Towards Building a Secure Blockchain-Based Architecture for Internet of Things (IoT)

نحو بناء بنية آمنة قائمة على سلسلة الكتل لإنترنت الأشياء

by

DEEPA PAVITHRAN

A thesis submitted in fulfilment

of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

at

The British University in Dubai

June 2020

Towards Building a Secure Blockchain-Based Architecture for Internet of Things (IoT)

نحو بناء بنية آمنة قائمة على سلسلة الكتل لإنترنت الأشياء

By
Deepa Pavithran

A thesis submitted to the Faculty of Engineering and IT

in fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE
at
The British University in Dubai
June 2020
Thesis Supervisor
Prof. Khaled Shaalan

Approved for award:

_____

_____

Name

Designation

_____

_____

Name

Designation

_____

_____

Name

Designation

Name

Designation

Date: _____

DECLARATION

I warrant that the content of this research is the direct result of my own work and that any use made in it of published or unpublished copyright material falls within the limits permitted by international copyright conventions.

I understand that a copy of my research will be deposited in the University Library for permanent retention.

I hereby agree that the material mentioned above for which I am author and copyright holder may be copied and distributed by The British University in Dubai for the purposes of research, private study or education and that The British University in Dubai may recover from purchasers the costs incurred in such copying and distribution, where appropriate.

I understand that The British University in Dubai may make a digital copy available in the institutional repository.
I understand that I may apply to the University to retain the right to withhold or to restrict access to my thesis for a period which shall not normally exceed four calendar years from the congregation at which the degree is conferred, the length of the period to be specified in the application, together with the precise reasons for making that application.

_____
Signature of the student

COPYRIGHT AND INFORMATION TO USERS

Abstract

IoT (Internet of Things) devices usually generate a large amount of data shared with a centralized cloud to provide various services. Traditional IoT architecture is heavily centralized, where data stored in a cloud environment, is prone to several kinds of threats. Blockchain is a very promising technology that spans many use-cases other than cryptocurrencies. For example, its implementation in the Internet of Things based networks (IoT) is still unclear and demands further research. The traditional adoption of the blockchain protocol for Bitcoin is common but it cannot be used for IoT because Bitcoin is a payment system, whereas the IoT eco-system has a different architecture. Implementing blockchain for IoT may still impose a variety of challenges. In this thesis, we proposed an architecture for the use of blockchain in event-driven IoT. In particular, we identified the key components along with their design considerations and challenges to consider while creating the blockchain architecture for IoT. We also defined gaps that hinder creating a secure blockchain framework for IoT.

Various literatures have proposed blockchain architectures for IoT; however, most of them are applicable to use-cases related to smart homes and healthcare. In addition, we identified that the existing architectures have additional overhead of key management. Hence, we proposed a privacy-preserving blockchain architecture for Traffic Speed radars using Hierarchical Identity Based Encryption (HIBE). The proposed architecture uses edge and cloudlet computing paradigm as well as HIBE to preserve privacy. The performance of the proposed architecture is evaluated by conducting extensive experiments. We created the blockchain network using Ethereum and evaluated the system performance. Network performance was evaluated by simulating the network using Contiki OS. Finally, we analyzed the security of the scheme through theoretical analysis and threat-modelling tool that considers the existence of a malicious adversary.

الخلاصة

إنترنت الأشياء عادة ما تولد كمية كبيرة من البيانات التي يتم تقاسمها مع سحابة مركزية لتوفير خدمات مختلفة.

الهندسة المعمارية التقليدية لإنترنت الأشياء هي مركزية لحد كبير، حيث تخزن البيانات في بيئة سحابية ، مما يجعلها عرضة لأنواع عديدة من التهديدات.

"البلوكتشين" أو سلسلة الكتل هي تقنية واعدة جدا ولها عديد من الإستخدامات-و التي تطبق في مجالات أخرى غير التشفير.

فعلى سبيل المثال، لا يزال تنفيذ تقنية البلوكتشن في الشبكات القائمة على الأشياء على شبكة الإنترنت غير واضح ويتطلب مزيدا من البحث.

ولا يمكن استخدام بروتوكول البلوكتشين التقليدي في شبكات إنترنت الأشياء لأن البيتكوين او العملات المشفرة هو نظام دفع مالي والذي يختلف عنها في البنية. ولا يزال تطبيق البلوكتشين في إنترنت الأشياء يفرض مجموعة متنوعة من التحديات.

في هذه الأطروحة ، نقترح بنية بسيطة لاستخدام البلوكتشين في إنترنت الأشياء. وعلى وجه الخصوص ، حددنا خمسة مكونات رئيسية جنبًا إلى جنب مع اعتبارات التصميم والتحديات التي يجب مراعاتها أثناء إنشاء بنية الأشياء. واستخدام البلوكتشين حدد أيضًا الفجوات التي تعيق إنشاء إطار آمن لإنترنت الأشياء.

في هذه الدراسة نقترح بنية البلوكتشين للحفاظ على الخصوصية للإنترنت الأشياء باستخدام أسلوب التشفير الهرمي القائم على الهوية والتي تستخدم البنية لنموذج مقترح للحوسبة السحابية والتي تحافظ على الخصوصية.

تتناسب البنية المقترحة بشكل جيد مع أجهزة إنترنت الأشياء التي تعتمد على الحدث. كما تم عرض دراسة حالة في اطار رادارات السرعة للمركبات باستخدام المعمارية المقترحة. وتم تقييم أداء العمارة المقترحة من خلال إجراء تجارب مكثفة. وأخيرا، تم تقييم أمن النظام من خلال تحليل نظري يأخذ في الاعتبار وجود خصم ضار.

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Prof. Khaled Shaalan for the continuous support of my Ph.D. study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

I would like to thank Dr. Cornelius Ncube, Professor Sherief Abdallah, Dr. Khalid Al Marri and Professor Halim Boussabaine for their valuable classes on advanced topics in computer science and research methodologies.

I would also like to thank Dr. Jamal Al-Karaki, Head of the Department, Abu Dhabi Polytechnic for his guidance, motivation and support in research. I thank my colleagues in Abu Dhabi Polytechnic and classmates at the British University in Dubai, Charles Christin Gabriel, Munavwar Mubarak Shaikh, Rajesh Thomas and Aisha Mohdhar for the valuable discussions and feedbacks.

Finally yet importantly, I would like to thank my husband and children for supporting me and understanding me throughout the research and my life in general.

Thanks for all your encouragement!

Table of Contents

List of Tables

List of Figures

Chapter 1 : INTRODUCTION

1.1 Problem Statement

The success of bitcoin indicates blockchain technology as a clear replacement for the centralized client server system in terms of security. However, how to utilize this technology for applications other than cryptocurrency is still vague. There are various architectures, platforms and consensus available in the literature. Nevertheless, there is no typical standard for any of those. This is mainly due to the resource-constrained nature of IoT devices, lack of proper identity management techniques and poor device interoperability between the devices. Public blockchain undoubtedly have greater consensus strength and visibility; however, it is not a secure approach to expose the organizational data publicly. Many industries intend to apply blockchain but they stepped back due to this privacy constraint and the lack of clarity of design approach for private blockchain. Furthermore, involvement of IoT also comes with many constrained issues in terms of security, integrity and scalability. Therefore, based on the literature study and after analysis of various Blockchain perspectives, we have taken a use-case on Traffic speed radars.

Speed radars are IoT devices that use Doppler Effect to detect the speed of a moving vehicle and generates traffic fine once the designated speed exceeds. Currently, this system uses a centralized database where a targeted attacker can modify the traffic fines or delete the database records. The centralized approach has many drawbacks, like single point of failure and is always a target of attack. In addition, the data generated by the sensors can be intentionally or unintentionally modified or deleted by internal or external people. For example, a disgruntled employee can manipulate the traffic fine generated by sensors. In addition, this system does not provide privacy. Administrators who have access to the database can know the location, time and fine amount of various vehicles as well as driver's information.

A report from Kaspersky Lab uncovered many security vulnerabilities in these devices. The report stated that “Criminals can get access to the entire database”(Makrushin, Denis 2016). In 2018, an anonymous hacker hacked and deleted the entire speed camera database in Correggio, Italy (Paganini 2018). Apart from the hackers, there were cases where officials manipulate the data generated by sensors. In the city of Flint, Michigan CNN reported claims that officials manipulated the data generated by water meters (CNN Editorial Research 2019). In addition, anyone who knows the traffic symbol of the vehicle can access the traffic fine history. These are some of the challenges reported for current radar systems that warrant the need to have a better system, which can deal with these challenges. To embark on the needed solution, the key problems with the current architecture are outlined below:

- Security of the data including confidentiality, integrity and availability of the data generated by sensors and
- The cost of maintaining a centralized cloud.
- Privacy concerns due to centralized architecture
- Weaknesses in the design of traditional system that has been in place for long time.

Blockchain technology has recently gained much attention as means for trust management. In particular, the problem of trusting a third party in the traditional payment system has been solved using blockchain. Apart from cryptocurrencies, blockchain can be applied into a variety of applications including Internet of Things (IoT) (Dorri, Ali and Kanhere, Salil S and Jurdak 2017). A majority of the blockchain applications for IoT are carried out on supply-chain management (Bocek et al. 2017)(Borah et al. 2020), asset management(Tran, Lu & Weber 2018)(Verma et al. 2017), health care(Meena & Dwivedi 2019)(Uddin et al. 2020) and provenance management(Ramachandran & Kantarcioglu 2018)(Kaku 2017). In contrast to the traditional way

of a central authority maintaining the ledger of all transactions, blockchain distributes the ledger to every node in the network. The verification and validation of these transactions are conducted through cryptographic techniques. Current centralized, cloud based IoT solutions are not scalable and not capable to meet security challenges. We believe that adoption of blockchain can help in solving challenges associated with current speed radar system.

Hence, we try to identify, develop and implement the best technology that can be used for IoT blockchain applications.

Blockchain is considered to be used as a solution to this problem, as it provides security in a decentralized environment. However, traditional blockchain protocols for cryptocurrency may not be used here due to the limitations in IoT devices with respect to expensive consensus mechanisms and IoT infrastructure has a different architecture than a payment system. As such, we propose blockchain architecture for traffic speed radars that utilize all the advantages of blockchain technology while making it suitable for IoT devices. Enterprises today see blockchain as a new foundation to the business world. It is expected to change the way the economy runs through some ways by organizing the data. However, blockchain technologies are still in constant change and development. Hence, it is difficult to provide its exact process and underlying technologies. Initially, the primary motive of developing blockchain was for financial applications. However, after the introduction of smart-contracts it expands its use-cases to other distributed peer-to-peer applications, where members can interact with each other irrespective of trust. Achieving privacy and confidentiality in a permissioned blockchain is extremely difficult (Vossen, 2018). While there is currently, no standard in the blockchain space, all the on-going efforts on blockchain involve some combination of database, transaction, encryption, consensus and other distributed system technology.

1.2 **Research Question and Hypothesis**

Our research questions are as follows

1. Identify the key components in implementing Blockchain for IoT devices. (Chapter 3)
2. Identify how PKI can be integrated into IoT Blockchain? (Chapter 5)
3. Identify how to select a consensus node for IoT devices? (Chapter 7)
4. How to design a privacy preserving protocol for IoT blockchain? (Chapter 6)
5. How to design an efficient architecture for Traffic Speed Radars using blockchain(Chapter 4)

Each research is identified in different Chapters.

Hypothesis

H1: It is possible to create a blockchain for event driven IoT with given constraints

H2: HIBE can be used in a blockchain environment to provide data confidentiality for IoT data

H3: The proposed blockchain architecture can improve the security of IoT data

H4: The system and network performance of the proposed architecture is better the traditional centralized architecture of IoT

1.3 **Aims and Objectives**

The main aims of this project is to provide a secure architecture for Traffic Speed Radar using Blockchain technology. In general, decentralizing the IoT network has various advantages like reduced cost associated with maintaining a central database for IoT transactions, improved security and privacy and eliminating the need for a third party.

On the other hand, implementing blockchain for IoT devices like speed radars impose various challenges including additional storage, delay in network communication, the consensus

mechanism to be adopted, confidentiality of the data and the distributed architecture for speed radars.

Our implementation is an edge-based architecture with distributed cloudlet nodes instead of traditional centralized cloud model. Instead of public blockchain infrastructure, we work on implementing cloud independent private blockchain framework integrated with smart contract. The research seek to achieve two goals. First, it is provides a clear exposure for those sectors who are planning to reshape their traditional architecture. Second, it provides a gateway for non-security experts to gain better understanding of private blockchain and smart contract. In addition, it will help specialists and blockchain enthusiasts explore innovative technologies related to blockchain, IoT and Cloud services.

Our approach includes solutions for:

1. Identifying the key components in implementing Blockchain for Speed radars?
2. Designing a PKI infrastructure that can be used in IoT Blockchain
3. Designing an efficient node selection process would be the best fit for lightweight IoT devices.
4. Designing an efficient encryption layer for traditional blockchain protocol?
5. Designing an efficient architecture for IoT blockchain by combining the solutions identified in 1, 2, 3 and 4.

1.4 **Contribution of the Thesis**

Contribution of the thesis is multifold as described below

1. A decision framework for building blockchain for IoT,
2. An architecture to provide Public Key Infrastructure in event triggered-edge node based IoT using Blockchain,

3. An architecture to provide data confidentiality in event triggered-edge node based IoT using Blockchain,
4. An Efficient consensus node selection protocol IoT using Blockchain,
5. Performance comparison of IoT device only and edge node based architecture, and
6. An architecture for event triggered- edge node based IoT using Blockchain by combining the above findings is proposed and evaluated.

1.5 Overall Structure of the Thesis

The main objective of this thesis is to create a secure blockchain architecture for IoT. Before reaching to that objective, we identified and solved some components that are necessary to build a secure architectures. These individual components are explained in Chapter 3, Chapter 5, Chapter 6 and Chapter 7. Hence the theoretical background, related works, proposed solution, implementation and conclusion related to the component is provided in all these Chapters. These results are used in creating the final architecture given in Chapter 4 and the Implementation and result of the architecture is briefly discussed in Chapter 8.

Chapter 2 surveys the literature in the field of blockchain, IoT and cryptography. It describes the most recent advancement in the field of blockchain by comparing it with the traditional blockchain protocol, which is bitcoin. It briefly describes how blockchain works based on the bitcoin protocol highlighting the transactions and security in bitcoin. It also provides recent advancement in IoT highlighting the generic IoT architecture and various types of IoT. Then it describes the current state of art on blockchain for IoT including various use-cases, architectures, design challenges etc. in creating blockchain for IoT. It also provides a comparison of the various platforms, architectures and consensus used in IoT blockchain

Chapter 3 provides the detailed description of the five key components to be considered while creating IoT blockchain along with their design considerations and challenges that should be considered while creating blockchain architecture for IoT. It also define gaps that hinder creating a secure blockchain framework for IoT. It also presents the simulation result of two different types of IoT blockchain architecture and its throughput is compared.

Chapter 4 Provides an architecture for IoT blockchain using the components and finding identified. The implementation, performance evaluation and Security analysis of the proposed architecture is provided in this Chapter.

Chapter 5 provides an architecture for creating PKI in IoT blockchain. It includes background in PKI, a proposed architecture, implementation details and comparison of the proposed system with the existing architecture.

Chapter 6 provides a detailed state of the art survey on data confidentiality for blockchain-based approaches for IoT. In particular, a comparison on various encryption schemes used for data confidentiality in IoT blockchain. Further, it provides examples of generic use-case of Traffic speed radar, showing how Identity Based Encryption can be used to provide data confidentiality in IoT blockchain.

Chapter 7 provide a simple and efficient node selection mechanism that can perform consensus without wasting energy. It provides a brief literature survey on consensus used in IoT blockchain along with the proposed approach and consensus node selection algorithm. The approach uses PBFT, where the nodes to participate in the consensus are not predetermined by a central authority. Nodes are selected based on their performance in the blockchain.

Chapter 8 provides results of implementation and evaluation of the proposed architecture.

Chapter 9 provides conclusion, limitations of the proposed system and future work

1.6 **Summary of Results**

Through our experiment, we identified that a blockchain architecture with mobile edge computing and cloudlet computing with hierarchical identity based encryption can provide an efficient privacy preserving architecture for IoT. The System parameters and the Network parameters we obtained through experiment show that it is possible to create a blockchain network for an event driven IoT like Traffic Speed radars. In addition, we identified that IBE has the potential to provide data confidentiality for edge-based IoT blockchain, when compared with the other encryption techniques. We contributed in creating a decision framework by identifying the key components that will help in creating blockchain architecture for IoT and creating an efficient lightweight consensus node selection process for IoT blockchain. We also provided a distributed architecture for PKI in IoT blockchain.

Chapter 2 : LITERATURE SURVEY

This Chapter review works in the field of blockchain and Internet of things. In addition, it also reviews on core topics in cryptography and the building blocks of blockchain and Internet of Things. It includes symmetric cryptosystem, asymmetric cryptosystem, digital signatures, hash function, edge computing and smart contracts. A detailed description of Identity based encryption (IBE) is provided, as IBE is used in the proposed architecture. Various integrations, challenges, design considerations of IoT and blockchain are reviewed in this Chapter.

2.1 **Internet of Things**

Information and communication technology is growing at a rapid pace. Advancement in semiconductor devices and communication technologies allows a multitude of devices to communicate through the internet. These devices enable a machine-to-machine and machine-to-human communication. Such a trend can be referred to by many terms, including Internet of-Things (IoT), Internet-of-Everything (IoE), Internet-of-Vehicles (IoV), Internet-of-Medical-Things (IoMT), Internet-of-Battlefield-Things (IoBT), and so on (Banerjee, Lee & Choo 2018). These devices usually have sensors that can detect data from the physical environment. The detected data is then stored into centralized cloud storage for analysis and processing by various applications. The data residing in the centralized cloud is vulnerable to various forms of attack.

IoT is not a technology nor an industry. It is a complex eco-system and comprised of an array of technologies, products, and services, and with the collaboration of many stakeholders. It is a concept that can be applied across industries and sectors. IoT could be viewed as a transformational enabler that brings economic and social benefits to society. Sensors collect data from the physical

environment. The data is accessible from anywhere and at any time. This data is learned, analyzed and turned into insights that enable to make data driven decisions.

Applications areas of IoT include healthcare, smart homes, smart city, wearables, energy management, supply-chain and asset tracking. It comprises of devices that can compute, communicate, and sense from physical environment. There are three kinds of data communication mode in IoTs. Event driven, periodic (time-driven) and on-demand (query-driven) reporting (Al-Karaki & Kamal 2004). In the event-driven mode, sensors communicate to gateways or sink when a particular event happens. In periodic reporting mode, sensors sense data in pre-determined time and periodically send the data to the gateways. However, in the on-demand mode, gateways can query the data when required. IoT architecture can be broadly classified into three layers: end-device layer, edge-device layer and server or back-end layer (Ramachandran & Krishnamachari 2018). End device layer comprises of sensors, low powered embedded platforms, and wireless communication technologies. These devices sense data from physical environment and pass it to edge-device layer, which acts as a network gateway. Data from multiple end-devices are processed in this layer to meet the real-time demands of application. These data are then passed to a server or back-end layer for further processing. This layer includes web-servers, databases, data-analytics engine, etc. End-users communicate to this layer to access the data.

The National Institute of Science and Technology, NIST, defines IoT as a concept based on creating systems that interact with the physical world using networked entities, such as sensors, actuators, information resources, and people. NIST further elaborates the IoT concept by introducing foundation concepts of IoT Component, IoT System and IoT Environment. There are several cybersecurity frameworks in industry and following standards, recommendations and best practices in vital for business operations. The NIST cybersecurity framework (Boeckl et al. 2019) was

updated to reflect the rise of IoT. Its cybersecurity framework includes five key components, which are: identify, protect, detect, respond and recover. The initial step is to identify the assets, business environment, risk assessment, governance and risk management strategy. Protect includes access control mechanism, data security, information protection, process and procedures and awareness training. The third step is to detect anomalies and events, continuous monitoring and detection process and procedures. The fourth step is to respond to anomalies and finally to recover.

The International Telecommunications Union, ITU, on the other hand, defines the IoT as a global infrastructure for the information society, enabling advanced services by interconnecting physical and virtual things based on existing and evolving interoperable information and communication technologies, ICT (ITU 2012). According to the U.S. Computer Emergency Response Team, US-CERT, the Internet of Things refers to any object or device that sends and receives data automatically through the internet (Lowne 2014). The IETF, the Internet Engineering Taskforce, defines the Internet of Things as the network of physical objects or things embedded with electronics, software, sensors and connectivity to enable objects to exchange data with the manufacturer, operator and/or other connected devices (Ivanov 2014).

An IoT solution can be realized with a number of technologies, from advances in chipset, modules, hardware, sensors and actuators, to digital connectivity, cloud computing, AI, blockchain, applications, and smart phones. It is appropriate to say that the Internet of Things is not made up of one enormous network of connected devices. Rather, it is made up of many different networks of devices. Some are private, some are public, and most are connected to the internet in one way or another. This is because IoT is applicable across many industry sectors, from smart cities, consumer and household businesses, automotive and transportation, to agriculture, healthcare, supply chain, retail, fleet management, and manufacturing.

There are fundamental characteristics that define IoT:

1. IoT endpoints is a term generically used to denote an IoT ”device,” which is capable of sensing (i.e., sensor) and/or executing an action as instructed (actuator). An IoT endpoint can be a simple sensor, a stand-alone device with sensor/actuator and communications capability, or as an embedded IoT ”function” within a larger device/machine (e.g., a vehicle, a washing machine, etc.).
2. Sensors and actuators are the key building components of an IoT device/object/thing, enabling it to generate data from its environment.
3. Connectivity refers to the communication activity between IoT devices/endpoints, from IoT devices to an IoT platform. There are varieties of connectivity technologies that can be applied.
4. Fundamental to the concept of IoT, this is the ability for an IoT device to produce and provide data. In general, “enabling technologies” refers to cloud computing technology that provides various services such as data storage, analytics, applications, SaaS, IaaS, and XaaS. Specifically, emerging technologies that increasingly play significant roles in making the IoT more valuable are Big Data Analytics, Artificial Intelligence, Machine Learning, Deep Learning, and Block Chain.

The key reason behind implementing any IoT solution in any industry sector is the desire to generate values, either for business or personal purposes, using information gathered to make observations or to take action. The value could come in the form of productivity improvement, efficiency gains, process automation, cost reduction, quality-of-life improvement, monitoring valuable assets, and so on. Though not an outcome or a characteristic of an IoT application, cybersecurity (including privacy, reliability, and resiliency) is critical to the business success of

IoT as well as to the privacy and safety of consumers. The concept of hundreds or thousands of IoT devices belonging to the same network, all connecting to the Internet and being vulnerable to cybersecurity exploitation could cripple the entire IoT network, rendering the business inoperable. In the recent years, we have seen a steady advancement in the wireless sensor networks, communication and information technology. IoT is a technology that is still underdeveloped and requires many improvements in its architecture, platforms, communication, security, etc. The devices are reducing in size, consumes less energy and hardware cost is reduced. This enabled them to be integrated into everyday objects (Mattern, Friedemann 2010). As cited in (Uckelmann, Harrison & Michahelles 2011) the term 'Internet of Things' came into attention in September 2003 when Auto-ID Centre launched their vision of a supply chain management that can be automatically tracked. This trend has created a huge number of tiny devices that are connected to the internet to serve specific functionalities. Such types of devices are collectively called Internet of Things. It is considered as a global network infrastructure where numerous devices are connected to each other through the internet (Xu, He & Li 2014). They are rapidly growing and have high impact on everyday life. These devices can be referred to as smart objects that have the ability to interact and communicate with each other, within themselves, with an end-user, or with an interconnected object (Atzori, Iera & Morabito 2010). These objects have minimal communication and computational facilities. They consist of sensors, actuators, mobile devices, and RFID tags. When the number of devices connected to the internet increased, the problem of addressing these devices with a unique address was a challenge. Identifying these devices with a unique address was made possible by the IPV6 remarkable decision to increase the address space. This helped in creating a fully functional IoT. The huge address space provided by the IPV6 can provide unique addresses to billions of devices (D. Foote 2016).

2.1.1 **Key Applications of Internet Of things**

IoT devices can be simple sensors that monitor one particular thing, such as temperature or moisture. Other IoT things can also be a stand-alone product such as a personal tracker, a connected camera, or a toy. IoT devices or functions can also be embedded within a household item such as a washing machine, a refrigerator, a vehicle, or an air conditioner.

According to a survey by GSMA (GSMA 2015), the top trending IoT applications of users' choice are smart appliances, smart energy meters, wearable devices, connected cars and smart health devices. These devices are mainly used in environmental monitoring, surveillance, smart cities, smart homes and industrial equipment (Miorandi et al. 2012). Some of these applications are briefly described below.

- *Smart Homes:* A smart home consists of various devices at home connected to a network, which can be controlled by the owner. This provides improved security and manages home appliances and energy efficiently. Example of such energy saving products for the smart home are smart bulb, air conditioners, refrigerators, washing machine, air pollution sensors (Gubbi et al. 2013).
- *Wearables:* Wearable IoT devices are mainly used for health monitoring, fitness and entertainment. These devices are small in size and include features that serve purposes like activity tracking, monitoring sleeping pattern and heart rate tracking.
- *Smart Cities:* A smart city is equipped with devices that can send and receive data or signals through the internet. For example, each street light can gather and send information. Parking slots can be shown to user in real time and can find charging stations for electric vehicles. Waste bin will be triggered when it is full. Watering system monitoring will be automatic.

Sensors will detect leaks and are triggered when necessary. It can plan its preventive maintenance activities and can monitor security activities (Hall, R. E., Bowerman, B., Braverman, J., Taylor, J., Todosow, H., & Von Wimmersperg 2000).

- *Industrial equipment:* IoT devices play a major role in many industries today. This includes automatic managing of workers through surveillance and alarming system to temperature sensors in the office buildings. Some of the industries that have adopted IoT include agriculture, food processing, environmental monitoring and Health Care(Xu, He & Li 2014).

2.1.2 **Architecture of Internet of Things**

IoT architecture can be considered as a system, which is a collection of numerous active physical things, sensors, actuators, cloud services, IoT protocols, communication layers, users. It can be physical, virtual or hybrid. As cited in (Ray 2018), the well defined form of IoT architecture as "a dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual 'Things' have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network"

IoT solutions are made up of multiple technologies, products, and services, and can be complex to navigate and define. To help understand the building blocks of any IoT solution, a number of organizations from around the globe have attempted to define IoT reference architecture. Below are few sample organizations from around the globe which have developed some sort of IoT reference architecture.

The Industrial Internet Consortium, IIC has developed a common architecture framework. Its aim is to develop interoperable industrial IoT (IIoT) systems in the public and private sectors (The Industrial Internet of Things Volume G1: Reference Architecture 2019).

The ISO/IEC 30141 standard proposes an IoT reference architecture that describes IoT systems from a number of different views: conceptual, system, domain, network, and functional, plus a cross-sectorial service ecosystem view(ISO/IEC 30141:2018 [ISO/IEC 30141:2018] Internet of Things (loT) — Reference Architecture 2018).

IoT Alliance Australia, IoTAA - IoT Reference Framework is vendor-and-technology neutral, which enables IoT solutions to be viewed end-to-end, from industry, to business, end-users, and technology building blocks. Depending on the specific needs of the organization, the framework is designed to be used as a foundation on which concrete IoT architectures, frameworks, and security can be built. Organizations can utilize the framework to adapt to their needs, whether it's security, organizational skills gaps, or technology evaluation (IoT Alliance Australia 2018).

IEEE 2413**,** is a standard for an architectural framework for the internet of things which is under development stages. (2413-2019 - IEEE Approved Draft Standard for an Architectural Framework for the Internet of Things (IoT) 2019).

All architecture are developed considering the security, privacy, safety, reliability and relationships between participants in the IoT ecosystem. Several architecture layers of IoT architecture are available in the literature. Architecture of IoT varies within devices due to the heterogeneity of the devices. These devices are manufactured by various companies with different specifications. Generally, the basic architecture of IoT is shown in `Figure 2.1`. This architecture consists of sensing/perception layer, networking layer, middleware layer, application layer and business layer (Khan et al. 2012). The perception layer consists of the physical object or the sensor devices. These

objects sense data from the physical layer and communicate to the middleware layer through the network layer. These objects can be 2D-Barcode, RFID, or infrared sensors. The information coming from barcode scan events, RFID-based locations, or data received from the sensors are passed through the network layer. The network layer uses ZigBee, Bluetooth, 3G, and WIFI as the transmission medium to pass these data to the middleware layer.

The middleware layer use database to store the data collected by the sensor. These data will be passed to a centralized database for further processing. The application layer collects the data from the middleware layer and integrates with smart apps. Business layer is responsible for the overall management of the IoT system and services. It builds business models, flowcharts and graphs based on the data received from application layer.

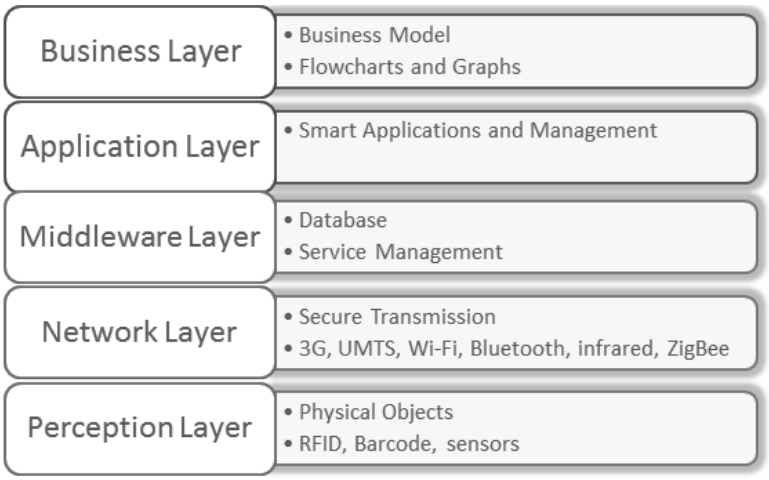


Figure 2.1:Architecture of IoT

An example of security architecture for IoT is provided in Figure 2.2(Yang et al. 2013). It includes a perception layer, transport layer and application layer. Perception layer includes the IoT devices

the key distribution center. The transport layer includes the CA, the sink nodes, Database and Key generation servers and the application layer includes the IoT data processing center.

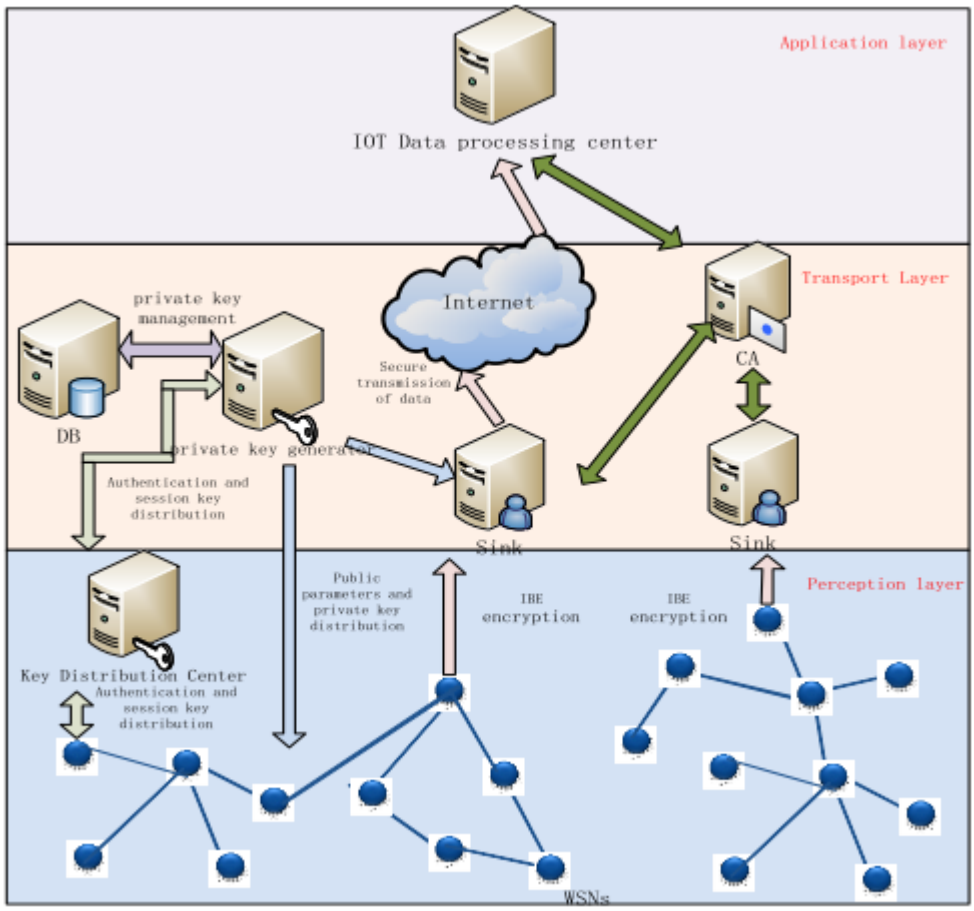


Figure 2.2:Security architecture for IoT (Yang et al. 2013)

The various functional blocks (Ray 2018) in IoT include, devices that can collect data from physical enviornment and are capable of controling and monitoring activities. These data are gathered locally or in a centralized server. It can also exchange data with other connected devices or cloud based applications or perform some tasks locally and other tasks within IoT infrastructure. Wired and wireless interfaces are used for communication. These include the interfaces for I/O connectivity for sensors, internet connectivity, audio/video inteface and memory and storage interfaces. Almost all IoT devices generate data using sensors and this data is further processed to

generate useful information. For example, data generated though soil moisture sensor can sense the moisture of soil in a garden, and when processed can help in determining the optimum watering schedules.

2.1.3 **Communications In IoT**

Key to the IoT concept is communication between IoT endpoints and the IoT platform. IoT communication protocols are used for communication between devices and remote servers. These protocols work in data link layer, network layer, transport layer, and application layer. There are many different technologies and protocols that can be implemented. The Connectivity technologies include Wired connection (Ethernet), Short range wireless connection (WiFi, Bluetooth, RFID), Long range wireless coneection known as LPWAN (Low Power Wide Area Networks), which includes LoRaWAN, SigFox, NB-IoT/Cat-M1 (cellular), Weightless, Ingenu, WiSUN and Cellular network that includes NB-IoT/Cat-M1, LTE, Cat-1, 3G, GSM

Some of the most common communication protocols used in IoT deployments today are:

MQTT : MQTT stands for MQ Telemetry Transport. It is the most comon protocol used in IoT communication. It is designed for constrained devices to minimize network bandwidth and to ensure reliability and some degree of assurance of delivery. It is an extremely simple and lightweight messaging protocol. It is mainly used in industrial and smart home applications. MQTT is a publish-subscribe based messaging protocol on top of the TCP/IP protocol. It requires a message broker that is responsible for distributing messages to interested clients based on the topic of a message. These principles make the protocol ideal for IoT and Machine to Machine world of

connected devices, and for mobile applications where bandwidth and battery power are limited (Liñán et al. 2015).

CoAP: The Constrained Application Protocol (CoAP) is an application layer communication protocol that allows small, low power sensors, switches, valves and similar devices to communicate interactively over the Internet. It supports multicast with low overhead and simplicity. It is is designed to easily translate to HTTP for integrating with the web. It make use of two message types, requests and responses, using a simple binary base header format and run on devices that support UDP and optionally to DTLS (Datagram Transport Layer Security), providing a high level of communications security.

Any bytes after the headers in the packet are considered the message body and when bound to UDP the entire message must fit within a single datagram. The length of the message body is implied by the datagram length. (Liñán et al. 2015).

HTTP: The Hypertext Transfer Protocol (HTTP) is the underlying protocol used by the World Wide Web. It is a request-response protocol that defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.

TCP: The Transmission Control Protocol (TCP) is one of the main protocols of the Internet protocol suite. It is a Transport Layer protocol. It is reliable and connection oriented protocol. It uses 3 way handshake to establish a connection.

UDP: User Datagram Protocol (UDP) is a Transport Layer protocol. It is a part of Internet Protocol suite, and is referred as UDP/IP suite. While the TCP is reliable and connection oriented

protocol, UDP is unreliable and connectionless protocol. Hence, there is no need to establish connection before transferring data.

SOAP: Simple Object Access Protocol (SOAP). It is a messaging protocol based on XML and is used for exchanging information among computers. SOAP can be delivered via a variety of transport protocols and can be used in a variety of messaging systems. Its initial focus is remote procedure calls transported via HTTP.

Apart from communications, an IoT system includes various services, management, Security and Application. An IoT system serves various types of functions for modeling of device, publishing data, controlling device, data analytics, and device discovery. It is the management block, that provides different functions to govern an IoT system that seek the underlying governance of IoT system. Authentication, authorization, privacy, message integrity, content integrity, and data security are provided by the security functional block. The most important in terms of users is the Application Layer. It acts as an interface that provides necessary modules to control, and monitor various aspects of the IoT system. Applications allow users to visualize, and analyze the system status and can make decisions and predictions.

2.1.4 **Challenges in Internet of Things**

The recent growth in IoT devices has imposed many challenges in the world of electronics and communications. Some of the key challenges in IOT are security and privacy, interoperability of IOT and identity management. Due to the limited computational power of IoT, it is inefficient to use some of the conventional public key cryptosystem. Hence, IoT require lightweight cryptography (Babar et al. 2011). The data from the sensor devices are transmitted through the network layer, which is vulnerable to many types of attack.

Manufacturers create devices using their own technologies and standards. Hence, standardizing these devices to work and collaborate with other devices is a key challenge. As far as naming and identity management are concerned, every IoT device requires a unique identity. As organizations rush to launch new IoT initiatives, they are less concerned about what level of access do these devices have on sensitive and non-sensitive data. Hence dynamically assigning identities for the IoT device is a challenge (Khan et al. 2012).

2.1.5 **Utilities of IoT**

(Ray 2018) have provided an example of how an IoT system should be self adapting. Surveillance IoT System comprising of several surveillance camera should be self adapting with the changing contexts and should be able to take actions based on their operating conditions, user's context, or sensed environment. It should be able to adapt to day and night light. It should change resolution from lower resolution to higher resolution modes while detecting motion and alert nearby devices. In this example, the surveillance system is adapting itself based on the context and changing (e.g., dynamic) conditions.

IoT devices may have self-configuring capability, with minimal manual or user intervention to configure themselves, setup the networking, and fetch latest software upgrades. It should allow a large number of devices to work together to provide certain functionality. IoT devices may support a number of interoperable communication protocols and can communicate with other devices and also with the infrastructure. Each of IoT device has a unique identity and unique identifier like a URI or IP Address. IoT device interfaces allow users to monitor their status, to query other devices, to control other devices remotely, and create association with the management, control and configuration infrastructure. It can adapt based on the context, allowing communication with users and environmental contexts through intelligent interfaces.

