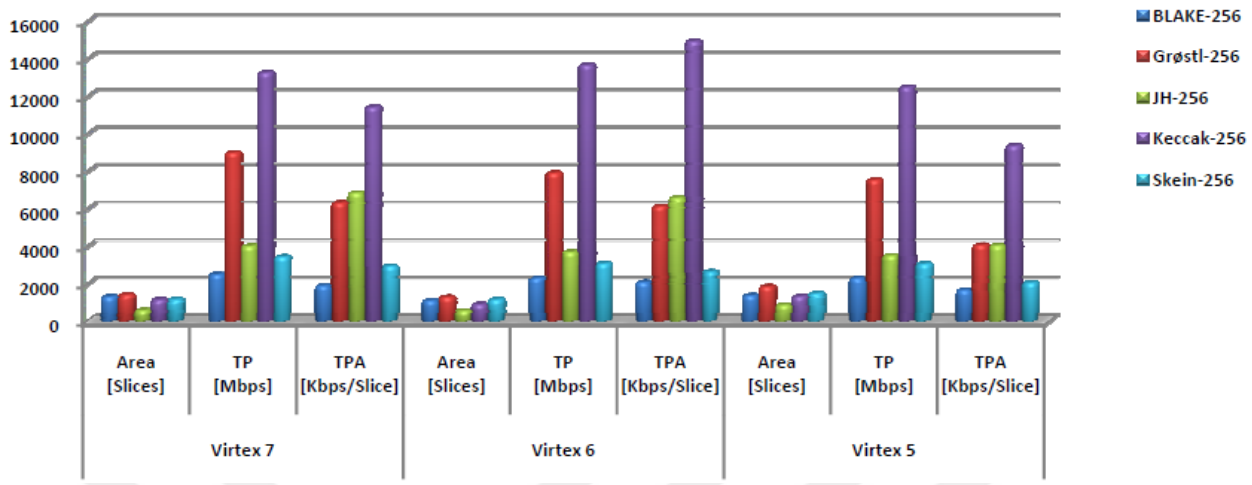

Figure 3.4: Hardware performance comparison of 256-bit SHA3 finalists

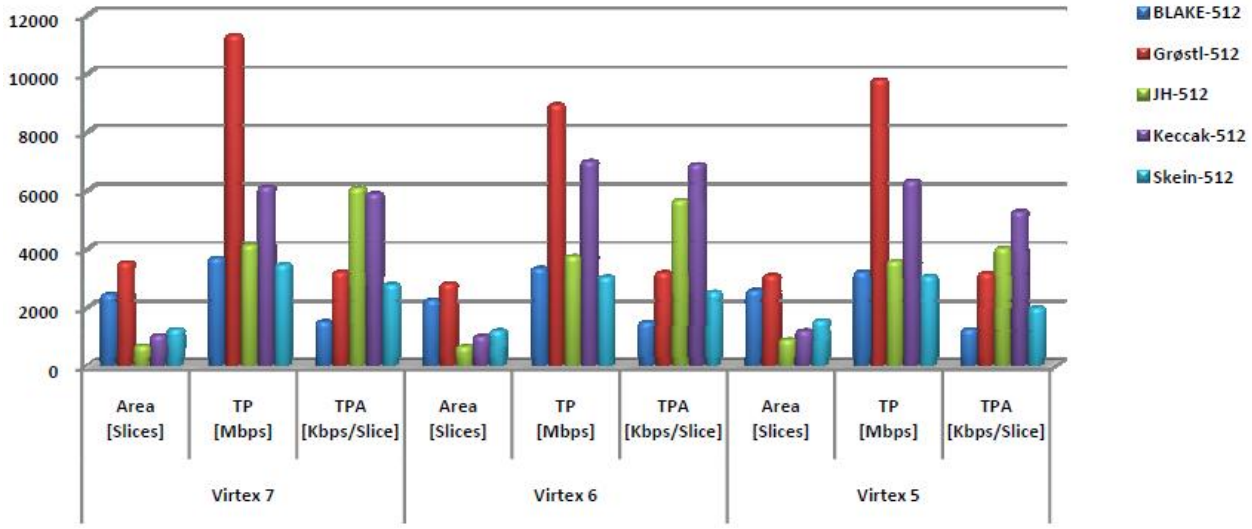

Figure 3.5: Hardware performance comparison of 512-bit SHA3 finalists

On the other hand, Keccak has an overall good software performance and is faster than SHA2 on all modern PCs. Following table offers a comparison of different algorithms in terms of cycles/byte and strength. Finally, it should be noted that SHA2 is the current NIST standard for hashing algorithms and any practical attack on SHA2 is not imminent. SHA3 was meant to be the fail-safe algorithm and is compatible with SHA2 message digests while offering better strength and efficient performance.

3.3.3. Kademlia

In Ethereum, the nodes communicate with each other through Recursive Length Prefix (RLP) protocol, an encrypted and authenticated transport protocol which is used as an encoding method to serialize objects. Ethereum uses a Kademlia-like peer-to-peer Distributed Hash Table (DHT) to discover and establish connections with peers. This table in each peer consists a maximum of 255 rows and each row consists information of 5 to 20 peers. The information about a peer might include the peer address, network address, timestamp, peer's signature and other meta-data. The identity of each node (node ID) is the 256-bit cryptographic hash of the node's public keys, while in the original Kademlia protocol, node IDs are random 160-bit numbers.

The original Kademlia protocol consists of four Remote Procedure Calls: PING, STORE, FIND_NODE, and FIND_VALUE but in the Ethereum version, FIND_VALUE and STORE packets are not implemented. The packets are signed in Ethereum unlike in the original Kademlia implementation. The receiver retrieves the public key from the signature and verifies the signature for authenticity. A request through these RPCs returns the *(IP address, UDP port, Node ID)* from the target node and the default port for peer discovery in Ethereum is 30303. In Ethereum, there exists few nodes which are always online and reliable. These nodes act as an entry point to the Ethereum network and all the new nodes that weren't a part of the network before, connect to these nodes first. These are called *bootstrap* nodes and are hardcoded into Ethereum clients. Bootstrap nodes store the details of all the nodes connected to them within a period of time (which is predefined) and share these details about other peers to the newly joined nodes. Go-Ethereum clients, for the frontier network, have 3 bootstrap nodes hardcoded into them:

Once a newly joined node establishes a connection with the nearby nodes, it may prune the connections to bootstrap nodes since they're no longer required in the peer discovery mechanism. If Universal Plug and Play (UPnP) is enabled on the router, an Ethereum node also accepts incoming requests and connections as well. The connections in Ethereum are established through an encrypted handshake which is carried out in two phases. The first phase is the key exchange involving the transmission of key material derived through a Key Derivation Function (KDF) with Elliptic Curve Diffie-Hellman Key Exchange (ECDHE) - derived keying material. This key exchange takes place through an Elliptic Curve Integrated Encryption Scheme (ECIES) encrypted message which includes ephemeral keys for perfect forward secrecy. The second phase is the authentication and protocol negotiation phase and the capabilities that each node supports are

exchanged during this phase. Two kinds of connections are possible in Ethereum, a node can connect to a known (previously connected) peer, or a node can connect to a new peer. In either case, the handshake should be successful i.e., if the handshake fails while initiating a connection to a known peer, the peer information should be deleted from the Distributed Hash Table and the connection to this peer should not be re-attempted. Once the connections are established, further communications are encrypted using the Advanced Encryption Standard (AES) 256-bit key in the counter (CTR) cipher mode. Kademlia is an efficient protocol for finding and connecting to peers. For a network with *n* nodes, a single node contacts a maximum of *O (log (n))* nodes during the search. Also, the decentralized structure of the Kademlia protocol helps in preventing Denial of Service (DoS) attacks.

3.3.4. Ethash

Ethash is Ethereum's Proof-of-Work system which is a combination of modified versions of Dagger and Hashimoto algorithms. This new system was developed to overcome the problems prevalent in other cryptocurrencies, the major one being ASIC-resistance. In most of the cryptocurrencies, mining is becoming more and more centralized due to the development of ASIC machines and certain companies building huge mining farms. The primary intention of Ethereum's Ethash was to make CPU mining feasible and ASIC mining unprofitable thereby keeping the network decentralized. Even if ASICs are developed, the idea was to minimize the speed at which the mining can be done with ASICs, so that it is marginally profitable for regular users to mine with CPUs. The Hashimoto algorithm is responsible for achieving ASIC resistance by making memory reads the limiting factor during mining. On the other hand, the Dagger algorithm is designed to achieve memory-hard computation and memory-easy verification (otherwise called as light client verifiability) by using directed acyclic graphs. For the PoW function to work, the client needs to select subsets of fixed dataset that is dependent on the nonce and block header. This dataset is called DAG and is approximately 1 GB in size. This is one of the differences between Ethash and the original Hashimoto algorithm. The Hashimoto algorithm uses the blockchain as the data source while Ethash uses this custom generated DAG dataset. Following are the parameters for the algorithm:

```
WORD_BYTES = 4                    # bytes in word
DATASET_BYTES_INIT = 2**30        # bytes in dataset at genesis
DATASET_BYTES_GROWTH = 2**23      # dataset growth per epoch
CACHE_BYTES_INIT = 2**24          # bytes in cache at genesis
CACHE_BYTES_GROWTH = 2**17        # cache growth per epoch
CACHE_MULTIPLIER=1024             # Size of the DAG relative to the cache
EPOCH_LENGTH = 30000              # blocks per epoch
MIX_BYTES = 128                   # width of mix
HASH_BYTES = 64                   # hash length in bytes
DATASET_PARENTS = 256             # number of parents of each dataset element
CACHE_ROUNDS = 3                  # number of rounds in cache production
ACCESSES = 64                     # number of accesses in hashimoto loop
```

Figure 3.6: Parameters for DAG

For each block, a seed value can be calculated by skimming over the block headers until that point. A pseudo random cache that is approximately 16 MB in size can be computed from the seed value. From this cache, the DAG dataset is generated such that each item in this dataset depends on a small number of items from the cache. During mining, random slices from this dataset are accessed and hashed together so that the resulting hash string will result below a threshold depending on the difficulty thereby making the PoW system memory-hard.

Initially, the DAG is 1GB in size and the size of this dataset grows linearly with every epoch period i.e., every 30,000 blocks or 100 hours. For every epoch, a new dataset is generated that is approximately 0.5 GB in size. So, unlike the original Dagger algorithm, the dataset in Ethash is semi-permanent and is updated periodically. These DAG files take some time to generate, and if DAG generation happens during the epoch transition, the network might experience a delay. If not mentioned explicitly, geth automatically generates and maintains DAGs for two epochs to ensure a smooth transition. Since DAG only depends on the block number, it should be pre-generated and stored beforehand to avoid long epoch transitions.

3.4. DEPLOYING A PRIVATE BLOCKCHAIN USING ETHEREUM ON RASPBERRY PI DEVICES

We are using Raspberry Pi devices to simulate nodes on an IoT network upon which a private blockchain network is deployed. The specifications of the devices are as follows:

3.4.1. Specifications

Number of processors : 4

Model name : ARMv7 Processor rev 4 (v7l)

BogoMIPS : 38.40

Features: half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstrm crc32

CPU implementer : 0x41

CPU architecture : 7

CPU variant : 0x0

CPU part : 0xd03

CPU revision : 4

Hardware : BCM2709

Revision : a22082

Serial : 00000000ae85d2bb

Raspberry Pi OS version for deploying a private blockchain using ethereum, we have used a multipurpose command line interface tool called Geth which can run a full ethereum node. This tool is implemented in python and supports mining, fund transfer, creating contracts, sending transactions and many more.

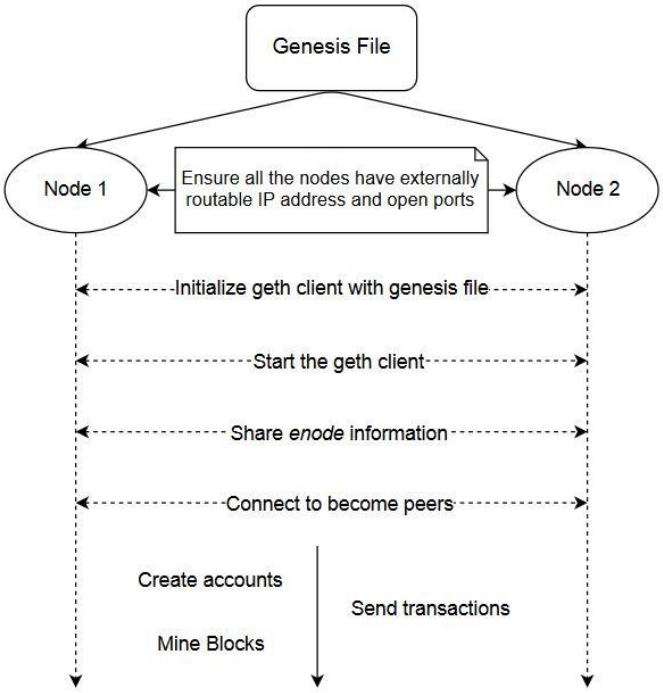

Figure 3.7: Working with geth

3.4.2 Custom Genesis Block

The starting block of a blockchain is called the genesis block and this does not point to any predecessor blocks. It is the entry point into the blockchain database and defines all the initial parameters of blockchain database. The genesis block is created by using a JSON file which contains all the required parameters.

```
{

"nonce": "0x0000000000000042",
"mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",
"difficulty": "0x200",
"alloc":
{
"bc28fe2c73fdb9cd023d84f98345e057384a5311":
{"balance": "7000000000000000000"}
},
"coinbase": "0x0000000000000000000000000000000000000000",
"timestamp": "0x00",
"parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
"extraData": "Capstone Research",
"gasLimit": "0xffffffff"
}
```

Figure 3.8: Custom Genesis. Json file

3.4.3 Operating on a Private Blockchain

While running multiple Ethereum nodes in a private blockchain:

a. All the nodes should have the same genesis file and same network ID.
b. Each node should have a different empty folder for data storage (*--datadir*). All the blockchain data including account names and nodes is stored in this directory.
c. Each node should run on a different ethereum (*--port*) and rpc port (*--rpcport*) if multiple nodes are deployed on the same subnet.
d. If the ipc interface is not disabled, each node should have a unique ipc endpoint. This can be mentioned using the *--ipcpath* option.
e. If a –datadir value is not specified at the launch of geth then the default values are used. The default directory for different OS are mentioned below:
    a. Mac: ~/Library/Ethereum
    b. Linux: ~/. ethereum
    c. Windows: %APPDATA%/Ethereum

3.5 CONFIGURATION ISSUES FACED WHILE RUNNING GETH ON PI

3.5.1 Network Connectivity Issues

a) Trouble connecting multiple nodes on the same network with nodes on a different network

At the time of this project, we faced difficulties in connecting nodes on the same private network with nodes on different external networks as nodes on the same private network share common public IP address. We tried running geth on different ports on these devices in the same network but still couldn't fix this issue. So, we have deployed all the nodes on different private networks as a work around and have proceeded with our research.

b) Issue with keeping all the nodes connected to the network at all times

In the private network, we need to add all the peers manually by using the *addPeer()* function. Often, those connections are lost due to many problems. The common problems are: 1) Out of sync local time and 2) Firewall configurations. All the Ethereum nodes which are connected to each other should maintain the same time. Subtle differences are acceptable but if the time difference between nodes is more than 10 seconds, they'll automatically get disconnected. Similarly, if the firewall configurations do not allow UDP traffic to flow through,

the nodes might get disconnected. Whatever the reason may be, these nodes need to be connected again and in the private blockchain, adding them manually is the only option. Usually, these nodes get automatically connected whenever there are minor disruptions. Ethereum provides some solutions for dealing with lost connections.

There is a feature called **static nodes**. If there are some peers a node always wants to connect to, details about those peers are placed in *static-nodes.json* file in the data directory (*--datadir*). These nodes are re-connected on disconnects. This is the structure of the file:


```
[
"enode://fa217e9f2e6b243d83e38f528b4b8482880771fbe688c083855a8966dfde3faa1b66dc623346f7aee5745ca00e65180a52a421b614f1f6cbdd360a0759f996d5@52.42.255.126:30303",

"enode://44b3af7a1afb837fdeeefa2fc2115c92349867d1ec3538e86e5fdd8edb64d24757b042adf055e4102c737958a8c4e1f405564d1640eceb939c050072f77fde3f@146.115.90.113:30303",

"enode://6ee65b5d66ae3c4abcda8c8d3947da3430ddb4bd8704029dcd92b301fac5b7f2866623fe4c5e93b88e21aa443fe7ec50a9937e346735df5c91b0c22165cff97a@146.115.82.109:30303",

"enode://65a55a7df7541099d49547ed54a46d78f29bdf5125912bbdde13e6688e42750dc2140d056fee006268ae404baa5861a495c4aa2f2d63b3065c39154b9da4e380@52.34.24.43:30303",

"enode://152930f71148b9ce03f50a11d05c5d69d9964cf478dff4dd51eaab51c5631f06eaf9b5bd68c47b73424c613e52cad700b5f2f11f421e2b1405276572ae50180a@216.15.125.24:30302",

"enode://8725980137c4c0c0366e842f24aa1657148f038153b5503fdfd687fdd5dff87cd4a49a1b548b5048c0fbf91fedb7f3ed9a5cdef9ab21377c971b49940d51ec01@73.69.55.205:30301"
]
```


Figure 3.9: Sample static-nodes.json

There is another feature called as **trusted nodes**. Usually, private blockchains have a *--maxpeers* option to limit the no. of connections to each node (25 by default). But connections from trusted nodes will always be allowed even if the peer limit is reached. These nodes cannot be blocked. Details about these nodes are placed in the *<datadir>/trusted-nodes.json* file. Connections to these privileged trusted nodes are reattempted every 15 seconds.

Also, there is a concept called **bootnodes**. These are a part of the peer discovery protocol in the Ethereum network. One or more geth instances can be nominated as bootnodes so that all the non-bootnode instances first connect to these, in order to find other peers in the private network. Bootnodes can be added in the following way:

geth --bootnodes enode://pubkey1@ip1:port1, enode://pubkey2@ip2:port2

3.5.2 Hardware limitations for mining on Raspberry pi devices

Due to computational limitations, raspberry nodes cannot perform mining and would throw the following errors.

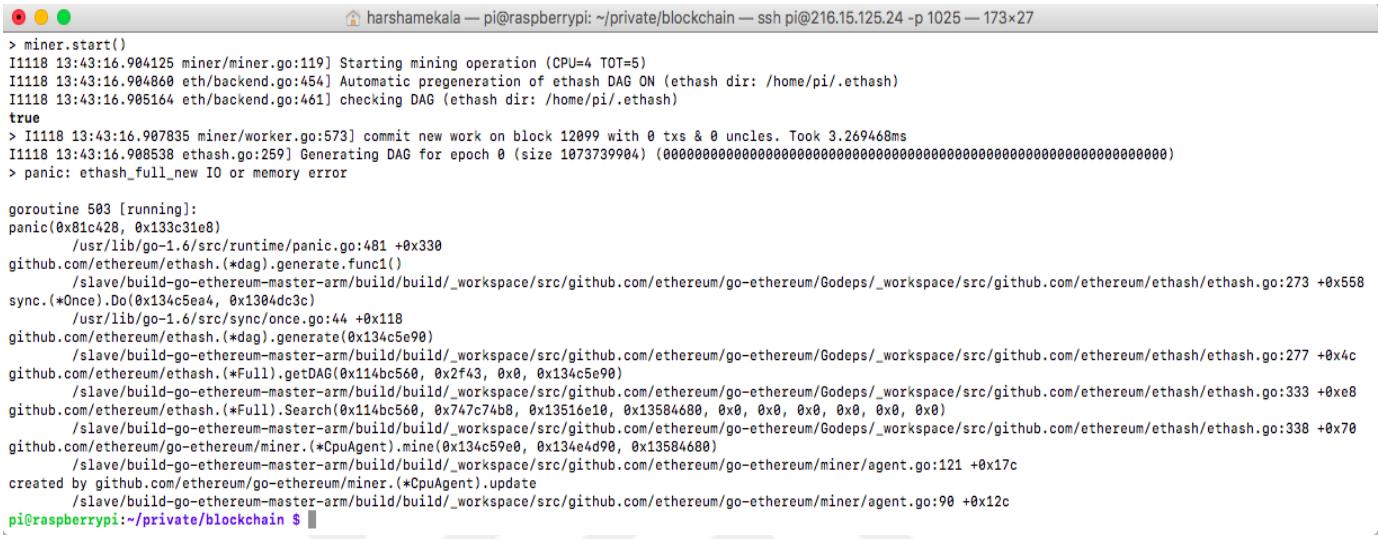

Figure 3.10: Error when mining on Raspberry pi

For this purpose, we are introducing a different node specifically for mining purpose and the system specifications of the miner are as below:

a. Make: Dell Inspiron 14z - 5423
b. Operating System: Windows 10
c. Memory: 4GB
d. Processor: 1.7 GHZ Intel Core i7
e. Graphics: AMD 1GB Radeon

3.5.3 Benchmarking Ethereum

For the purpose of benchmarking Ethereum functionality on IoT devices, we have deployed a private blockchain using three Raspberry Pi devices, one Windows Server 2016 running on Amazon EC2 and a Windows laptop which serves as the miner.