2.1.6 **Security in IoT**

In the context of IoT security, Three elements are concerned with IoT applications and platform security. The first one is the protection of data while it is in transit. The data transferred within different applications and platforms should be protected. The second is the application access security, which includes the perimeter security protection, these could include web server, hosting, mobile application server, web portal, development environment, analytical platform, visualization platform, user identity management. The third is the software and firmware security which concerns the integrity of hardware and software of various applications in the platform. This is also concerned with the connectivity and device management.

2.2 **Blockchain**

Blockchain is essentially as distributed, immutable ledger where data are distributed as transactions to all nodes in the network. Transactions are stored in blocks where each block points to an immediate previous block via an inverse reference that is essentially the hash value of the previous block. The block structure contains the block version, hash of previous block, timestamp, a nonce value staring from 0 and increasing for every hash calculation, the number of transactions and MerkleRoot (i.e., the hash value of the root of a Merkel tree with concatenating the hash values of all the transactions in the block) (Nakamoto 2008). Blockchain allow nodes to agree to a consensus without the use of centralized authority. Blockchain originally came up as a solution for double spending problem in crypto-currency. Hence majority of its current applications are on digital currency. Apart from being a payment protocol, the introduction of smart contracts in blockchain opened its way to other applications including Internet of Things, supply-chain management, asset tracking, certificate management. Blockchain became more suitable for applications where multiple stake holders are involved. Even-though the first blockchain bitcoin,

was launched a decade ago, blockchain applications on other use-cases other than crypto-currency are not fully developed.

Distributed computing is witnessing a new era with the growing popularity of blockchain. Internet allows to communicate data or information between two parties, However, in some cases like purchase of items, a trusted third party is required to validate the transaction. Usually, banks and credit card companies acts as trusted third parties. Blockchain allows to transfer value without or purchase items without the need for a trusted third party. It provides transparency, where transactions are visible to every party. An example of a Blockchain transaction is provided in Figure 2.3. The figure shows two transactions where the first transaction has a sender and two receivers. Sender and receivers are identified by a long string. The amount transferred is 0.05574799 BTC. In that 0.05296059BTC is transferred to the first account, while 0.00165740 BTC is transferred to the second account. In the second transaction, a sender is sending 22.22743709 bitcoins to 3 different receivers.

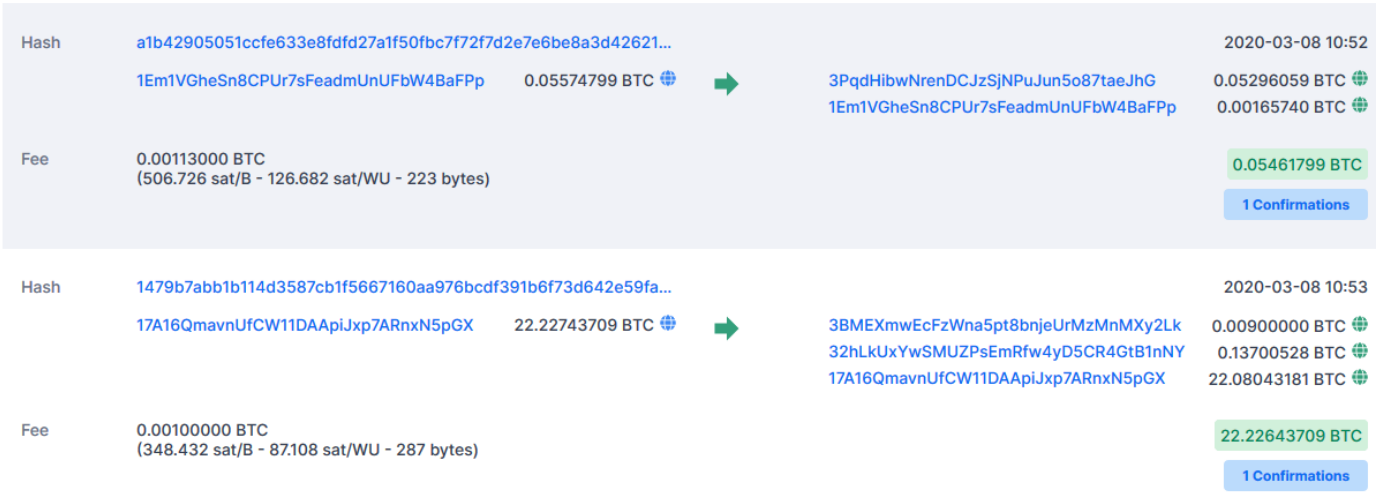


Figure 2.3: Example of bitcoin transaction

These transactions are validated by miners who has to solve a cryptographic puzzle to win mining rewards. Bitcoin mining rewards get reduced by half every four years. Figure 2.4 shows an example of mining reward of 12.57616122 BTC awarded to an addresses who solve the puzzle.

Figure 2.4:Mining reward

Conceptually, blockchain is essentially a distributed database, where assets can be stored and exchanged through decentralized network of computers while still providing security and anonymity. The distributed records are tied to their history using hash values, which restricts the record to be altered or modified. The only kind of update allowed is to extend the blockchain by creating new transactions. Still, the old transactions will be available in the blockchain. A trustable workflow is maintained by ensuring that a majority of the validators behave honestly and hence all transactions are transparent and auditable. It also confirms that each unit of value was transferred only once. Even though the asset is distributed, only the owner who has the private key can make transactions on this asset. The other computers in the network act as validators for the transaction. It securely records transactions into a public ledger among nodes without the need for a trusted third party. In the centralized cloud approach when an asset is owned, it is either stored in the custody of the owner or with a trusted intermediary or a centralized authority like a bank. Blockchain eliminates this risk by storing the data in a distributed fashion which are auditable. Bitcoin is an example of a public blockchain, which can also be referred to as a permissionless

blockchain. It does not require an access control or a trusted authority to manage the blockchain. Anyone can join and leave the blockchain. However, a permissioned or a private blockchain enhances collaboration among various industries and partners by providing a trusted database that is transparent to the participating entities. The entities can restrict its access and can decide on the consensus process. Consensus in a permissioned blockchain follows a Byzantines fashion.

Byzantines General problem is a situation to avoid catastrophic failure in a distributed environment. Nodes in a distributed environment, can be reliable or unreliable. For example a malicious node can act as failed or as functioning to different observers. So it is difficult for other nodes to determine the state of the malicious node. Hence they have to reach to a consensus to determine its state. Byzantines fault is any fault presenting different behavior to different systems. The objective of Byzantines fault tolerant is to defend against the malicious nodes and reach to a consensus among other nodes for the correct operation of the distributed system.

Formal Definition (Byzantine fault 2020): Given a system of 'n' components 't' of which are dishonest, and assuming only point-to-point channel between all the components.

Whenever a component 'A' tries to broadcast a value 'x' the other components are allowed to discuss with each other and verify the consistency of 'A's broadcast, and eventually settle on a common value 'y'

Property:

The system is said to resist Byzantine Faults if when a component 'A' broadcast a value 'x':

1. If 'A'is honest, then all honest components agree on the value 'x'.
2. In any case, all honest components agree on the same value 'y'.

Permissioned blockchains are designed to tolerate Byzantines failure and offers strong security. When compared with the traditional database system used in financial and assent management applications, blockchain system incur less human labor and lower cost of infrastructure. In addition, it reduces human errors and the need for manual interventions due to conflicting data.

Apart from public and private blockchain, hybrid architectures(Ramachandran & Kantarcioglu 2018)(Sagirlar et al. 2018)(Sharma & Park 2018) are emerging that combines public and private blockchains. It permits delicate information to be opened on a private blockchain at the same time using public blockchain to make the winner announcement and payments accountable.

Some of the popular applications that use blockchain are smart contracts, distributed cloud and digital assets (Olleros, Zhegu & Pilkington 2016). Some of the industries that can benefit from blockchain are finance, cross-border transactions, Insurance, Government, Supply chain management, Healthcare and Internet of Things

Bitcoin (Nakamoto 2008) launched in 2008 was the first decentralized digital currency that is built on the blockchain technology. Followed by the success of Bitcoin, several cryptocurrencies, tokens and ICOs have emerged. After much speculation and hype, a significant number of them have become problematic or worthless(Vossen, 2018.). It was Ethereum, that emerged with smart contracts to create blockchain for asset management. Hyperledger project is under the Linux foundation for permissioned blockchain. It is currently the largest and the most popular permissioned blockchain. Bitcoin is also known as cryptocurrency because security and anonymity of the currency is provided through cryptography. The value of the currency is created and stored in transactions. What differentiates bitcoin from traditional currencies and payment card system is that bitcoin is a data structure that is replicated in many different nodes that are part

of the network. There is no central authority or central server that stores the user's asset value making it difficult for cyber attackers to target a single machine. Bitcoin allows only values to be exchanged. Transactions are hashed and added to the block. Identity of the customer is verified through a public-private key pair where a customer can have more than one public-private key pairs. Each user maintains public-private key pair where public key is shared with other agents whereas private key is maintained as private in the wallet. To make a transaction, the sender uses the public key of the receiver and digitally signs the transaction using senders private key to provide authentication.

2.2.1 **How blockchain works**

Blockchain records the transactions in units of block. Each block contains the hash of the previous block, hash of the current block, time stamp, other information and transactions for that block. When a sender node creates a transaction, it distributes it to all other nodes in the network. The receiving nodes validate this transaction and perform proof of work. The node that succeeds the proof of work will broadcast it to all other nodes and add the block to the chain (Nakamoto 2008). The transaction includes the public key of the receiver and is signed by the sender. Hence every other node can validate the authenticity of the transaction. Each block contains a hash of the previous block which means every block is linked to each other as shown in Figure 2.5(Nakamoto 2008) making it difficult for an attacker to modify the transactions or blocks.

The underlying technology behind Bitcoin protocol is cryptography. In cryptography, the integrity of a message is obtained through hash functions and authentication through digital signatures. Public key cryptography is used in digital signatures which generate a public-private key pair. The keys are related to each other. It is computationally infeasible for an adversary to

compute the private key, if the public key is known. A Bitcoin address is Base58 encoding of hash of ECDSA public key which is computed as

V = 1 byte of 0 (zero); on the test network, this is 1 byte of 111

Key hash = V ||RIPEMD-160(SHA-256(public key))

Checksum = 1st 4 bytes of SHA-256(SHA-256(Key hash))

Bitcoin Address = Base58Encode(Key hash || Checksum)

The sender signs the transactions with the private key and receivers validate the authenticity of the transaction with the sender's public key. The Transactions are recorded in units of block. Each transaction contains the receiver's public key and Hash of previous transactions signed with the sender's private key as shown in Figure 2.5. Each block contains a record of recent transactions, a reference to the previous transaction, and the answer for the mathematical puzzle. Miners validate the transaction and add the block to the blockchain after solving a mathematical puzzle. The mathematical puzzle is difficult to compute but easy to validate. As a reward for the work, miners are provided with new Bitcoins. Out of the 21 million available Bitcoins, 14 million are already generated. The rate at which Bitcoin is generated will be reduced to half every 4 years. Miners were rewarded with 50 Bitcoins from 2009 to 2012 where the exchange rate during at the time was 1 USD = 1,309.03 BTC. In 2012 November, the mining was reduced to 25 Bitcoins with 1 USD per Bitcoin. In 2016, it was around 12.5 Bitcoins to approximately 733 USD and from June 2020, the mining reward is further reduced to half to 6.25.

2.2.2 **Bitcoin Transaction**

The Bitcoin system operates on a list of blocks. Each block contains a header and transaction data. Each transaction is linked to each other as shown in Figure 2.5. The header is 80 bytes that

contain a 256-bit hash of the previous block H_{i-1}, the time stamp T_i, the 32-bit Nonce Ni (used to generate block), the hash of the transaction data Tx_i and the difficulty parameter di. For a block to be valid, the double hash of the block header should be less than the linear function of difficulty parameter(Biryukov, Alex, Dmitry Khovratovich 2014). A block will be valid only if these conditions are met.

$$H_i = SHA\text{-}256(SHA\text{-}256(H_{i-1} \,||\, T_i \,||\, Tx_i \,||\, d_i \,||\, N_i)) \; < f(d_i)$$

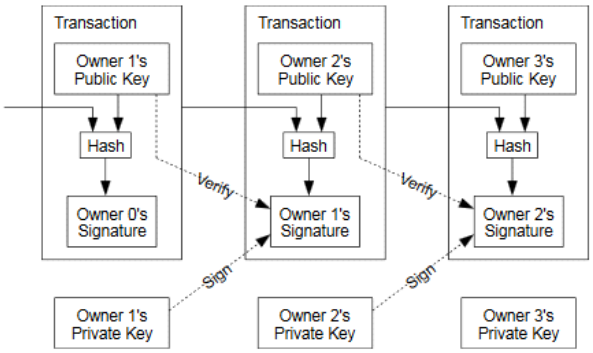


Figure 2.5:Bitcoin transactions (Nakamoto 2008)

A Bitcoin transaction consists of input, output, the time stamp, and the amount of Bitcoins transferred. The transaction can be of two types, either a minted transaction that generates new Bitcoins to the miner Figure 2.6 or transfer of Bitcoins from sender to receiver Figure 2.7. New Bitcoins are added to the address of the miner who first solves the complex mathematical puzzle.

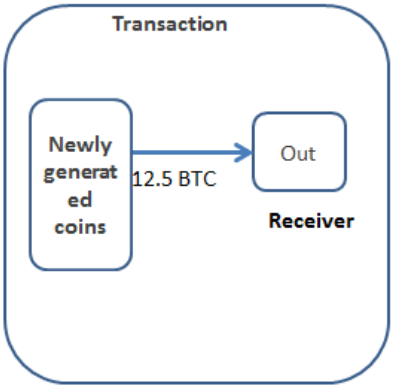


Figure 2.6:Newly generated coins

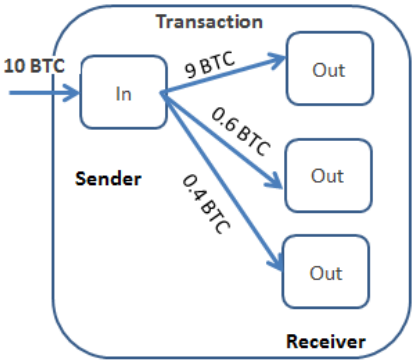


Figure 2.7: Transaction from sender to receiver

Transaction input represents all the senders, and transaction output represents all receivers. There can be more than one sender and receiver. All transactions can be viewed from blockchain.info site. The entire Bitcoin system is based on the transactions. There is no coin exchanged between users, whereas coins are implicitly represented by flow of transactions (Di Francesco Maesa, Marino & Ricci 2018). When new transactions are created, they are broadcasted to all nodes in the Bitcoin system.

2.2.3 **Attacks on Blockchain**

Although a number of attacks are documented for blockchain most of them are not relevant in practice (Buccafurri et al. 2017). Some of the attacks available in the literature are:

- Malwares: The distributed nature of blockchain architecture introduces the spreading of malwares. With the development of newer protocols with the ability to store and compute data, it would be possible to store malicious data within the blockchain (Cermeño 2016). Malware effects on the devices in blockchain will result in its propagation to other nodes in the blockchain. This can result in crashing of the nodes.
- Distributed Denial of Service Attacks (DDOS): The study conducted by (Vasek, Thornton & Moore 2014) found that 7.4% bitcoin related services have experienced DDOS. In these eWallets, financial services, mining pools are more likely to be attacked. Just like, in the case of traditional wallet, bitcoin wallet also needs to be protected. It is recommended to use two-factor authentication to protect the bitcoin wallet. For additional layer of security, wallet should be encrypted and backup to be taken.
- Phishing attacks on bitcoin wallets: Several phishing attacks on bitcoin wallets and on blockchain.info site were reported in 2018 (comodo 2018). Hackers created a site similar to blockchain.info and tried to steal the wallet information. In another case, hackers impersonated legitimate recipients and persuaded the investors to send bitcoins to their address. Once the bitcoin was sent, it could not be recovered.
- Majority Attacks: This type of attack is also known as 51% attack. Group of miners can decide which transactions should be approved or not if they can control the majority of the network mining power. This would allow them to reject other transactions or double spend their own transactions. If the blockchain network is free and open, this could be made possible especially with the rise of mining pools. However, the attack doesn't give full control over the bitcoin network. Similarly, in a private or permissioned blockchain, proof-

of-work will be implemented under the regulator's direction; therefore regulator will have an authority to control the network (Cermeño 2016)

- Sybil Attack: Sybil attack (Newsome et al. 2004) is controlling a peer to peer network using multiple identities. A single entity creates multiple fake identities to control the network. If an attacker is possible to control the majority of mining nodes in the blockchain, then he can create a fake transaction and add it to the blockchain.
- Eclipse attack (Heilman, Kendler & Goldberg 2015) It is a targeted attack on distributed system, where a malicious attacker isolates a specific node and cut off all its inbound/outbound connections with its peers. So attackers try to gain the 51% of the mining power by trying to isolate some of the mining nodes.

2.2.4 **Ethereum Virtual Machine and Smart Contracts**

In November 2013, the Ethereum concept was introduced by Vitalik Buterin. The Ethereum Virtual Machine (EVM) is the runtime environment for smart contracts in Ethereum. The code running inside the EVM has no access to network, file system or other processes as it is a sandbox and is completely isolated. Smart contracts even have limited access to other smart contracts. The operations of the blockchain in a smart contract based blockchain can be summarized as follows: All the Applications share a common state and common program code. Transactions are created to call the stored programs. Each participants can create transactions to call the program. Consensus Process is used to validate and verify all the transactions. After validating and verifying the transaction, a block is created. Based on the implemented consensus mechanism, the parties agree on the next received transaction block. Each new block contains a hash of the previous block, thereby forming a linked list.(Teslya & Ryabchikov 2018).

Smart Contracts are an ideal technology that can securely store contracts. They are computer programs that are used to transfer, monitor assets or digital currencies among parties under certain rules. It does not only determine the conditions and penalties but can also enforce those policies or agreements. Whenever a transaction is suppose to happen, smart-contract determine where the transaction was actually originated or to where the transaction should be transfer (Ali et al. 2019).

```
pragma solidity >=0.4.0 <0.7.0;

contract StorageContract {

   uint IoTData;

   function set(uint x) public {

      IoTData = x;

   }

   function get() public view returns (uint) {

      return IoTData;

   }

}
```

The first line indicates the version. This program can be compiled to a Solidity version between 0.4.0 and 0.7.0. All other versions are incompatible. The name of the contract is StorageContract. It first declares a variable 'IoTData' as an Unsigned integer. This can be considered as a slot in

database where you can query for the data. The function ‘set’ and ‘get’ are used to modify or retrieve the value of the data. The function ‘set’ is used to insert the data into the database, where as function ‘get’ is for retrieving the data from the database. Anyone could call set again with a different value and overwrite your number, but the number is still stored in the history of the blockchain.

Ethereum consists of external accounts and contract accounts. External accounts that are controlled by public-private key pairs and contract accounts which are controlled by the code stored together with the account. Contract address is determined at the time contract is created, whereas public key is used as the address of external accounts(Solidity 2016). For our experiment, we implemented Private Ethereum Blockchain in Linux. We used Ubuntu on Vmware. The installation, setting up and the smart contract execution steps are provided in Appendix1.

2.3 **Integration of Blockchain and IoT**

Current approaches in IoT implementations are largely centralized, which raise several security concerns like single point of failure, trust and privacy. In addition it limits their scalability. This alarmed the need for a decentralized trust mechanism in IoT. Blockchain can provide trust through cryptographic techniques without the need for a central authority. Recently, several blockchain based applications for IoT have gained attention due to its potential for improving security and privacy. A recent study by Juniper research (Research 2019) predict that a combination of IoT and blockchain on food industry can save billion dollars by reducing the retailers cost, simplifying regulatory compliance and tackling fraud. Giants in food industry like Carrefour, Nestle and Cermaq have already started using Hyperledger Fabric, a blockchain application developed by IBM (Carrefour 2019)(Nestlé 2019)(Cermaq 2019)

Blockchain is essentially a decentralized platform where a copy of each transaction is kept by all parties (Nakamoto 2008). The transactions are transparent and any modifications in them can be easily detected. Consider the example of a smart city where parking spaces are shown to users in real-time. Once sensors detect a free parking space, they update the centralized database. It is possible for a system administrator who manages this database to reserve a parking space for himself without showing this slot to others. In this case, integrity of the data from the sensor is compromised. The purpose of a blockchain network of interconnected devices is to eliminate the use of a third party and; hence, ensure that the real-time data provided by the sensor can reach every user in the network without any modification. In addition, blockchain allows IoT devices to communicate among themselves and make decisions automatically.

Decentralizing the IoT network has various advantages, including reduced costs associated with maintaining a central database for IoT transactions, as well as improved security and privacy, which eliminates the need for a third party. However, it remains unclear as to how these features can be implemented in IoT. This is mainly due to the limitations of IoT devices in terms of computational capacity, power and storage. For this reason, the blockchain protocol designed for cryptocurrencies cannot be used for IoT applications. Various IoT applications that can benefit from blockchain are shown in Figure 2.8. This includes supply chain management, health care, smart city, home equipment automation, energy management and asset tracking.

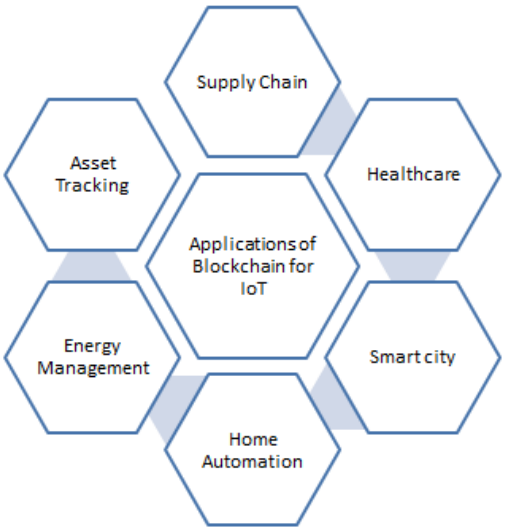


Figure 2.8 Applications of Blockchain for IoT

In the traditional supply-chain management there is no traceability and accountability. The price of goods can be artificially crafted. Blockchain can help the supply-chain industry to keep tamperproof ledgers and can keep track of products without an intermediary (Borah et al. 2020) (Abeyratne and Monfared, 2016). This ensures greater transparency and reduces corruption in the supply chain industry. In healthcare, the combination of IoT and blockchain help to easily collect patient data, monitor in real time and store data securely (Simic et al. 2017). Home equipment and IoT in smart cities can be automated using blockchain enabling device-to-device communication between equipment. Energy sectors are moving to implement blockchain because of its ability to lower cost and reduce harmful environmental impacts (Consensus 2019). Blockchain can help asset tracking by providing transparent, secure and accountable data collected from IoT devices attached to assets. Therefore, energy efficiency is one of the relevant issues that should be addressed when blockchain and IoT are integrated.

2.3.1 **How Blockchain can Address IoT Challenges**

IoT devices in the cloud architecture are connected through a cloud server. It processes and store the data sent and received by the devices. However, devices connected to the cloud are vulnerable to various attacks. Each block of IoT architecture could act as a bottleneck or single point of failure(Swan 2015). The cloud model is susceptible to manipulation. For Example: In the city of Flint, Michigan smart water meters were used to measure the quality of water. The authorities were insisting on the fact that water in the city is safe to drink whereas CNN article asserted that officials might have altered sample data to lower the lead level in water (CNN Editorial Research 2019). It reported that two of the collected samples were discarded by the officials. Such types of malpractice can be avoided by implementing blockchain for IoT. This is because the data generated by the sensors could not be modified.

In blockchain, devices rely on smart contract to exchange messages. Authentication is done by digitally signing the message with the private key of the owner which ensures that the message originated from the owner itself. This eliminates the possibility of man-in-the-middle, replay and other types of attacks (Swan 2015). Some of the advantages of using blockchain for IoT are:

- *Reduced cost:* According to Gartner (Gartner 2017)8.4 billion IoT devices will be in use in 2017 which is 31 percent increase when compared with 2016. This radically increases the storage and network capacity required by these devices. Using blockchain, devices can communicate with each other and can execute actions automatically. Hence cloud storage and administrative staff for maintaining cloud storage will not be required (Sun, Yan & Zhang 2016).
- *Single Point of failure:* Each entity in the IoT architecture is independent in their functions. Hence malfunctioning of any device can create a single point of failure. In a blockchain, all the devices are connected to each other and all transactions are copied to every node in

the blockchain; hence, malfunctioning of a single device does not affect the operations of other devices.

- *Resistant to Malicious Attack*: IoT devices are vulnerable to many types of attacks due to its centralized architecture. Some examples of attacks are distributed denial of service, deception attack, and data theft. These can be avoided with the blockchain architecture for IoT whereas blockchain is vulnerable to some other types of attacks as described in section II-H.
- *Trust:* A trusted third party is used in centralized architecture of IoT, whereas in blockchain, trust is provided automatically using cryptographic protocols.
- *Security and Privacy:* Due to centralized architecture of IoT, information is likely to be manipulated where as in blockchain, devices are interlinked and hashed. Hence, manipulation of data on one device cannot be propagated to other devices in the blockchain.

2.3.2 **Edge based architectures for IoT blockchain**

Edge computing is an emerging technology in Internet of Things. This was mainly introduced to overcome the scalability issues in IoT. It pushes the computing resources, storage, networking and intelligence decision making towards the edge and provide various benefits such as reducing the network traffic, easily managing the data and faster processing. This helps to reduce the computation load of the data centers and some computations are processed in the edge itself. Hence requests can be processed in real time and reduces the latency. Typical applications that can benefit from edge computing are emergency response, augmented reality, video surveillance, speech recognition, computer vision, and self-driving. Edge devices also support mobility due to the abundant availability and geo-distributed nature (Dolui & Datta 2017) (Pan et al. 2018).

Based on the network and communication protocols used, the intermediate edge nodes can be implemented as an edge layer between the end devices and the cloud. Dolui and Datta (Dolui & Datta 2017) classifies such kind of implementation of the edge layer into three types, Mobile Edge Computing (MEC), Fog Computing (FC) and Cloudlet Computing (CC).

Fog Computing includes devices like edge routers and gateways. They are used to store and compute data from end devices locally before forwarding to the Cloud. Whereas, Mobile Edge computing, uses radio network of intermediate nodes with storage and processing capabilities thus offering Cloud Computing capabilities within the base station network.

The Cloudlets are present in logical vicinity to the consumers and is similar to a data center based. They are dedicated devices with computation and storage capacity accessible to users. It allows end devices to offload computing to the Cloudlet devices with resource provisioning similar to that of a data center.

In most of the IoT deployments sensors send data directly to the cloud or use a gateway. Hence Cloud hosted storage architecture and data processing are mostly deployed. However, this type of architecture suffer from high latency and data transmission cost. The introduction of technologies like the fifth generation cellular network with advanced hardware capabilities, IoT devices would be able to send data at much higher rate imposing considerable scaling challenges for the cloud based central system(Mendki 2019).

Edge and Fog computing architectures enable using resources available close to the edge of IoT network. The connections between the mobile devices and the edge network are facilitated through radio network which normally employ a 4G long-term evolution and distribute a wide geographical area in smaller clusters, which are then controlled by radio network controller. The

network management functions and controlling the base station nodes are carried out by radio network controller (Bhattacharya et al. 2020).

(Stanciu 2017) has stated that for many IoT applications, including large-scale sensor networks that provides intelligent services and that monitor the environment, there is a need of a new platform, that can provide computational resources, mobility support, location awareness and low latency. Edge computing provides compute, storage and networking services towards end devices and can be seen as members of a decentralized network. To improve latency, computations should be close to data. Hence, for such applications edge nodes can provide the first steps for data processing, thus limiting the volume of data that should be transferred to the central cloud services. Integrating the IoT system into a three layer architecture including edge nodes and cloud services can reduce the network latency and improve performance. However, Edge computing has not replaced the two-layer architecture that includes just the IoT device and the cloud. To reduce the data transfer and storage requirements, the three-tier model for edge computing is necessary. For Applications that produce huge volume of data, data processing in the edge nodes is thus extremely important and should provide real-time response to end users (Stanciu 2017).

Fog computing (FC) can be described as a virtualized platform that seeks to provide various technological services such as computing, storage, control and networking services between end users and the Cloud. (Frimpong et al. 2020)The FC implementation is a decentralized Computing infrastructure based on Fog Computing Nodes (FCNs) include routers, switches, access points and IoT gateways that could be placed at any point of the architecture between the end devices and the cloud. The heterogeneity of the nodes is hidden from the end devices by creating a Fog abstraction layer. This layer performs resource allocation and monitoring, security and device management

along with storage and compute services. It supports devices at different protocols including non IP based technologies to communicate between the FCN and the end-device.

Mobile Edge Computing: MEC bring computational and storage capacities to the edge of the network within the Radio Access Network. It can be seen as an implementation of Edge Computing to reduce latency and improve context awareness. It has the capability run on a virtualized interface with multiple instances and has the capability to perform computations and storage on it. The service provider deploys the local data centers and servers at the edge within a mobile network. Eg: base stations of mobile network.

They can provide real time information on the network itself including the load and capacity of the network while also offering information on the end devices connected to the servers including their location and networking information. Mobile devices can access the edge servers to enhance their computing capability, e.g., IoT sensing data processing. Edge computing becomes a promising solution for mobile blockchain applications, because it can incorporating more miners, hence the robustness of the blockchain network is naturally improved. The mobile users have an incentive from the reward obtained in the consensus process(Xiong et al. 2018)

Cloudlet Computing: They are trusted cluster of computers, which are connected to the Internet, with resources available to use for nearby users. A Cloudlet can be treated as ”data center in a box” (Satyanarayanan et al. 2006) that run on a virtual machine that is capable of provisioning resources to end- devices and users in real time over a WLAN network. Cloudlets are provided over a one-hop access with high bandwidth, thus offering low latency for applications. A comparison of edge computing implementations is provided in Table 2.1(Dolui & Datta 2017)

Table 2.1: Comparison of Edge computing implementations

	Fog Computing	Mobile-Edge Computing	Cloudlet Computing
Node devices	Routers, Switches, Access Points, Gateways	Servers running in base stations	Data Center in a box
Node location	Varying between End Devices and Cloud	Radio Network Controller/Macro Base Station	Local/Outdoor installation
Software Architecture	Fog Abstraction Layer based	Mobile Orchestrator based	Cloudlet Agent based
Context awareness	Medium	High	Low
Proximity	One or Multiple Hops	One Hop	One Hop
Access Mechanisms	Bluetooth, Wi-Fi, Mobile Networks	Mobile Networks	Wi-Fi
Internode Communication	Supported	Partial	Partial

One of the pioneering work on Cloudlet computing was done by (Satyanarayanan et al. 2006). They identified the difference between the cloud and cloudlet computing. While the cloud is hosted by a public provider, the cloudlet are deployed within office premises and a managed locally with little or no professional attention. The public cloud needs to be professionally administered with a 24X7 operator. Cloudlet can have decentralized ownership, whereas cloud us centralized ownership by Amazon, Microsoft, Yahoo, etc. Cloudlet can be implemented though a local area network and can accommodate only few users at a time. The difference between cloud and cloudlet computing is given in Table 2.2 (Satyanarayanan et al. 2006).

Table 2.2: Difference between Cloud and Cloudlet Computing

	Cloudlet	**Cloud**
State	Only soft state	Hard and soft state
Management	Self-managed; little to no professional attention	Professionally administered, 24x7 operator
Environment	"Datacenter in a box" at business premises	Machine room with power conditioning and cooling
Ownership	Decentralized ownership by local business	Centralized ownership by Amazon, Yahoo!, etc.
Network	LAN latency/bandwidth	Internet latency/bandwidth
Sharing	Few users at a time	100s-1000s of users at a time

Due to the resource constrained nature, many security protocols cannot be used with IoT. Due to this, many security protocol has become inefficient while adopted for IoT. Blockchain combined

with smart contract can provide a trustless environment, and can provide unique features such as data/transactions persistence, tampering resistance, validity, traceability, and distributed fault tolerance. There are limited works on applying blockchain into decentralized IoT and edge computing systems. In a typical edge based system, there will be three tiers(Pan et al. 2018).It includes an IoT device tier, Edge tier and cloud tier. Edge tier consist of various edge nodes located within the IoT devices and the data from the edgenodes are passed to centralized cloud for further processing and storing. In this architecture the EdgeChain is located between the IoT applications and the edge cloud platforms in the shared infrastructure. It can run on different edge cloud platforms such as Cloudlet. It uses an internal currency system to link the edge cloud resource pool with the account of IoT device.

Several integrations of blockchain for IoT are available in the literature (Ali, Vecchio, et al. 2018)

1. Gateway devices/ Edge Nodes with IoT

   IoT devices will be registered to the gateways and gateways issues transactions in the blockchain. All communications go through the gateways. Hence Gateways acts as end-point in the blockchain. Degree of decentralization is less in such architectures. It needs external storage in cloud. The main disadvantage is it needs increased bandwidth, storage requirements and scalability.

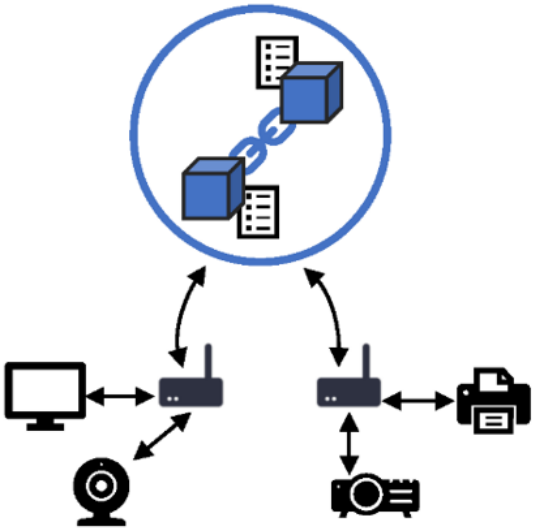


Figure 2.9:Gateway devices/ Edge Nodes with IoT (Ali, Vecchio, et al. 2018)

2. IoT Device only approach:

   In this case of architecture, devices issues transaction to the blockchain, however it does not store the transactions. The device itself is assumed to have the cryptographic capabilities for performing blockchain process. However, the main disadvantage is this type of architecture requires ability to highly scalable, incorporate different types of IoT devices, interoperability with heterogeneous devices, and increased computational complexity of IoT hardware. It is not suitable for devices with low computational capabilities. In terms of Latency, this approach is fastest, since it can work offline, it provides security. Only a part of IoT data will be stored in blockchain whereas the IoT interactions take place without using the blockchain

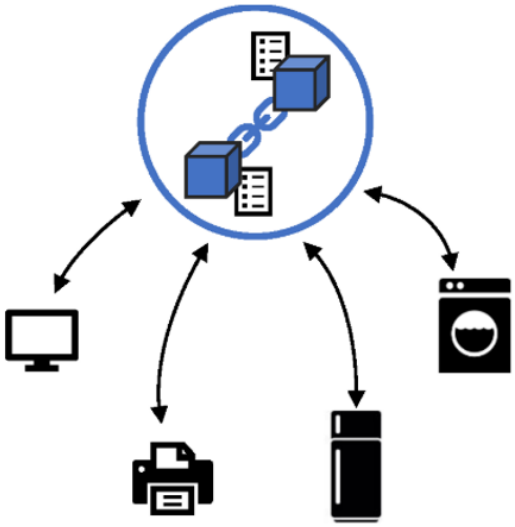


Figure 2.10:IoT Device only approach (Ali, Vecchio, et al. 2018)

3. Interconnected edge devices as end-points to the blockchain: In this approach, both the device and gateways can issue transactions and participate in blockchain process. This approach will have low latency and high throughput. This is more suitable in use-cases where reliable IoT data is required.

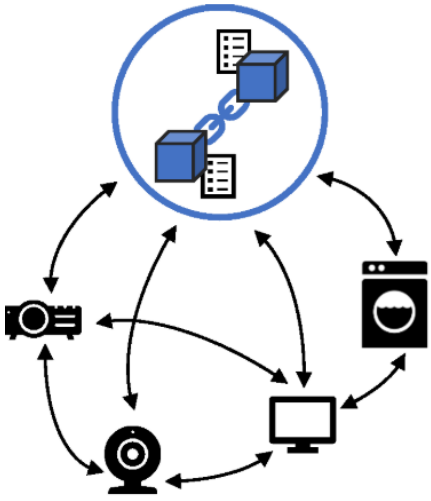


Figure 2.11:Interconnected edge devices as end-points to the Blockchain

(Ali, Vecchio, et al. 2018)

4. Cloud-blockchain hybrid with the IoT edge: This is a hybrid approach where IoTs, edges and cloud participate in blockchain. It can also utilize fog computing to reach to a certain level of decentralization.

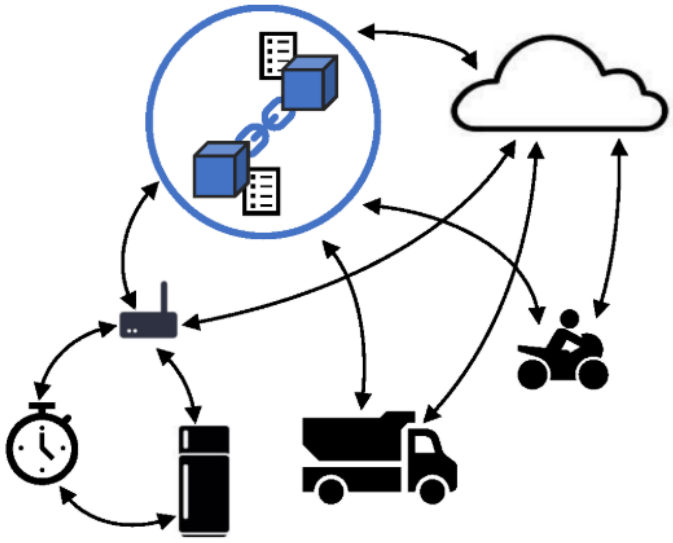


Figure 2.12:Cloud-blockchain hybrid with the IoT edge (Ali, Vecchio, et al. 2018)

2.3.3 **Drawbacks of edge computing**

If the edge node is overloaded or is malfunctioning, computing can increase the latency beyond tolerable limits. All the security techniques that are leveraged on cloud should also be implemented in the edges. This will include, Authentication, distributed intrusion detection techniques and access control mechanisms. The need for standardization is required across various vendor products for interoperability (Bagchi et al. 2020). Scalability is another issue with edge level processing. It is not easy to scale the resource infrastructure deployed at the edge as compared to the cloud based computing (Mendki 2019)

2.4 **Cryptography**

2.4.1 **Symmetric and Asymmetric Cryptography**

Cryptography is the technique of applying various transformation on data to provide Confidentiality, Integrity and Authentication. Confidentiality is the assurance that information is only available to authorized parties. Integrity says that information is only modifiable by authorized parties. Authentication provides assurance about the identity of an entity or the validity of a message. Mainly authentication is about two types which are Entity authentication and Data origin authentication. Entity authentication could be single factor authentication or multifactor authentication. In single factor authentication, usernames and passwords have been provided by users in order to gain access to the platform. However this kind of authentication is not much secure as it is prone to password leakage. In multifactor authentication, users use biometric features like fingerprint, retina, iris or hand geometry to provide additional factors for authentication. In order to discuss about data origin authentication, it is also known as message authentication which provides an assurance that the source of information is verified. Data origin authentication implies data integrity because if a source is confirmed, then the data must not have been altered. Message Authentication Code and Digital Signature are most commonly used.

Symmetric Encryption: It is single key encryption where same key is used for the encryption and decryption process. It is also called conventional or single key encryption. Because the same key is used for encryption and decryption, the communicating parties should exchange key before the communication. This is a major drawback of Symmetric encryption. Symmetric Encryption can be classified based on the type of encryption process as stream ciphers and block ciphers. Stream ciphers encrypt bit by bit whereas block ciphers encrypt a block of data at a time. An example of stream cipher is RC4 and example of block cipher is AES. Most of the stream ciphers and block ciphers performs simple substitution and permutation. Fiestal cipher structures are used to build block cipher.

Plaintext : It is the original message given as input to a cryptographic algorithm

Encryption algorithm: Algorithm that takes input as Plaintext and performs various operations on it.

Secret Key: It is also an input to the encryption algorithm. It is independent of the plaintext and the algorithm. Based on the key used, the output of the algorithm differs. The secret key should be kept secret. The substitutions and transformations performed by the algorithm depend on the key

Cipher text: The scrambled message produced as output is called the cipher text. The output depends on the plaintext and the key. Different keys and different plaintext produces different cipher texts. Cipher text is a random stream of data that is unintelligible.

Decryption algorithm: It takes the cipher text and the secret key as input and produces the original plaintext.

Symmetric Cryptosystem:

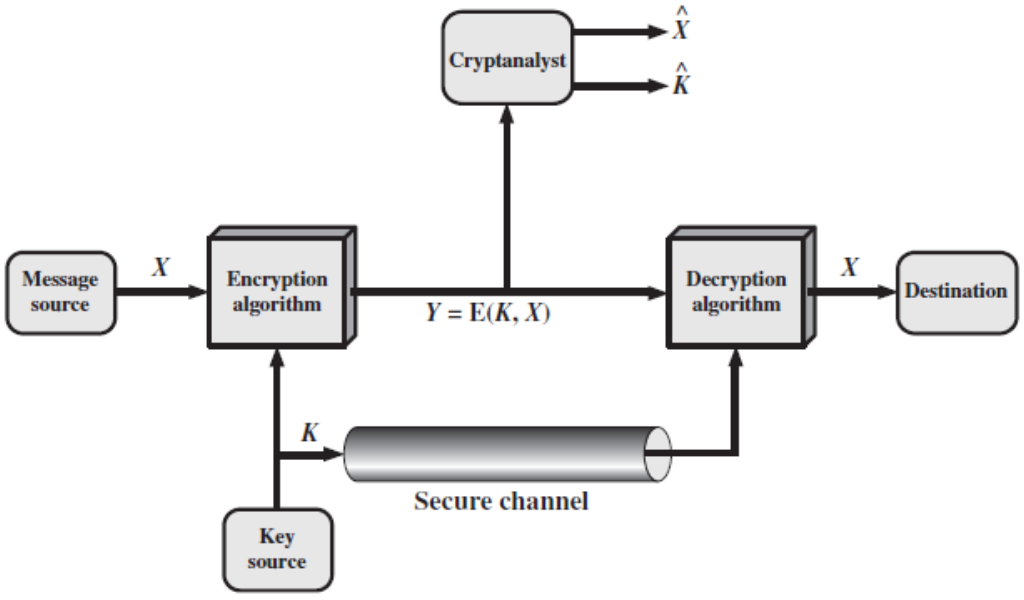


Figure 2.13:Model of symmetric cryptosystem (Stallings 2006)

The algorithm used for cryptosystem is not key secret, whereas the key is kept as secret. In the Figure 2.13 (Stallings 2006)the message X is the plaintext, which is passed through encryption algorithm. It takes input the key and the plain text. It is then transformed to $Y = E(K, X)$ where Y is obtained through a series of substitution and permutation. The ciphertext, Y is then passed through an insecure channel to the receiving end. At the receiving end, the cipher text and the same key is passed to the decryption algorithm which converts back the cipher text to the plaintext X. In symmetric encryption, the key shared by the sender and receiver is same and should be communicated through a secure channel. Hence the principle security problem in symmetric encryption is maintaining the secrecy of the key. A source produces a message in plaintext, $X = [X1, X2, \ldots\ldots XM]$. The M elements of X are letters in some finite alphabet.

For encryption, a key $K = [K1, K2, \ldots\ldots KJ]$ is generated. This Key must be shared to the receiving end through a secure channel. With the message X and the encryption key K as input, the encryption algorithm

forms the ciphertext $Y = [Y1, Y2, \ldots\ldots\ldots YN]$. This can be written as

$$Y = E(K, X)$$

Y is produced by using encryption algorithm E as a function of the plaintext X, with the specific function determined by the value of the key K. The intended receiver, in possession of the key, is able to invert the

Transformation as follows:

$$X = D(K, Y)$$

An attacker will try to recover the plain text or the Key or both. The encryption algorithm (E) and the decryption algorithm (D) are assumed to be public, which means the attacker have knowledge about it. If the attacker wants only a particular message, he/she will try to recover the plaintext

without trying to break the key. However, if the attacker is interested in all future messages, the attacker will try to recover the key.(Stallings 2006).

A cryptographic scheme is basically characterized along 3 independent dimensions.

1. Based on the type of transforming the plaintext to the cipher text, it is classifies as substitution cipher or transposition cipher. Substitution cipher is where the plain text characters or bit is replaced with another character or bit. Frequency analysis techniques are commonly used to attack such type of ciphers. Transposition ciphers involves, rearranging the letters or bits in the plaintext to form the cipher text. The fundamental requirement for both the method is that no information should be lost.
2. The number of keys used. Based on the number of keys used, it can be classified as symmetric or asymmetric encryption. Symmetric encryption uses single key for both encryption and decryption, whereas asymmetric encryption uses one key for encryption whereas another key for decryption. In this one of the key will be public, whereas the other key is kept secret. Asymmetric encryption is also referred to as public key encryption.
3. The way in which the plaintext is processed. Based on the way the plaintext is processed, it can be classified into stream cipher and block cipher. In stream ciphers, plaintext is processed bit by bit wheras, in block ciphers plaintext is processes block by block. Usually a block of 128 bit is used.

Asymmetric Cryptosystem:

It is also known as Public Key Cryptography, which is mainly used in applications to provide authentication, confidentiality and Key sharing. It uses two keys for encryption and decryption. Both the keys can be applied in any order. Various asymmetric cryptography schemes are in use, such as RSA, DSA etc.

Asymmetric Keys

They are two related keys, a public key and a private key, that are used to perform complementary operations, such as signature generation and signature verification or encryption and decryption.

Public Key Certificate

It is a digital certificate issued to an owner of a public key. The certificate is issued by a certificate authority indicating that that the one who possess this certificate is the owner of a particular public key and it corresponding private key is with the owner. This is basically used to provide authentication of public key.

Public Key (Asymmetric) Cryptographic Algorithm

A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that deriving the private key from the public key is computationally infeasible.

Public Key Infrastructure (PKI)

A set of policies, processes, server platforms, software and workstations used for the purpose of administering certificates and public-private key pairs, including the ability to issue, maintain, and revoke public key certificates.

Asymmetric algorithms rely on using one key for encryption and another key for decryption. One of the key can be public whereas other key is kept secret. It is computationally infeasible to determine the private key with the knowledge of encryption algorithm or public key. Both the private and public keys can be applied in either way and both keys are related to each other. An example of Asymmetric algorithm is RSA.

A public-key encryption scheme has six ingredients

The essential steps in Asymmetric encryption are as follows.

1. Each user generate the public and private key pair.

2. They share the public keys in a public register and keeps the private key as secret.

3. If User A want to send a confidential message to User B, User A will encrypt the message with the Public Key of User B

4. When User B receives the encrypted message from User A, User B will decrypt the message using the Private Key of User B

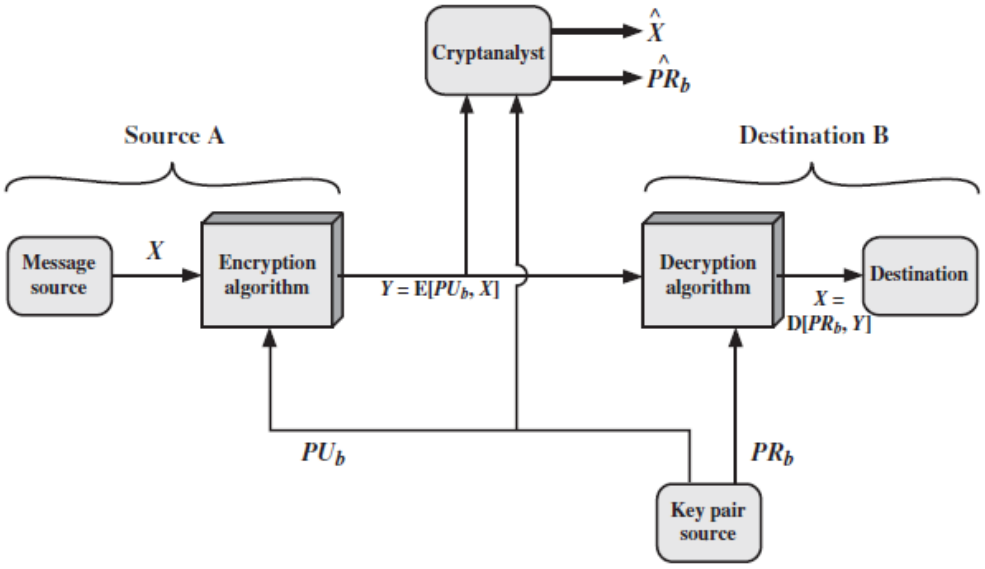


Figure 2.14:Public Key cryptosystem : Secrecy (Stallings 2006)

Asymmetric encryption can provide secrecy and authentication. To provide secrecy, User A encrypts the message using the public key of User B.

$$Y = E[PUb, X]$$

This message is then shared through an insecure channel. Upon receiving the ciphertext, User B decrypts the message using his private key.

$$X = D[PRb, Y]$$

An attacker, will have access to the public key and the Cipher text, and it is assumed that the attacker have knowledge about the encryption and decryption algorithm. Hence he/she will try to

find the plaintext or the private key. If attacker is interested only in a single message, then he/she will try to get the plain text, whereas, of the attacker is trying to get all the future messages, then he/she will try to break the private key (Stallings 2006)

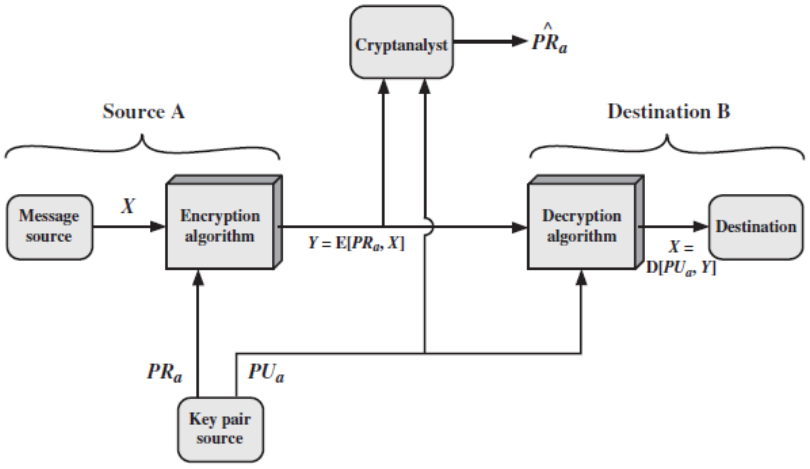


Figure 2.15:Public key cryptosystem: Authentication (Stallings 2006)

To provide authentication, the message X is encrypted with the private key of the source A.

$$Y = E[PRa, X]$$

The destination B decrypt the message using the public key of A.

$$X = D[PUa, Y]$$

In this case, authentication is achieved by encrypting the message with the private key of the sender and then the receiver will verify the message by decrypting the cipher text with the public key of the sender. In this way authentication is achieve. This also serves as the basis for digital signature, which is explained in the next section.

2.4.2 **Digital Signatures and Hash Functions**

Digital signature

A signature scheme is “secure” if no party can force a valid signature without having the secret key. In, (Katz 2007) the formal definition of a digital scheme is as follows

Definition 1. A signature scheme is a tuple of three probabilistic polynomial-time algorithms (Gen, Sign, Verify) satisfying the following:

1. The key-generation algorithm Gen takes as input a security parameter 1κ and outputs a pair of keys (vk, sk). These are called the public key and the private key, respectively. We assume for convenience that vk and sk each have length at least κ, and that κ can be determined from vk, sk.
2. The signing algorithm Sign takes as input a private key sk and a message $m \in \{0,1\}*$. It outputs a signature σ, denoted as $\sigma \leftarrow Signsk(m)$.
3. The verification algorithm $Verify$ takes as input a public key vk a message m, and a signature σ. It outputs a bit b, with $b = 1$ meaning valid and $b = 0$ meaning invalid. We write this as

   $b = Verifyvk(\sigma, m)$.
4. It is required that for every κ, every vk, sk output by $Gen(1\kappa)$, and every $m \in \{0,1\}*$, it holds that

   $$Verifyvk(m, Signsk(m)) = 1$$

   If$(Gen, Sign, Verify)$ is such that for every (vk, sk) output by $Gen(1\kappa)$, algorithm $Sign\ sk$ is only defined for messages $m \in \{0,1\}l\kappa$ (and $Verifyvk$ outputs 0 for $m \notin \{0,1\}\ l\kappa$ then we say that (Gen, Sign, Verify) is a signature scheme for messages of length l(n)).

   A signature scheme is used as follows: a singer S runs Gen(1^{κ}) to get the public and private keys (vk, sk). Then publicly announce the public key vk belonging to the signer S. If S needs to send a message m to other party, it computes $\sigma \leftarrow Signsk(m)$ and gives

(m,σ) to the receiver. The receiver knowing the public key vk can verify the authenticity ofm by checking whether $Verifyvk(\sigma, m) = 1$.

Intuitively, we want to ensure that no malicious party can force a valid signature and message pair without knowing the corresponding secret key. In other words, only the owner of the key pair can produce the valid signatures.

Hash Functions

Hash functions are used for providing integrity of the data. Suppose User A send a message 'm' to User B, an attacker can modify the message to m' and send it to User B. Hence to prevent such attacks and provide integrity to the message Hash functions are used. User A will calculate the Hash of the message H(m) and append it along with the message. The message along with the appended hash value are send to User B. User B, when received the message will calculate the hash of the message and compare the calculated hash with the appended hash. If both the hash values are equal, then message is not modified.

A hash function H accepts a variable length message and create a fixed length hash value.

$$h = H(m)$$

A "good" hash function has the property that the results of applying the function to a large set of inputs will produce outputs that are evenly distributed and apparently random. In general terms, the principal object of a hash function is data integrity. A change to any bit or bits in M results, with high probability, in a change to the hash code. It is computationally infeasible for an adversary to find the message from the hash function. Hence Hash functions have the one-way property.

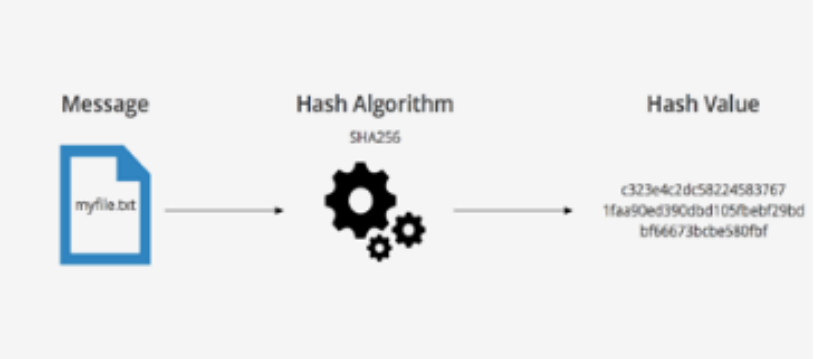


Figure 2.16: An example of message to hash value

2.4.3 **Identity Based Encryption**

In 1984, Shamir (Shamir 1985) proposed a concept of identity-based cryptography. In this type of cryptography users identifier information can be used as public key instead of a long computer generated key. This avoids the use of a digital certificate for public key authentication. The public key can be users email address, phone number etc. This significantly reduces the system complexity and cost for establishing and managing the Public key authentication process known as Public Key Infrastructure (PKI). IBE was an open problem until 2001, when Boneh and Franklin (Boneh & Franklin 2003) and Cocks (Cocks 2001).Although Shamir easily constructed an identity-based signature (IBS) scheme using the existing RSA function, he was unable to construct an identity-based encryption (IBE) scheme.

Basic Concepts of Identity Based Encryption

In (Shamir 1985) Shamir asked for a public key encryption scheme where the public key can be an arbitrary string. There should be four algorithms in such scheme: (1) setup: generates the global system parameters and a master key, (2) extract: uses the master key

to generate the private key corresponding to the arbitrary public key string (3) encrypt: encrypts messages using the public key ID, and (4) decrypt: decrypts messages using the corresponding private key.

The original motivation for IBE was to simplify certificate management in e-mail systems. If Alice want to sends mail to Bob at bob@company.com she simply encrypts her message using the public key string "bob@company.com". Alice does not have to obtain the public key certificate of Bob. When Bob receives the encrypted mail he contacts Private Key Generator (PKG) which is a third party. Bob has to authenticates himself to the PKG in the same way he would authenticate himself to a CA and PKG will provide his Public Key. Bob can then read his e-mail. In this system, Alice can send encrypted mail to Bob even if Bob has not yet setup his public key certificate.

Practical Difficulties with Public key Encryption

Originally, public key was designed to be stored in a public directory like a universal phone book, However practical implementation of this idea still doesn't exist. Public keys are distributed as digital certificates stored in LDAP directory where the complexity and difficulty of using the protocol is well-documented.

Key recovery is another problem that makes deployment of public-key technology difficult. When the recipient of an encrypted message is not available, private keys have to be recovered to decrypt the message. This could be in such case where, the recipient is no more working for the company, or might be on vacation. Also there could be regulations to store unencrypted copy of messages or to scan the message for Viruses, spam or phishing attack. In any of these cases private key is required. The only way to achieve this is to securely archive copies of all of the private keys. This requires additional

infrastructure cost for creating backups. Figure 2.17 provides a traditional PKI system. Bob should initially register with the Certificate Authority by providing its public key and credentials for authentication. The CA will verify it and sign the public key of Bob and store the digital certificate in the Certificate Server. If Alice wants to send a confidential message to Bob, Alice will first have to lookup the certificate server for the digital certificate of Bob and then use the public key to encrypt the message to Bob. Bob after receiving the message will decrypt with his private key. Bobs private key will be stored in a recovery server.

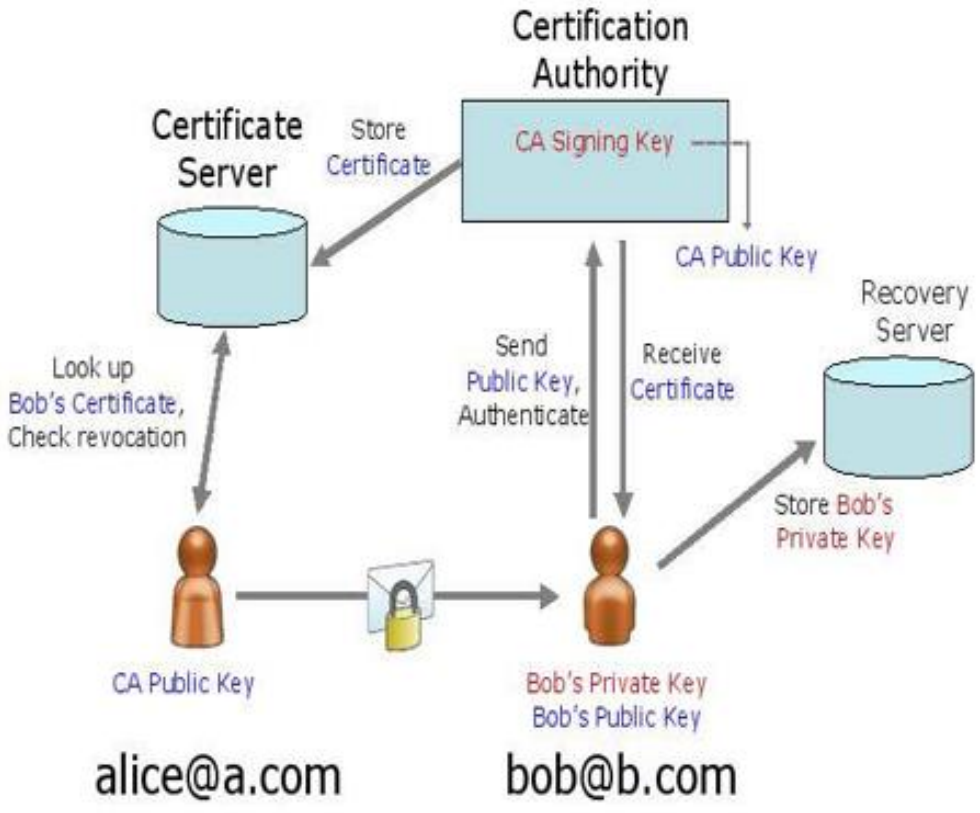


Figure 2.17: Traditional PKI

If identity Based Encryption is used, it replaces the role of CA with a PKG and in addition it eliminates the need for a certificate server and a Key recovery Server. This is given in Figure 2.18.

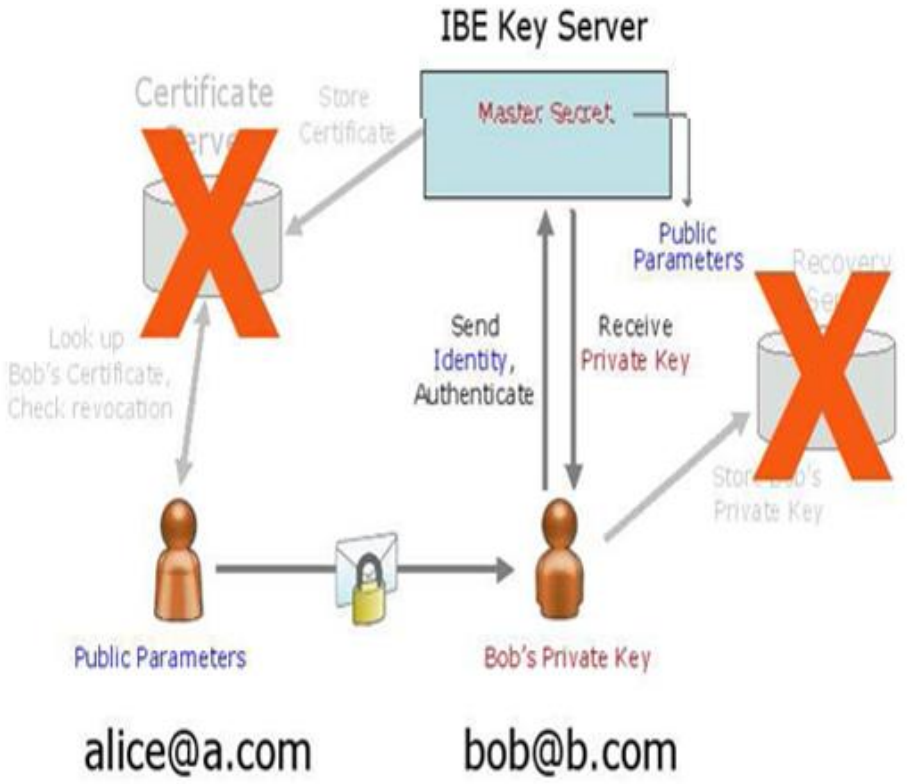


Figure 2.18:The Identity Based encryption System

In Figure 2.19 the operations of IBE is described. When Alice want to send a message to Bob, She first calculates the public key of Bob. This could be the email id or phone number of Bob. She encrypt the message with this public key. When Bob receives the message, he contact the PKG for the private key. Bob authenticates himself to the PKG

and securely receives the private key. Using this private key, Bob can decrypt the encrypted message.

Summing up, an IBE scheme is described using the following steps.

1. Setup: The PKG creates a master key and public key pair for itself. The master key is the private key of PKG.

2. Private Key Extraction: Bob, the receiver of the message will authenticates himself to the PKG to obtains a private key associated with his identity ID*Bob*.

3. Encryption: The sender Alice encrypts her plaintext message *M* and obtains a ciphertext *C* using Bob's identity ID*Bob* and the PKG's public key.

4. Decryption: After receiving the ciphertext *C* from Alice, Bob decrypts it using his private key *to* recover the plaintext *M*.

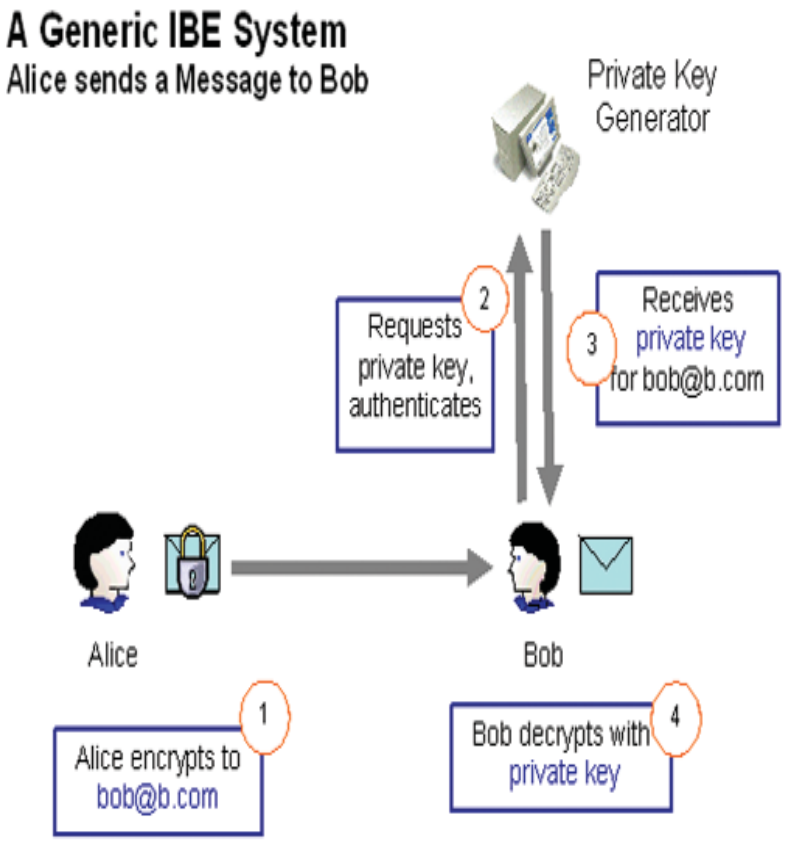


Figure 2.19: Generic IBE system with PKG

Identity Based Encryption Scheme based on Quadratic Residues

Cocks' identity-based cryptosystem (Cocks 2001) uses quadratic residues modulo a large composite integer. The security of this scheme is related to the difficulty of solving the quadratic residuosity problem. Cocks IBE scheme is constructed using a variant of integer factorization problem. The scheme invented by Cocks relies on the *quadratic residuosity* problem for its security, a mathematical problem that relates to determining whether or not a particular integer has a square root module or a large composite integer. However, the scheme is inefficient in that a plain text message is encrypted bit-by-bit and hence the length of the output cipher text becomes long.

Hierarchical IBE scheme. In an IBE system heavy workload is imposed on a single PKG. To Avoid this drawback, Horwitz and Lynn (Horwitz & Lynn 2002)suggested incorporating a hierarchy of PKG, where the PKGs have to compute private keys only to the entities immediately below them in the hierarchy. In this hierarchical IBE scheme, the users are no longer identified by a single identity, but by a tuple of identities which contains the identity of each of their ancestors in the hierarchy.

Implementation and applications of IBE

'MIRACL' is a cryptographic library that that includes Boneh and Franklin's IBE scheme. It contains fast experimental implementations of IBE. By the group of people including Boneh and Franklin (Boneh & Franklin 2003), the IBE scheme was designed, which they call "Stanford IBE system", was implemented. Both of Stanford and Shamus's library were developed using C/C++.

Timings include all number theoretic components of encrypt/decrypt processing. Tate Pairing is the most time-consuming component in the calculation . The discrete logarithm bit-length security of a pairing-based system is a function of the product of the *security multiplier k* and the bit length of the base field.

The notable real world applications of IBE include the IBE email system developed by Voltage Security, which provides secure email, secure file,securing networks and securing key management server. Also, researchers from Hewlett Packard Lab in Bristol,UK developed a health care information system that facilitates an IBE capability.

Advantages of Using IBE

There is no pre-enrollment required for IBE. Using IBE, it is easy to communicate with a person who has not already enrolled in the system. If Alice want to send an encrypted message to Bob, Alice just have to calculate the public key of Bob then use that key to encrypt a message. When Bob receive the message, he will contact the PKG to obtaine the private key. Since we can calculate a key for any recipient, there is no pre-enrollment required for users of an IBE system. Since keys are calculated, there is no requirement for looking up public keys, and one of the big practical difficulties that has been associated with public-key cryptography is no longer an issue.

IBE is a public-key technology, so it has all the benefits that other public key technologies have, but it also brings other benefits, since IBE keys are calculated instead of being randomly generated. Techniques like e-mail answerback can be used in situations like this to authenticate users, allowing us to easily create *ad hoc* groups that can communicate securely, but with minimal set-up cost, making it ideal for use in situations where low-overhead dynamic groups need to be created.

2.5 **Chapter Conclusion**

In this Chapter, we provided a brief literature review on all the topics we used in our thesis. We started with defining internet of things and proceeding towards identifying the key applications of IoT and the different architecture standards for IoT. We also provide how the communications happen in IoT and security in IoT. We then proceed to defining blockchain by starting with how blockchain works. In this section, we briefly describe the working of Bitcoin, which is the first successful blockchain application. We then introduce some attacks on blockchain and smart contracts and ethereum virtual machine. We then proceed to linking the relationship between blockchain and IoT. We then proceed towards defining basic term in cryptography including

Symmetric and asymmetric encryptions, Digital signatures and hash functions and Identity based encryption. We surveyed various literatures to identify how the blockchain can be utilized for IoT. While analyzing the requirements for creating a blockchain for IoT, we identified some gaps in existing literatures. This includes, what are the different types of IoT devices and what are the challenges in integrating blockchain into it, what are the different types of IoT and blockchain applications, what are the different ways of storing data in the IoT blockchain, what are the various security requirements, and what parameters of blockchain should be used for IoT. We identified these requirements and created a decision tree model, which is explained in Chapter 3.

Chapter 3 : KEY COMPONENTS IN CREATING BLOCKCHAIN FOR IOT

In this Chapter, we provide the key components that we identified that should be considered while creating blockchain for IoT. We identified from various literature, that there is no decision tree that includes all the components for creating blockchain for IoT. We identify this as a gap in the existing literature.

3.1 Introduction

While the blockchain trend is spreading to use-cases other than cryptocurrencies, it is still not clear on its implementation for IoT. This is mainly due to the limited constraints of IoT and the ledger based design of blockchain protocol. IoT can benefit massively if blockchain features can be balanced to fit for IoT. This could help in solving many current problems in IoT. Implementing blockchain for IoT impose variety of challenges. We identified the key components along with design considerations and challenges that should be considered while creating a blockchain architecture for IoT. We create a decision tree based on the identified components. These components includes, Identifying the type of IoT device, Identifying the type of application, Identifying data and storage requirement and Identifying security requirements, and Identifying blockchain parameters. We list these gaps that will help in providing insights into creating a secure framework of blockchain for IoT.

3.2 Related Work

Majority of the work on IoT blockchain are that proposes architecture, consensus and security. We compared some of the existing architecture under Section 3.6. Performance and scalability are the main problems in IoT blockchain (Lo et al. 2019). This is due to the large volume of data generated by the devices. Several papers identified potential challenges and technologies in IoT blockchain

(Yang et al. 2017) (Dedeoglu et al. 2019). Authors in (Dedeoglu et al. 2019) identified key challenges and potential applications for IoT blockchain. They provided a detailed description of various challenges, types of blockchain and consensus used in blockchain. A detailed description of variety of Byzantines Fault Tolerance (BFT) techniques with its negative and positives aspects are summarized in this Chapter. A variety of literature use variant of Byzantines fault Tolerance consensus for IoT blockchain (Sousa et al. 2018). Proof of Work based consensus is not widely used in IoT blockchain due to the resource constrained nature of IoT devices. Various use-cases of blockchain beyond cryptocurrencies are provided in (Conoscenti, Vetro & De Martin 2017). They also provide a detailed list of type of data that are stored in blockchain and the implementation differences in IoT blockchain and cryptocurrencies. A detailed description of various blockchain based consensus methods, platforms and implementations for IoT are surveyed in (Salimitari & Chatterjee 2018) . In (Pahl, El Ioini & Helmer 2018) authors provide a decision framework to choose when to use blockchain and what platform to choose while creating blockchain for IoT.

3.3 **Identify the Type of IoT device**

The first step for creating a blockchain architecture for IoT is to identify the type of IoT devices. This is provided in Figure 3.1. Some devices have only the sensor functionalities with computations specifically to share the sensor data to a database. Whereas other devices will have sensor functionalities along with computation capabilities to encrypt or process data. In the first case, devices are capable of performing blockchain computations; hence blockchain of edge nodes or gateways can be created. A full node can carry the full copy of blockchain and can perform the computation required in blockchain whereas a light node does not hold the blockchain data instead, refer to a full node.

There are three kinds of data communication mode in IoTs. Event driven, periodic (time-driven) and on-demand (query-driven) reporting(Al-Karaki & Kamal 2004). In the event-driven mode, sensors communicate to gateways or sink when a particular event happens. In periodic reporting mode, sensors sense data in pre-determined time and periodically send the data to the gateways. However, in the on-demand mode, gateways can query the data when required.

Internet of things (IoT) is a concept where physical devices communicate with each other through internet using IP Address without the need for human interference. Internet of Everything(IoE) is about things, people, Process and data. Internet of Nano Things (IoNT) is a communication network of Nano devices, which uses traditional communicational with classical network protocols. Internet of Mobile Network (IoMT) is using sensors within the mobile devices. Internet of Mission Critical Things (IoMCT) are devices that are used in critical areas like military applications. These devices should be of high precision and accurate. (Srinivasan et al. 2019).

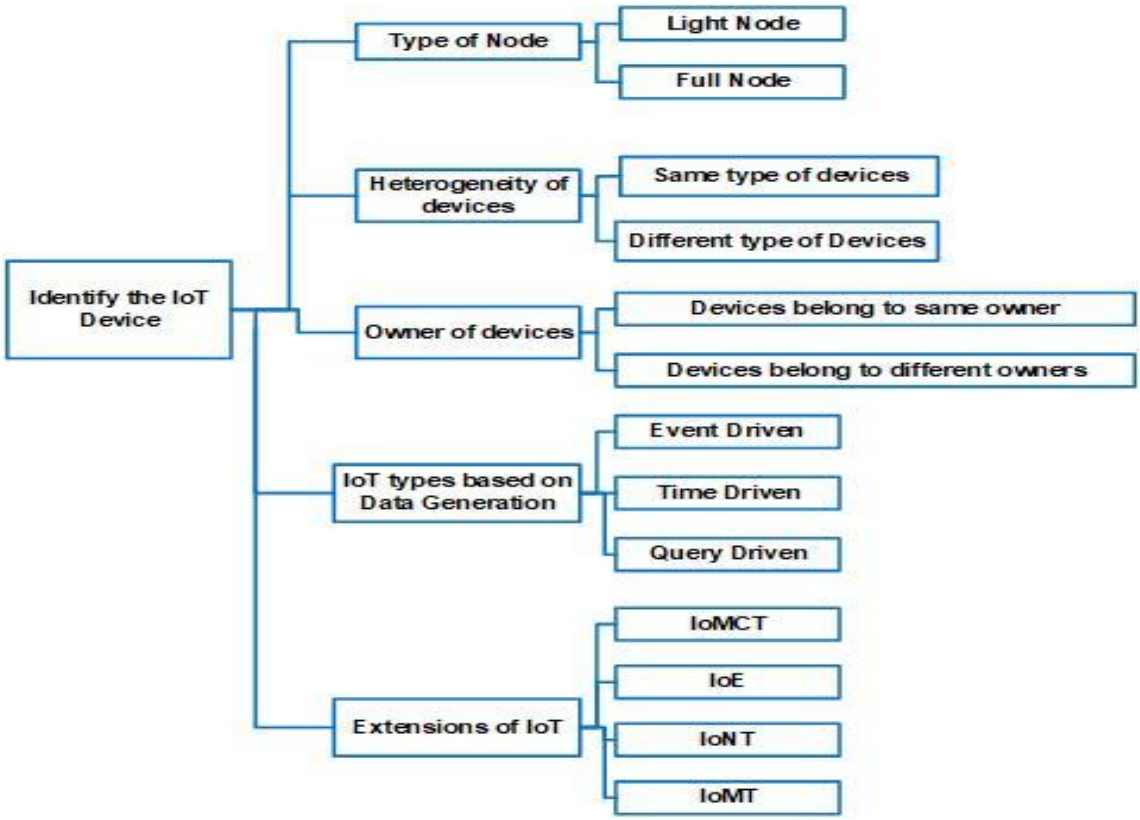


Figure 3.1: Identify the IoT device type

As IoT devices are different in their design and architecture, interoperability within these devices under a common blockchain will be a challenging issue. Bringing different types of devices under the same blockchain can be a trivial task. This issue can be addressed if organizations like ISO can come up with standards for IoT manufacturing and blockchain implementation. Devices owned by different entities or owners will need standardized policies on the data that could be accessed and stored. The blockchain should be linked with the regulatory authorities to adapt consistent regulations. In order to provide efficiency, certain security and privacy controls should be in place such as risk management process. In addition, there should be rules to govern the interactions between participants.

3.4 Identify the Type of Application

While building applications based on blockchain, we need to systematically consider the features and configurations that are required and assess the impact and quality of these with IoT. Requirements to identify the application types are provided in Figure 3.2.