The final state of the blockchain with 12098 blocks upon which the benchmarking is performed is given below.

```
pi@raspberrypi:/home $ geth --datadir "/home/pi/private/blockchain/" --networkid 3141592 --nat extip:216.15.125.24 --port
30302 console
I1117 13:00:23.639161 ethdb/database.go:82] Alloted 128MB cache and 1024 file handles to /home/pi/private/blockchain/chain
data
I1117 13:00:23.686907 ethdb/database.go:169] closed db:/home/pi/private/blockchain/chaindata
I1117 13:00:23.695219 ethdb/database.go:82] Alloted 128MB cache and 1024 file handles to /home/pi/private/blockchain/chain
data
I1117 13:00:23.829000 ethdb/database.go:82] Alloted 16MB cache and 16 file handles to /home/pi/private/blockchain/dapp
I1117 13:00:23.845430 eth/backend.go:172] Protocol Versions: [63 62], Network Id: 3141592
I1117 13:00:23.846337 eth/backend.go:201] Blockchain DB Version: 3
I1117 13:00:23.857242 core/blockchain.go:206] Last header: #12098 [ff43930e…] TD=9740797499
I1117 13:00:23.857947 core/blockchain.go:207] Last block: #12098 [ff43930e…] TD=9740797499
I1117 13:00:23.858397 core/blockchain.go:208] Fast block: #12098 [ff43930e…] TD=9740797499
I1117 13:00:23.867905 p2p/server.go:313] Starting Server
I1117 13:00:23.868966 p2p/nat/nat.go:111] mapped network port udp:30302 -> 30302 (ethereum discovery) using ExtIP(216.15.1
25.24)
I1117 13:00:24.225193 p2p/discover/udp.go:217] Listening, enode://152930f71148b9ce03f50a11d05c5d69d9964cf478dff4dd51eaab51
c5631f06eaf9b5bd68c47b73424c613e52cad700b5f2f11f421e2b1405276572ae50180a@216.15.125.24:30302
I1117 13:00:24.226045 p2p/server.go:556] Listening on [::]:30302
I1117 13:00:24.226249 p2p/nat/nat.go:111] mapped network port tcp:30302 -> 30302 (ethereum p2p) using ExtIP(216.15.125.24)
I1117 13:00:24.232524 node/node.go:296] IPC endpoint opened: /home/pi/private/blockchain/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.4.10-stable-5f55d95a/linux/go1.6.2
coinbase: 0x9c9dd18429b014c295c1afd722020ebd5d6dfa80
at block: 12098 (Sat, 01 Oct 2016 13:08:56 EDT)
 datadir: /home/pi/private/blockchain
 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 3.11: Final State of the Blockchain with 12098 blocks

4. RESULTS

Blockchain application platform enables the building of decentralized applications and smart cities. This platform believes in the notion that bitcoins or any monetary asset values are not essential for the existence of blockchains. Smart cities perceive blockchains as “distributed rulebooks” which can keep a track of the modifications made and the list of permissions assigned by using private keys for a secure data management. Smart city aims at providing developers with blockchain structures that include expansive coding to foster communication. Contrasting smart city for deployment, Ethereum as we discussed is designed to use Ether crypto currency and make transactions on the blockchain. However, smart city focuses on building permissioned blockchains enabling users to define properties for their use cases and achieve secure transmission of data.

4.1 CPU UTILIZATION

4.1.1. Miner

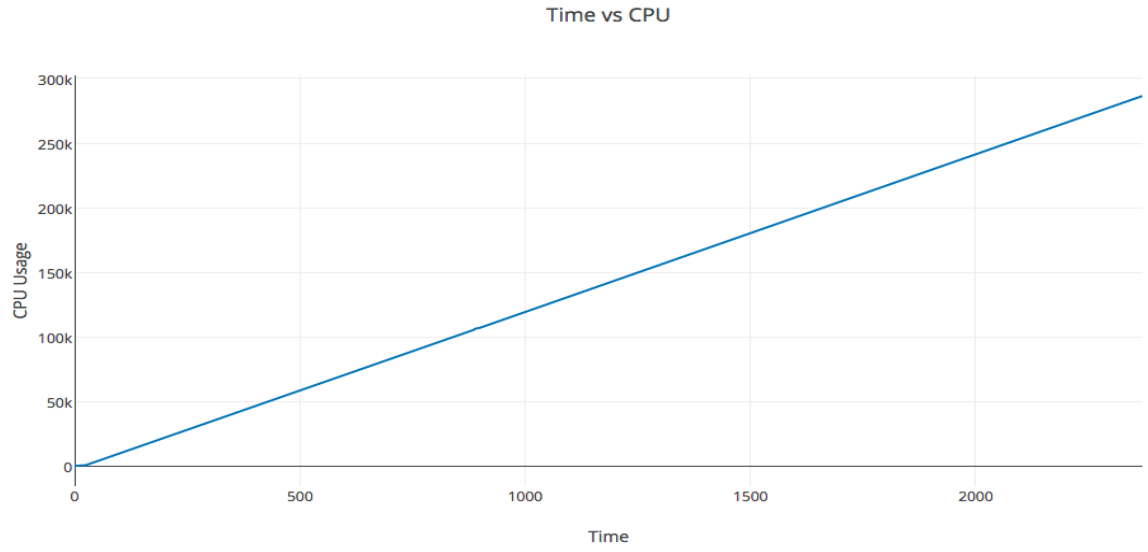

Figure 4.1: Time (2-minute interval) vs CPU resource consumption for Miner Node

We have initiated only CPU mining with a single thread by issuing miner.start(1) and observed that the CPU utilization by process grows exponentially as the blockchain size increases.

4.1.2. Raspberry Pi Nodes

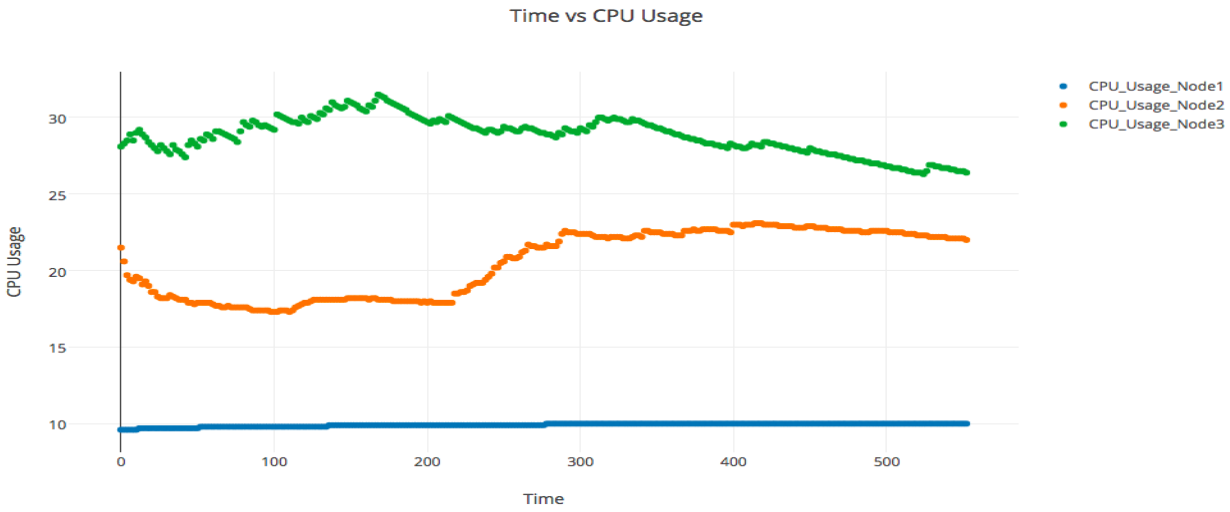

Figure 4.2: Time (2-minute interval) vs CPU resource consumption graph for 3 Raspberry Nodes.

Table 4.1: Statistics for the CPU resource consumption of 3 Raspberry Nodes

Descriptive statistics	Node 1 CPU	Node 2 CPU	Node 3 CPU
N	277	277	277
Min	9.6	17.3	26.3
Max	10	23.1	31.5
Mean	9.913718412	20.50722022	28.81624549
First Quartile	9.9	18.1	28
Median	9.9	21.6	29
Third Quartile	10	22.5	29.7
Standard Deviation	0.105581936	2.164786747	1.208658437
Variance	0.011147545	4.686301659	1.460855218
Standard Error	0.006343804	0.13006943	0.072621247

We have observed that the CPU utilization of process on the raspberry pi nodes didn't fluctuate much and displayed a maximum standard deviation of 2.16 units.

4.2 RAM UTILIZATION

4.2.1. Miner

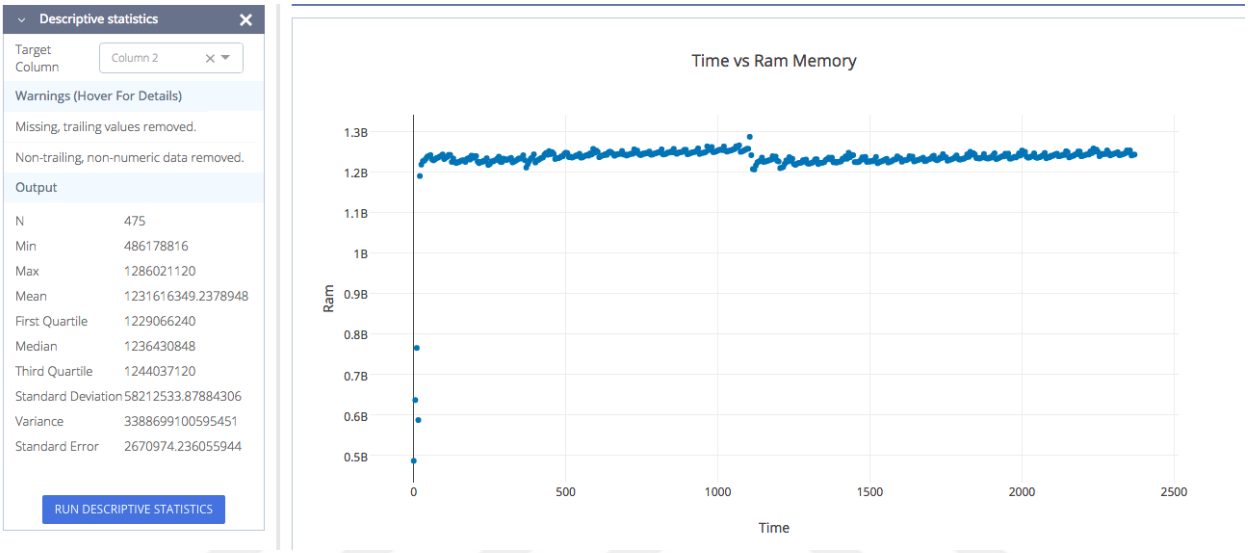

Figure 4.3: Time (2-minute interval) vs RAM (Bytes) consumption graph for Miner Node

It was observed that the consumption of RAM memory by the miner was linear and neither the blockchain size or block difficulty has any effect on the RAM consumption.

4.2.2. Raspberry Pi Nodes

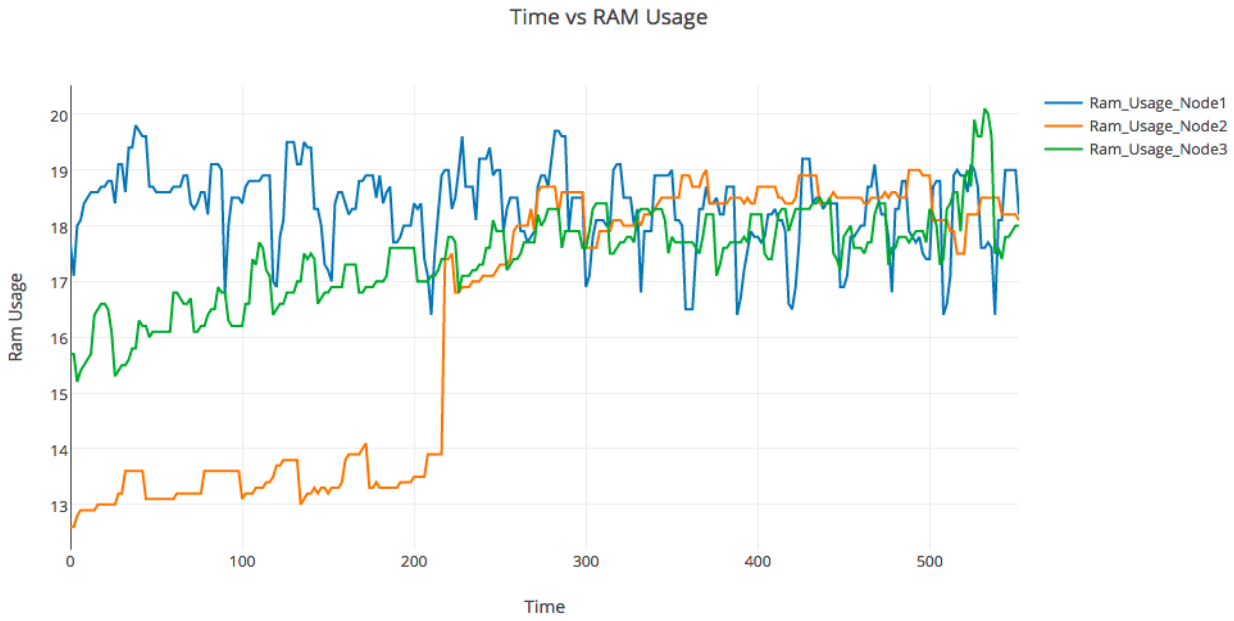

Figure 4.4: Time (2-minute interval) vs RAM (Bytes) consumption graph for 3 Raspberry Nodes.

Table 4.2: Statistics for the RAM (Bytes) consumption of 3 Raspberry Nodes

Descriptive statistics	Node 1 Ram	Node 2 Ram	Node 3 Ram
N	277	277	277
Min	16.4	12.6	15.2
Max	19.8	19	20.1
Mean	18.34115523	16.33068592	17.43429603
First Quartile	17.9	13.5	16.9
Median	18.5	17.9	17.6
Third Quartile	18.9	18.5	17.9
Standard Deviation	0.72587761	2.427362945	0.846492418
Variance	0.526898304	5.892090865	0.716549414
Standard Error	0.043613759	0.145846104	0.0508608

It was observed that even though the ram usage at the start was different on each node, they start to converge from a point and the RAM usage displayed similar trends on all the devices from this point. Two out of the three nodes have not displayed any noticeable fluctuations in RAM usage and the standard deviation was less than one unit.

4.3 CPU TEMPERATURE

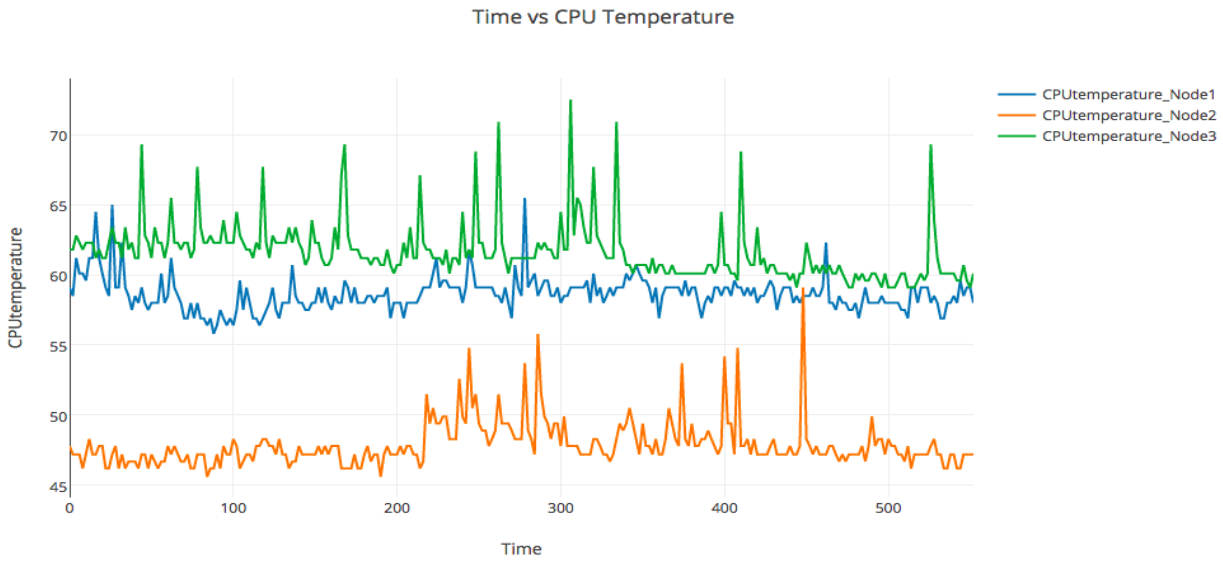

Figure 4.5: Time (2-minute interval) vs CPU core temperature (°C) graph for 3 Raspberry Nodes.

Table 4.3: Statistics for CPU core temperature (°C) data of 3 Raspberry Nodes

Descriptive statistics	Node 1 CPUTemperature	Node 2 CPUTemperature	Node 3 CPUTemperature
N	277	277	277
Min	55.8	45.6	59.1
Max	65.5	59.1	72.5
Mean	58.63465704	47.88736462	61.64476534
First Quartile	58	47.2	60.1
Median	58.5	47.2	61.2
Third Quartile	59.1	48.3	62.3
Standard Deviation	1.222188503	1.626235466	2.059340708
Variance	1.493744738	2.644641791	4.240884151
Standard Error	0.07343419	0.097711019	0.123733791

4.4 BLOCKCHAIN EXPLORATION

4.4.1. Mining Difficulty

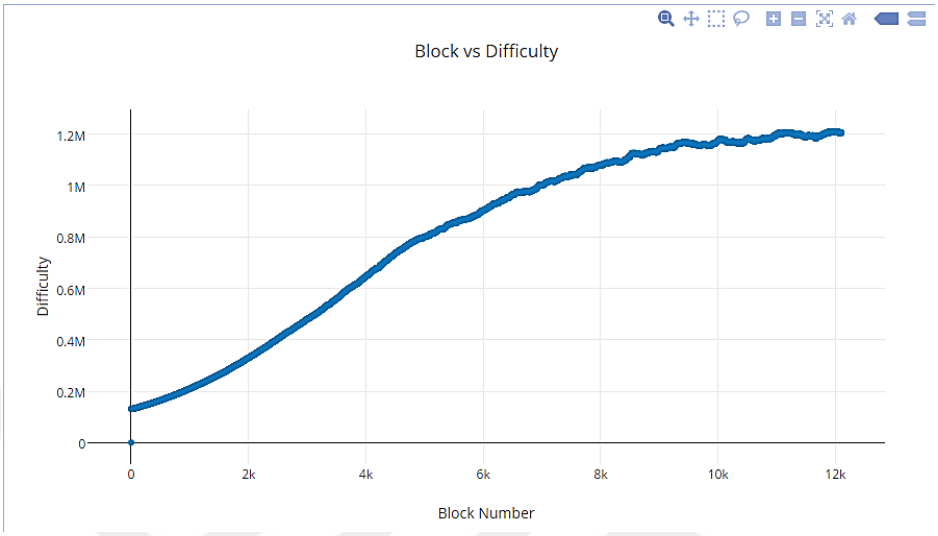

Figure 4.6: Block Number Vs Block Difficulty graph for blockchain

Initially, we have started the private blockchain with the difficulty set to **0x200**. This value controls the block generation time in our private blockchain. However, this difficulty is designed to scale dynamically. Over time, this difficulty increases so that the block generation time will fall to a default ~15 seconds (block time in Ethereum public blockchain). This is due to the design of the difficulty adjustment algorithm.

Formula:

*block_diff = parent_diff + parent_diff // 2048 * max (1 - (block_timestamp - parent_timestamp) // 10, -99) + int (2**((block.number // 100000) - 2)).*

Where // is the integer division operator, e.g. 6 // 2 = 3, 7 // 2 = 3, 8 // 2 = 4.

What does this mean?

If the timestamp difference (block_timestamp - parent_timestamp) is:

A. < 10 seconds, the difficulty is adjusted upwards by parent_diff // 2048 * 1
B. 10 to 19 seconds, the difficulty is left unchanged
C. >= 20 seconds, the difficulty is adjusting downwards proportional to the timestamp difference, from *parent_diff // 2048 * -1* to a max downward adjustment of *parent_diff // 2048 * -99*

So, according to the algorithm, the difficulty level is dynamically modified and is directly dependent on the timestamp difference between the blocks.

"This guarantees to keep block time in the 10-20 range and according to simulations restores the target 15 second blocktime."

Making the mining difficulty static for a private blockchain:

Since the difficulty algorithm is present in the implementations of Ethereum (geth in our case), it can be modified to return a static block time.

func CalcDifficulty(<parameters>) {return big.NewInt(0x4000) }

This sets the difficulty target of every block to a constant value of 0x4000. Geth should be built from source for this to take effect. However, the network should be tested for uncle rate and orphan blocks.

4.4.2. Block Time

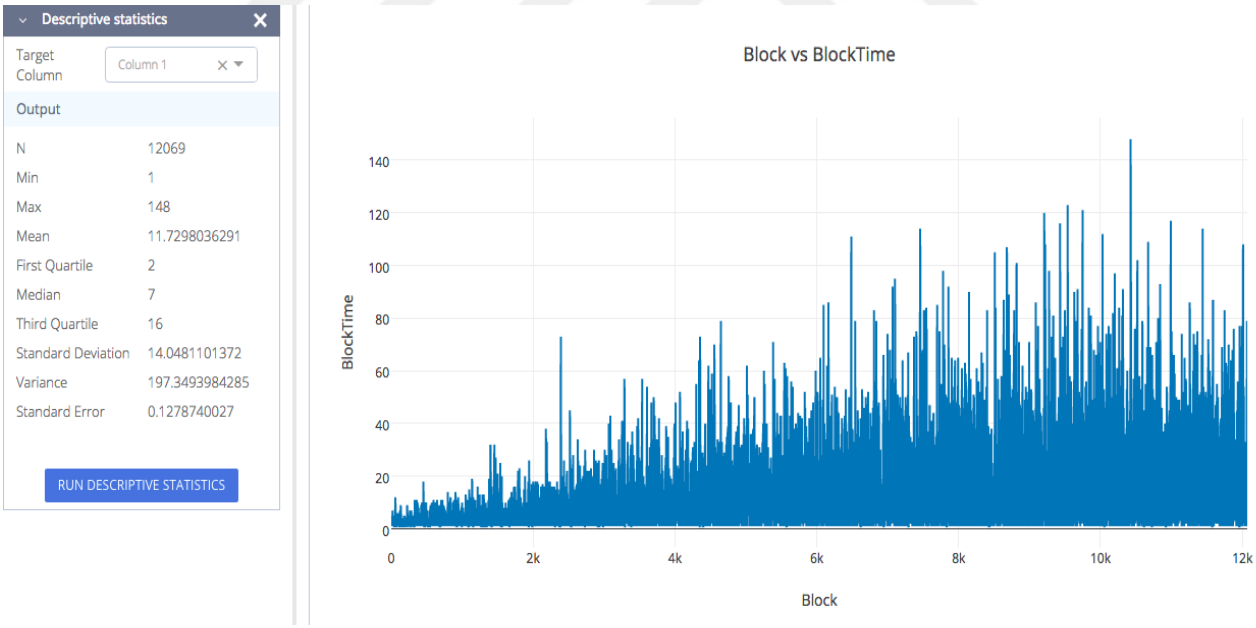

Figure 4.7: Block Number Vs Block Time for blockchain

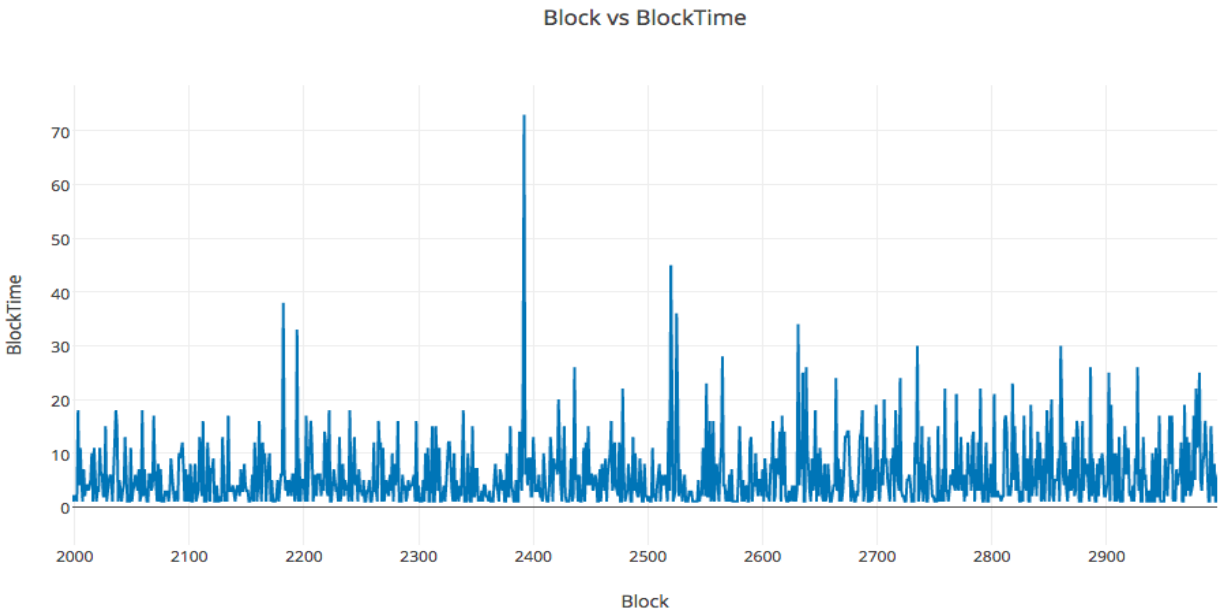

Figure 4.8: Sub-part (block number 2000 - 3000) of above Graph

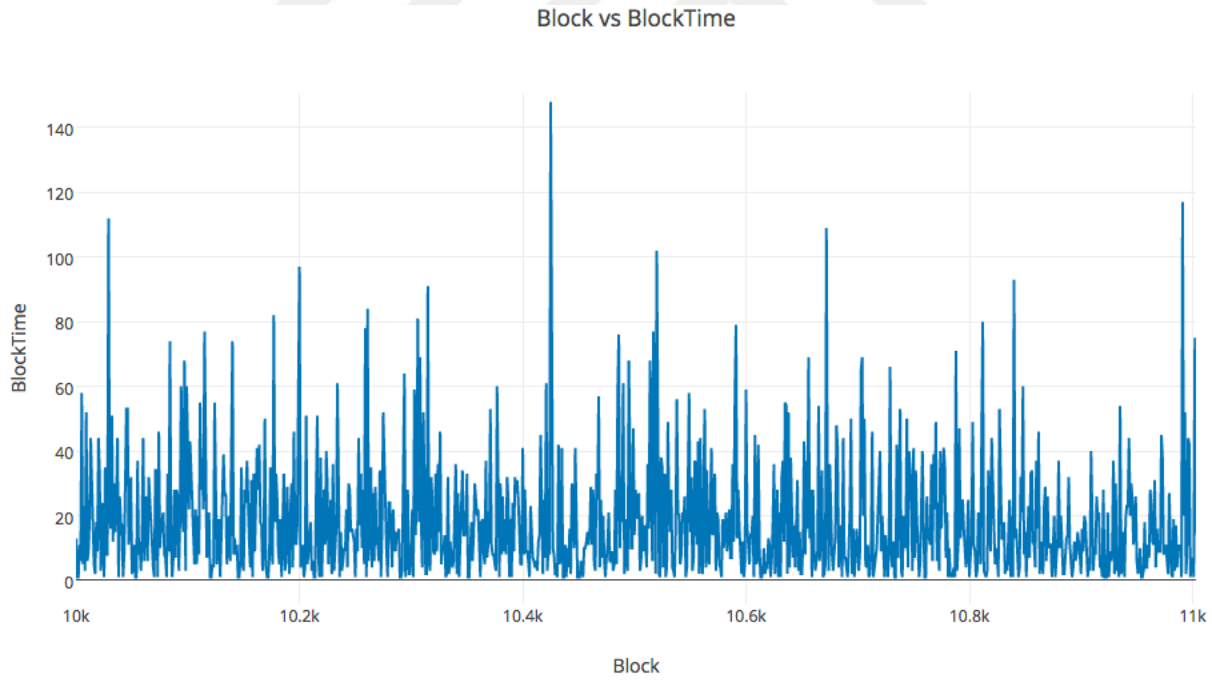

Figure 4.9: Sub-part (block number 10000 - 12000) of above Graph

The graph plotted between Block Vs Block Time displayed similar trends as observed in the Block vs Difficulty graph. As the difficulty of the block increased, the time taken to add the block to the chain also increased till a point and as the difficulty is auto adjusted so was the time taken to add a block to the chain.

4.4.3. Total Difficulty

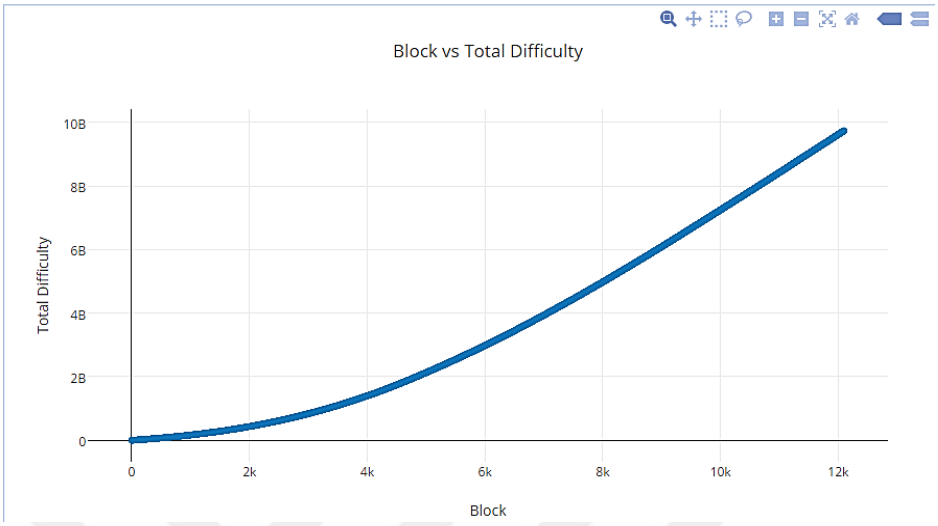

Figure 4.10: Block Number Vs Total Block difficulty for blockchain

The accumulated sum of all the blocks difficulties prior to a block gives the total difficulty of a block. This value is taken into consideration to determine the longest chain by the client in case of a fork. The chain with the highest difficulty is chosen in such a case.

5. DISCUSSION

5.1. ANALYSIS

Blockchain in smart cities is a modular research and platform that simplifies building, testing and running ecosystem applications with the help of numerous tools and connected services. The main features of blockchain in smart world are:

5.1.1. Chains

Eris chains is invaluable for unlocking permission able blockchains optimized by smart cities. Blockchain in smart cities dB client is the industry leader for this. Blockchain chains provides developers with a wide variety of options to create, administer and operate different kinds of blockchains. These are operated from a base of chain definition files. The command: 'blockchain chains' in a docker QuickStart terminal gives various options for which it can be used.

5.1.2. Packages

Blockchain in smart cities pkgs facilitates in unlocking the power of smart cities systems. Smart cities are deployed to blockchain in smart cities chains and blockchain in smart cities pkgs provides various options to developers to create, build, test, and deploy complex systems of these smart cities. Pkgs can be used to test smart cities both on blockchain networks as well as on throwaway chains. Throw away chains are solely for testing smart cities packages. The command: 'blockchain in smart cities pkgs' in a docker QuickStart terminal gives various options for which it can be used.

5.1.3. Keys

Blockchain in smart cities keys expose their developers to get up to speed with modeling the ecosystem applications. It provides the reference API implementation for wallet makers as well as signing solutions to be able to work on the blockchain in smart cities platform. This is intended only for development use. For production use, a different keys service should be made and fully audited before it is implemented. The command: 'blockchain in smart cities keys' in a docker QuickStart terminal gives various options for managing the key pairs.

5.1.4. Files

Eric files leverages the power of distributed data lakes. It provides functionality for working with content addressable and distributed management systems. It is used for quick file sharing from the

host. The command: ‘blockchain in smart cities files’ in a docker QuickStart terminal gives various options for distributed file sharing.

5.1.5. Services

Blockchain in smart cities services unlocks the services that an application relies on. This is primarily used for integrating micro services into an eco-system application. The service definition files are mainly used while starting a Docker container. The command: ‘blockchain in smart cities services’ in a docker QuickStart terminal gives information on blockchain in smart cities interacts with services.

5.2 BLOCKCHAIN COMPONENTS

5.2.1. Blockchain CLI

Blockchain’s is a tool that is used by developers to test, build, manage, and operate smart cities applications. It is a wrapper around the Docker’s API. This is the entry point for the Blockchain platform.