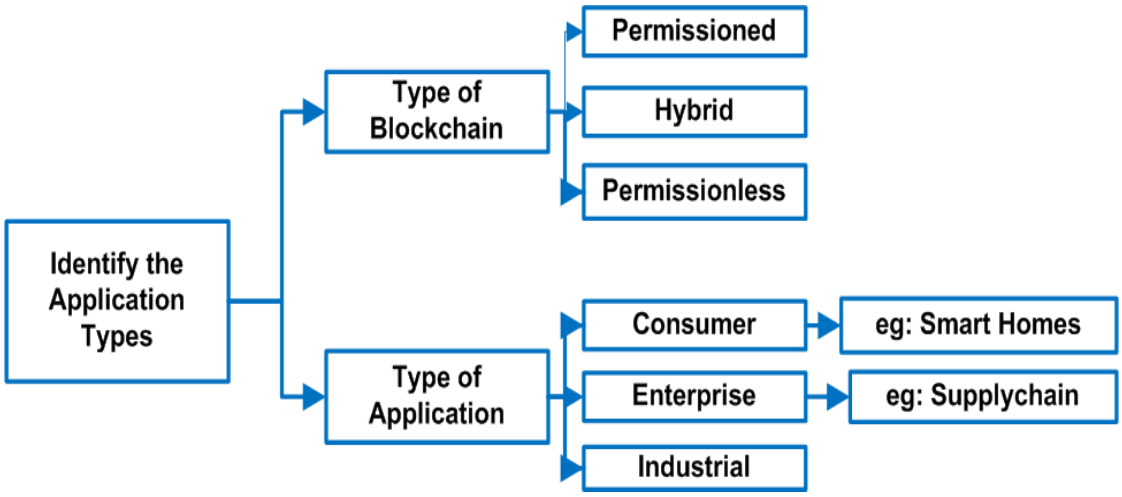


Figure 3.2:Identify the type of Applications

Based on the type of implementation, blockchain can be classified into permissionless, permissioned or Hybrid blockchain (Vukolić 2017). In case of permissionless blockchain, anyone can join the network and can participate in consensus procedure. It has open read/write access to the database. Bitcoin is an example of permissionless blockchain. Whereas in case of permissioned blockchain, only selected participants can be part of consensus procedure. IBM's Hyperledger blockchain is an example of permissioned blockchain. Hybrid blockchain is a combination of permissioned and permissionless blockchain. It will have a public facing network for the customers and an internal private blockchain network. In a permissionless network, all the full nodes will be running all the applications. In case of IoT this will affect the performance of the IoT device due to the resource constrained nature of these devices. In permissioned blockchain every node will only

need to perform the computations required for a given application. A comparison of permissioned and permissionless blockchain is provided in Table 3.1

Table 3.1:Permissioned and permissionless blockchain

Permissionless	Permissioned
No restriction on who can perform transactions	Restriction on who can perform transactions
No restrictions on adding as a node	Restrictions on adding as a node
No Restriction to participate in consensus mechanism	Restriction to participate in consensus mechanism
Low Performance when compared with Permissioned	High Scalability and faster
Less cost effective	Cost effective
More chance of spreading malwares	Security depends on the access control system implemented
Fully decentralized	Not fully decentralized

Depending on the type of application, IoT devices can be classified into consumer, enterprise or industrial IoT. Consumer IoT are solutions made for individual non-commercial usage. IoT devices in smart home are a consumer based IoT. Solutions created for large commercial buildings or in enterprise are classified under enterprise IoT. Example is IoT used in supply chain industry, IoT in street light etc. Industrial IoT are devices used in the factory or farm to monitor fuel levels, trigger when fuel is empty etc. A selection of the blockchain use-cases for IoT available in the literature is provided in Table 3.2.

Table 3.2 Use-cases of Blockchain for IoT

Use-cases example	Reference
Home Automation	(Dorri, Ali and Kanhere, Salil S and Jurdak 2017)
Blockchain based sharing services towards Smart Cities	(Sun, Yan & Zhang 2016)

Blockchain ready : manufacturing supply chain using Distributed ledger	(Abeyratne, Saveen A. 2016)
Pharma Supply chain	(Bocek et al. 2017)
Supply chain traceability System for Food safety	(Tian 2018)
Access Control Framework	(Ouaddah, Abou Elkalam & Ait Ouahman 2016)
Logistics and Supply Chain	(Abeyratne, Saveen A. 2016) (Korpela, Kari, Jukka Hallikas 2017)
Energy Management	(Imbault et al. 2017)
Data storage management	(Zyskind, Nathan & Pentland 2015)
Trade of items and data	(Zhang, Yu 2015) (Wörner, Dominic 2014)
E-business model for smart property management	(Zhang, Yu 2015)
Power generation and distribution	(Lo3Energy 2019)
Modum framework for supply chain	(Modum 2018)

In this thesis, we will present architecture for event driven edge based IoT with Traffic speed radar as use-case example. A public blockchain requires high resources when compared to permissioned blockchains. This is mainly due to the cost required for performing consensus. For an enterprise application where each node is owned by different organizations, a permissioned blockchain is used (Thakkar, Nathan & Viswanathan 2018). Speed radars are enterprise applications and hence we used permissioned blockchain for the proposed architecture.

3.5 **Identify Data and Storage Requirements**

Identifying what data should be stored in the blockchain is a major component while designing blockchain. Figure 3.3 provides an overview of this requirement. These can be IoT sensor data, device identity, public key, or reference to data stored in cloud.

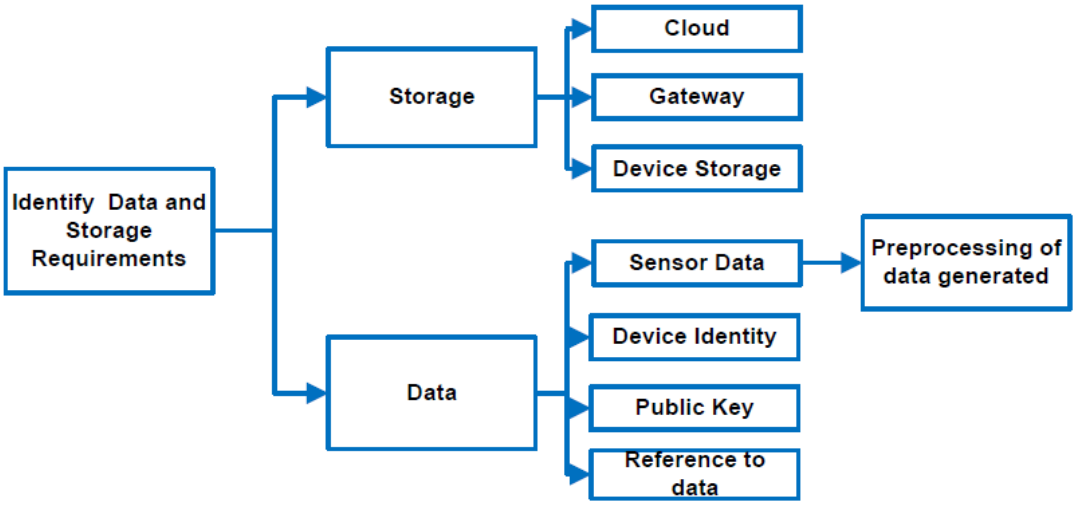


Figure 3.3:Identify Data and Storage Requirements

Each node in the blockchain maintains a distributed ledger, which is basically a database that requires storage space. To add a new device into the block, the device should download all the transactions from the first block. Hence, in such architectures, IoT devices should have enough storage capacity to maintain a copy of the transactions. IoT sensors generate vast amount of data. Replicating this data to many different nodes require high storage capacity for the nodes and high-speed data transfer facilities. One of the major challenges would be on how to avoid the large amount of unwanted data generated by sensors without being replicated to other nodes. AI techniques should be used to parse the raw data and remove the unwanted data. Blockchain, on the other hand, usually processes limited number of transactions per second; therefore, this may create a gap between the data being generated and the capability of processing the data.

Every transaction in blockchain is signed using a private key, which should be kept securely. One of the main challenges in designing blockchain for IoT would be finding solution on how to store the private keys securely within IoT. Most of the IoT devices reside in public places and; hence; it could be compromised easily. In bitcoin, private keys are stored securely in the owner's bitcoin

wallet. If an owner loses his bitcoin wallet, he will lose all the bitcoins associated with that wallet. Majority of the attack on bitcoin are due to stolen wallet. Hence, private keys within the IoT devices should be stored securely. Hardware embedded secure keys should be used in such a case.

3.6 **Identify Security Requirements**

Blockchain is capable of solving the security challenges in IoT. The traditional bitcoin protocol provides integrity, authentication and pseudo-anonymity. However, in case of IoT the confidentiality of the data generated by sensors depends on the sensitivity of data. Providing confidentiality for the generated data is a trivial task. Highly sensitive data generated by IoT device needs to be protected from unauthorized people. The distributed nature of blockchain stores all transaction in all the participating nodes. Controlling access to the data within devices should also be considered.

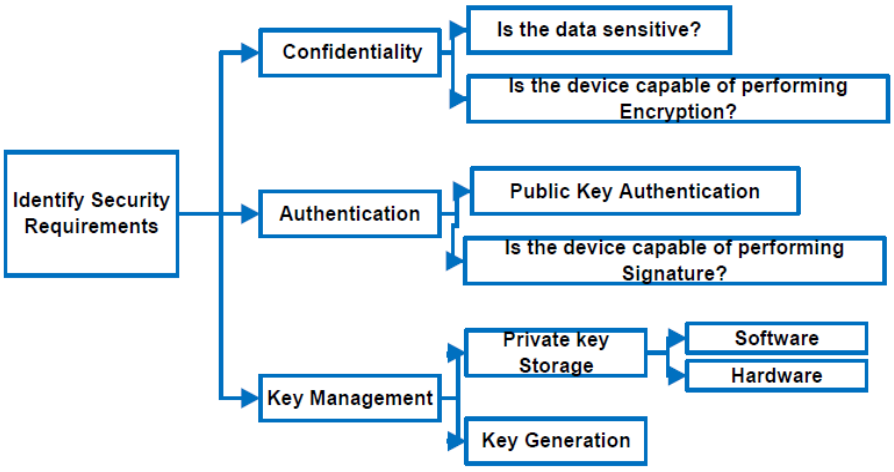


Figure 3.4:Identify Security Requirement.

Figure 3.4 identifies the security requirement while creating IoT blockchain. Even though blockchain technology reduces the potential risks in traditional centralized architecture, still

security breaches are unavoidable. If a user's private key is compromised, the attacker can perform transactions on user's behalf. Security is provided in blockchain through asymmetric cryptography which requires huge computational efforts to break the cipher. This is because classical computers encode information as bits. Quantum computing takes new approach in processing information which will be much faster than the classical approach. If the information can be processed much faster, then the computation efforts to break the asymmetric cryptography will be easy. Hence, with the arrival of quantum computing, asymmetric cryptography will become obsolete. Hence quantum resistant cryptography for blockchain (Yunusov et al. 2018) will be required in future.

Another issue related to security, is the reliability of the IoT data. Since blockchain can only ensure reliability of data stored within the chain; however, if this data is already malicious from IoT source, then it will remain as is within the blockchain. Finally, several IoT devices rely on existing complex and centralized security protocols that are based on PKI, such as TLS and DTLS; therefore, integrating these devices with decentralized blockchain enabled systems may raise several concerns about interoperability.

3.7 **Identify Blockchain Parameters**

It is trivial to identify the participating IoT and trusted nodes that verify the transaction.

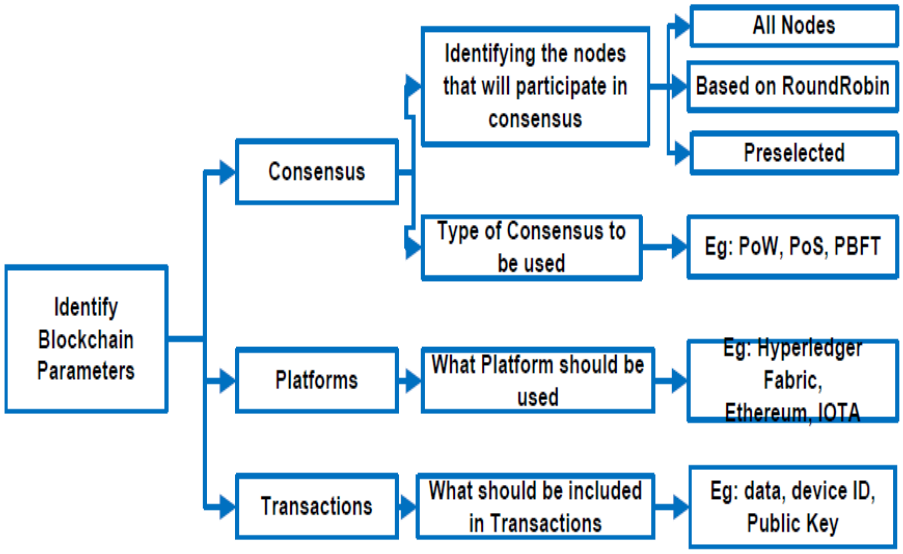


Figure 3.5:Identify Blockchain Parameters

A central authority or a group of stakeholders can decide on the nodes that will be added to the network. Such type of design will be like a hybrid blockchain that uses the basic features of blockchain and mining will be done by one or more trusted party. A variety of blockchain parameters are provided in Figure 3.5. Identifying the optimal consensus and optimal platform for implementation is an important task.

A. Consensus

Consensus in the literal terms means agreement. (Seibold S and Samman G 2018) define consensus mechanism as a method of authenticating or validating a value or transaction on a Blockchain or a distributed ledger without the need to trust or rely on a central authority. In a distributed or decentralized network, for nodes to reach to a common agreement, consensus algorithms are used. Bitcoin use proof of work based consensus which consumes high energy. Such

kind of consensus cannot be used for IoT. Blockchain platforms use a range of consensus model which are built on Byzantines Fault tolerance.

In a decentralized environment where there is no central authority to keep the ledger, this process is done through consensus mechanisms that allows secure updating of a distributed shared state. Cryptocurrencies powered by blockchain uses decentralized environment, where each ledger is distributed among all nodes in the network. The process of validating the transactions and adding it to the ledger are done by nodes in the network. But how do we trust these nodes? What if some validating nodes are malicious? They may be trying to perform double spending or trying to discard some transactions. Such type of problems can be considered as Byzantines generals Problem. A byzantine node can mislead other nodes involved in the consensus mechanism. Hence the consensus mechanism should be able to operate correctly and reach consensus even in the presence of byzantine nodes. A solution to Byzantine generals' problem is PBFT (Practical Byzantine Fault tolerance) (Castro and Liskov 1999). Permissioned blockchain platforms mainly use PBFT. In PBFT, Each party maintains an internal state. When a transaction is received, each party uses their internal state and run computations to validate a transaction. This computation will lead to party's decision about transaction. This will be shared to all other nodes in the blockchain. The final decision is based on the total decision from all parties. When enough responses are reached, a transaction is verified to be a valid transaction.

B. Blockchain Platforms for IoT

- IOTA (IOTA 2019) is a permissionless distributed ledger which uses 'Tangle' consensus. It is based on Directed Acyclic Graph (DAG), where the vertices in the DAG represent transactions and edges represent approvals. Tangle uses lightweight consensus specifically designed for IoT. It does not use block to store data, instead each transaction is a unique block. To create a

transaction, nodes initially sign the transaction and randomly choose two previous transactions to approve. When a node issues a new transaction, it must approve two previous nodes. The newly created node is then called ‘tip’. This node will remain as ‘tip’ until it is approved by a newly created node. As most of the other protocols use cryptographic algorithms that will be obsolete with quantum computing, IOTA uses quantum resistant cryptography, curl-p’ for hashing and Winternitz signature for authentication. It is fast and scalable. However the main drawback is that, there is no rule in Tangle on how to choose the two nodes for approval. All the tokens are generated in the genesis transaction and hence there is no mining for generating tokens. All the nodes contribute to provide network security by approving two other transactions. There is no rule that which two transactions should be selected for approval. For a node to issue a valid transaction, the node must solve a cryptographic puzzle similar to bitcoin. This is achieved by finding a nonce such that the hash of that nonce concatenated with some data from the approved transaction has a particular form(Popov 2018).

- Hyperledger Fabric is an open source blockchain platform developed by IBM. This is the most widely used blockchain platform which is used across different industries and use-caseuse-cases. It is used in 400 prototypes, proof of concepts, and in production distributed ledger system. Hyperledger fabric is a permissioned blockchain with pluggable consensus. It is one of the projects of Hyperledger (Hyperledger 2018)which is under the Linux Foundation. It is the first blockchain system that allows the execution of distributed applications written in standard programming. While the traditional blockchain uses order-execute-validate architecture, Hyperledger Fabric uses execute- order-validate architecture. It uses an endorsement policy that is evaluated in the validation phase. Endorsement policy is managed by designated administrators and act as a static library for transaction validation. Examples of

endorsement policies are "Three out of five" or "(A∨B) ∧ C". Custom endorsement policies can also be written. One of the disadvantages is that a central authority is managing the endorsement policy and will be implementing it in the network forcing all others to accept it. This is due to the fact that the BFT used in Hyperledger Fabric simply assumes certain parties of the network to be trustworthy. Within an organization, it assumes that all peers to be trustworthy. This reduces the transaction processing as, not all nodes need to execute the transaction. Hyperledger Fabric allows writing smart contract in a general purpose language. The framework cannot be used for large scale applications similar to public blockchain due to the network overhead caused when the number of nodes is increased (Salimitari and Chatterjee 2018).

- Ethereum (Ethereum 2019)is a project that can built a generalized technology on which all transaction based state machine concepts can be built. Ethereum enables developers to build and deploy centralized applications. Thousands of different applications can be created using Ethereum platform. Its core innovation, the Ethereum Virtual Machine (EVM) helps in creating blockchain applications easier. Developers do not have to start coding from the scratch, Instead they can use Ethereum platform and can create their own transaction formats, rules and state transition functions (Wood 2014). A comparison of these platforms is provided in Table 3.3.

  We simulated the proposed architecture using ethereum because it is an open source platform that allows to create smart contract. Another reason is (Dinh et al. 2017) have identified that Hyperledger fabric fails to scale more than 16 nodes. Hence it will not be suitable for our use-case.

Table 3.3:Comparison of Ethereum, Hyperledger fabric and IOTA

Characteristics	**Ethereum**	**Hyperledger Fabric**	**IOTA**

Description of platform	Permissioned/permissionless	Permissioned	Permissionless
Type	Open source	Open source	Not Fully Open source
Governance	Ethereum developers	IBM	IOTA Foundation
Consensus	Customizable	Pluggable consensus	Tangle
Smart Contract	YES	YES	NO
Data Confidentiality	NO	YES	NO
Advantages	Allow public and private blockchains	Allow writing smart contract in a general purpose language	use quantum resistant cryptography
Drawbacks	Does not provide confidentiality of data	Framework cannot be used for large scale applications	There is no rule in Tangle on how to choose the two nodes for approval.

3.8 **Comparison of existing Architecture for IoT Blockchain**

In this section, we compare various architectures available in the literature.

Figure 3.6 shows a generic blockchain for IoT architecture with support for several types of IoT devices as well as different infrastructures. The integration of IoT devices must involve cloud system, edge computing, gateways, and different types of IoT devices that range from simple

sensors that can only communicate through nearby gateways to devices with computational and processing capabilities.

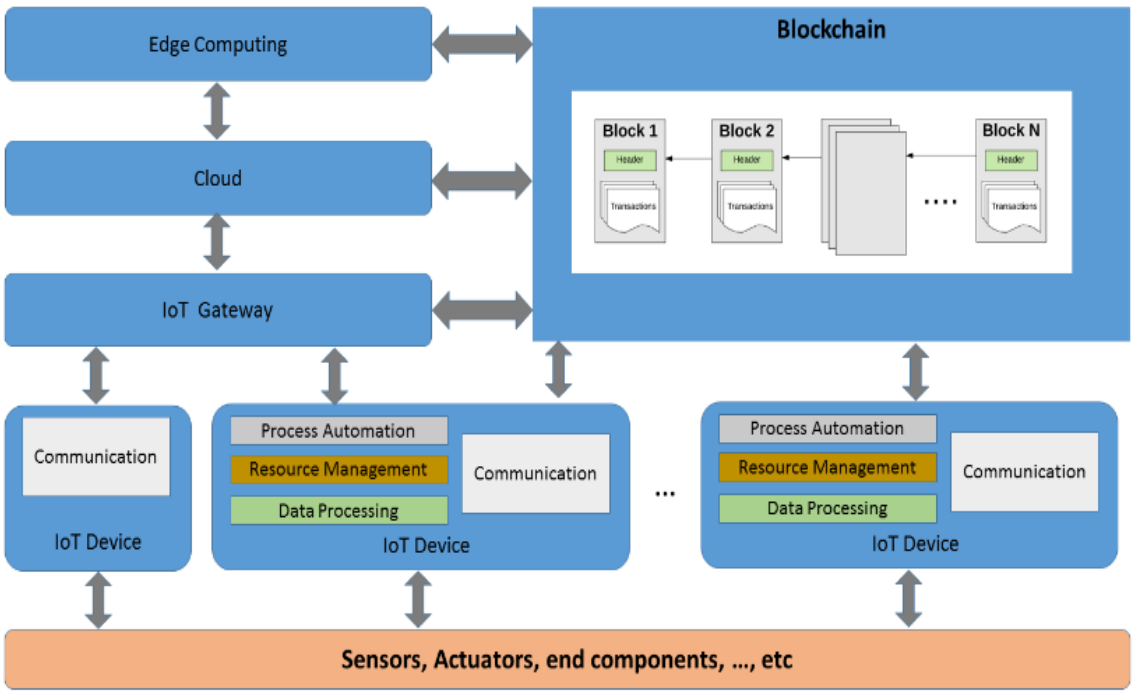


Figure 3.6:A Generic Blockchain for IoT architecture

Table 3.4 show a comparison of various architectures available in the literature, based on the type of storage used, consensus and security. IOTchain (Bao et al. 2018) is three-tier blockchain based IoT security architecture. The three layers are authentication layer (Certification layer), blockchain layer and application layer. It is designed to achieve identity, authentication, access control, privacy protection, lightweight, fault tolerance, DOS attack resilience and storage integrity(Bao et al. 2018) Hardware security model (HSM) is used to generate, store and handle key pairs and hashes are stored as Merkle tree. Any lightweight consensus can be used with IoT chain, it can be Practical Byzantine Fault-Tolerance Algorithm (PBFT) or Proof of stake (PoS). Initially, nodes register through the certification layer which provides the key pair after a valid authentication step. The keys are then added to the HSM to prevent tampering the key.

Hybrid IoT use both proof of work and BFT. Proof of work based sub blockchain are created which are then interconnected using BFT (Sagirlar et al. 2018). In Hybrid IoT, Proof of work based sub blockchain are formed which are then connected using a BFT inter-connector. They use separate centralized storage for each sub blockchain.

Blockchain based framework for edge and fog computing is proposed in (Tuli et al. 2018). Fog computing brings the network and cloud computing resources closer to the edge. Hence computations can be performed near to the IoT devices, instead of sending it to cloud datacenter (Kotb et al. 2019). Fog Bus can integrate different IoT systems to fog and cloud infrastructure. It functions as a platform-as-a-Service model where developers can build different types of IoT applications, customize the services and manage resources. A case study of health monitoring is provided in their paper. It also provides authentication and encryption techniques to protect the data.

A novel blockchain based scheme with a proxy re-encryption scheme to ensure confidentiality is proposed in (Manzoor et al. 2019). The architecture includes IoT device, miners, cloud server and data requester connected through the internet. The IoT sensors capture and transmit the data to cloud storage. This data will be encrypted and stored in the cloud. The sensor owner activates the sensor and registers them on the blockchain. Blockchain executes smart contract on the sensor transactions and provide the required key to the sensor to encrypt the data. According to the architecture the data are not stored in the blockchain whereas is it stored encrypted in a central cloud which is a centralized architecture and also single point of failure. Their architecture is efficient in terms of providing encryption layer to sensor data. In (Samaniego & Deters 2019), a blockchain system is implemented using multiple nodes including an Arduino in- order to illustrate an IoT–blockchain application.

Table 3.4:Comparison of Architecture for IoT Blockchain

Name	Architecture type	Consensus Used	Storage Used	Encryption Layer	References
IoT Chain	Three layer architecture	Any lightweight Consensus	Distributed Storage	No	(Bao et al. 2018)

Hybrid IoT	Proof of work based sub blockchain interconnected with BFT	Proof of work and BFT	A transaction pool for each sub blockchain	No	(Sagirlar et al. 2018)
Fogbus	Platform independent interface Scalable cost efficient	Proof of work	Distributed repository nodes and later backup to cloud infrastructure	YES	(Tuli et al. 2019)
Proxy re-encryption scheme	Without the involvement of trusted third party, IoT data is encrypted and stored in cloud	Ethereum Smart contract	Data stored in cloud and Address of the data stored in blockchain	YES	(Manzoor et al. 2019)
Multichain and arduino	Two layers: FOG and IoT	Round robin	Data processed in FOG	NO	(Samaniego & Deters 2019)

3.9 **Comparison of Device only and Gateway Based type of architectures through simulation**

To evaluate the performance of the framework, we conducted simulation using Cooja simulator for Contiki operating system (David & Hsu 2018). We used Z1 motes generated at random location. We simulated a network of 5, 10, 20 and 40 nodes. The nodes use IPV6 over low power wireless personal regional network (6LoWPAN) to connect to each other. In this simulation, we have not considered the

computation and storage procedures. We considered only the communication process. Assuming 72 bytes for elliptic curve signature size and 32 bytes for SHA-256 hash functions and an average transaction size of 77 bytes. We fixed the transaction size and varied the number of nodes.

We compared the average time of communication between nodes in the blockchain on an IoT-device-only architecture and gateway architecture. Based on the result from throughout, a graph was plotted. The X-axis in the graph shows the number of nodes and Y-axis shows the transactions in bytes. We collected the number of transactions in a period of 5 seconds and 10 seconds. From the results, we identified that the throughput is low while using gateways as shown in Figure 3.7 and Figure 3.8.

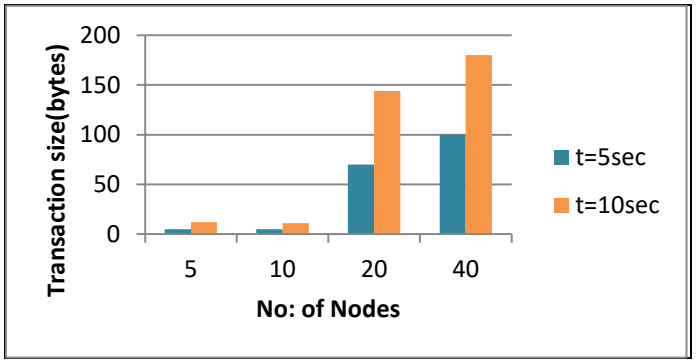


Figure 3.7:Average throughput for IoT-device-only type of architecture

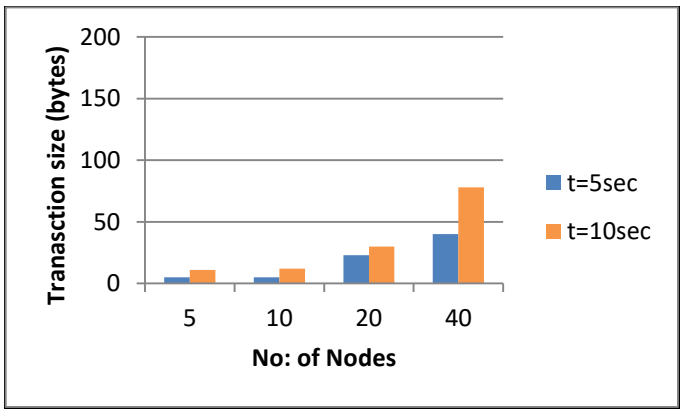


Figure 3.8:Average throughput for device with Gateway type of architecture

3.10 **Chapter Conclusions**

IoT devices participating in blockchain technologies enable a lot of challenging applications including supply chain management, health care, weather predictions, and food safety. This could be a clear replacement for the untrusted cloud technology providing security and privacy for the user's data. While creating an architecture for IoT, we identified that five components should be considered. They are IoT device types, type of applications, blockchain types and nodes, data and storage, and security. Blockchain based IoT requires energy efficient design along with security and ability to scale. IoT devices should be equipped with scalable storage solutions and computational power required to hash the transactions and verify the digital signatures. Implementation should address the challenges of both IoT and blockchain. We conducted and experiment that compared the most widely used platforms for IoT blockchain, which are Ethereum, Hyperledger Fabric and IOTA. We got some indicative results. We identified that Hyperledger Fabric is the most preferred platform due to its pluggable consensus and provides confidentiality to the data, which is most important in case of IoT due to the sensitive nature of the data. We identified that (PBFT) is the most widely used consensus for IoT blockchain due to the minimal requirement of computation than other consensus.

We also compared the architectures and frameworks for IoT Blockchain. Designing the storage and confidentiality of data are the crucial components that should be done carefully in IoT blockchain. Most of the architectures we analyzed use centralized cloud storage, which contradicts with the original objective of blockchain and can be a single point of failure. However, some of them have used distributed storage, which does not have any protection on data. This reason is that providing confidentiality for a distributed storage is not an easy task. Hence, we identified that an efficient architecture for IoT blockchain is still not available. Considering the vast advantages that blockchain can provide for IoT, we believe that blockchain will overhaul cloud computing systems. Our research delivers insight into how changes in IoT, due to blockchain technology, progress and in what directions

firms have to think while changing their business model. Hence this Chapter answers the Research Question 1.

Chapter 4 : ARCHITECTURE

In this Chapter, we propose a privacy-preserving blockchain architecture for IoT using Hierarchical Identity Based Encryption (HIBE). The proposed architecture uses edge and cloudlet computing paradigm as well as HIBE to preserve privacy. The proposed architecture fits well with event driven IoT devices. A case study of traffic speed radars is also demonstrated for the use of the proposed architecture. The performance of the proposed architecture is evaluated by conducting extensive experiments. Finally, the security of the scheme is proven through a theoretical analysis that considers the existence of a malicious adversary.

4.1 **Introduction**

Internet of Things (IoT) is becoming an integral part of human life. It is used in a wide variety of applications. For example, the use of IoT helps in reducing and detecting air pollution, detecting fraud activities, improving agriculture by minimizing crop losses and increase productivity, predicting the weather, improving the food supply chain, patient health monitoring are just a few examples. It relies on sensors that capture observation from the physical environment and helps to make decisions and predictions based on the observed data. The concept is largely based on a network of connected computers equipped with sensing devices that can collect environmental data, process it, and act according to the decision made from the collected data. The most common architectures of IoT include (1) sensing layer that relates to sensors, (2) network layer, that relates to communications within the IoT devices and gateways and (3) application layer, that relates to data processing and the interaction between service providers and users (Gu, Wang & Sun 2014). The most common communications model for IoT is device to device, device to gateway, and device to cloud (Sklavos & Ioannis 2016). Different systems and technologies have to be combined to provide safety and security in the IoT eco-system. As the collected data plays an integral part in many systems, it has always been a target of attack. Hence protecting it from unauthorized access and unauthorized modification is still a big

challenge. This is mainly due to the heterogeneity of the devices. Apart from that, a majority of the current IoT architecture uses a centralized cloud as an effective solution to store and manage data that can be accessed from anywhere. However, a centralized system has two significant drawbacks. It can be a single point of failure, and the data owner has little control over the data residing in the Cloud. Currently, most IoT infrastructures are heavily centralized. Besides, several security threats have emerged to disrupt the operations of IoT infrastructure. Some of the threats associated with centralized architecture are (1) DoS Attack- An attack on the centralized server can affect the security of the data. It will create a single point of failure, (2) Sensitive information stored in centralized servers does not have control on its accessibility, (3) Users have limited control on how their data is stored or is accessed by whom, (4) Many organizations outsource the data storage to cloud provides. With the growing number of these devices, the centralized approach will not be efficient in storing and processing of these data.

Decentralized approaches can solve many of the threats related to IoT. Blockchain is a promising technology that can secure the IoT communication system. Blockchain was developed as a solution for the double-spending problem in Bitcoin. Therefore, bitcoin is strongly supported by blockchain that creates an immutable record of network transactions. It incorporates a distributed ledger to accommodate every transaction of the network. Participants on the network are identified using their Public Key (PK). New records are added as blocks through a process called mining, and a set of nodes called miners conducts this process. The bitcoin mining consumes energy because mining is a competition where miners try to find the hash of a value that is less than a target. The miner who finds this value first will be rewarded with bitcoins. However, such kinds of mining techniques are impractical in the case of IoT, whereas its key features like anonymity, decentralization, and security can benefit IoT devices. Hence, a full-fledged architecture of IoT using blockchain is still in its infant stage.

The edge layer between the IoT devices and the cloud can be implemented in different ways depending on the devices, which acts as the intermediary layer. This implementation can be classified into 3 types, Mobile edge computing (MEC), Fog computing (FC) and Cloudlet Computing(CC) (Dolui & Datta 2017). FC utilizes Fog nodes, which includes devices like gateways and wireless router. They are used to compute and store data from the edge devices and forward it to the cloud. MEC uses intermediate nodes that works within the Radio Area Network (RAN). A key aspect of MEC is resource virtualization enabling MEC nodes to run applications in containers offering Platform-as-a-Service(PaaS). Cloudlets are dedicated devices that are located within the logical vicinity to the consumers. It supports distributed edge computing by processing content on edge devices such as base stations, radio networks, hotspots, local data centers, routers switches, and WiFi access points . It can be developed using open-source hardware and software. Satyanarayanan et al. (Satyanarayanan et al. 2006), define Cloudlet as a mobility enhanced small scale data center that is a trusted, resource-rich computer or cluster of computers that are well-connected to the internet and available for use by nearby mobile devices. Using a cloudlet simplifies meeting the peak bandwidth demand of multiple users trying to access high-resolution images or videos. A cloudlet can be viewed as a "datacenter in a box"(Satyanarayanan et al. 2006). It is self-managing and can be deployed in small business premises without requiring high setup costs.

Due to the increasing number of IoT devices, security is a paramount concern (Weber 2010). Protecting the sensor data is a trivial task because most devices are located in public places where attackers can gain access to data, falsify the data, or hijack the device itself (Deogirikar & Vidhate 2017) . Symmetric encryption like AES (Advanced Encryption Standard) are used to provide confidentiality of data. Symmetric encryption techniques use a single key for encryption and decryption. The encrypted key should be shared with the miner or validator for the decryption purpose. However, using an Asymmetric encryption has two limitations. Firstly, it needs a Public Key Infrastructure (PKI) for certificate management and verification. Secondly, most public-key encryption schemes like RSA

Encryption needs high computations, which could not be possible in a resource-constrained environment like the IoT eco-system. However, lightweight encryption schemes such as elliptic curve cryptography are proposed to use in such an environment (Malan, Welsh & Smith 2004). This facilitated the development of numerous public-key cryptography for IoT devices.

The need for lightweight cryptography for resource-constrained devices has gained wide attention from the research community. Attribute-based encryption (Goyal et al. 2006) is identified as a potential technology for providing data confidentiality in distributed systems. Identity-Based Encryption (IBE)(Boneh & Franklin 2003) is lightweight, supports massive users, simple key management, and flexible key applications. It provides identity authentication, digital signatures, and privacy protection for electronic contracts (Zhu & Fan 2019). Homomorphic encryption (Gentry et al. 2013)is a promising field that allows computations on encrypted data. However, Homomorphic encryption is still in its developing stage. IBE is proved to reduce computational complexity when compared with other Asymmetric encryption. A major advantage of IBE when compared with other public key encryption scheme is that, in IBE the public key can be user's identity instead of a long string. In Public Key cryptosystems, public key is generated using the key generation algorithm, whereas in IBE it could be email address, phone number etc. It is the Private Key Generator (PKG) that generates the secret based on the public key. Security mechanisms using IBE have shown lesser overhead than public-key cryptography due to the reduced key size. Although being asymmetric encryption, IBE based on bilinear pairing reduces the computational complexity which can be adopted into resource-constrained IoTs. IBE has the disadvantage that a root PKG computes the secret key for the user using its master secret. In a large network, a single PKG can be overloaded with multiple requests. Hence Gentry and Silverberg(Gentry & Silverberg 2002) proposed HIBE which allows root PKG to distribute its workload to lower level PKG.

Various literatures propose several blockchain integration schemes in IoT(Jita & Pieterse 2018)(Dorri, Kanhere & Jurdak 2017). This typically depends on the type of IoT architecture which is based on the type of application or the type of data reporting in IoT.

Fog Computing: Fog computing is a computing paradigm, that takes place closer to connected devices. It was introduced to address the issues of High-bandwidth. It is an extension of cloud computing services to the edge of the network to decrease latency and network congestion. It is characterized by wider spread and geographically distributed nodes to support mobility and user interaction. It has a distributed architecture which targets services and applications that has widely dispersed deployment(Osanaiye et al. 2017)

Mobile Edge-Computing: Mobile edge computing is an implementation of edge computing where computations and storage capacities are brought to the edge of the network within the Radio Access network. These nodes are collocated within the radio network controller or the macro-base station(Dolui & Datta 2017).

Even though, edge computing increases real time system capacity through reduced end to end latency, overloaded or malfunctioning edges can drive latency beyond tolerable limit(Bagchi et al. 2020). In addition, decentralized management can lead to security and privacy breaches.

4.2 **Detailed Problem statement**

Blockchain was developed as a solution for double spending problem in Bitcoin. Therefore, bitcoin is strongly supported by blockchain that creates an immutable record of network transactions. It incorporates a distributed ledger to accommodate each and every transaction of the network. Participants on the network, are identified using their Public Key (PK). New records are added as blocks through a process called mining and this process is conducted by a set of nodes called miners. The bitcoin mining consumes energy, because mining is basically a competition where miners try to find hash of a value that is less than a target value. The miner who find this value first, will be rewarded

with bitcoins. However such kind of mining techniques are impractical in case of IoT, which is particularly resource constrained devices. Whereas, its key features like anonymity, decentralization and security can benefit IoT devices.

IoT consist of networked objects that can sense and gather data from their surroundings, which is then used to perform automated functions. It includes devices that range from small, localized system in specific locations to large complex systems that are geographically distributed. The data gathered by IoT device may be sensitive which should be protected. Currently most IoT infrastructures are heavily centralized which acts as single point of failure. In addition; several threat security threats have emerged to disrupt the operations of IoT infrastructure. Decentralized approaches, can solve many of the threats related to IoT. Some of the threats associated with centralized architecture are

1. DoS Attack- An attack on the centralized server can affect the security of the data. It will create a single point of failure
2. Sensitive information stored in centralized servers does not have control on its accessibility. Users have limited control on how their data is stored or is accessed by whom.
3. Many organizations outsource the data storage to cloud provides.
4. With the growing number of these devices, centralized approach will not be efficient in storing and processing of these data.

Several integration schemes for blockchain and IoT are proposed in various literature. Achieving absolute decentralization for Iot is problematic. A major of current IoT architecture make use of centralized cloud for storing and processing of data. Full nodes are participants in the blockchain that can host the entire copy of blockchain and have enough processing power to operate blockchain. It can also act as validator in blockchain. Light nodes run a light client. They can issue transaction and can perform the operations in blockchain but they are not equipped with enough storage to store copy of data. In addition, it does not engage in block creation or validation process.