5.2.2. Blockchain DB

Blockchain DB is the blockchain client of Blockchain. It constitutes the smart city consensus, an implementation of Ethereum’s virtual machine, and a permissions layer. Blockchain-db is used through the command line interface by the blockchain-chains command.

5.2.3. Blockchain Chain Manager Tooling:

It is a high-level tool which is used for executing complex operations on Blockchain chains. On the design, nature, and level lines, it is similar to blockchain-pm (blockchain package manager). This is used in providing an interface for the modular components on the blockchain platform.

5.2.4. Blockchain Package Manager Tooling

This is a utility which is mainly used for testing and deploying smart cities packages. This provides access to blockchain-db tooling. Blockchain package manager is primarily used for deploying and testing suites of smart cities. This also defines what jobs should be run and the order in which they should be run.

5.3 POTENTIAL DEPLOYMENT IN SMART CITY

Blockchain has various components for creating, deploying, and managing blockchains. It also has some unique features that are not present in Ethereum (e.g. permissioned blockchains). But, as far

as the internal mechanisms are concerned, Blockchain is essentially running the Ethereum Virtual Machine combined with a smart city consensus in the world. As opposed to Ethereum's Proof-of-Work consensus, Blockchain achieves consensus through a Proof-of-Stake mechanism i.e., Smart city. In the PoW model, users are asked to perform complex cryptographic calculations and solve intense mathematical problems to validate transactions and create blocks. These calculations cannot be performed by simple computers and miners need to purchase special hardware, which is worth thousands of dollars, to keep the crypto-currency network stable. Also, these "mining rigs" consume a lot of energy. This means that a cryptocurrency system working on a PoW mechanism is dependent on costly hardware and uses up a lot of electric power. Most of the miners are interested in these projects because of the financial incentives. But, as the incentive decreases and the difficulty increases, miners will opt out of the network and mining through PoW might not be a feasible option for keeping the network secure. Due to these disadvantages, PoS is considered a better option for achieving consensus in a distributed system.

In a PoS mechanism, a node which validates a block has to provide a proof that it has access to a certain amount of coins. It requires users that have a high stake at the currency (i.e. hold a lot of coins) to determine the next block. As opposed to miners in Bitcoin and Ethereum, the Smart city consensus engine consists of validators. Each validator owns some stake in the network which they bond. Bonding stake means you deposit some money into the network, and in some sense use it as a collateral to vouch for a block. In PoW, a valid chain is identified by the amount of work put behind it, while in PoS, a chain with the highest collateral is trusted and considered to be valid. However, this doesn't mean that an account with the highest balance always gets to confirm all the transactions as it would lead to centralization of the system because, the person with the most no. of coins can control the transactions. So, PoS should have a way of defining the next valid block in any blockchain. In Smart city, this is achieved through voting and some other mechanisms in the Smart city state.

In distributed systems, Byzantine faults are considered dangerous because the presence of a Byzantine fault breaks the harmony of the system i.e., the Byzantine nodes in a distributed system will no longer be able to secure the system because they don't perform in an expected manner due to conflicting and incorrect results. The term Byzantine was used because this situation is compared to a group of generals from the Byzantine army are communicating with other only through messengers to agree upon the same battle plan. There is a possibility that one or more of

the generals are traitors who will try to confuse others. These faults are different from fail-stop failures. In fail-stop failures, if a single node in a distributed system starts to malfunction, that node simply halts and doesn't communicate with other nodes. In the case of Byzantine faults, a malicious node sends conflicting messages to other nodes and these situations are difficult to handle.

Smart city is a high-performance enterprise ready consensus for implementation which can be used to develop and run Byzantine fault tolerant applications. The validators sign the block to say that they approve that this is the correct block. So, a block gets added to the blockchain when it gets signed by the majority of validators. As long as $2/3^{rd}$ or more of the voting power of validators sign the block, it becomes officially committed. The $2/3^{rd}$ majority is chosen because, in case a fork occurs, at least $1/3^{rd}$ of the validators have signed for both the blocks so this means that the system can tolerate up to $1/3^{rd}$ of participants. So, if more than $1/3^{rd}$ of the nodes are byzantine nodes, the consensus cannot be guaranteed. In Bitcoin and Ethereum, anybody can mine blocks. They just need to have the required hardware and the supply of electricity. This is called permission-less validation. The Smart city ideology considers this a bug, not a feature. Particularly in the Enterprise industry, this kind of system might lead to the violation of regulations.

Some key advantages of Smart city used blockchain and are that the block time can be as low as 2 seconds. There is no block size limit and the network can handle 10,000 transactions per second if the transaction size is 250 bytes. Due to the usage of public keys for signing the blocks, accountability is maintained and liability can be determined when a blockchain fork occurs.

5.4. DEPLOY BLOCKCHAIN BLOCKCHAIN ON IOT (ARM) DEVICES

5.4.1. Build The Docker Blockchain Installation Package

```
git clone https://blockchain.com/hypriot/rpi-docker-builder.git

cd rpi-docker-builder

sudo sh build.sh

sudo sh run-builder.sh
```

5.4.2. Install The Docker Blockchain Installation Package

curl -sSL http://downloads.hypriot.com/docker-hypriot_1.10.3-1_armhf.deb >/tmp/docker-hypriot_1.10.3-1_armhf.deb

sudo dpkg -i /tmp/docker-hypriot_1.10.3-1_armhf.deb

rm -f /tmp/docker-hypriot_1.10.3-1_armhf.deb

sudo sh -c 'usermod -aG docker $SUDO_USER'

sudo systemctl enable docker.service

For the latest version follow the below mentioned steps:

apt-get install -y apt-transport-https

wget -q https://packagecloud.io/gpg.key -O - | sudo apt-key add –

echo 'deb https://packagecloud.io/Hypriot/main' | sudo tee /etc/apt/sources.list.d/hypriot.list

apt-get update

apt-get install -y docker-hypriot

systemctl enable docker

To check if docker is properly installed give to get the below output: sudo docker info

```
pi@raspberrypi:~/private $ sudo docker info
Containers: 0
 Running: 0
 Paused: 0
 Stopped: 0
Images: 0
Server Version: 1.11.1
Storage Driver: devicemapper
 Pool Name: docker-179:7-266913-pool
 Pool Blocksize: 65.54 kB
 Base Device Size: 10.74 GB
 Backing Filesystem: ext4
 Data file: /dev/loop0
 Metadata file: /dev/loop1
 Data Space Used: 305.7 MB
 Data Space Total: 107.4 GB
 Data Space Available: 24.94 GB
 Metadata Space Used: 729.1 kB
 Metadata Space Total: 2.147 GB
 Metadata Space Available: 2.147 GB
 Udev Sync Supported: true
 Deferred Removal Enabled: false
 Deferred Deletion Enabled: false
 Deferred Deleted Device Count: 0
 Data loop file: /var/lib/docker/devicemapper/devicemapper/data
 WARNING: Usage of loopback devices is strongly discouraged for production use. Either use `--storage-opt dm.thinpooldev` or use `--storage-opt dm.no_w
arn_on_loop_devices=true` to suppress this warning.
 Metadata loop file: /var/lib/docker/devicemapper/devicemapper/metadata
 Library Version: 1.02.90 (2014-09-01)
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge null host
Kernel Version: 4.4.21-v7+
Operating System: Raspbian GNU/Linux 8 (jessie)
OSType: linux
Architecture: armv7l
CPUs: 4
Total Memory: 925.5 MiB
Name: raspberrypi
ID: E47E:4CDT:TACI:RBD5:L4PA:Z7TD:BR7R:CGHR:BG6Y:RO4H:WOAP:G7XT
Docker Root Dir: /var/lib/docker
Debug mode (client): false
Debug mode (server): false
Registry: https://index.docker.io/v1/
WARNING: No swap limit support
WARNING: No kernel memory limit support
WARNING: No cpu cfs quota support
WARNING: No cpu cfs period support
WARNING: No cpuset support
pi@raspberrypi:~/private $
```

Figure 5.1: sudo docker info output

At this point, we are currently blocked by an issue corresponding to getting the ARM device specific Blockchain installation files. Moreover, the documentation for Blockchain is ambiguous and not up to date. Installation of Blockchain is dependent on various other components like Docker and issues being faced with these dependencies is further delaying the process. We have logged an issue with Blockchain community and we tried out various other alternatives suggested.

```
Fetched 9,290 kB in 15s (592 kB/s)
W: Failed to fetch https://eris-iot-repo.s3.amazonaws.com/eris-deb/dists/raspbian/experimental/binary-armhf/Packages  HttpError403