Authors in Ali et al. 2018 presents various integration schemes in IoT blockchain. This include Gateway devices as end-points to the blockchain, where gateways, issues transactions and act as end-points to the blockchain. The second approach is the devices as transaction-issuers to the blockchain. In this type of architecture, devices can issue the transactions and can perform cryptographic operations, however the data is not stored within the device. This approach is more suited where high throughput, low latency reliable IoT data is required. The third approach is cloud-blockchain hybrid with the IoT edge. This approach use the blockchain for certain IoT interactions and remaining events occur directly between IoT devices.

Edge Computing: Edge and fog computing are widely used techniques, that provide same functionalities where data is processed in edge nodes or fog nodes which are situated either on, or close to where the data originated from. Edge nodes are a collection of servers comprising a distributed network. Performing computations at the edge of the network reduces the network traffic which reduce the risk of data bottleneck.

According to a recent report, nearly half of the companies that use IoTs have faced security breaches and the cost to deal with these attacks are more than the traditional breaches. The cost associated with securing IoT devices is expected to rise in the coming years (Kawamoto 2017). Companies with less than 45M in company revenue reported a potential loss of $255K on average due to IoT security breaches(Bewley, S. Dean, R. Feddida, A. Maxwell 2017).

Mirai botnet attack is estimated to infect around 2.5 million devices. It causes a DDoS against a set of target servers and propagating to weekly configured IoT device. It communicates to a report server through an anonymous tor network. While Mirai was using a centralized architecture, Hijami botnet which is avariant of Mirai uses distributed architecture. The impact of DDos attacks by botnets highlight the risk of IoT devices pose to internet (Stavrou, Voas & Fellow 2016). Hence IoT devices can act as a gateway for malwares for entering into the corporate network.

Typically, an IoT infrastructure is not a standalone device. It include various layers which can be broadly classified into 4 layers. The sensing layer, network layer, Middle-ware layer and application layer. Each of these layers use a variety of diverse technologies. Sensing layer includes sensors and actuators that gather data or information and pass it to the second layer which is the network layer. The network layer uses a communication network to transmit this date to the third layer which is Middle-ware layer. This layer acts a bridge between the network and the fourth layer, which is Application layer. The Application layer includes various IoT based end-to-end application layer.

IBE is a public-key cryptosystem where the public key is the user's Identity, and the private key is generated by a Private Key generation (PKG) center based on the master private key and user's ID. The concept of IBE was first presented by Shamir in 1984 (Shamir 1985). The idea was proposed to use the user's identity as a public key instead of a digital certificate. Identity can include the user's email address, phone number, etc. Initial works on IBE proposed by Boneh and Franklin(Boneh & Franklin 2003) is based on bilinear pairing and Cocks(Cocks 2001) based on quadratic residuosity problem.

A Hierarchical Identity based Encryption(Gentry & Silverberg 2002) scheme is specified by five randomized algorithms: Root Setup, Lower Level Setup, Extraction, Encryption and Decryption.

1. Root Setup: The Root PKG, takes a security parameter k and returns $params$ (System parameters) and root secret key. The system parameters include a description of finite message space Ḿ and finite cipher text space C. The system parameters are public whereas only the Root PKG knows the root secret.

2. Lower Level Setup: The Lower level Users uses the system parameters from the root PKG. It may generate a lower level secret or a random one-time secrets for each Extraction.

3. Extract: A Root PKG or a Lower Level PKG with ID-tuple $(ID1, \ldots, IDt)$ may compute a private key for any of its children (e.g., with ID-tuple $(ID1, \ldots, IDt, IDt + 1)$) by using the system parameters and its private key.

4. Encrypt: Takes as input params, ID and $M \in$ Ḿ and returns the ciphertext C$\in C$

5. Decrypt: Takes as input params, ID, C$\in C$ and a private key d. It returns plain text $M \in$ Ḿ

4.3 **Related Work**

Depending on the use-cases, several blockchain based architectures for IoT are available in the literature. This includes blockchain architecture for smart homes (Dorri, Kanhere, et al. 2017)(Mohanty et al. 2020)(K, Babu & Manoj 2020), health care (Simic, Sladic & Milosavljević 2017)(Meena & Dwivedi 2019)(Dagher et al. 2018)(Badr, Gomaa & Abd-Elrahman 2018) smart Cities (Lazaroiu & Roscia 2017)(Michelin et al. 2018)(Sun, Yan & Zhang 2016)(Sharifinejad, Dorri & Rezazadeh 2020)(Seyed et al. 2020)(Paul et al. 2018) and energy trading (Imbault et al. 2017)(Dorri, Luo, et al. 2019)(Jain & Dogra 2019)(Silva et al. 2019). Mainly two type of integration schemes are proposed in majority of the literature. This includes

1. IoT device performing Blockchain and connecting to a centralized cloud. This type of architecture requires an IoT device to be capable of performing blockchain functions, including validation and verification of nodes and reaching a consensus. Such kind of implementation is inappropriate for many use-cases.

2. IoT devices connected to the edge node and blockchain process is performed in the edge node, with a cloud layer used for the storage and user interface. However, this type of architecture requires centralized cloud storage.

Uddin et al. (Uddin et al. 2020) propose a decentralized IoT health framework where Blockchain is deployed to perform task migration on the Edge network. The blockchain consensus mechanism is

executed on the MEC. Ali et al. (Ali, Vecchio, et al. 2018) present various integration schemes in IoT blockchain. This includes Gateway devices as end-points to the Blockchain, where gateways, issues transactions, and act as end-points to the Blockchain. The second approach is the devices as transaction-issuers to the Blockchain. In this type of architecture, devices can issue the transactions and can perform cryptographic operations, however the data is not stored within the device. This approach is more suited where high throughput, low latency reliable IoT data is required. The third approach is cloud-blockchain hybrid with the IoT edge. This approach uses the Blockchain for certain IoT interactions, and remaining events occur directly between IoT devices. Bhattacharya et al. (Bhattacharya et al. 2020) survey blockchain mining framework for IoT devices using Mobile Edge Computing.

Jararweh et al. (Jararweh et al. 2016) have identified that the integration of Mobile edge computing and Cloudlet computing is the future of Mobile cloud Computing. The advantage of this integration includes avoiding a single point of failure; it reduces the delay as the requests are served locally, increased scalability, and reduces global network traffic. A blockchain architecture using fog and cloud computing for IoT devices is proposed by Kafhali et al. (Kafhali et al. 2019).

Identity Based Cryptography has been extensively applied in variety of domains including IoT (Markmann, Schmidt & Wählisch 2015)(Hengartnert & Steenkiste 2005)(Oliveira et al. 2011). Integration of Identity Based Encryption for IoT is provided by Sankaran (Sankaran, Sanju & Achuthan 2018). The Author proposed IBE for the hierarchical topology of IoT that can massively scale. An optimized Identity based encryption for lightweight devices was proposed by Guo et al. (Guo et al. 2017). Several literatures provide the mainly, two types of architecture

1. IoT device performing blockchain and connecting to a centralized cloud. This type of architecture requires, IoT device to be capable of performing blockchain functions including

validation and verification of nodes and reaching to a consensus. This is impractical in case of several devices.

2. IoT devices connected to edge node and blockchain process is performed in the edgenode, with a cloud layer used for the storage and user interface. However, this type of architecture requires a centralized cloud storage.

Challenges:

1. Sensitivity of data: In an IoT infrastructure there are various layer of encryption and decryption. Different encryption techniques are used at different layers.
2. Consensus
3. Growing ledger
4. Cloud: Risk caused by the cloud is a challenge. Users should know who can access the data stored in the cloud
5. Sensor Errors: Sensor errors will be easy to handle in a centralized architecture, however it can become a bottle-neck in distributed architecture
6. Autonomous system:

4.4 Use-case Example and Existing System

The commonly used use-case examples of IoT blockchain includes Supplychain, home automation, health care and industrial equipment. However, we provide the example of traffic speed radars, that are located at different geographical locations to detect vehicles that are over-speeding. The traffic-radars are located within a country and is controlled by the government. Its architecture and functions varies from the commonly used use-case examples for IoT blockchain. Currently this system uses a centralized database, where the data generated by the sensor is passed to a centralized cloud system. A radar gun, works by using Doppler effect to detect the speed of the vehicle. If speed of the vehicle is higher than the government issued

road speed, then it capture the vehicle ID and generate a fine. The transaction includes Vehicle ID, Time, location, speed and fine amount.

The system components of these equipment mainly consist of speed measuring component, data processing component and image capture component (Administration & Administration 2008). The speed-measuring component will have the ability to detect and discriminate individual vehicles and measure their speed in real-time. This is achieved by using sensors. The data processing and storage component is a computer that receives data from the measuring unit and compare it with the threshold that was set to define violations in real-time. If a vehicle's speed exceed the threshold, it triggers the camera to take the vehicle's photograph. Additional information such as time, date etc is also recorded along with the speed. This unit then trigger the image capture unit in case of any violation detected. The image capture unit takes the image of the license plate, and if driver identification is required, a clear image of the drivers face.

However, this system in inefficient due to the following reasons

1. Integrity: Data generated by the sensor can be tampered by an inside employee, cloud provider or by an attacker. An employee who has access to data can modify the fines generated for him.
2. Availability: Data in the cloud acts as single point of failure. Centralized databases are always a target of attack.
3. Confidentiality: Sensor data can viewed by parties who have access to the cloud or the database. Hence if a vehicle is detected for overspeeding, the data recorded by the sensor such as location, time, amount etc are publicly available. Privacy of the user is not considered.
4. User does not have any idea on who can access the data generated by sensor.

Existing System: The existing system architecture consist of IoT devices connected to the server in the cloud using a gateway node. The data collected by the sensors are passed to a Gateway, which could be an edge node or fog node. The gateway further process the data and pass it to the cloud server.

Users connect to the server in the cloud to access the data. A major problem with this architecture is the centralized management of data.

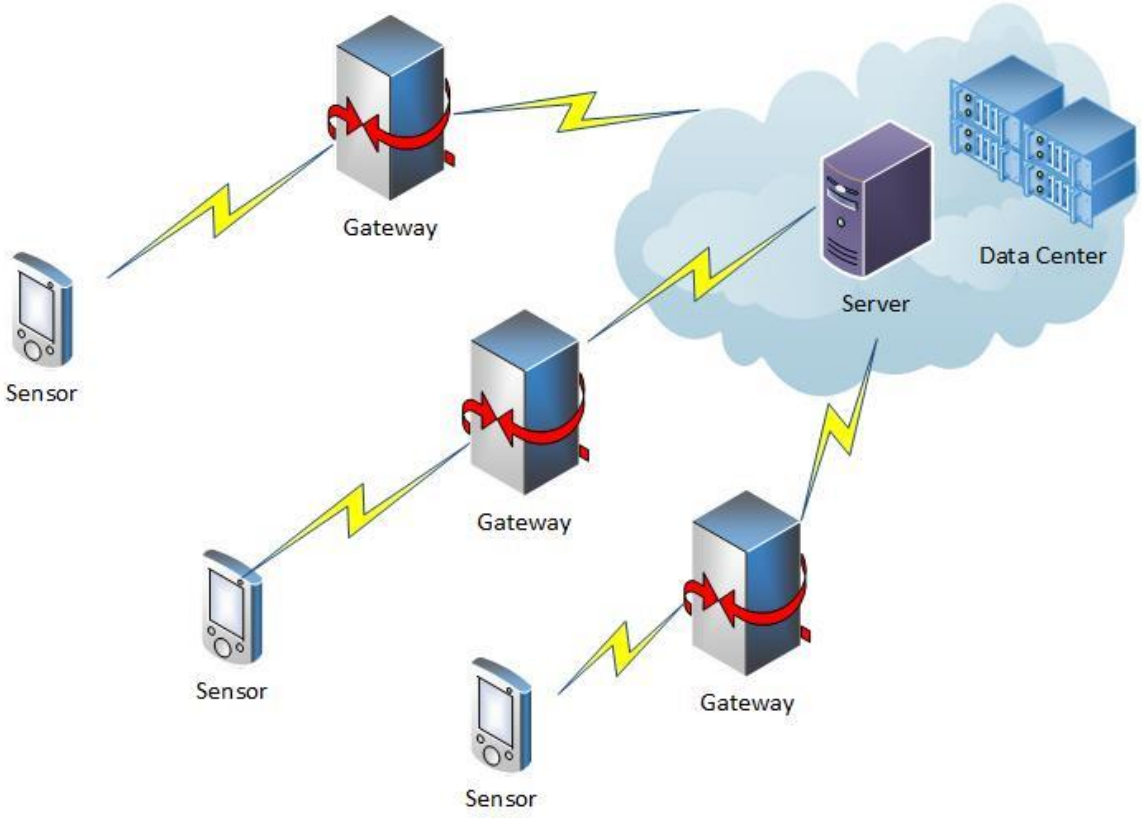


Figure 4.1: Centralized architecture using Gateway

4.5 **Proposed System Architecture**

In this section, we provide a description of our system model by defining the main entities along with their respective capabilities. We model a three-tier IoT network, as shown in Figure 4.2. These are the sensor nodes/devices layer, the mobile edge layer, and cloudlet layer. This type of architecture is applicable for event-driven IoT network, where IoT devices are connected to Mobile edge devices which are located within a radio network. These Mobile Edge devices are nodes that can perform blockchain computations. They are connected to cloudlet nodes. Cloudlet nodes CN={CN1,CN2,CN3…..CNn} form the Upper-tier closer to the users. These nodes can be considered as a data center in a box. The Middle tier is a set of MEC Nodes = {MEC1,MEC2,MEC3… MECn}

where these nodes are located within a base station or at the gateway and is within radio network of IoT sensors. The lower tier is sensors S={S1,S2,S3….Sn} detects data from the physical environment. Physical security should be implemented within the sensors, as Hardware Security Modules (HSM) that can perform cryptographic operations and protect sensors from physical tampering.

Each layer is dependent on other, While the IoT layer collects the data from physical environment, this data is passed to the Mobile Edge Layer which performs the blockchain process. The blocks that are generated by this layer is passed to cloudlet layer where each cloudlet node keeps a copy of the data. Users communicate with cloudlet node to retrieve the information.

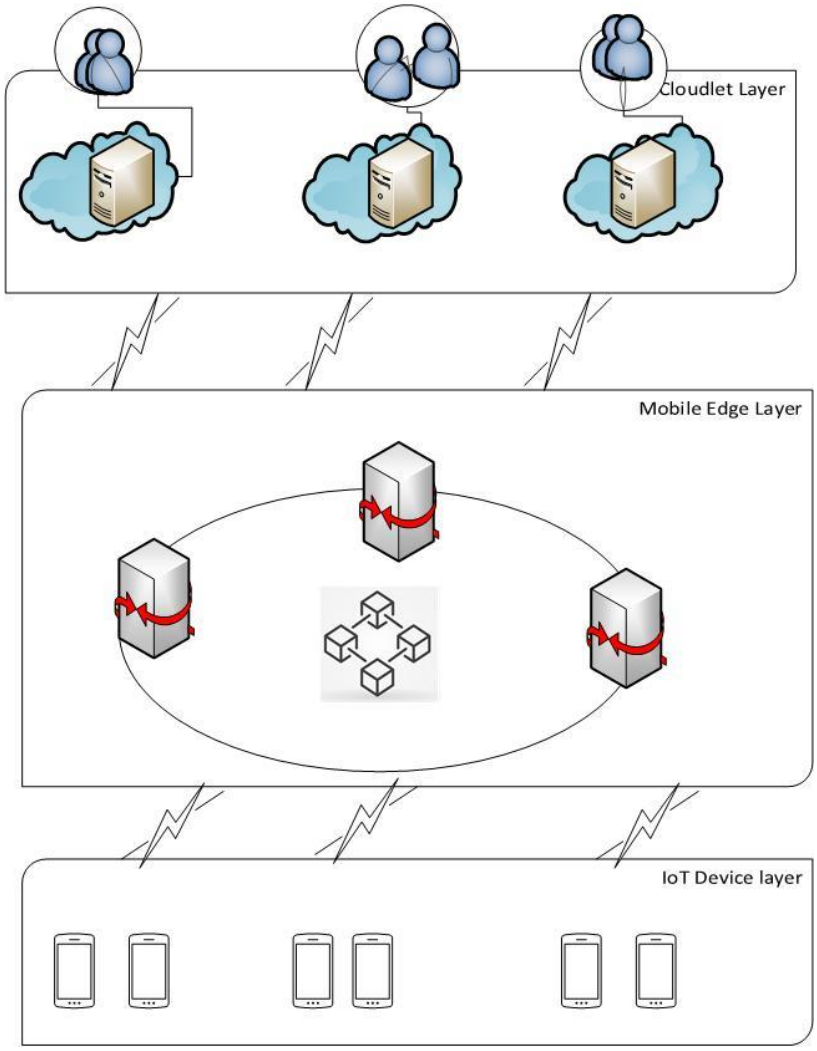


Figure 4.2: Three Tier Architecture

4.5.1 **Proposed System Architecture**

The main objective of this thesis is to provide a tamper-proof, confidential ledger where even an inside employee will not be able to tamper the data generated by sensors. To achieve this goal, a blockchain based system is proposed to support the operations of IoT infrastructure. The proposed system will use edge nodes for the blockchain process, as shown in Figure 4.3. We use edge-based architecture with Mobile Edge Layer (MEL) and Cloudlet Layer (CL). MEL is the closest to the IoT devices and is assumed to be within a radio network. The CL is closest to the users, which can also be viewed as a distributed cloud network. While the nodes in the MEL perform the verification and validation of transactions, the data are stored and copied into the CL. Nodes in the MEL are grouped as clusters based on their geographic location. When an IoT device sends an event-transaction to its nearest nodes in MEL, it collects that transaction and distributes to all Peer Endorser nodes, within its cluster network. The node that is selected using the Miner-selection algorithm (Pavithran & Shaalan 2020) will validate the transaction and add it to the blockchain, which is stored in MEL, and a copy is replicated to Cloudlet Nodes (CLN). By doing so, we create an abstraction layer by separating the MEL and CL. Clients connect to the CLN in their respective geographic location to access the transaction. CLN also contains PKG (Private Key Generator), where PKG will generate the secret key of the users.

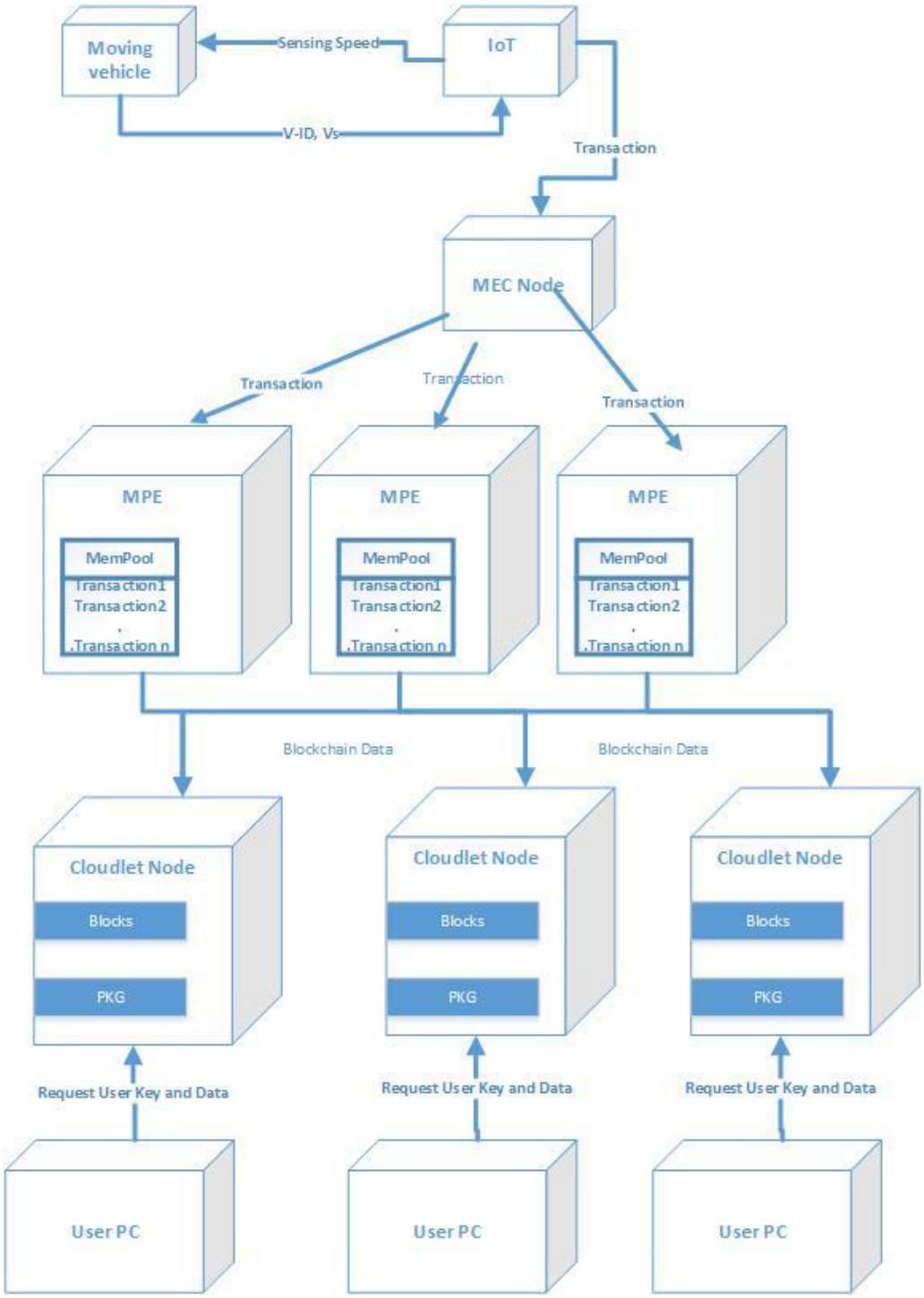


Figure 4.3: Overview of System Architecture

The proposed architecture is composed of IoT-devices, Mobile Edge Node, Peer Endorser Node, Cloudlet Node, PKG and User

1. IoT: These are resource-constrained devices that operate within a radio network. They have the ability to perform ECDSA signature and IBE encryption. The private key of the device is stored within the device as embedded hardware which should be secured through hardware security implementation (Chakraborty et al. 2018).

2. Mobile Edge Node(MEC): These nodes are computers that are not resource-constrained. They are assumed to have enough processing power to perform cryptographic operations and have enough storage capacity to store all the transactions in the blockchain. This acts as a gateway for IoT.

3. MEC Peer Endorser (MPE): This is a Peer MEC node that is responsible for Validating and verifying transactions. The MPE nodes prevent unstable or non-deterministic transactions from passing through the network. A transaction is sent to an endorser in the form of a transaction proposal. All endorsing peers are also committing peers (i.e., they write to the ledger).

4. Cloudlet Node(CLN): A copy of all the transactions is broadcasted to all nodes in the CN. However, they do not participate in the validation and verification of transactions. These nodes could be web servers or storage networks that retain a copy of all the transactions. They act as an interface for the users to connect. Each user initially registers to a particular CLN. All images and videos captured by the IoT devices of a particular user will be stored in the CLN, where the user is registered.

5. User PC(Client): A Client is a computer device that starts communication by requesting the MEC nodes to get the transactions for a particular vehicle ID. Initially, the client registers with the TA nodes providing its vehicle ID and identity proof. TA node, after verifying the identity provides the client secret key through PKG. The client decrypts the transaction to see the fine details for that particular vehicle ID.

6. PKG: PKG is a private key generation server that generates the secret key when provided with a public key. It is located within the CLN.

7. TA node: The client initially registers its vehicle Id with a TA node. It acts like a verifying authority, that request for the secret key from PKG after verifying the identity provided by the client.

4.5.2 **System Interactions**

When a radar detects a vehicle for a traffic violation, it records the vehicle details and generates a fine amount. The steps performed by various components are given in Figure 4. Let the Identity of vehicle be $V - ID$, the time of speed violation is t, the location of speed violation be loc, the fine amount for violating the speed be amt, the public key be denoted by P_k, the secret key is S_k and edge Node Id as EID.

The speed sensor in the IoT device detects a vehicle with speed above the predetermined road speed. It captures the vehicle Id, time, speed, time, location, and calculate the fine amount. It encrypts the data using IBE and creates tr. The tr is signed and send to its nearest MEC's.

$$tr = E_{P_k V-ID}[V - ID, t, loc, amt]$$

After receiving the transaction from IoT devices located within the radio network, MEC verifies the signature, and it adds EID and timestamp to the tr creating $tr1$

$$tr1 = EID || tr || Timestamp$$

MEC maintains a list of all MPE within its cluster. Hence it broadcast the $tr1$ to all MPE's that are predetermined. When the number of transactions reaches a threshold, endorser node selection algorithm (Pavithran & Shaalan 2020) is used to select the endorser node. Selected MPE will verify the signature and validate the timestamp and add it to the blockchain, which will be broadcasted to all nodes in the MEC and CLN.

Clients connect to the CLN node within its geographical area to receive its transaction. It requests PKG for its secret key and decrypt the transaction. $D_{K_S}[tr1]$

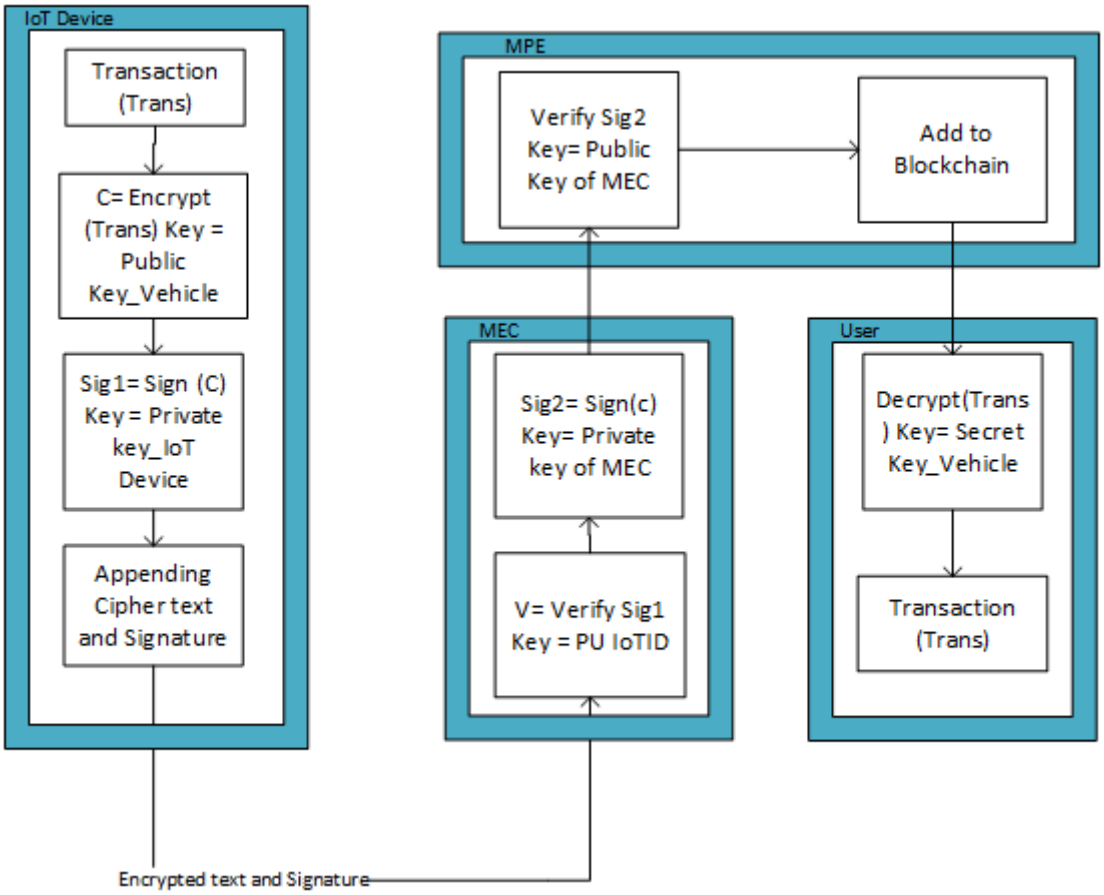


Figure 4.4: The proposed system interactions

4.5.3 **Overall System Work flow**

When a sensor detects a vehicle for speed violation, it creates a transaction encrypted with the Identity of the vehicle. The transaction includes the device-ID and other params described in the previous section. The encrypted record is forwarded to its edge node, which adds a header and forwards to the miner nodes. The miner node that is selected for validation and verification validates the signature and add the transactions to a block and is distributed to all CLN. The vehicle owner connects to PKG in CLN to retrieve the secret key. The overall workflow is given in Figure 4.5.

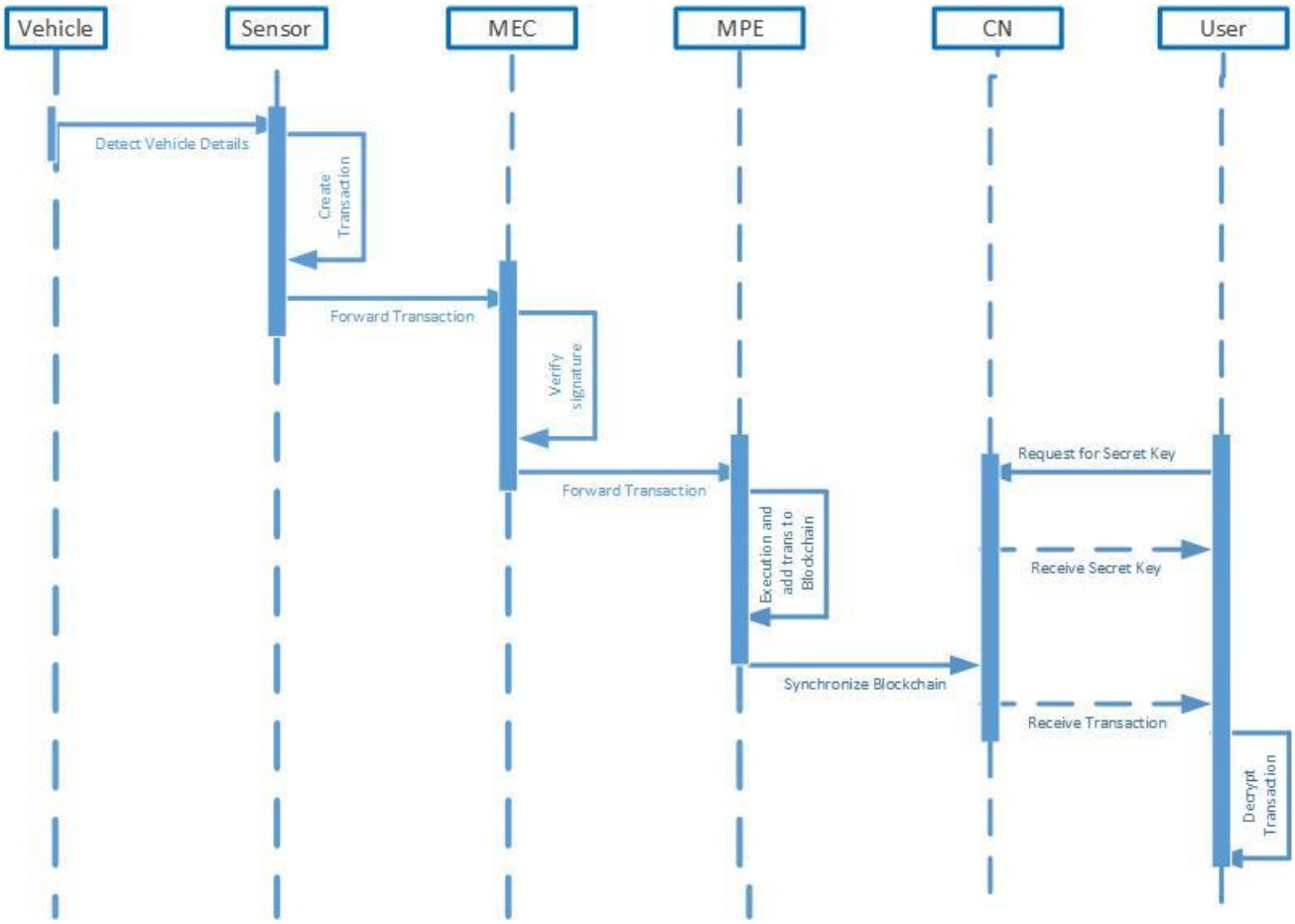


Figure 4.5: Overall System workflow

4.5.4 **HIBE Setup and Encryption at Various Layers**

PKG initially generates a master public key and master private key. User public key can be computed from the master public key. To obtain the corresponding private key, the owner of the identity should contact the PKG to generate private key. PKG generates users private key from the master private key. The HIBE allows a root PKG to distribute the workload by delegating private key generation and identity authentication to lower-level PKGs. To encrypt the message the IoT device only needs the Public parameters from the root PKG and the vehicle ID. The HIBE proposed by Gentry and Silverberg (Gentry & Silverberg 2002) have the advantage higher level PKG secret cannot be compromised if the domain PKG secret is disclosed. We use this approach with Root PKG at Level0 and CLN at Level1, Vehicle Owners at Level2.

$$ID_{Root} = ID_R$$

$$ID_{CLN} = ID_C \;||\; ID_R$$

$$ID_{Vehicle} = ID_V \;||\; ID_C \;||\; ID_R$$

Root Node Setup:

Let G be a group with prime order $q, e: G_1 \times G_1 \rightarrow G_2$ be a bilinear map and g be a generator of G. It chooses an arbitrary generator

$$P_0 \in G_1$$

and picks a random secret

$$s_0 \in \mathrm{Z}/q\mathrm{Z}$$

and sets

$Q_0 = s_0 P_0$;

Chooses cryptographic Hash functions H_1 and H_2 which are used to generate public keys corresponding to identities.

$H_1 = \{0,1\}^* \rightarrow G_1$ and $H_2 = \{0,1\}^n \rightarrow G_2$ for some n

The system parameters are

$params = (G_1, G_2, e, P_0, Q_0, H_1, H_2)$.

Public Key of Root Node

$$P_0 = H_1(ID_R)$$

Cloudlet Node Setup Performed by Root Node:

We consider all CLN at Level1, picks a random secret

$$s_1 \in \mathbb{Z}/q\mathbb{Z}$$

Public key of CLN is computed as

$$P_1 = H_1(ID_C \,||ID_R) \in G_1$$

Generate the secret key for CLN

$$S_1 = s_0 \; P_1$$

Generate secret element s_1 which will be shared with CLN and Root Nodes.

Compute

$Q_1 = s_1 P_0$;

The secret keys are securely retained, whereas the public key and public parameter Q_1 is made public

Vehicle key generation performed by Cloudlet Node

Compute Public key of vehicle

$$P_2 = H_1(ID_V \;||\; ID_C \;||\; ID_R)$$

Generate the secret key for Vehicle

$$S_2 = S_1 + s_1 \; P_2$$

Generate secret element s_2 which will be shared with CLN and Vehicle Owner.

Compute Public Parameter

$Q_2 = s_2 P_0$;

The secret keys are securely retained, whereas the public key and public parameter Q_2 is made public

Encryption by IoT Device:

Generate the Cipher text, Choose a random

$$r \in \mathbb{Z}/q\mathbb{Z}$$

Let $M \in \mathcal{M}$ be the Message to be encrypted

Cipher text $C = [rP_0, rP_2, M \oplus H_2(g^r)]$

where $g = e(Q_0, P_1) \in G_2$

$$C = [U_0, U_1, V]$$

Decryption by Vehicle Owner

To decrypt C, the Vehicle Owner computes

$$V \oplus H_2\left(\frac{e(U_0, S_2)}{e(Q_2, U_1)}\right) = M$$

4.6 **Research Methodology**

4.6.1 **Implementation and Performance Analysis**

In this section, we present the results of experiments we conducted to demonstrate the feasibility of our proposed architecture. Our experiments focused primarily on evaluating the performance of the system with respect to the case study described in section 4. For this purpose, we created and deployed smart-contract in Ethereum (Sianturi 2019) in order to test the blockchain network. Smart contracts are automated contracts that are self-executing with specific instructions written in its code, which get executed when certain conditions are made. We used Solidity for creating smart contract. To test the

system performance, we created a four-node network using Ubuntu 16.04 LTS. The Encrypted information received from the IoT along with the other parameters, are passed through the Ethereum blockchain. We tested the network by creating 10, 20 and 30 transactions at various time intervals, as provided in Table 1. We used geth version of Ethereum and geth metrics are used for calculating the performances. We measured the disk and memory activities. The average read and write count is given in Figure 4.6 and average read and write data is given in Figure 4.7. Figure 4.8 provides average memory used for executing 10, 20 and 30 transactions.

Table 4.1:No: of transactions passed at various time intervals

	Time interval t1	Time interval t2	Time interval t3
Node1	3	7	5
Node2	4	3	8
Node3	1	5	10
Node4	2	5	7

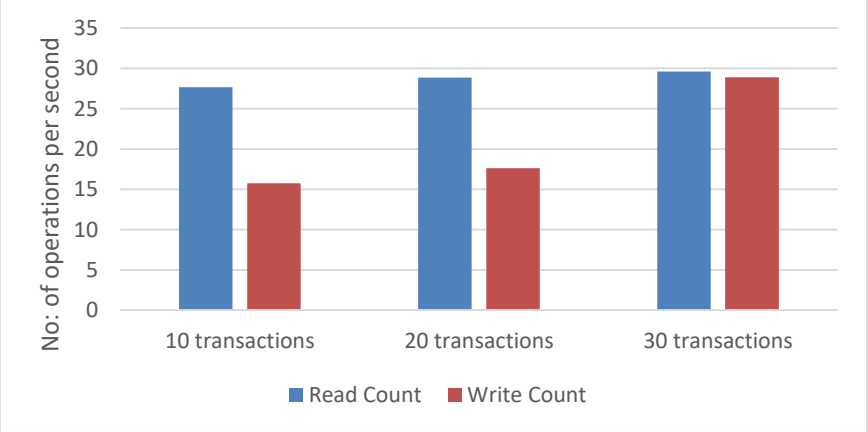


Figure 4.6: Average Disk Read Write count for executing 10, 20 and 30 transactions

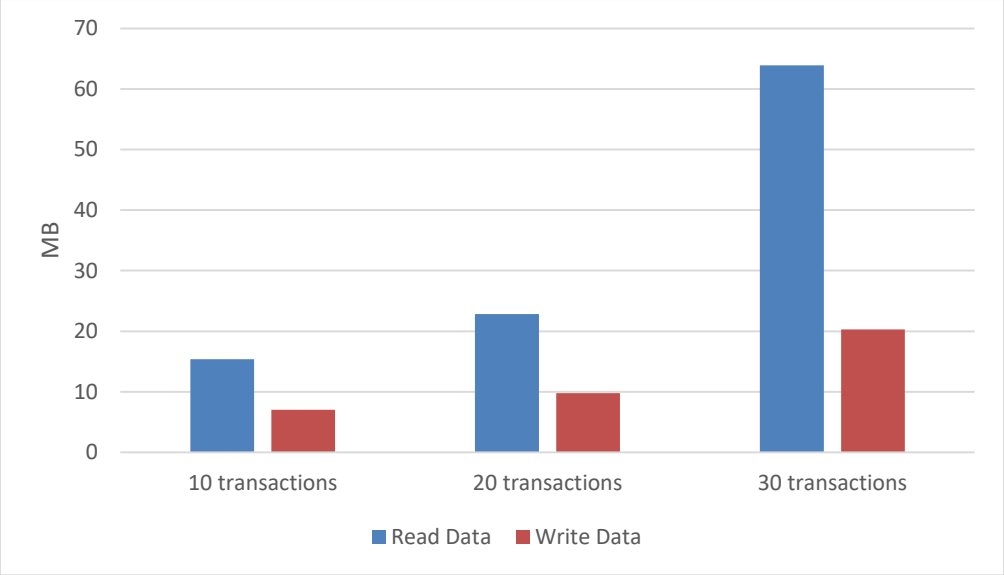


Figure 4.7: Average Disk Read Write Date for executing 10, 20 and 30 transactions

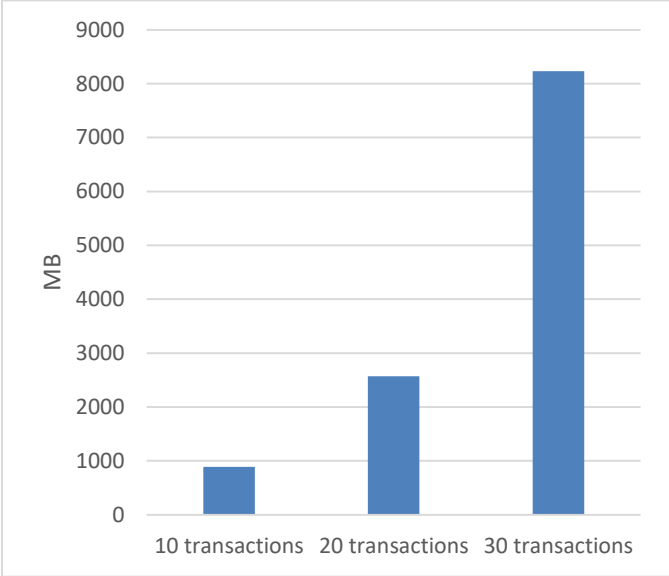


Figure 4.8: Average memory used for executing 10, 20 and 30 transactions

To measure the network performance, We simulated the network using Contiki OS (David & Hsu 2018) to calculate the latency and throughput. We used Z1 motes generated in a linear position with

edge nodes within its radio network. The latency between the IoT device and the edge node is calculated and is given in Figure 9, and the average throughput is given in Figure 10.

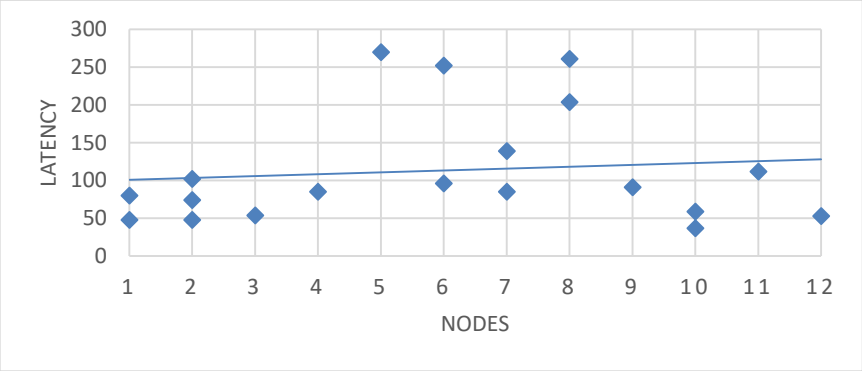


Figure 4.9: Latency within the IoT device and Edge node

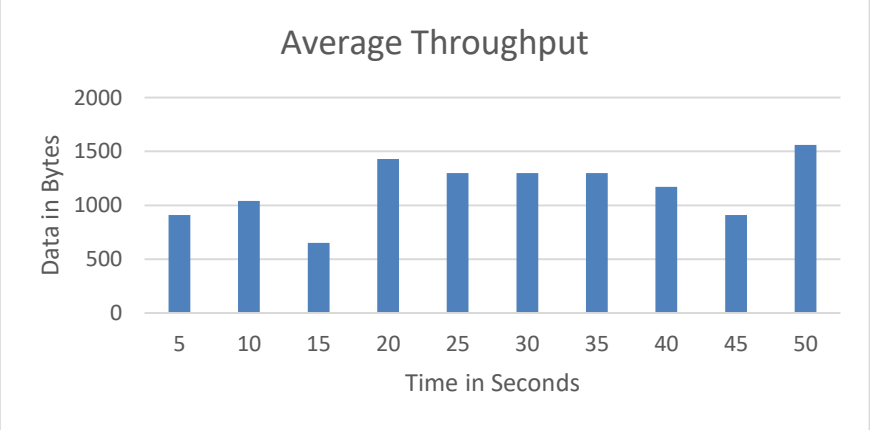


Figure 4.10: Average Throughput

4.6.2 **Security Analysis and Threat Model**

To ensure efficient communication within devices and within devices in different layers, the IoT architecture must fulfill various security requirements. If a device is compromised, an attacker can deny the device from sending the transactions (DoS). However, it is not possible to authenticate a fake transaction, unless the private key of the device is compromised. Compromising sensor data, during the sensing process can create false transactions. If an MEC node is compromised, it can deny the transaction. However, it cannot create fake transaction. Table 2 provides how this architecture provides security objectives and defend against some major security threats.

Table 4.2:Security objectives and security threat analysis

	Description	**Defense**
Confidentiality	Data generated by the sensors should be protected from unauthorized users. This in-turn, ensures the privacy of data.	Data is end-to-end encrypted. Hence it provides confidentiality.
Integrity	Maintaining the integrity of data is a crucial requirement. The data generated by the sensors should be protected from modification by malicious parties. These parties could be an outside attacker or an inside employee. In the case of IoT, the data is generated by the sensor from the physical environment. Hence this data should not be subject to any modification.	Blockchain provides integrity of data by hashing the transactions and linking blocks with the previous blocks making it harder for an attacker to modify the transactions.
Availability	The data should be made available to legitimate users. Hence denial of service attack should not impact the availability of data.	Data is distributed to several nodes which provides availability
Authentication	Only authenticated users or processes should be able to access or modify the data.	Devices sign the transactions with its private key embedded within the device as HSM. Signatures are verified by the MPE Nodes

Forgery	Fake identities and profiles, fake information to mislead the user.	Transactions are signed and public key of all IoT devices are shared among the LEN. Hence Forgery can be prevented.
Tampering	It is the act of deliberately modifying the data through unauthorized channels	The data generated by sensors are hashed and signed. Hence modifying the data can be prevented. However dropping or delaying the transactions cannot be prevented
Sybil attack	It is a type of attack where a node becomes malicious and can create and operate multiple fake identities. This is mainly done in a peer-to-peer network to gain control of the network. If an attacker manages to control the majority of the mining power in a blockchain network, then he/she will be able to create fake transactions.	Created multiple fake identities can be prevented using this architecture, as all the neighbouring nodes within the cluster have the pre-determined pubic key of all other nodes. If an attacker wants to create a fake identity, he/she will have to add the public key of the device in all the neighboring nodes. If a new device is added to the network, it should be authorized by the TA.

Eavesdropping:	Capturing of transmitting packets and try to read the contents. Any type of smart environment can be a victim of these attacks.	It can be prevented as data is end to end encrypted.
Man-in the middle	Manipulating the exchanged data between the sender and receiver.	The data shared between the devices and nodes are encrypted, signed and hash, Hence it is computationally impractical to manipulate it.
Data Privacy	Exposure of user data to unreliable parties.	User data is encrypted, which can only be decrypted with the secret key of the user.

4.7 **Chapter Conclusion**

The Chapter has presented blockchain architecture for IoT using Mobile edge and cloudlet computing. The presented model is deployed in a use-case of traffic speed radars, where the integration of Blockchain and HIBE is evaluated. The architecture provides confidentiality, integrity and availability of data through a partially decentralized network. Our architecture provides Confidentiality without the need for sharing secret key among nodes. A detailed simulation is carried out using the Contiki OS. In Addition, a 4 node blockchain is created using ethereum, and a smart-contract is executed. The system and network performance is evaluated. The obtained experimental outcome indicates relatively

low latency and high throughput under several evaluation conditions. Hence this Chapter answers Research Question 5.

Assessment was validated in a step by step process, by initially collecting the requirements from various literatures and then created a simulated environment using Raspberry Pi as IoTs and Ubuntu Machine as edgenodes. The data generated by the devices follows the Blockchain process where ethereum platform was used. The system performance was collected using Geth Metrics by passing 10, 20 and 30 transactions at various time intervals. The network performance was collected using Contiki Simulator. The performance of the system during this process was then compared with the traditional system. The obtained result was documented and graphical representation was provided.

Chapter 5 : PUBLIC KEY AUTHENTICATION FOR IOT BLOCKCHAIN

Besides confidentiality and privacy, trust is an important factor for any IoT system. When a sensor sends data that is signed with its private key, the receiving nodes verify it using the public key of the sensor. Hence, it is understood that authenticating the public key of the system is part of creating trust within the system. Traditionally, trust is maintained using Public Key Infrastructure (PKI) where a centralized Certificate Authority (CA) is used for authenticating the public keys. However, a centralized system can result in single point of failure where CA can be compromised or can act maliciously. Decentralizing this system using blockchain and by automating the process of certificate authentication without the need for a central third party can overcome the above-mentioned limitations. In this Chapter, we identify the challenges in creating such a system and propose a generic framework for PKI in IoT infrastructure using blockchain that can provide the functions of a CA**.**

5.1 **Introduction**

Internet of Things is a promising technology with millions of intelligent devices collaborating with each other to create a smart world. They can sense information from a physical environment, communicate within different devices and can take appropriate decisions. It generally consists of five components (Horrow & Sardana 2013): (1) sensors to collect information from surroundings, (2) Actuators to trigger a device for particular functions, (3) Computing node that process the information sensed by the sensor (4) Receiver to receive message from other devices and (5) Communicator to pass message to other devices.

In a network of IoT devices that communicates with each other, before sending any data, the receiving device should be authenticated within the network. This authentication can be effectively done by providing certificates to the devices. However, the traditional certification process includes a centralized Certificate Authority (CA) to generate and issue the certificate. A major drawback of this approach is that the certificates are issued by a third party which could lead to single point of failure

within the system, where the third party could potentially act maliciously or can be compromised. According to a recent report, thousands of malware samples uploaded to VirusTotal was given valid certificates by a well-known Certificate Authority(Rashid 2019). Another major weakness of this centralized approach is the cost and complexity of deployment. The complexity of a possible solution and intense computations of asymmetric cryptography makes it less suitable for IoT devices. In addition, dedicated team of experts are required to handle such PKI implementation.

Blockchain enables all devices to communicate with each other and automatically verify the transactions generated by the devices. In case of IoT, the data is sensitive and should be protected. Hence, when a transaction is carried out, the device should verify to whom it is sending the data. A centralized management and a trusted authority entirely contradicts with the concept of blockchain which is basically a decentralized, transparent solution that provides trust through cryptographic techniques. Compared with the conventional PKI, blockchain based PKI have the advantage of eliminating single point of failure, provides certificate transparency and reliable transaction record. It can prevent man in the middle attack and minimize the control of third parties on users keys. Although the concept of blockchain originated as a tool for cryptocurrency, the concept can also be utilized for other applications including IoT (Dorri, Ali and Kanhere, Salil S and Jurdak 2017). Several modifications of the traditional bitcoin protocol are available in the literature for IoT. But it is understood that the challenges faced in IoT-blockchain use-caseuse-cases differ from the cryptocurrency use-caseuse-cases.

Asymmetric encryption uses public and private key. Private key is kept secret within the device while, public key is made public. Even though the public key is public, it should be distributed in a trusted manner otherwise a phishing attack can lure the devices to use the public key of an attacker and will result in sending confidential data to the attacker. Compared to the asymmetric key system, the symmetric key system has advantages in computational complexity, but key management and security are insufficient. For example, the difficulties to certify between the neighboring nodes and the joining

and the leaving nodes are not flexible enough. Particularly in the Internet of Things environment, how to integrate key management systems and other networks to explore. For this reason, the asymmetric key system is also applied to Internet of things(Yang et al. 2013).

In this work, we use all the advantages of blockchain technology to create a framework for public key authentication system for IoT devices. We assume, the device has the capability to perform basic cryptographic functions and the private keys are stored as hardware embedded cryptographic chips attached to the device.

The rest of the Chapter is organized as below. We provide a brief background on Public key infrastructure and blockchain in Section5.2. Related works are listed in Section 5.3. Section 5.4 highlights the challenges in replacing a centralized PKI system to a decentralized system. Section 5.5 elaborates the proposed framework and Section 5.7.15.7 provides implementation details and comparison of the traditional approach to our approach. Finally, Chapter conclusion is provided in Section 5.8.

5.2 **Public key Infrastructure and Blockchain**

Public key Infrastructure is used as an underlying technology to support confidentiality, integrity, authentication and non-repudiation. It manages the digital certificates and encryption keys of a given identity in a permanent and reliable way. Currently, the most commonly used approach for PKI is the CA based PKI. The CA based PKI is centralized which includes a trusted third party and hence has the disadvantage of single point of failure. Apart from the CA based model, The Web of Trust Model is a decentralized PKI, where public keys are signed by trustworthy users. This concept is used in PGP (Garfinkel 1995). However, it is less popular due to the fact that, a new user will have to meet someone in person to verify its identity and get its public key signed.

The three core functional components to a PKI are certificate authority, repository and management function (RSA Data Security 1999). A Certificate Authority is a trusted third Party that issues certificates, repository is an LDAP enabled directory services to store keys, certificates and the

certificate revocation list (CRL). Management Functions are a set of policies and procedures that is typically implemented. In addition to the core functions, additional components of PKI are Key recovery service and registration authority (RA). Key recovery service provides automated key recovery in case of lost or stolen keys and registration authority services includes collecting user information, verifying user identity, register the user according to policies and accept certificate requests. Systems performing CA and RA series are often referred to as certificate servers and registration servers. The core functions of a PKI are issuing certificates, revoking certificates, storing and retrieving certificates. In a traditional PKI, the keys can be generated by the client or the other option is CA generating for the client. This depends on the application for which the key will be used. If a key is used only for signing purpose, then it is better that this key is generated by the client itself. In case if the key is used for decryption purpose to provide confidentiality of the message, then it might be prudent to have the CA generate or have access to the key (Kenneth G Paterson 2003). This is to ensure if a client loses the keys, the encrypted information can still be decrypted. After generating the key, client provides a copy of the public key to the CA to certify. To verify the identity of the user and to ensure it is a legitimate user, Registration Authority is used. The functions of RA and CA can be combined and the generated certificates are stored in a certificate directory. Users can look up the directory for validity of the certificates. Figure 5.1, illustrates the steps of PKI. Suppose Bob wants to send a message to Alice using her public key. Alice should initially generate public-private key pair and submit the public key along with her identity to RA. It verifies the identity and request for certificate to CA. A certificate is basically, signing the public key of the user with the private key of the CA. CA issues and stores the certificate in the directory. Alice can get a copy of the certificate from CA. When Bob wants to encrypt a message, he checks the directory for Alice's Certificate and in turn verifies the signature of CA and the respective validity of the certificate.

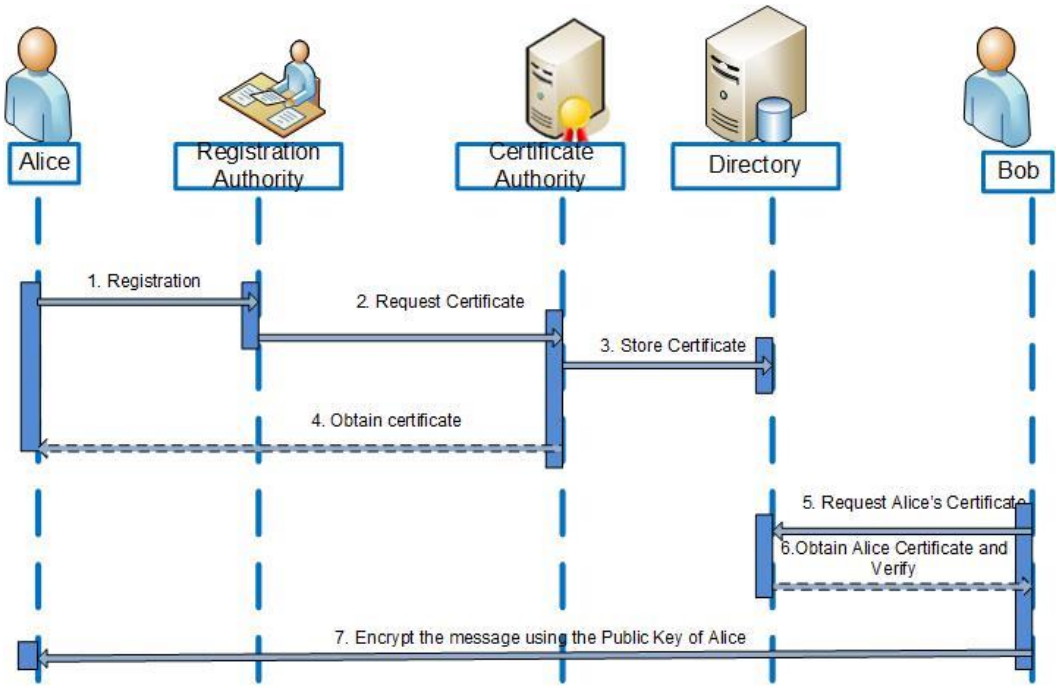


Figure 5.1:CA based Public Key Infrastructure

Blockchain is basically a distributed ledger, where a collection of data is shared and synchronized among different nodes in the network that are geographically spread through distinct locations (Pinto, Dias & Ferreira 2018). It consists of a growing list of records, called blocks that are linked together using cryptography. The concept of blockchain was introduced with bitcoin, which is a decentralized payment system that eliminates the need of central authority and provides solution to double spending problem. Each block in the blockchain is linked to its previous block using hash functions making it tamper proof ledger of records. The first block is called the genesis block. Transactions or records are stored in each block. Generally, each block contains hash of previous block, timestamp and transactions. In the bitcoin blockchain, the validation of the transactions is done by a group of miners who competes to solve cryptographic puzzle in order to get reward as bitcoins. This process is called Proof of Work (PoW). However, bitcoin mining consumes a good portion of the world's energy. This is due to the fact that mining involves calculation of trillions of hashes per second (*www.blockchain.com/charts/hash-rate* 2019), which is continuously increasing per year. Such kind of validation methods are not suitable

for IoT devices which are resource constrained. There are several alternatives for PoW in IoT. An example is IOTA, a decentralized protocol typically for IoT which uses Tangle consensus (Popov 2018).

5.3 **Related Work**

PKI using blockchain was proposed in several prior works. Blockstack (Ali et al. 2016) is blockchain based naming and storage system. It enable users to register unique, human-readable usernames and its public-keys along with additional data. Blockstack ID is PKI and identity system based on Namecoin. Namecoin (Kalodner et al. 2015) is a cryptocurrency which is a fork of bitcoin and offers the same features as bitcoin with the addition of a name/value store that can be used to hold arbitrary data. It uses PoW. The main motivation for starting Namecoin was to decentralize DNS using blockchain technology.

Emercoin (Konashevych 2015) is a blockchain based PKI which uses a hybrid system by combining PoW and Proof of Stake consensus. However several limitation of this approach was identified in (Yakubov et al. 2018). Certcoin (Fromknecht, Velicanu & Yakoubov 2014) is a decentralized PKI system based on bitcoin. Axon and Goldsmith (Axon & Goldsmith 2017) propose Blockchain based PKI that provides privacy. Certchain (Chen et al. 2018) design a certificate management system based on four layer blockchain architecture. It uses a rank based consensus protocol. But this is mainly for SSL/TLS connections and is not applicable for IoT. Blockchain based system to secure messages is proposed by Khacef and Pujolle (Khacef et al. 2019). It uses smart contract to verify the identities and its public keys and validates user certificates. It is not based on Namecoin and they use their own system but it provides only registration and does not provide key recovery or key updates. In Yakubov et al. (Yakubov et al. 2018), authors adds blockchain features to extensions field of X509V3 digital certificates to make it a hybrid certificate.

However, all the above approaches are for providing certificates for SSL/TLS for domain names and servers. These approaches cannot be directly applied for IoT, as IoT's architecture and operations differ from these. Blockchain based PKI for crowdsources IoT sensor information is proposed in Pinto

et al. (Pinto, Dias & Ferreira 2018). Their approach is based on keybase and revocation and updating of keys in key base is a trivial task is not identified.

A set of use-cases for privacy preserving blockchain based PKI can be used are identified in (Axon & Goldsmith 2017). These include:

Ubiquitous computing and IoT: User interactions with a computer system can occur across multiple devices, such as laptops and smartphones, and IoT devices such as portable devices. A user's actions and location can be traced if they are linked to multiple devices.

Vehicular networks: PKI is required for secure inter-vehicular communications, but its use must not enable remote tracking of a vehicle's actions. The identity corresponding to a public key must not be publicly disclosed, or keys linked at update.

Anonymous forums and networks: Such networks requiring user anonymity need a PKI in which users can verify network membership, but need disclose no further information pertaining to their identity or linking their separate actions. In this case, an entity's public keys should be frequently updated, and key updates not linkable. Identity should not be linked with public key.

Smart cards: Smart cards have multiple uses, authenticating payments, and proving credentials or identity. A single smart card may be used in multiple locations and for multiple purposes, so its use should not be traceable by repeated use of the same public key, or by linkable updates.

5.4 **Challenges**

Some of the challenges we identified in creating PKI for IoT blockchain are as follows:

Private key Storage: In a bitcoin blockchain, private keys are stored with the users in their wallet. If the wallet is lost all bitcoins are lost. However in case of IoT, private keys should be secured by either embedding it in the hardware or through softwares. Cryptographic keys can be stored securely in chips which provide physical and electronic security mechanisms. Hardware embedded cryptographic chips are available that can store the private key and can also perform the required mathematical operations.

Scaling PKI for Billions of Devices: IoT devices are mostly deployed in public. Grouping and managing certificates for billions of devices is a key challenge. Only authorized devices should be provided access to the network.

Heterogeneity of the devices: Devices are manufactured by different manufactures without specific standards. Hence bringing all these devices under a common rule is a challenge today.

Storage: Blockchain is a growing ledger where data is stored in a distributed fashion. Distributing the data within light node is impractical in the long run. Hence the storage should be designed within edge nodes or separate cloud storage should be used.

Consensus: Bitcoin blockchain performs expensive consensus mechanism which consume a good portion of energy. However for a resource constrained IoT, this is impractical. Hence designing an energy efficient consensus is a challenge.

5.5 **Proposed Approach**

Our approach is to create a blockchain including edge nodes and IoT devices, where each edge nodes are not resource constrained devices and is located in different geographic locations, and connected to IoT devices within the location.

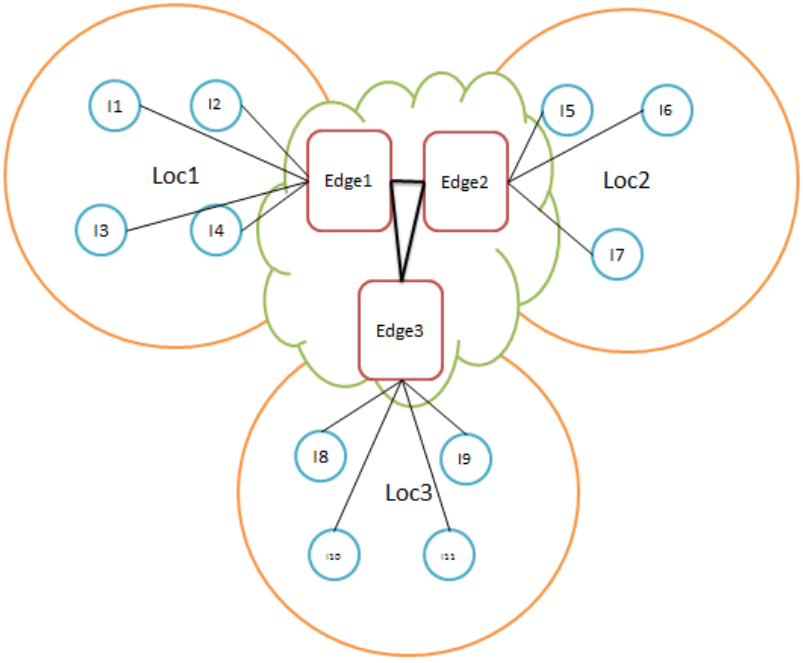


Figure 5.2:Device connectivity with edge nodes

Figure 5.2, is a generic figure illustrating the devices in various locations and its connectivity to the edge nodes. Different locations are identified as {loc1,loc2,…loc_n}, edge node as {Edge1,Edge2…$Edge_n$} and IoT device as {I1,I2,I3…I_n} Figure 5.3 provides the block structure and the representation of data to be added in the blockchain. Each block contain block header, hash of previous block, Merkle root and transactions. Transactions include Device ID, Public key of the device, certificate, location of device, time stamp and expiry data.

Initialization:

1. IoT devices are grouped based on the locations where each device is registered to an edge node that are not resource constrained devices.

2. Private keys of each IoT device are embedded within the device and the device is capable of performing cryptographic operations.

3. Edge nodes generate their own private/public key pair and their certificates are validated through Web of Trust model. These transactions are initially added to the blockchain and distributed to all the nodes.

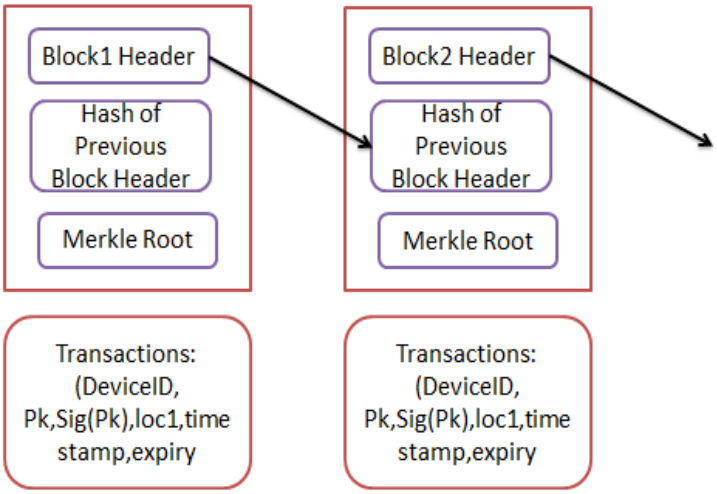


Figure 5.3:Block Structure of the proposed PKI for IoT blockchain approach

Registration:

1. When a new IoT device is added to the network, the device is registered to its edge-node with its Device ID, Public key and timestamp

Blockchain process:

1. Edge node create transaction

Device ID, Pk, Sig(Pk), loc1 and share the transaction with its neighboring nodes

2. The neighboring nodes elected through a cluster head selection process validate the transaction and adds the same to the blockchain. The structure of the block is given in Figure 5.3. It includes Block header, Hash of previous block, Merkle root and transactions.

3. The validated transactions are created as blocks and added to the blockchain, which is distributed among all the edge nodes. When an IoT device in loc_i tries to establish a communication with a device in loc_j; it contacts its edge node, which would in turn verify the device's public key signed by corresponding edge-node based on the latest transactions.

Key Revocation/Key updates:

1. When a device private key is compromised, the new key can be registered to the edge node, which follows the same steps as above. Only the latest transactions for the device will be used and hence the older key will automatically become invalid.

2. When the private key of an edge node is compromised, it will have to create new transactions for all the devices within its network.

5.6 **Overall System Workflow**

Devices initially register with its edge node, which then creates transactions that is shared to the neighboring Miner nodes. Miner nodes validate the transaction and store the validated transaction in the edge nodes. When an IoT device wants to send an encrypted message to another device, it first

request with the edge node for the device certificate. After receiving the certificate, it encrypt the message using the public key.

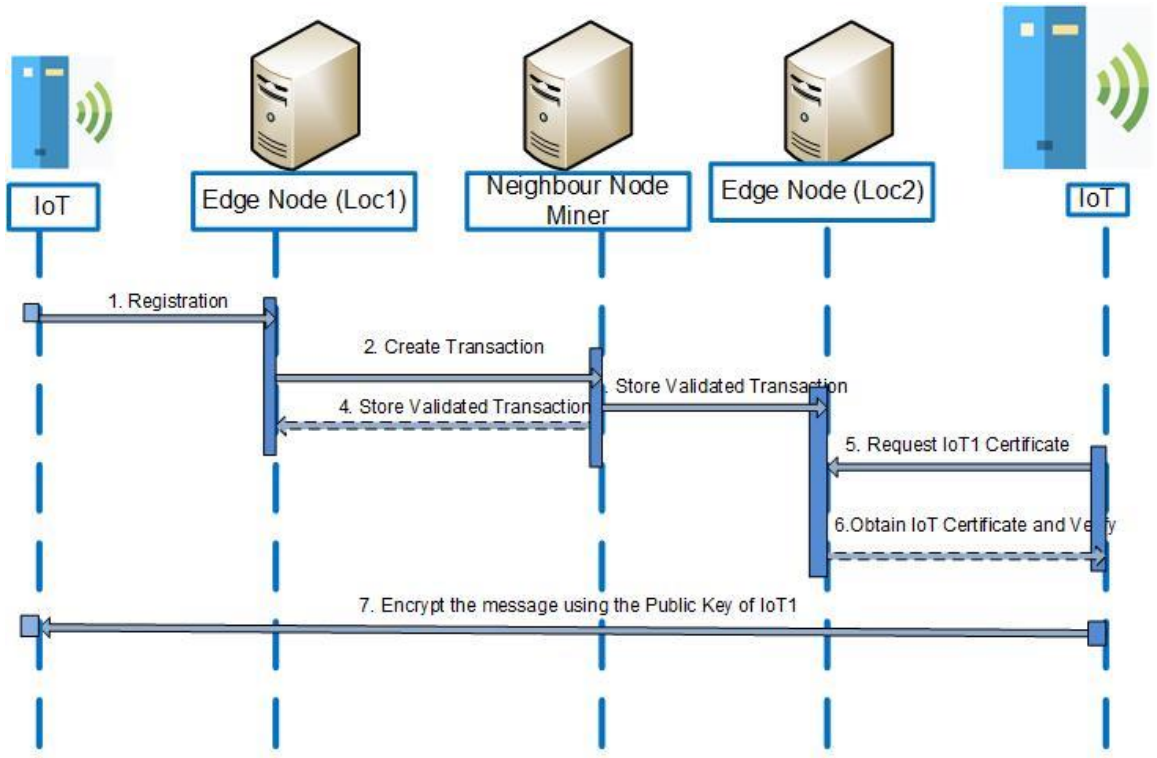


Figure 5.4:Steps in the proposed PKI for IoT blockchain approach

5.7 Research Methodology

5.7.1 Implementation

RaspberryPi with a sensor can be used to simulate the IoT. We implemented the blockchain process on two edge nodes. We used Ubuntu 16.04 LTS and Geth was used to run a full Ethereum node implemented in GO(*Official Go implementation of the Ethereum protocol* 2019). We used solidity, to program the smart contract and deployed it in the network. Smart contract store device_ID and public key of IoT in the blockchain. The screenshot of the Ethereum wallet executing smart contract is shown in Figure 5.5 and Figure 5.6

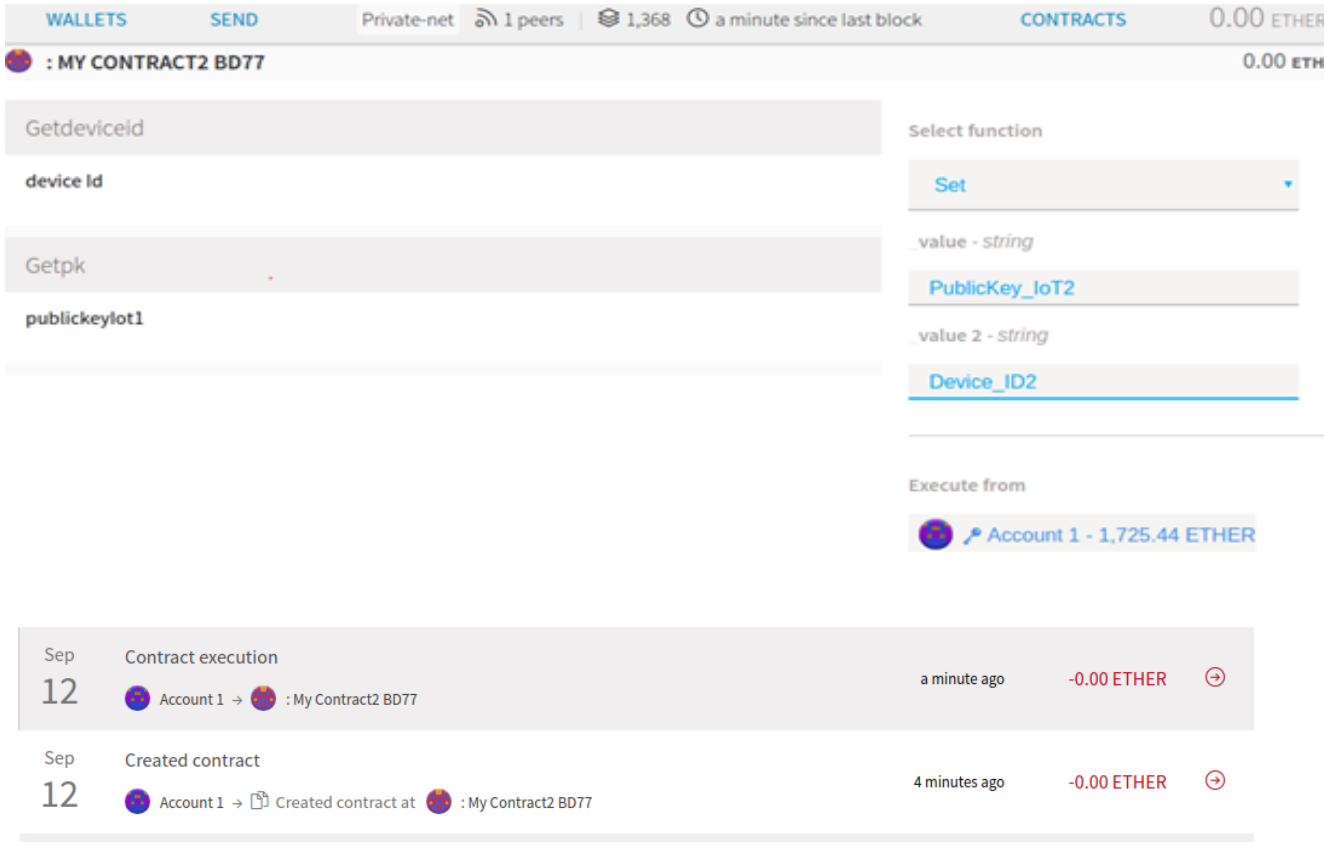


Figure 5.5:Screenshot of Smart contract deployed in Ethereum for public key

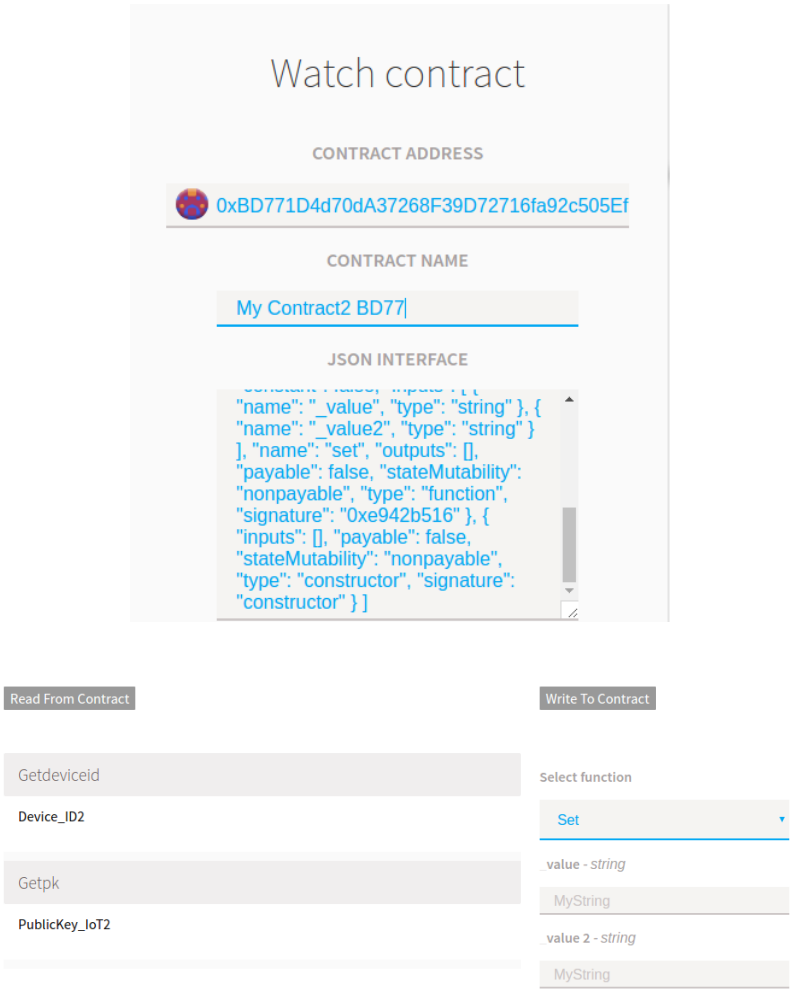


Figure 5.6:Screenshot of Smart contract deployed in Ethereum for public key-1

A snippet of the code used for creating smart contract is given in Table 5.1

Table 5.1 Code used for creating smart contract

```
pragma solidity ^0.4.24;


contract MyContract2 {


string value1;

string value2;
```

```
constructor() public{
value1 = "publickeyIot1";
value2="device Id";
}
function getpk() public view returns(string){
return value1;

}
function getdeviceid() public view
returns(string){
return value2;

}
function set(string _value, string _value2)
public {
value1 = _value;
value2=_value2;
}
}
```

5.7.2 Comparison with existing System

A comparison of our approach with the functions of traditional CA based PKI is given in Table 5.2.