E: Some index files failed to download. They have been ignored, or old ones used instead.
pi@raspberrypi:~/private $ sudo apt-get install eris
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package eris
pi@raspberrypi:~/private $
```

Figure 5.2: Error faced while installing Blockchain on Arm devices

5.5. COMPARISON & ANALYSIS

Table 5.1: The comparison of major blockchain for potential deployment in smart cities.

Applications / Parameters	Ethereum	Bitcoin	Litecoin
Block Time	15 Seconds	2 Seconds	N/A
Block Size	No Limit	No Limit	N/A
Txps	25	10,000	To be added
Smart cities	Yes	Yes	To be added
Type of Blockchain	Public, Private	Private, Permissioned	N/A
Distributed Consensus	Proof-of-Stake	Proof-of-Work	Proof-of-Work
Explorer API	Yes	Yes	Yes (Web UI also available)
Signing Algorithm	ECDSA	Ed25519	To be added
Hashing Algorithm	Keccak 256	RIPEMD 160	To be added
Minimum Nodes	1	4	To be added
Supported OS'	Windows, Linux, MAC, ARM	Windows, Linux, MAC	N/A (Java 64-bit)
Robust Codebase	Medium - High	Low - Medium	Reference Implementation
Fault Tolerance Level	1/2 of the nodes	1/3 of the nodes	To be added
Documentation	Mostly Clear	Ambiguous	Very little
Implementation Complexity	Easy – Medium	Medium – Hard	Easy
Accountability	No	Yes	To be added
Deployment in Smart Cities	Yes	Yes	No
Target Usecases	Distributed Apps	Enterprise applications for regulatory environments	IoT applications

Normal transactions have been around for a long time when compared to blockchain, so it is subjected to more scrutiny and has been tested by many researchers and developers. It is safe to say that blockchain is well developed, has a robust codebase, and more importantly, a good developer community. It is rapidly improving in stages by documenting the limitations in each stage through blockchain improvement proposals. Since there is a strong debate that PoS is better than PoW in many areas, research is being conducted and blockchain is planning to release their own version of PoS algorithm called Casper. This algorithm claims to reduce the block time by starting with 4 seconds and going further down depending on how the network reacts. Since the implementation is not concrete yet, it is difficult to compare with smart city. The major advantage of blockchain lies in the Smart city algorithm. If not for that, bitcoin is just an blockchain virtual machine with a meta-permission layer operating on top of the blockchain. This permission layer is not for read/write operations on blockchain. These permissions determine which accounts can participate in the consensus process and which accounts have the ability to create smart cities in a blockchain network. So, in a smart city application, one can control who can validate blocks and who can make changes to code logic. Some of the advantages of blockchain in smart city are given by:

a) Energy efficient since there is no hashing involved.
b) CPU mining is possible because signing and verification operations are simple.
c) Requires less computational capacity and memory.
d) Block time is approximately 2 seconds while in normal online transaction, it is 15 seconds.
e) Smart city can support 10,000 transactions per second while normal transaction supports 15 transactions per second. These stats are based on simple transactions (from, to, value)
f) Smart city uses blockchain algorithms for signing and verifying blocks. Users can sign 100,000 blocks/second and can verify 70,000 signatures/second without a compromise in security. Normal transaction is not as fast.

6. CONCLUSION

6.1. CONCLUSION

Blockchains and IoT are still at an early stage of development and there is certainly a lot of hype around both the technologies. The current state of blockchain is ambiguous and everything is still in the exploration phase. Bitcoin and Ethereum remain the most popular and novel projects in this space while others are just forks with subtle changes. Projects on blockchain keep emerging but most of these projects are abandoned even before they materialize. This is because people fail to realize that blockchain is not the solution for every problem we have, although it does offer some interesting possibilities that can improve the current situation of some ecosystems. Once the internal mechanisms are understood, we learn that not much can be solely achieved by the blockchain technology itself. Instead, the blockchain technology can be incorporated into current solutions to make them robust and more secure. The strength of blockchain lies in financial technology (fintech) because that is what it was invented for and that is where its impact is becoming apparent. We are yet to observe a major breakthrough in other applications of blockchains like IoT, supply chain, healthcare, manufacturing, and digital asset management. As far as our research is concerned, Ethereum combined with the smart city proof-of-stake mechanism seems to be the most optimal solution for building the majority of IoT applications. This is because litecoin is still in the beta phase and only has a reference implementation. Nothing concrete can be inferred from this yet. Also, it has a completely different structure for storage which is not yet proven to be robust. However, there are some interesting projects like Hyperledger and IBM Blockchain to watch out for. IBM is rapidly developing and testing Proof-of-Concept applications in various industries. The Hyperledger consortium is backed by a lot of big players and is managed by the Linux foundation group. Both these projects claim to support IoT use cases. The ongoing research in the blockchain field will soon reveal how these projects will develop and impact the IoT ecosystem.

6.2. FUTURE RECOMMENDATIONS

The blockchain deployment should observe that all these advantages are due to the smart city algorithm which is not native to normal transaction. Smart city doesn't depend on other components of the blockchain i.e., it acts as a portable module and can work as a consensus engine for any blockchain architecture. It has a special protocol to communicate with the other components of a blockchain. So, there is no strong reason to use Eris if we can incorporate Smart city into Ethereum or if Ethereum should release Casper. Moreover, Eris is not subjected to the same scrutiny as Ethereum so it is still not robust. Also, from the features, we can observe that Eris is primarily developed for regulatory environments which need accountability and liability. So, it is not even pseudo-anonymous and doesn't comply with the fundamental principles of a blockchain. On the other hand, Litecoin is yet to prove its mettle in this space. Currently, only a reference implementation written in Java is available to experiment with the technology and there is no proper client to complement the unique features described in the Litecoin whitepaper. The plan is to ultimately replace this reference implementation with something more suitable for low powered IoT devices. The documentation for bitcoin and Litecoin is scattered all over Twitter, and Slack and it will take a significant amount of time to make a proper collaboration. So, Ethereum is one step ahead in this space as well. Due to all these advantages, ubiquity, and a wide range of applications, Ethereum seems to offer a better platform for IoT applications. Unless Litecoin makes some significant advancements with major upgrades, Ethereum with a PoS mechanism seems to be the architecture of choice for developing blockchain applications in IoT for now.

REFERENCES

[1] Malomo, O.O.; Rawat, D.B.; Garuba, M. Next-generation cybersecurity through a blockchain-enabled federated cloud framework. J. Supercomput. 2018, 74, 5099–5126.

[2] Adebayo, A.; Rawat, D.B.; Njilla, L.; Kamhoua, C.A. Blockchain-enabled Information Sharing Framework for Cybersecurity. Blockchain Distrib. Syst. Secur. 2019, Chapter 7.

[3] Divya, M.; Biradar, N.B. IOTA-next generation block chain. Int. J. Eng. Comput. Sci. 2018, 7, 23823–23826.

[4] Available Online: https://www.skalex.io/support/blockchain/blockchain-technology/

[5] Shrestha, R.; Bajracharya, R.; Shrestha, A.P.; Nam, S.Y. A new type of blockchain for secure message exchange in VANET. Digit. Commun. Netw. 2020, 6, 177–186.

[6] Syed, T.A.; Alzahrani, A.; Jan, S.; Siddiqui, M.S.; Nadeem, A.; Alghamdi, T. A comparative analysis of blockchain architecture and its applications: Problems and recommendations. IEEE Access 2019, 7, 176838–176869.

[7] Shrestha, R.; Nam, S.Y. Regional blockchain for vehicular networks to prevent 51% attacks. IEEE Access 2019, 7, 95021–95033.

[8] Velliangiri, S.; Karunya, P.K. Blockchain Technology: Challenges and Security issues in Consensus algorithm. In Proceedings of the 2020 IEEE International Conference on Computer Communication and Informatics (ICCCI), Nagoya, Japan, 25–27 June 2020; pp. 1–8.