Table 5.2:Comparison of CA based PKI with Blockchain based PKI for IoT

PKI Functions	CA Based PKI	Blockchain Based PKI for IoT (This work)

Issuing Certificates	CA sign and stamp the certificate after authenticating the identity of the requestor. The certificate may be returned back to the owner or posted in a repository.	Edge node within the network sign and stamp the identity of the IoT device. It create transaction that is shared to other edge nodes for validation and then added to blockchain
Revoking Certificates	CA revoke the certificate before its expiry in case of lost or stolen private keys. The revoked certificates will be included in the Certificate revocation list (CRL)	The new key can be registered to the network. As only the latest transactions for the device will be used, the older key will automatically become invalid
Storing and Retrieving Certificates	Certificates are stored via directory services	Certificates are stored as distributed data structure
Updating Keys	Certificates have a validity of usually one year or two. After the expiry of the key, Keys are updated to new key	After the expiry of the key, Keys are updated to new key
Backing up Keys	Keys should be backed up in case if user forgets the password or in case of a virus attack	Private keys are embedded within the IoT device Hardware

5.8 **Chapter Conclusions**

Blockchain is a promising technology that can provide trust among ‘things’ without the need for a central authority. We provide a framework for managing public keys of IoT devices in a decentralized way. We initially identified the challenges and gaps in various literatures on creating a public key authentication process for an edge driven IoT. Our approach provides solution for issuing certificates,

revoking certificates, storing and retrieving certificates, updating keys and backing up keys for edge based IoT devices. Our approach use edge nodes for verifying and validating the transactions. We identify the challenges and provide comparison of the CA based PKI functions with the Blockchain based PKI for IoT. We provided the work flow of the process how this can be implemented using smart contract. Hence this Chapter answers Research Question2.

Chapter 6 : DATA CONFIDENTIALITY IN IOT BLOCKCHAIN

Recently, the blockchain technology has attracted tremendous interest from both academia and industry. The technology currently spans several applications such as healthcare, Internet of Things (IoT), and cloud storage. However, implementing blockchain for IoT possess several challenges which have been identified in many recent researches. One of the major challenges is the sensitivity of the data generated by sensors in the IoT. Since blockchain produces transparent and auditable records, it is still not clear on how the confidentiality of data can be maintained. In this chapter, a detailed state of the art survey is conducted for blockchain-based approaches for IoT. In particular, a comparison on various encryption schemes used for the confidentiality in IoT blockchain. Further, this Chapter provides examples of generic use-case showing how Identity Based Encryption can be used to provide data confidentiality in IoT blockchain**.**

6.1 **Introduction**

The number of embedded devices are increasing in an exponential rate. These devices range from small low power devices like sensor nodes, wearables, RFIDs, smart phones to large devices such as smart vehicles and Industrial control system (ICS). Some of these devices have low battery life, low computational capability and less storage space, whereas others have powerful computations and good storage space. Blockchain can be used in IoT to trace the data sensed by devices and prevent forgery or modifying data. In addition it can communicate with other devices and make decisions without the need for a trusted central authority.

The process of converting information into non-readable form is called encryption. Traditionally, confidentiality of data is attained by encryption. However, bitcoin does not use any encryption process for data confidentiality (Nakamoto 2008). Every transactions made by users are readable and can be publicly accessible from the *blockchain.info* website. It provides pseudo-anonymity such that the sender

will be identified by a public address. In the case of Internet of Things (IoT), the data is sensitive; the data generated by IoT should be protected while sending it to other device, gateway, or cloud.

Due to the increasing number of IoT devices, security is a paramount concern (Weber 2010). Providing protection for the sensor data is a trivial task due to the fact that most devices are located in public places where attackers can gain access to data, falsify the data or hijack the device itself (Deogirikar & Vidhate 2017). To provide confidentiality of data, many works use symmetric encryption like AES (Advanced Encryption Standard). Symmetric encryption techniques use single key for encryption and decryption. The encrypted key should be shared with the miner or validator for the decryption purpose.

Another approach is to use public key cryptosystem. However, it has two limitations. Firstly, it needs a Public Key Infrastructure (PKI) for certificate management and verification. Secondly, most public key encryption schemes like RSA Encryption needs high computations which could not be possible in a resource constrained environment like IoT eco-system. However, lightweight encryption schemes such as elliptic curve cryptography are proposed to use in such an environment (Malan, Welsh & Smith 2004). This facilitated the development of numerous public key cryptography for IoT devices.

The need for lightweight cryptography for resource constrained devices has gained wide attention from the research community. Attribute based encryption (Goyal et al. 2006) is identified as a potential technology for providing data confidentiality in distributed systems. Identity Based Encryption (IBE) is lightweight, support massive users, simple key management and flexible key applications. It provides identity authentication, digital signatures, and privacy protection for electronic contracts (Zhu & Fan 2019). Homomorphic encryption is a promising field that allows computations on encrypted data. However choosing the best encryption scheme and the suitability for blockchain based IoT system still remains vague.

Our contribution in the current study is to:

1. Provide a brief literature survey on encryption schemes used in IoT Blockchain.
2. Provide a comparison of encryption schemes with respect to its applicability in IoT blockchain.
3. Provide a use-case example to show how IBE can be used in IoT Blockchain.

6.2 **Encryption in IoT Blockchain**

In this section we provide description of commonly used encryption techniques in Blockchain based IoT.

6.2.1 **Identity Based Encryption**

Identity based Encryption (IBE) is a public key cryptosystem where public key is users' identity and private key is generated by a Private Key generation (PKG) center based on the master private key and users' ID. The concept of IBE was first presented by Shamir in 1984 (Shamir 1985). The idea was proposed to use users identity as a public key instead of a digital certificate. The identity can include user's email address, phone number, … etc. Initial works on IBE are proposed by Boneh and Franklin (Boneh & Franklin 2003) based on bilinear pairing and Cocks (Cocks 2001) based on quadratic residuosity problem.

An Identity based Encryption scheme is specified by four randomized algorithms: Setup, Extract, Encrypt and Decrypt (Boneh & Franklin 2003).

The initialization process consists of two parts. First calculate the public parameters, and select the master key s; according to the identity of each wireless node ID, calculate HASH and using the master key to generate the appropriate key K; transmit parameters and the node private key K to the wireless nodes, such that each node has its own public key and the associated parameters.

1. Setup: Takes a security parameter k and returns params (System parameters) and master-key. The system parameters includes a description finite message space Ḿ and finite cipher text space Ć. This process is done by the Private Key Generator (PKG).

2. Extract: Takes as input params, master-key and an arbitrary ID ∈ {0,1}* and returns a private key ‘d’. Here ID is an arbitrary string that is used as the public key of the client and ‘d’ is the corresponding private key. This process is also carried out by PKG, where it extracts the private key from the given public key.

3. Encrypt: Takes as input params, ID and M ∈ Ḿ and returns the cipher text C∈ Ç

4. Decrypt: Takes as input params, ID, C∈ Ç and a private key d. It returns plain text M ∈ Ḿ

6.2.2 **Homomorphic Encryption**

Homomorphic encryption (Gu 2015) is an encryption mechanism that allows computations on encrypted data, without decrypting it. There are basically three types of Homomorphic encryption: Fully Homomorphic Encryption (FHE), Partially Homomorphic Encryption (PHE) and Somewhat Homomorphic Encryption (SHE) (Shrestha & Kim 2019). In PHE, only one type of mathematical operation is allowed on the encrypted message, either addition or multiplication operation, with unlimited number of times. In SHE, both addition and multiplication operation is allowed but with only a limited number of times. In FHE, it allows a large number of different types of evaluation operations with unlimited number of times. Fully Homomorphic encryption (FHE) is a cryptographic primitive that facilitates arbitrary computations on encrypted data. Integrating IoT with blockchain and Homomorphic encryption can provide secured decentralized IoT system.

Homomorphic encryption gained wide attention due to the security vulnerabilities in cloud computing. When organizations started outsourcing the data to cloud providers, they encrypt the data and stored in the cloud to provide confidentiality. However, if any computation on stored data needs to be calculated, then the entire data should be decrypted. Homomorphic encryption can produce result to a user’s query without the need for decrypting the entire data.

The general Homomorphic Encryption is a set of four functions(Armknecht et al. 2015) as described below

H = {Key generation, Encryption, decryption, Evaluation}

Let Ć be a set of circuits. A Ć –evaluation scheme for Ć is a tuple of probabilistic polynomial–time algorithms (Gen, Enc, Eval, Dec) such that:

1. Key Generation: Gen(1^{λ}, α)- generates public key (p_k), secret key (s_k) and evaluation key (ev_k) where λ is security parameter and α is auxiliary input

   (s_k, p_k, ev_k)← KeyGen($)

2. Encryption: Enc(p_k,m) encrypts a message (m) with the public key (p_k) and outputs a ciphertext (cϵC),

   $c \leftarrow Enc_{pk}(m)$

3. Decryption: Dec(s_k, c) decrypts a ciphertexts with the secret key (s_k) and recovers message (m) as the output,

   $m \leftarrow Dec_{sk}(c)$

4. Evaluation : Eval(ev_k,C, c_1, c_1,…, c_n) produces evaluation output by taking 'ev_k' key as input, a circuit C ϵ Ć and tuple of input ciphertexts, i.e., $c_1 \ldots c_n$ and previous evaluation results,

   $c* \leftarrow Eval_{evk}(ev_k, C, c_1, c_2, \ldots c_n)$

6.2.3 **Proxy re-encryption**

Proxy re-encryption provides solution to the delegation of decryption rights (Nuñez, Agudo & Lopez 2017), (Ateniese et al. 2006). If Alice wants to delegate a message she received, which was encrypted with her public key, to Bob without sharing her private key, she will forward the message to a proxy that will convert the encrypted message to a form that could be decrypted by Bob. In this case, Alice doesn't have to share her private key with proxy or Bob and proxy will not have any knowledge about the plaintext. The technique is basically applied in email forwarding, law enforcement monitoring and

content distribution, secure file network storage and outsourced filtering of encrypted spam (Ateniese et al. 2006).

6.2.4 **Attribute Based Encryption**

Attribute Based Encryption (Sahai & Waters 2005) is a public key cryptography where the private key is based on a set of attributes. It is a variant of identity based encrpyion that allows users to encrypt the message without obtaining the public key certificate. This is comonly used in applications where there are muliple entities to decrypt the message. Any one with the correct set of attributes can decrypt the message.

6.3 **Related Work - Integration of Encryption schemes in IoT Blockchain**

In one of the pioneer work on IoT blockchain (Dorri, Kanhere, et al. 2017) Authors proposed symmetric encryption for providing confidentiality of data. Even though, symmetric encryption is fast and needs less computations when compared with asymmetric encryption, sharing the secrete key is a trivial task. Asymmetric encryptions require Public Key Infrastructure (PKI) for certificate management which can be avoided by using Identity Based Encryption (IBE). IBE has been applied in variety of domains including IoT, healthcare and cloud computing. Author in (Sankaran 2016) proposed IBE for hierarchical topology of IoT that can massively scale. Applications of IBE in IoT blockchain are very few(Polyzos & Fotiou n.d.). However attribute based encryption is used in many papers on IoT blockchain. Sahai and Waters (Gentry et al. 2013) proposed attributed based encryption, where data owner can decide the access policy of cipher text. IoT devices has to encrypt the messages using the attributes of the decryptors'. Authors in (Rahulamathavan et al. 2018) propose attribute based encryption (ABE) to provide privacy and confidentiality of data in IoT blockchain. Their architecture include IoT sensors that are connected to cluster head, where cluster head is assumed to be more powerful in terms of computation when compared to sensors. It also include a set of miners for validating the transactions and attribute authorities for providing ABE. They provide credentials to users and miners. The data generated by the sensors is encrypted by cluster head which will be verified by

miners, who have the right attribute. One of the disadvantages of their attribute of based encryption is; if the attributes are large then there will be only less number of miners. They provide use-case in Healthcare where attributes will be “Doctors”, “Nurses” etc. Their scheme is resistant to Sybil attack. IoT blockchain with Homomorphic encryption is considered to have synergic effect. A novel blockchain based IoT system with Homomorphic encryption is proposed by Zhou et al. (Zhou, Wang, Sun, et al. 2018)(Zhou, Wang, Ai, et al. 2018). Servers store the encrypted data from IoT and perform computations on encrypted data without decrypting it. Hence, servers cannot learn anything from encrypted data. Proxy Re-encryption (PRE) scheme to secure transfer of sensor data to the user is proposed in (Manzoor et al. 2018). Their analysis verify that combining proxy encryption scheme with blockchain enables a secure platform for trading and sharing sensor data. In (Eltayieb et al. 2019), authors used certificate-less proxy re-encryption scheme. Other encryption schemes that are used in IoT Blockchain are searchable encryption(Tahir & Rajarajan 2018) and Pseudo-anonym based encryption(Badr, Gomaa & Abd-Elrahman 2018). Table 6.1 provides a summary of encryption schemes used in blockchain based IoT architectures.

Table 6.1:Encryption schemes used in IoT Blockchain

Paper Name	**Type of Encryption Used**	**Ref.**
Towards an Optimized BlockChain for IoT	Symmetric Encryption	(Dorri, Ali and Kanhere, Salil S and Jurdak 2017)
A Decentralized Privacy-Preserving Healthcare Blockchain for IoT	Symmetric encryption (ARX ciphers) for data and Asymmetric encryption (Diffie- Hellman Key exchange) for secret key	(Dwivedi et al. 2019)
BeeKeeper: A Blockchain-Based IoT System with Secure Storage and Homomorphic Computation	Homomorphic Encryption	(Zhou, Wang, Sun, et al. 2018)(Zhou, Wang, Ai, et al. 2018)

Proxy Re-Encryption Scheme for Secure IoT Data Sharing	Certificate Based Proxy Re- Encryption (CB-PRE)	(Manzoor et al. 2018)
A Secured Proxy-Based Data Sharing Module in IoT Environments Using Blockchain	Proxy Re-Encryption (PRE) scheme based on Attribute Based Encryption	(Agyekum et al. 2019)
FogBus: A Blockchain-based Lightweight Framework for Edge and Fog Computing	Attribute based encryption	(Tuli et al. 2018)
Multi-tier Blockchain Framework for IoT-EHRs Systems	Pseudo-anonym based encryption	(Badr, Gomaa & Abd-Elrahman 2018)

The use-case that we propose is about traffic speed radars with IoT devices that measure the speed of vehicles and generate fines. Currently, this system uses a centralized architecture, which has several drawbacks including single point of failure, target of attack and have to trust the central authority who is maintaining the databases. Implementing this system using blockchain can solve the above mentioned problems. Currently the system works as follows: When a speed radar captures a vehicle for over-speeding, it records its Device ID, time and location and calculates its fine amount which is stored in a centralized database. In this Chapter, we provide a framework on how to provide data confidentiality in a blockchain based architecture for speed radars.

6.4 **System architecture using IBE**

We provide example on how IBE can be used to provide data confidentiality in IoT blockchain. For that we consider the example of a traffic speed radar that is implemented using blockchain. The detailed process is listed in Figure 6.1.

Step1: When a device is sensed for over-speeding, its vehicle ID is scanned which will be used as the public key.

Step 2: Speed radar generates the transaction = E(fine amount, time, location)Pk

Step 3: The data is sent to edge nodes. A set of edge nodes acts as miners which verify the signature of the transaction and add it to the blockchain. Miners can be selected using round-robin or cluster head selection process.

Step 4: When the vehicle owner wants to know the fine details, it request PKG, which will be maintained by the traffic authority. PKG can provide the secret key for the vehicle and; hence, data can be decrypted. As the traffic authority also has the secret key, they can also prove the validity of the message

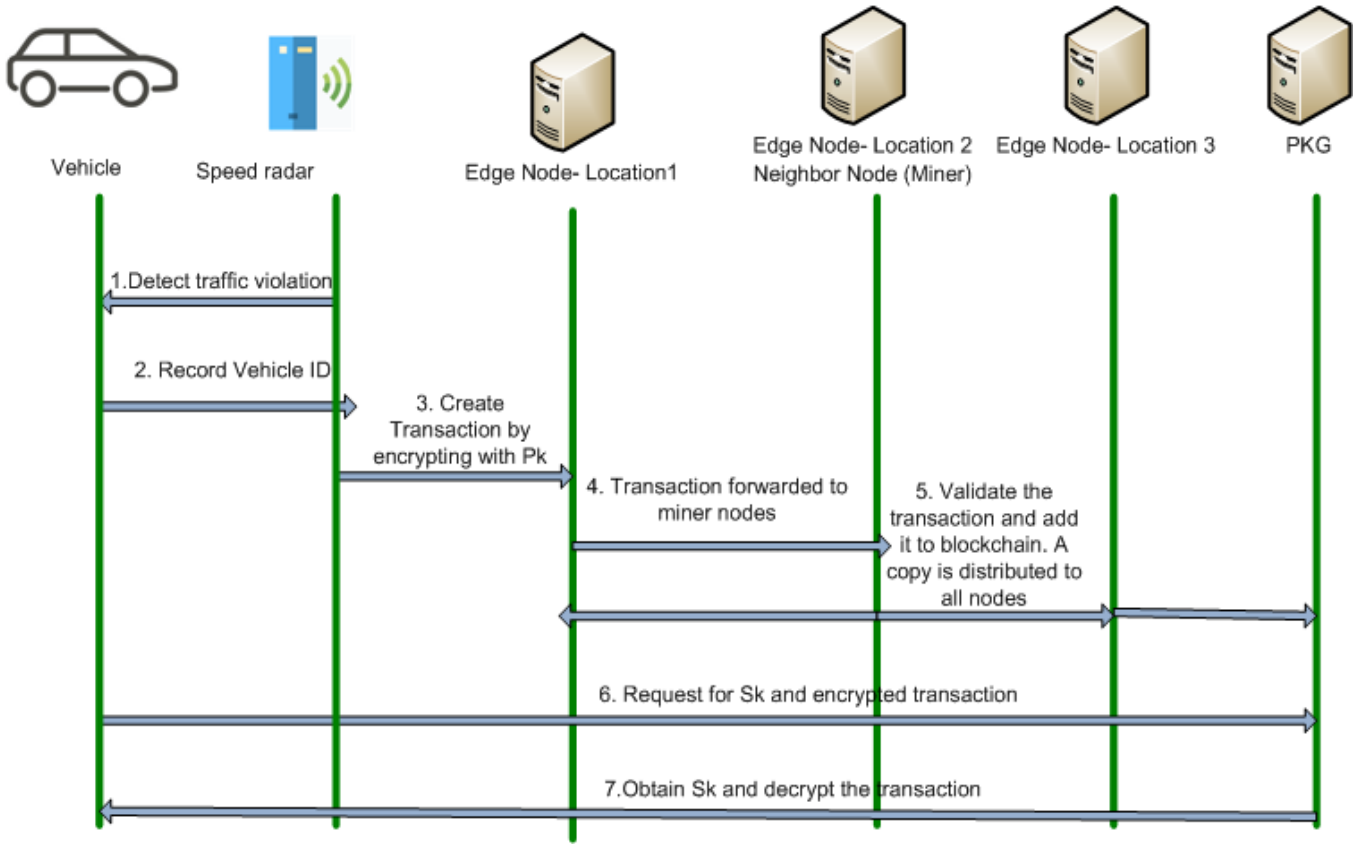


Figure 6.1:Integrating IBE with blockchain process for traffic speed radars

6.5 **Research Methodology**

6.5.1 **Implementation and Performance Evaluation**

We implemented Identify based encryption based on RSA approach. The parameters we passed are, The public key of the Vehicle, recorded time of speed violation, location, and the calculated fine amount. Figure 6.2 shows the parameters and the encrypted message. The screenshot of the IBE implementation is provided in Figure 6.3. We then passed this encrypted message along with the signed hash of the message to IoT devices simulated using Contiki OS, and calculated the average throughput. We simulated a network with four edge devices, where each edge device is connected to 5 IoT devices. More details of the experiment are provided in Chapter 8.

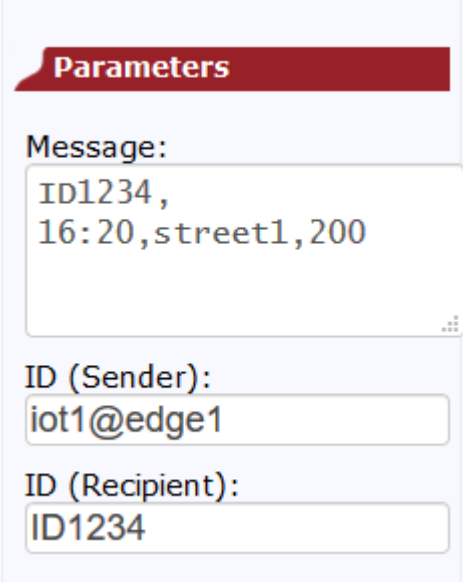


Figure 6.2: IBE parameters for IoT blockchain

```
ID (sender): iot1@edge1, ID (recipient): ID1234
Message: ID1234, 16:20,street1,200
Recipient Public Key: 551134583

Encrypted message:
41-91-2665-7343-59834-271759-118-24112-120-1006411-31-83-71-7667115-114511069-90104-12066-83792010746-9-11443-7511
6-105-9956-72-127052-50-12-51483390-41-103-40-9682-8268-406740-71-1165832-9044-12-47-9038-23-62-4557-852893-28-96-
10816-7132-2056-77-105-38464784116-63-6959-9-38-53-75-101-53-109-35-5-4945-569010520-128-81-16-107-10591-4932-5010
7-123-71-118-65-6498-21118-12-122-61117-57834066-11245114-6173-89-8-18-646594-41-100121-23-5061114-692-54116-6828-
17107187-60-1870724110936-10056-106-9863101-4512-85127-72-1166-7443-29-35-126-20-74-806-107-106-18-40-11-60-693347
2135851-111-59276692-14-4793535740-286197-171180-17-9096-127-48-5223-97-128-10494-83-11384996-107-1214-32-66127-69
958-1019

==== Server
id_sender (recipient): ID1234
Private Key is:
730020887441710548290869379896248597245273592758546778974979248372372605125309278908345576384425533918722241346370
851457822459900364151577447803540873660701821241826860002581493279375765153250824341560188509778279191745856540811
867038794582413422577671419274520540944163529829785777173201805638012681529530729148856991818731769775475427478101
705278555020600201636247352019317470267648579040725438486055441126923766207780583359609948366764009198490967157141
302911543238033186651158598346374748993693297848346752782684156317453055806270508820650801495507410979654293662214
27840444835787487000656422625670130502056142 47

Decrypted message:
ID1234 has sent this message:
ID1234, 16:20,street1,200
```

Figure 6.3: Screenshot of IBE Implementation for IoT blockchain

6.5.2 Security Analysis and Comparison of existing encryption techniques

IBE is proved to reduce computational complexity when compared with other Asymmetric encryption. Security mechanisms using IBE have shown lesser overhead than public key cryptography due to the reduced key size. Although being an asymmetric encryption, IBE based on bilinear pairing reduces the computational complexity which can be adopted into resource constrained IoTs.

Confidentiality: The sensor data is securely distributed to the database, where only the vehicle owner or the traffic authority can decrypt it. Consequently, no outsiders or cloud provides have access to the data.

Integrity: The content of the data cannot be modified as, every transaction is hashed and stored as a block, which is linked to the previous block in the blockchain.

Authentication: Speed sensors sign the message using its private key, which could be verified by miners within the edge node. However, we have not considered the certificate management process of public key which is out of scope of this thesis.

Availability: The ledger is shared and synchronized in real time with all nodes in the network which provides availability.

Attacks: The system is resistant to well know security attacks like Man-in-the-middle, Sybil attack and replay attack.

Table 6.2:Comparison of Encryption schemes for IoT Blockchain

Encryption Type	Advantages	Disadvantages
		• Extremely slow and computationally expensive

Encryption Type	Advantages	Disadvantages
Homomorphic Encryption	• Can derive intelligence form sensitive data without decrypting the entire data • Miners can verify the transaction without decrypting the entire data	• Fully Homomorphic Encryption is currently impractical to use in real world applications • Increases cost, as it need specialized client server application to run • Cipher text generated is much larger than the plain text. Adding large data to blockchain is not efficient, as this data will be replicated to all nodes in the network which creates additional storage overhead
Identity Based Encryption	• No Need to validate a certificate to send an encrypted message. No Need for a PKI, instead use a PKG.	• Requires a secure channel between sender and receiver to share the secret key • Requires a centralized server for Key generation. • Key revocation: If the private key is lost or stolen, the receiver will have to change the public key which is basically email id or phone number
Attribute Based Encryption	• It provides a secure access control mechanism. Blockchain can utilize it by encrypting with keywords like “miners”, “partners” etc	• The IoT device will have to use every authorized user’s public key to encrypt data. It has high computational overhead

6.6 **Chapter Conclusion**

IoT has high potential if integrated with blockchain and the right security and consensus mechanisms. Choosing the right encryption scheme can reduce the computational overhead of IoT, reduce cost of deployment and provide adequate security. The type of encryption scheme should be selected based on the sensitivity of data. Homomorphic encryption is still in its developing stage, it has high potential and can solve data confidentiality issues in blockchain based IoT system. Miners can

verify the transactions by making computations on encrypted data without fully decrypting it. We provided the use-case of traffic speed radar using identity based encryption. When compared with the traditional centralized architecture for speed radars, the blockchain based system provides data confidentiality, integrity and availability in a more efficient way. Hence this chapter answers research question 4.

Chapter 7 : AN OPTIMAL CONSENSUS NODE SELECTION PROCESS FOR IOT BLOCKCHAIN

Blockchain and Internet of Things (IoT) are two trending technologies, when combined together can strengthen the security of various applications. However, security of blockchain depend on its consensus mechanism. The consensus mechanisms used by cryptocurrencies requires high computations and hence cannot be applied to IoT. Applying lightweight consensus, such as PBFT (Practical Byzantine Fault Tolerance) requires an authority or a protocol to select leader nodes and the nodes to be involved in consensus. However, the current node selection process in many blockchain applications involves a central authority or is based on traditional round robin techniques. Hence, we propose a simple and efficient node selection mechanism that can perform consensus without wasting energy. Our approach uses PBFT, where the nodes to participate in the consensus are not predetermined by a central authority. Nodes are selected based on their performance in the blockchain.

7.1 Introduction

Blockchain has recently gained attention due to its adoption into a variety of applications. The problem of trusting a third party in the traditional payment system has been solved using blockchain. Apart from cryptocurrencies, blockchain can be applied to other applications including Internet of Things (IoT) (Dorri, Ali and Kanhere, Salil S and Jurdak 2017). In contrast to the traditional way of a central authority maintaining the ledger of all transactions, blockchain distributes the ledger to every node in the network. The verification and validation of these transactions are conducted through cryptographic techniques by a set of miners. Digital signatures are used for verifying the authenticity of transactions and transaction integrity is maintained through hash functions. Blockchain platforms can be classified into permissionless and permissioned (Baliga 2017). Any node in a permissionless blockchain can conduct a transaction as well as take part in consensus process, e.g. bitcoin. The method of validating the transactions in the blockchain without relying on a central authority is called

consensus. Due to the open ended nature of permissionless platform, achieving consensus is extremely complicated, e.g. Bitcoin's Proof of Work (PoW). It requires high computations resulting in wastage of energy. Studies prove that the electricity wasted in bitcoin mining is comparable to the average electricity consumption of Ireland (O'Dwyert & Malone 2014). In permissioned blockchain, adding node to the network and nodes that participate in consensus are decided by central authority. Nodes participating in consensus are considered to be trusted nodes that are verified by an authority. Here high computation PoW are not required and alternative consensus mechanisms are used. Some examples are Paxos, RAFT and variants of Byzantines Fault tolerance algorithms (Baliga 2017).

Poorly designed consensus mechanisms can interrupt the operation of whole blockchain. For example the Neo network was halted due to node failure. One of the consensus node was disconnected during consensus process and all other nodes were waiting for this node to create the block(Buntinx 2018). It caused delay of several hours until the neo authorities had to forcefully restart all the nodes. Problems in consensus can also cause the blockchain to fork or can give the blockchain control to few users, such as the case of the 51% attack (Kroll, Davey & Felten 2013).

IoT devices are low power devices that have minimal computational resources. Consensus like bitcoins PoW cannot be used in IoT blockchain. Moreover IoT are usually implemented in permissioned platform. Implementing IoT in a distributed nature faces lot of challenges including the additional storage and computations required by IoT. Currently IoT architecture is cloud based where a central cloud provider manages and maintains the database. Such type of architecture is subject to single point of failure and is always a target of attack. Also a cloud manager or a central authority may modify the data generated by the sensor. Implementing blockchain for IoT can overcome these problems. It can provide data integrity, availability and trust. Considering these advantages, IoT can benefit massively from blockchain. But implementing blockchain for IoT poses many challenges due to the low computational capacity of these devices. Hence implementing blockchain on IoT needs a blockchain protocol that can adjust with the limitations of IoT.

Achieving consensus in a blockchain relates to solving an old problem called the Byzantines General problem(Lamport, Shostak & Pease 1982). Consensus node selection is an important process in blockchain. Bitcoins PoW (Nakamoto 2008), Ethreum's Proof of Stake (PoS) and Byzantines fault tolerance (Moore 2016) are all different solutions for Byzantines general problem. While bitcoin select miners who can solve a cryptographic puzzle for the block creation, PoS selects block creators pseudo randomly. However variants of Byzantines fault tolerance techniques use round robin technique or a central authority for node selection. In this Chapter, we provide simple and efficient consensus node selection that can be used to implement blockchain for IoT. Our approach is to identify the nodes that are less busy and use it as consensus nodes.

In section 6.2, we briefly describe various consensus mechanism used in blockchain. In section 6.3 we introduce blockchain for IoT considering a case study of speed radar generating traffic fines. In section 6.4, we briefly describe our approach for achieving consensus in speed radar for generating traffic fine.

7.2 **Consensus Algorithms**

Centralized banks maintain ledgers to validate a transaction. If Alice sends 100 USD to Bob, bank checks the ledger to verify if Alice has enough amount in her account to transfer to Bob. If there is enough amount, then they update the ledger by reducing 100 USD from Alice's account and adding 100 USD to Bob's account. In a decentralized environment where there is no central authority to keep the ledger, this process is done through consensus mechanisms that allows secure updating of distributed shared state. Cryptocurrencies powered by blockchain uses decentralized environment, where each ledger is distributed among all nodes in the network. The process of validating the transactions and adding it to the ledger are done by nodes in the network. But how do we trust these nodes? What if some validating nodes are malicious? They may be trying to perform double spending or trying to discard some transactions. Such type of problems can be considered as Byzantines generals Problem(Lamport, Shostak & Pease 1982). A byzantine node can mislead other nodes involved in the

consensus mechanism. Hence the consensus mechanism should be able to operate correctly and reach consensus even in the presence of byzantine nodes.

Consensus in the literal terms means agreement. Seibold and Samman (Seibold S and Samman G 2018)define consensus mechanism as a method of authenticating or validating a value or transaction on a Blockchain or a distributed ledger without the need to trust or rely on a central authority. In a distributed or decentralized network, for nodes to reach to a common agreement, consensus algorithms are used (Sankar, Sindhu & Sethumadhavan 2017). Blockchain platforms use a range of consensus model which are built on Byzantines Fault tolerance. This includes PoW (Proof of work), PBFT (Practical Byzantine Fault Tolerance), Proof of Stake (PoS). Most of the cryptocurrencies use a variant or hybrid of these consensus mechanisms. Consensus mechanisms used by some cryptocurrencies are shown in Table 7.1.

Table 7.1:Consensus Used in Cryptocurrencies

Crypto currency	**Consensus used**
Bitcoin	Proof Of work
Ethreum	Proof Of Stake
XRP	Ripple consensus Algorithm
Bitcoin Cash	Proof Of work
EOS	BFT-DPOS
Litecoin	Proof of work
Cardano	Ouroboros – Proof of stake
Steller	Federated BFT

7.2.1 **Proof of Work**

Proof of work introduced by bitcoin (Nakamoto 2008) is the widest deployed consensus mechanism in existing blockchains. Proof of work is a cryptographic puzzle that is difficult to solve but easy to

verify. Bitcoin uses hashcash proof of work system. Hashcash was proposed by Adam Back(Back 2002) to limit sending spam emails. To send an email, user has to compute a hash value which requires some amount of processing power. The receiver can verify this with a single hash. For a normal user, sending one or two emails does not take much time or computations. However, it delays a spammer who wants to send 100,000 emails per minute. This technology is used in bitcoin mining to compute the hash functions. Miners calculate the hash of the header to find a value less than the given target. The fields in the bitcoin header are as follows.

The first field in the block header is the protocol version. The next field is the hash of the previous block, and then is Merkely root, which is a special hash of all transactions in the block. It is followed by timestamp of the block and mining difficulty value bits. Finally nonce is an arbitrary value that is incremented on each hash attempt to provide a new hash value. To successfully mine a block the following steps are performed.

Header = Version +Previous block header hash + Merkle root hash + timestamp +nBits+ nonce

Hash = SHA-256(SHA-256(Header)

The miners increment the nonce value to find a hash value that is less than a target. Target is a 256-bit number that is shared by all bitcoin clients. To successfully mine a block, the SHA-256 hash of block's header must be lower than or equal to the current target for the block. The lower the target, it is more difficult to generate a block. Difficulty is a measure of how difficult it is to find a hash below a given target. This indicates how much work has to be done by the node to find a hash that is smaller than the target. For example, if the difficulty is set to 4 then the target should have 4 leading zeros. Time taken to generate a block is set to 10 minutes. The network difficulty changes every 2016 blocks. 2016 block would take exactly two weeks to find. Difficulty increases if the time taken to generate the last 2016 blocks is less than 2 weeks or it decreases if the time taken to generate last 2016 bocks is greater

than 2 weeks (bitcoin 2018). Due to these properties, bitcoin's proof of work requires huge number of hash calculation that consumes high energy.

When two miners simultaneously mine a block of transaction, blockchain will fork leading to more than one chain. Bitcoin uses longest chain rule. In case of a fork the longest chain will be the valid chain.

7.2.2 **Proof of Stake**

Proof of Stake algorithms are designed to overcome the disadvantages of PoW in terms of high electricity consumption involved in mining operations. Instead of buying mining equipment and paying electricity cost, the user can spend that amount buying cryptocurrency and use it as a stake to buy proportionate block creation chances in the blockchain system (Baliga 2017). Mining in bitcoin's PoW is for creating a new block whereas Proof of stake isn't about mining, it is about validating. Block creators are selected pseudo-randomly thereby ensuring that no validator can predict its turn in advance. Ouroboros-Proof of Stake, delegated Proof of Stake different variants of proof of Stake algorithms (Kiayias et al. 2017),(Larimer 2014).

7.2.3 **PBFT (Practical Byzantine Fault tolerance)**

A solution to Byzantine generals' problem is PBFT. Permissioned blockchain platforms mainly use PBFT. An example is hyperledger fabric by IBM that uses hyperledger blockchain platform(Github 2018). When a client sends a request to the leader node, it distributes it to other nodes. Each node maintains an internal state. When a transaction is received, each node uses their internal state and run computations to validate a transaction. This computation will lead to party's decision about transaction. This will be shared to all other nodes in the blockchain. The final decision is based on the total decision from all parties. When enough responses are reached, a transaction is verified to be a valid transaction. In XRP cryptocurrency, there should be a recommended validator list. The validators are preselected by the Ripple foundation (Schwartz, Youngs & Britto 2014) whereas in Steller it uses Federated BFT where there is no recommended validator list chosen by central authority. Whereas each validator

decides which other validator they trust. Their list of trusted validators are called quorum slice. The new validators can then choose their set of quorum slice (Mazieres & Mazières 2015).

Some of the attacks on the consensus protocols are 51% attack (Kroll, Davey & Felten 2013), stake-grinding-attack and nothing-at-stake attack (ethereumwiki 2018).

7.3 **Consensus for Traffic Spped Radar**

In this section we will consider the example of a speed radar system that generates traffic fines. The centralized architecture of this system is shown in Figure 7.1. This architecture uses a trusted third party. The basic workflow is summarized as follows.

Radar acts as the sensor that senses the speed of the moving vehicle. When a moving vehicle is detected for over speeding, the vehicles license plate number, date and time, location are captured and send to the centralized database. This information is processed and fines are generated. A radar with a camera acts as the sensor device here. Speed of the moving object is measured using Doppler Effect. The radar transmits radio signal which bounces back after hitting the moving vehicle. There will be difference in the frequency when the object is approaching the radar and when it is moving away from radar. This difference in frequency is used to calculate the speed of the moving vehicle. The speed is given by (1) (Radar Gun Wiki n.d.)

$$v = \frac{\Delta f}{f}\frac{c}{2} \qquad \text{----------(1)}$$

Where f is the emitted frequency of radio waves, Δf is the difference in frequency transmitted and then received back and c is the speed of light. An image recorder attached to the sensor captures the vehicle license plate and send it to the middleware layer through the network layer. Middleware layer consisting of database and servers process this information and generate fine for the vehicle owner.

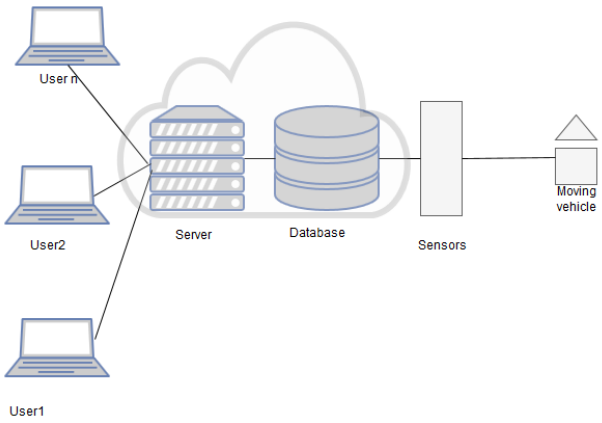


Figure 7.1: Centralized architecture for Speed radars

Some of the problems associated with this architecture are

- Single Point of failure: Cloud vendors often offer additional hardware and software to provide high reliability at an additional cost. But still if the cloud service is managed by a single company, this in-fact is a single point of failure. They might have different datacenters in different locations but the underlying infrastructure and software used will be the same. Also the cloud provider may go out of business. All these cases creates single point of failure in cloud architecture (Singh et al. 2018).
- Trust: In cloud model we trust the authorities maintaining the cloud. The administrators who have access to the database can manipulate the data. An internal employee can manipulate the data generated by the sensor. For example in the city of Flint, Michigan IoT devices were used to measure the quality of water. The authorities were insisting that the water in the city was safe to drink but it was later revealed that officials would have manipulated two of the collected samples that were contaminated (Swan 2015).
- Target of attack: The server where the data resides is always a target of attack for the attackers. A Denial of service attack on the server can make the data unavailable.

7.4 **Proposed Approach**

To implement blockchain on IoT device itself, the device should be capable to perform basic cryptographic functions, perform consensus, should be able to store growing record of transactions and should be able to communicate between devices. Otherwise for a light node, that is capable of only sensing and basic operations, edge nodes or cloud based nodes should be used. We assume our nodes are of the first type and are implemented as a permissioned blockchain, but nodes belong to different entities.