[9] Somy, N.B.; Kannan, K.; Arya, V.; Hans, S.; Singh, A.; Lohia, P.; Mehta, S. Ownership preserving AI Market Places using Blockchain. In Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 14–17 July

2019; pp. 156–165.

[10] Vyas, S.; Gupta, M.; Yadav, R. Converging blockchain and machine learning for healthcare. In Proceedings of the 2019 IEEE Amity International Conference on Artificial Intelligence (AICAI), Dubai, UAE, 4–6 February 2019; pp. 709–711.

[11] Shrestha, R.; Nam, S.Y.; Bajracharya, R.; Kim, S. Evolution of V2X Communication and Integration of Blockchain for Security Enhancements. Electronics 2020, 9, 1338.

[12] Shrivastava, V.; Kumar, S. Utilizing Block Chain Technology in Various Application Areas of Machine Learning. In Proceedings of the 2019 IEEE International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 14–16 February 2019; pp. 167–171.

[13] Shrestha, R.; Kim, S. Integration of IoT with blockchain and homomorphic encryption: Challenging issues and opportunities. In Advances in Computers; Elsevier: Amsterdam, The Netherlands, 2019; Volume 115, pp. 293–331.

[14] Wang, X.; Zha, X.; Ni, W.; Liu, R.P.; Guo, Y.J.; Niu, X.; Zheng, K. Survey on blockchain for Internet of Things. Comput. Commun. 2019, 136, 10–29.

[15] Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X.; Wang, H. Blockchain challenges and opportunities: A survey. Int. J. Web Grid Serv. 2018, 14, 352–375.

[16] Panarello, A.; Tapas, N.; Merlino, G.; Longo, F.; Puliafito, A. Blockchain and iot integration: A systematic survey. Sensors 2018, 18, 2575.

[17] Lin, I.C.; Liao, T.C. A survey of blockchain security issues and challenges. IJ Netw. Secur. 2017, 19, 653–659.

[18] https://www.activism.net/cypherpunk/manifesto.html “A Cypherpunk's Manifesto".

[19] Narayanan, A.; Bonneau, J.; Felten, E.; Miller, A.; Goldfeder, S. Bitcoin and

Cryptocurrency Technologies: A Comprehensive Introduction; Princeton University Press: Princeton, NJ, USA, 2016.

[20] Available Online: https://www.weforum.org/agenda/2017/07/how-can-creative-industries-benefit-from-blockchain/

[21] Augusto, J.C.; Callaghan, V.; Cook, D.; Kameas, A.; Satoh, I. Intelligent Environments: a manifesto. Human-Centric Comput. Inf. Sci. 2013, 3, 12, doi:10.1186/2192-1962-3-12.

[22] Roalter, L.; Kranz, M.; Möller, A. A Middleware for Intelligent Environments and the Internet of Things. Ubiquitous Intelligence and Computing; Yu, Z., Liscano, R., Chen, G., Zhang, D., Zhou, X., Eds.; Springer: Berlin, Germany, 2010; pp. 267–281.

[23] Ryu, M.; Kim, J.; Yun, J. Integrated semantics service platform for the Internet of Things: A case study of a Smart Office. Sensors 2015, 15, 2137–2160.

[24] Kim, H.M.; Laskowski, M. Toward an ontology-driven blockchain design for supply-chain provenance. Intell. Syst. Account. Finance Manag. 2018, 25, 18–27.

[25] Available Online: https://www.gminsights.com/industry-analysis/blockchain-technology-market

[26] Stojkoska, B.R.; Trivodaliev, K. A review of Internet of Things for smart home: Challenges and solutions. J. Clean. Prod. 2017, 140, 1454–1464.

[27] Dorri, A.; Kanhere, S.S.; Jurdak, R.; Gauravaram, P. Blockchain for IoT security and privacy: The case study of a smart home. In Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kona, HI, USA, 13–17 March 2017; pp. 618–623.

[28] Available Online: https://blockgeeks.com/guides/what-is-bitcoin/

[29] Lv, Z.; Halawani, A.; Feng, S.; Ur Réhman, S.; Li, H. Touch-less Interactive Augmented

Reality Game on Vvision-based Wearable Device. Personal Ubiquitous Comput. 2015, 19, 551–567.

[30] Available Online: https://medium.com/@blairlmarshall/how-does-a-bitcoin-transaction-actually-work-1c44818c3996

[31] Christidis, K.; Devetsikiotis, M. Blockchains and Smart Contracts for the Internet of Things. IEEE Access 2016, 4, 2292–2303

[32] Available Online: https://yos.io/2018/11/16/ethereum-signatures/

[33] Stafford, T.F.; Treiblmaier, H. Characteristics of a blockchain ecosystem for secure and sharable electronic medical records. IEEE Trans. Eng. Manag. 2020, 1–23.

[34] Available Online: https://en.wikipedia.org/wiki/Merkle_tree

[35] Makhdoom, I.; Zhou, I.; Abolhasan, M.; Lipman, J.; Ni, W. PrivySharing: A blockchain-based framework for privacy-preserving and secure data sharing in smart cities. Comput. Secur. 2020, 88, 101653.

[36] Available Online: https://medium.com/coinmonks/the-bitcoin-blockchain-a3eb996f7140

[37] Boulos, M.N.K.; Wilson, J.T.; Clauson, K.A. Geospatial blockchain: Promises, challenges, and scenarios in health and healthcare. Int. J. Health Geogr. 2018, 17, 25.

[38] Aggarwal, S.; Chaudhary, R.; Aujla, G.S.; Kumar, N.; Choo, K.-K.R.; Zomaya, A.Y. Blockchain for smart communities: Applications, challenges and opportunities. J. Netw. Comput. Appl. 2019, 144, 13–48.

[39] Dwivedi, A.D.; Srivastava, G.; Dhar, S.; Singh, R. A decentralized privacy-preserving healthcare blockchain for IoT. Sensors 2019, 19, 326.

[40] Liao, D.-Y.; Wang, X. Applications of blockchain technology to logistics management in integrated casinos and entertainment. Informatics 2018, 5, 44.

[41] Available Online: https://www.upgrad.com/blog/what-is-blockchain-how-to-create-networkcode-its-architecture/

[42] Available Online: https://blockgeeks.com/guides/what-is-hashing/

[43] Casado-Vara, R.; Corchado, J. Distributed e-health wide-world accounting ledger via blockchain. J. Intell. Fuzzy Syst. 2019, 36, 2381–2386.

[44] Rahman, M.A.; Rashid, M.M.; Shamim Hossain, M.; Hassanain, E.; Alhamid, M.F.; Guizani, M. Blockchain and IoT-based cognitive edge framework for sharing economy services in a smart city. IEEE Access 2019, 7, 18611–18621.

[45] Park, L.W.; Lee, S.; Chang, H. A sustainable home energy prosumer-chain methodology with energy tags over the blockchain. Sustainability 2018, 10, 658.

[46] Chaudhary, R.; Jindal, A.; Aujla, G.S.; Aggarwal, S.; Kumar, N.; Choo, K.-K.R. BEST: Blockchain-based secure energy trading in SDN-enabled intelligent transportation system. Comput. Secur. 2019, 85, 288–299.

[47] Jindal, A.; Aujla, G.S.; Kumar, N. SURVIVOR: A blockchain based edge-as-a-service framework for secure energy trading in SDN-enabled vehicle-to-grid environment. Comput. Netw. 2019, 153, 36–48.

[48] Bernal Bernabe, J.; Canovas, J.L.; Hernandez-Ramos, J.L.; Torres Moreno, R.; Skarmeta, A. Privacy-preserving solutions for blockchain: Review and challenges. IEEE Access 2019, 7, 164908–164940.

[49] França, A.S.L.; Amato Neto, J.; Gonçalves, R.F.; Almeida, C.M.V.B. Proposing the use of blockchain to improve the solid waste management in small municipalities. J. Clean. Prod. 2020, 244, 118529.

[50] Marsal-Llacuna, M.-L. Future living framework: Is blockchain the next enabling network?

Technol. Forecast. Soc. Change 2018, 128, 226–234.

[51] Sittón-Candanedo, I.; Alonso, R.S.; Corchado, J.M.; Rodríguez-González, S.; Casado-Vara, R. A review of edge computing reference architectures and a new global edge proposal. Future Gener. Comput. Syst. 2019, 99, 278–294.

[52] Rejeb, A.; Rejeb, K. Blockchain technology in tourism: Applications and possibilities. World Sci. News 2019, 137, 119–144.

[53] Rejeb, A.; Keogh, J.G.; Treiblmaier, H. The impact of blockchain on medical tourism. In Proceedings of the WeB 2019 Workshop on e-Business, Munich, Germany, 14 December 2019; pp. 1–12.

[54] Rejeb, A.; Keogh, J.G.; Treiblmaier, H. How blockchain technology can benefit marketing: Six pending research areas. Front. Blockchain 2020, 3, 3.

[55] Rejeb, A.; Keogh, J.G.; Treiblmaier, H. Leveraging the internet of things and blockchain technology in supply chain management. Future Internet 2019, 11, 161.

[56] Rejeb, A. Halal meat supply chain traceability based on HACCP, blockchain and internet of things. Acta Tech. Jaurinensis 2018, 11, 1–30.

[57] Rotună, C.; Gheorghit,ă, A.; Zamfiroiu, A.; Smada Anagrama, D. smart city ecosystem using blockchain technology. Inform. Econ. 2019, 23, 41–50.

[58] Salah, K.; Rehman, M.H.; Nizamuddin, N.; Al-Fuqaha, A. Blockchain for AI: Review and open research challenges. IEEE Access 2019, 7, 10127–10149.

[59] Wang, Q.; Zhu, X.; Ni, Y.; Gu, L.; Zhu, H. Blockchain for the IoT and industrial IoT: A review. Internet Things 2020, 10, 100081.

[60] Gong, S.; Tcydenova, E.; Jo, J.; Lee, Y.; Park, J.H. Blockchain-based secure device management framework for an internet of things network in a smart city. Sustainability

2019, 11, 3889.

[61] Mehmood, Y.; Ahmad, F.; Yaqoob, I.; Adnane, A.; Imran, M.; Guizani, S. Internet-of-things-based smart cities: Recent advances and challenges. IEEE Commun. Mag. 2017, 55, 16–24.

[62] Available Online: http://financeindex.co/investing/cc8765456/

`