Implementing Blockchain for traffic fine system can overcome the challenges with the cloud architecture. Since the data is distributed to all nodes there is no single point of failure in blockchain. Similarly If an attacker wants to manipulate the data, he/she will have to manipulate blocks preceding the block. An internal employee within an entity will not be able to manipulate the data. In blockchain, each IoT device will be connected to all other devices in the network as shown inFigure 7.2. The IoT device along with its sensors and storage is called a full node.

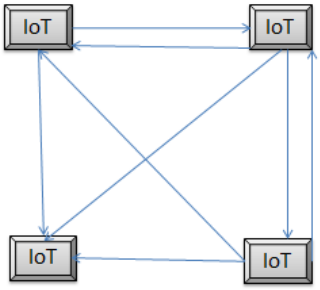


Figure 7.2:Distributed Nature of blockchain

When the sensor detects a traffic violation, it records the plate number, time and date, location and generates the fine. The details such as license plate number, time and date, location will be replicated to all nodes in the blockchain as shown in Figure 7.3.

To validate a transaction and add it to the block, PBFT can be used. But the problem with PBTF is that, a central authority has to provide the list of nodes that can participate in consensus. But this

approach will not be efficient, as some IoT nodes will be busier than others and the nodes belongs to different entities. Hence to manage efficiently, our approach is to identify the node that is less active, to participate in the consensus process.

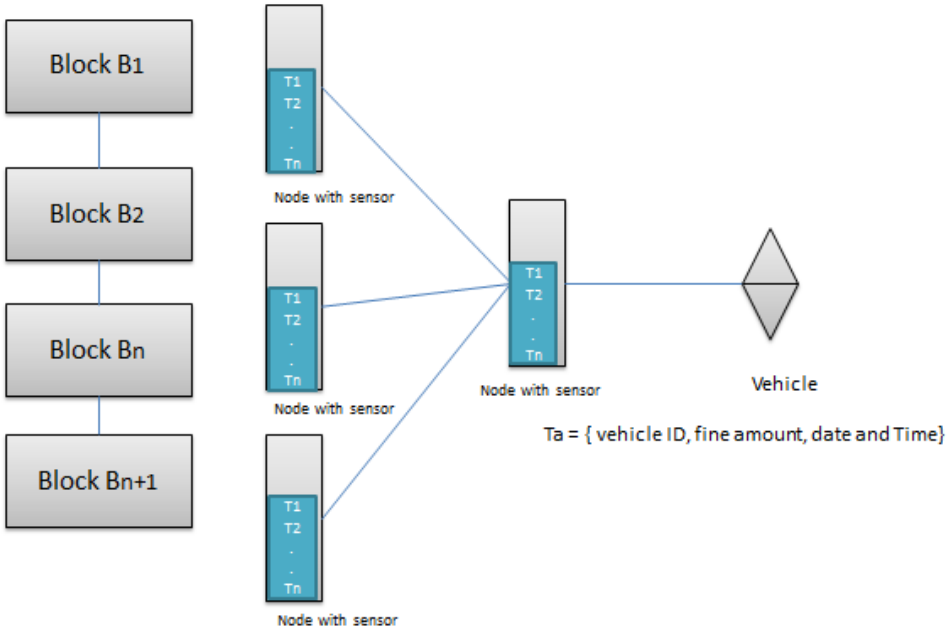


Figure 7.3: A generic blockchain architecture for traffic speed radars

For example when a sensor detects a traffic violation, it creates a transaction and distribute it to all other nodes in the network.

Let trans = $\{t_1, t_2, t_3 \ldots t_n\}$ where t_n contains {vID, td, loc, amt}.

trans= transaction set ;t_n= n^{th} transaction; vID = vehicle ID;

td = time and date; loc = Location; amt = Fine amount.

Every node collects all 'trans' generated by other nodes. When the number of 'trans' reaches a threshold 'r', it checks for the nodes that has the least number of transactions within a time frame. 'n' number of nodes with the lowest number of transactions will be selected for validating the transactions

and the least busy node will be selected for creating the block. This node creates the block and links it with the previous block.

Consider the example of a 3 node network as shown in table 2. Let $t_1, t_2, .. t_{na}$ be the transactions generated by Node A. If $t_{na} < t_{nb}$ and $t_{na} < t_{nc}$ then Node A is selected for participating in consensus. This indicates that Node A was less busy than other nodes. Hence the ideal candidate to choose for consensus is Node A.

Table 7.2: Transaction in three node network

Node **A**	t1	Node **B**	t1	Node **C**	t1
	t2		t2		t2
	.		.		.
	t_{na}		t_{nb}		t_{nc}

When the number of transactions reaches a threshold, the consensus node selection algorithm can be used. Algorithm takes input each node and the transactions generated by the node. It counts the number of transactions generated by all nodes. If the block is not created within a particular time, the node with the next least number of transactions will be elected for consensus. In case of a fork, the longest chain rule will be used.

Algorithm1: Consensus node selection

Input: node, transaction

Output: consensus node

1. **Struct** consensus

```
2. {Node; Transaction; }
3. Cons_Node = 0; Prev=0;
4. For i= each node in the network
5.      count= count (transactions in 'i' )
6.     If count< prev
7.        Prev = count;
8.        Cons_Node =i
9.    endif
10. end for
11. Return Cons_node
```

Applying this consensus for IoT can move the additional computational overhead of blockchain to nodes that are less active. Hence the normal routine of other IoT devices will not be interrupted and at the same time it provides all the advantages of blockchain technology. However to implement blockchain, IoT devices should be equipped with additional storage.

7.5 **Chapter Conclusion**

Blockchain has shown its potential for transforming traditional industry with its key characteristics: decentralization, persistency, anonymity and transparency. In this Chapter, we have shown how the potential of blockchain can be applied to IoT by implementing an efficient consensus node selection process that can be used by speed radars for generating traffic fines. Using this system, validating the transactions and creating blocks are done by nodes that are less active than other nodes. Hence without interrupting the normal service of the sensors, blockchain technology can be embedded into IoT. Hence this Chapter answers research question 3.

Chapter 8 : RESULTS AND DISCUSSION

In this Chapter, we will present the details of the implementation and evaluation methodology we used to evaluate the proposed system.

8.1 Introduction

This section clarifies the results of the experiments conducted by the research. We used three approaches to measure the performance of the proposed architecture. The first approach is to measure the system performance to identify the performance of the system during the blockchain process. This is to identify the computational requirements of the proposed approach. For this, we measured the system disk read/write data, memory performance and ethereum gas measurement. The second approach is to identify the network performance. For this, we simulated the architecture and measured the throughput and latency. The third approach is to identify the security of the proposed system. For this, we performed a security analysis and calculated the results with the threat modelling tool to support our findings.

8.2 Measuring System performance – First Approach

We implemented our blockchain solution on a real network using 2 Raspberry Pi4 - Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz with 4GB SDRAM and Lenovo T440 Desktop machine Intel(R) Core (TM) i5-4300M CPU @ 2.60GHz with four Ubuntu 16.04 LTS running on VMware. This type of implementation and evaluation is being used in various literatures (Tuli et al. 2018)(Xu et al. 2018)(Hang & Kim 2019) (Novo 2018)(Ouaddah, Abou Elkalam & Ait Ouahman 2016)(Lunardi et al. 2018)(Magnusson 2018)(Cha et al. 2018). The implementation setup is shown in Figure 8.1 . We used two RaspberryPi and connected sensors to it. We installed Ethereum light node in the Raspberry Pi. The difference between Ethereum light node and a full node is that a full node can perform mining whereas a light node cannot perform. For our architecture, the IoT device does not perform mining.

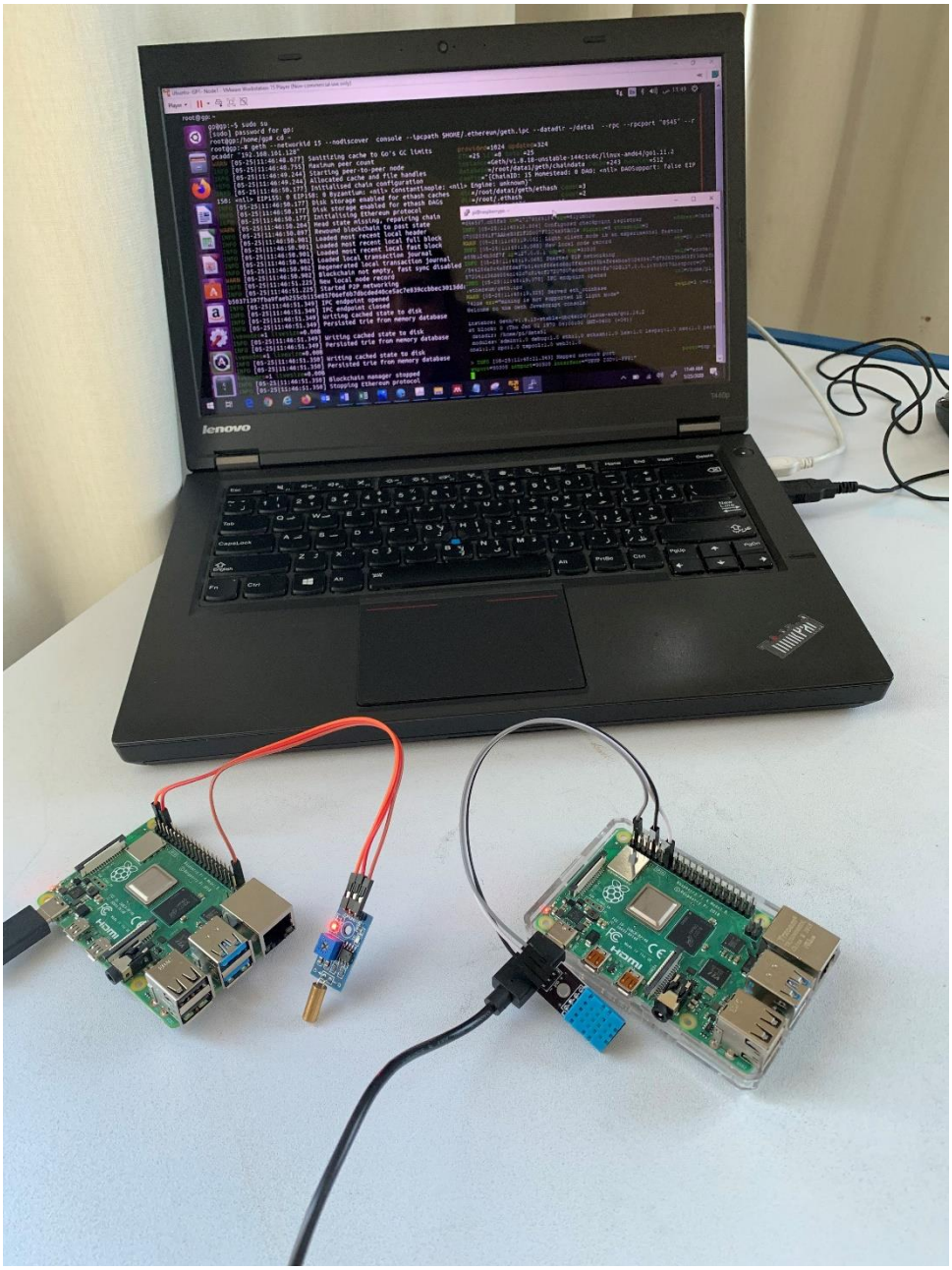


Figure 8.1: Experiment Setup

We installed geth in raspberry Pi and configured it using the same genesis file as our 4 node blockchain. The geth command we ran on raspberry pi is given below and it started listening as shown in Figure 8.2.

Geth - -datadir =~/data51 - -network =15 - - port =30308 - - nodiscover –console = - - ipcpath /home/pi/.ethereum/geth.ipc - - synmode light

We used 15 for the network ID and used port 30308 for the raspberrypi.

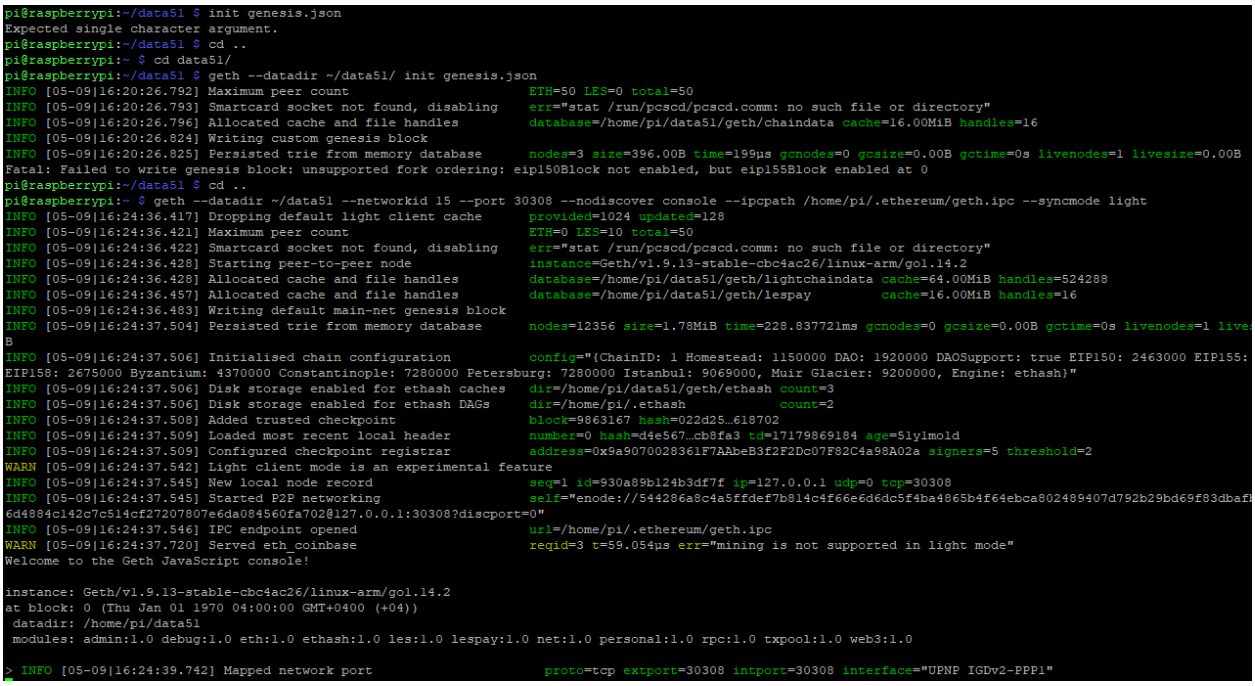


Figure 8.2:Runing Geth light node in Raspberry Pi

To create the smart contract, we used solidity compiler. The smart contract code was written in solidity and generated the output as ABI. The solidity compiler and the ABI output is given in Figure 8.3 and Figure 8.4.

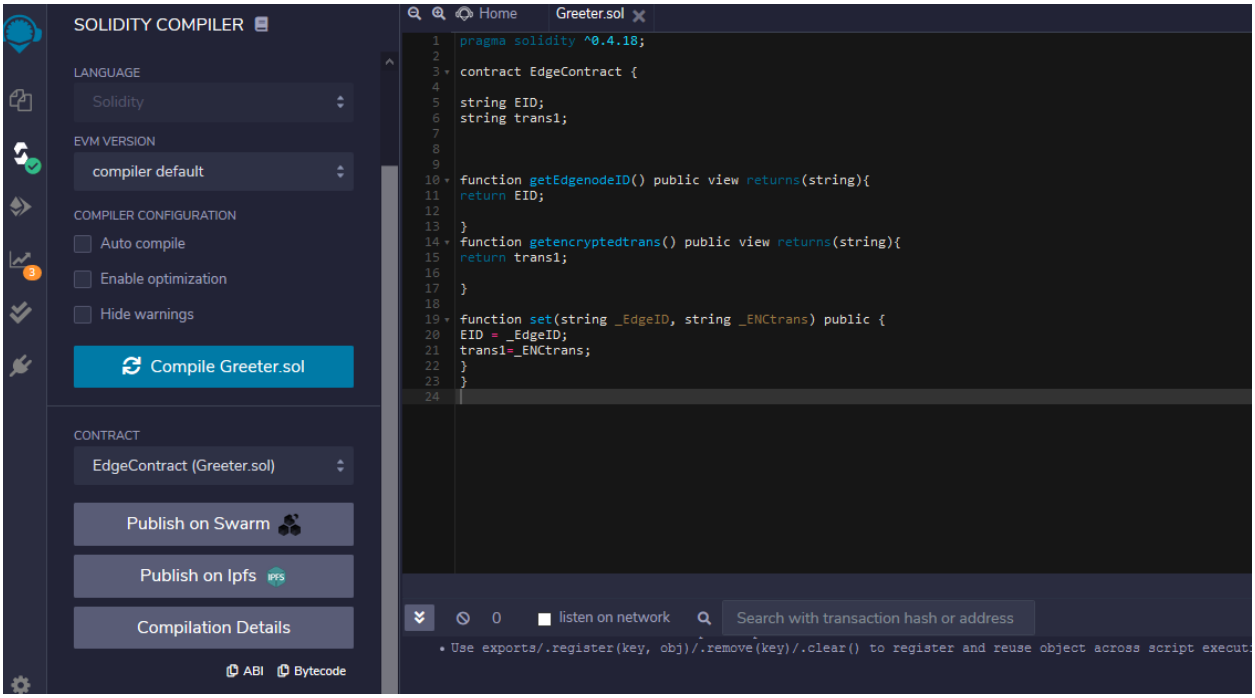


Figure 8.3: Solidity Compiler

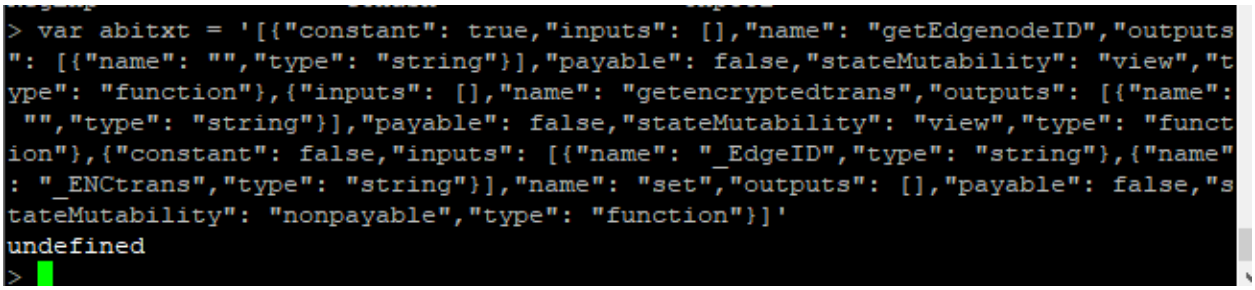


Figure 8.4:ABI code generated

We used Four Ubuntu machines to create a blockchain. The geth command used on all the 4 nodes are as follows.

geth --networkid 15 --nodiscover --datadir ~/gethDatadir --maxpeers=0 --ipcpath $HOME/.ethereum/geth.ipc

Mining was enabled on all 4 nodes. Figure 8.5 shows the ethereum network with 4 connected nodes and mining process is enabled on all nodes.

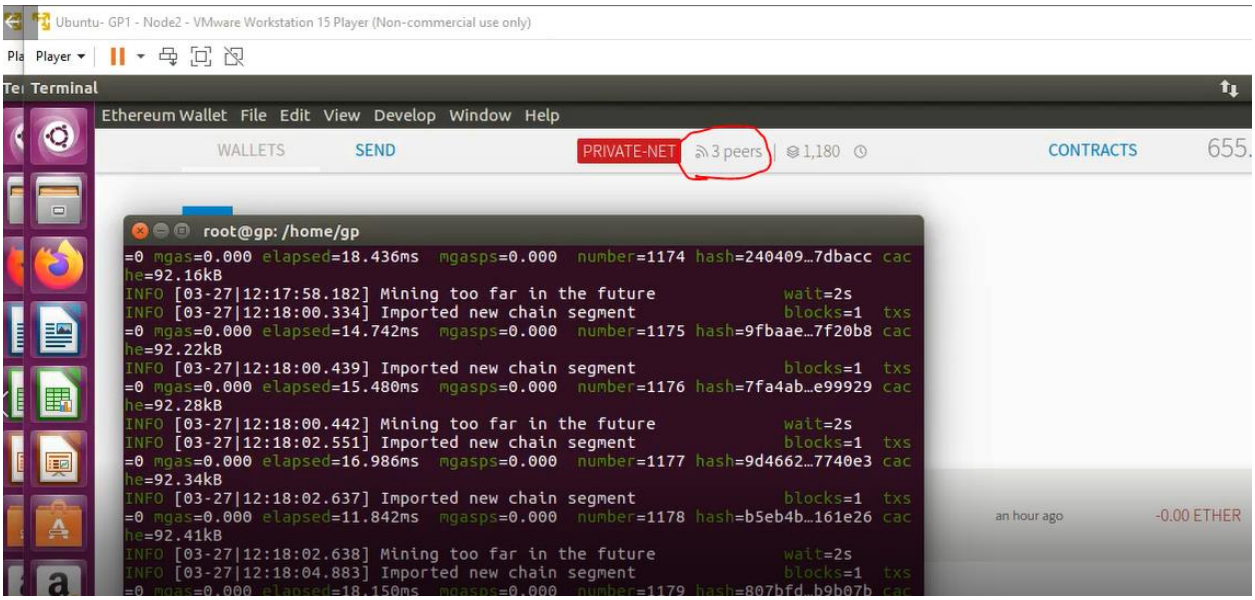


Figure 8.5: Ethereum network with 4 connected nodes

8.2.1 **Evaluation Metric- First Approach**

To evaluate the performance of the proposed approach, we measured the system performance, by using geth metrices. Geth has a logging system that is capable of creating leveled log entries which are helpful while debugging to see exactly what the system is doing while running the blockchain. Geth metrics is similar to logs, we should add arbitrary metric collection to any part of the code. Adding the metrics will collect the data automatically, surfaced through the APIs, queryable and visualizable for analysis. There are two types of metrics implemented in Geth.

Meters: The rate at which the amount of things that pass through are measured using meters. It counts arbitrary events instead of a specific unit of measure. It reports the total number of events that passed through the meter, Mean throughput rate of the meter and weighted throughput rate in the last 1,5 and 15 minutes(events/ second)

Timers: It measure arbitrary of events but in a particular duration. It is an extension of meters, where duration is collected not just the occurrence of events.

meter := metrics.NewMeter("system/memory/allocs")

timer := metrics.NewTimer("chain/inserts")

A geth time metric has additional reporting in perentiles *(5, 20, 50, 80, 95)*, that some percentage of the events took less than the reported time to execute (e.g. Percentile 20 = 1.5s would mean that 20% of the measured events took less time than 1.5 seconds to execute; inherently 80%(=100%-20%) took more that 1.5s)

Geth automatically exposes all collected metrics in the debug RPC API, through the metrics method, hence these can be queried simply from the console in:

```
root@gp:~# geth attach http://127.0.0.1:8545 --exec
"debug.metrics(false)"
WARN [05-11|12:26:35.306] Sanitizing cache to Go's
GC limits       provided=1024 updated=324
{
  chain: {
    inserts: {
      Avg01Min: "44 (0.73/s)",
      Avg05Min: "241 (0.80/s)",
      Avg15Min: "319 (0.35/s)",
      Maximum: "52.767336ms",
      Minimum: "135.169µs",
      Overall: "368 (2.63/s)",
      Percentiles: {
        20: "172.11µs",
        5: "154.72µs",
```

```
      50: "204.664µs",
      80: "349.491µs",
      95: "1.626236ms"
    }
  }
},
```


Using the data generated by geth metrices, we calculated the average Disk utilization and memory utilization.

Evaluation parameters

Disk Read Data - Displays the average size in bytes of read operations on the volume. This counter can be used in conjunction with Avg. Disk Sec/Read to determine the average read throughput of the volume. This counter can be used to form a baseline of storage usage.

Disk Write data- Displays the average size in bytes of write operations on the volume. This counter can be used in conjunction with Avg. Disk Sec/write to determine the average write throughput of the volume.

Disk Read Count – Number of disk read commands completed on each disk on the host, per second. It indicates the total number of read operations completed successfully.

Disk write count - Number of write operations completed successfully. Number of disk write commands completed on each disk on the host, per second.

Memory usage- Amount of memory currently being used. An Average of 1 min, 5 min and 15 minute was collected.

Figure 8.6 give the command we used for generating the disk and memory utilization. It shows how the data is collected through geth metrics. An example of the disk utilization data generated by the geth metrics in given in Figure 8.7 and an example of the memory utilization data is given in Table 8.1.

Figure 8.6: Collecting data through geth metrics

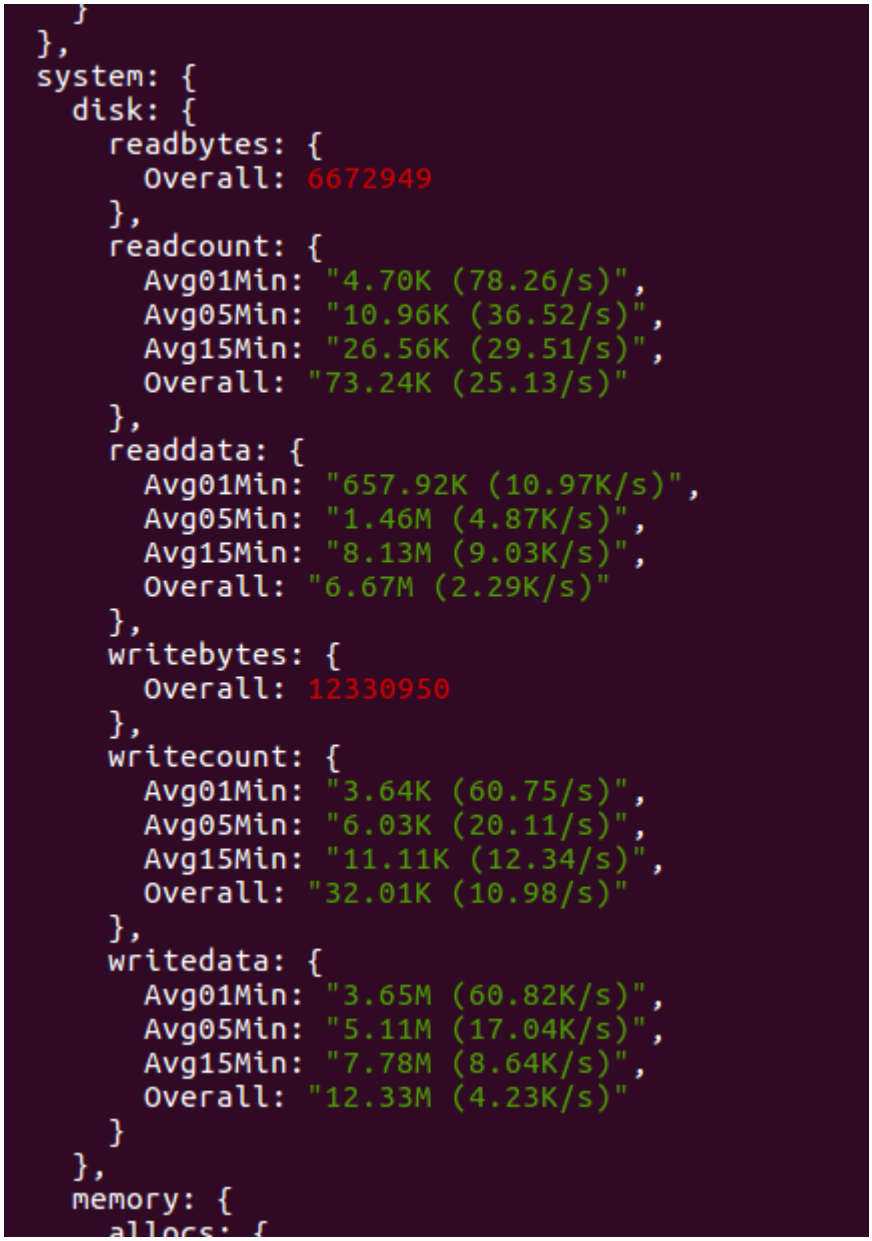


Figure 8.7: Sample output of Disk read write data

system/memory/

- allocs: number of memory allocations made
- frees: number of memory releases made
- inuse: memory currently being used
- pauses: time spent in the garbage collector

Table 8.1: Sample data collected for memory performance

```
memory: {
allocs: {
Avg01Min: "955.94K (15.93K/s)",
Avg05Min: "26.38M (87.94K/s)",
Avg15Min: "106.03M (117.81K/s)",
Overall: "1.10M (7.76K/s)"
},
frees: {
Avg01Min: "800.34K (13.34K/s)",
Avg05Min: "13.43M (44.75K/s)",
Avg15Min: "52.73M (58.59K/s)",
Overall: "941.97K (6.67K/s)"
},
inuse: {
Avg01Min: "94.76M (1.58M/s)",
Avg05Min: "3.49G (11.64M/s)",
Avg15Min: "14.20G (15.77M/s)",
Overall: "82.06M (581.22K/s)"
},
pauses: {
Avg01Min: "206.66K (3.44K/s)",
Avg05Min: "5.64M (18.79K/s)",
Avg15Min: "22.74M (25.27K/s)",
Overall: "180.21K (1.28K/s)"
}
```

```
  }
},
```

Ethereum gas is a unit that measures the amount of computational effort that it will take to execute a particular operation. It generally measures the effort to execute, create, and approve transactions. Every single operation that takes part in Ethereum, be it a transaction or smart contract execution, requires some amount of gas. We measure the amount of computation effort it takes by varying the number of transactions at a different time interval, each of 10 minutes. The computational efforts to execute the transactions in million gas is given in Figure 8.8. Gas usage in ethereum is collected by parsing the data generated while mining. A snippet of the parsed data is given below

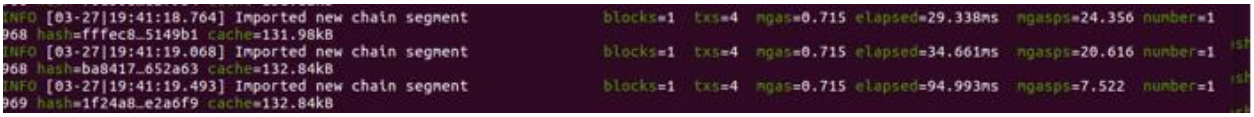


Figure 8.8: data generated for calculating ethereum Gas

INFO [03-27|13: 05: 39.432] Commit new mining work number= 1323 sealhash= 38d6a0…0cfe2a uncles= 0 txs= 10 gas= 1981260 fees= 0.07527764 elapsed= 385.334ms

INFO [03-27|13: 05: 39.457] Commit new mining work number= 1323 sealhash= 09b467…d3db9e uncles=1 txs=10 gas= 1981260 fees= 0.07527764 elapsed= 21.426ms

INFO [03-27|13 :05:39.744] Commit new mining work number= 1323 sealhash=d566d4…37f32a uncles= 2 txs= 10 gas= 1981260 fees= 0.07527764 elapsed= 873.949µs

INFO [03-27| 13: 05: 39.954] Successfully sealed new block number= 1323 sealhash=d566d4…37f32a hash= 48f978…7f0795 elapsed=210.020ms

INFO [03-27| 13: 05: 39.954] ⛓ block reached canonical chain number= 1316 hash=4a4bcc…38f217

8.2.2 **Summary of Result- First Approach**

Figure 8.9 indicates the system performance of the proposed Method (PM) and the traditional method (TM). Graph is plotted based on the experiment in Section 8.2.1. The result of system performance indicates that the proposed approach required more computations when compared with the traditional method. It also indicates that the proposed approach is practical with the system requirements used for the experiment which is detailed in Section 8.2.1.

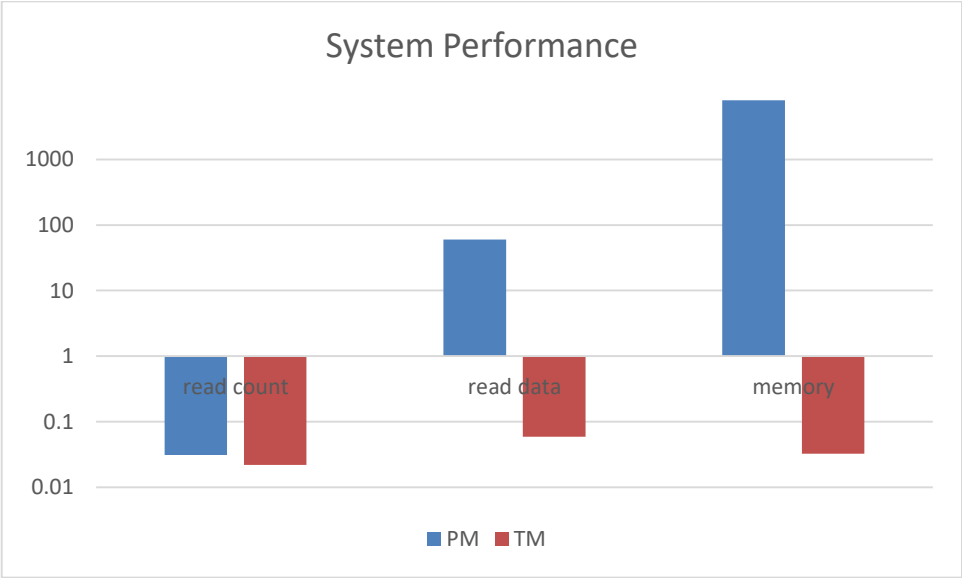


Figure 8.9: System Performance PM Vs TM

The Figure 8.10 indicates, gas usage for the proposed approach is 171,000. For a normal transaction, the standard ethereum gas limit is 21,000. The proposed approach requires a higher gas limit than the normal transaction. However, our experiment prove that, it is possible to create a transaction with 171,000 gas consumption with a system requirements as mentioned in Section 8.1.

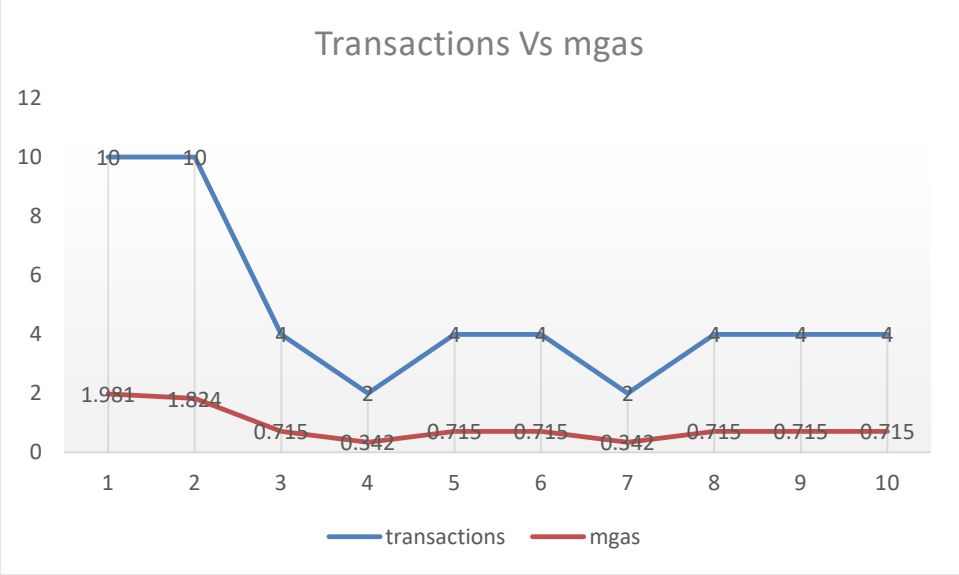


Figure 8.10: Ethereum Gas used for the proposed Method

8.3 **Measuring Network Performance – Second Approach**

To evaluate the network performance, we simulated our proposed scheme using Contiki and RELIC. This type of implementation and evaluation is used in various literatures (Sankaran 2016)(Bao et al. 2018)(Zoican et al. 2018)(Dorri, Kanhere, et al. 2019a)(Frimpong et al. 2020) (Kaku 2017)(Dorri, Steger, et al. 2017). Contiki is an opensource, multi-tasking operating system for embedded system and wireless sensor network. Contiki communication stack supports uIP and RIME. uIP is a small RFC-compliant TCP/IP stack that makes it possible for Contiki to communicate over the Internet. Rime is a lightweight communication stack designed for low-power radios. It is written in C language and can simulate embedded devices developed for various platforms such as MicaZ, TelosB, AVR, Z1 etc. RELIC is cryptographic toolkit. We arranged our nodes in a Linear fashion, IoT devices representing the blue nodes, edge nodes representing the purple nodes and cloudlet nodes as green nodes as shown in Figure 8.11. We used Z1 motes generated in a linear position with edge nodes within its radio network.

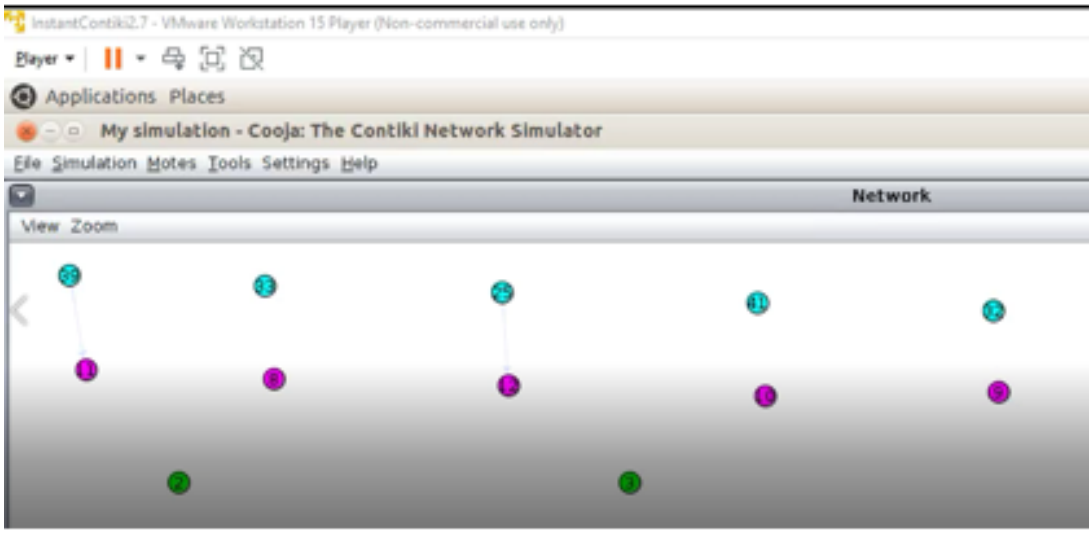


Figure 8.11: Simulation using Contiki OS

8.3.1 **Evaluation Metrics – Second Approach**

Latency: It is the time taken by a packet to reach the edge node from the IoT device. It is measured in milliseconds.

Throughput**:** It is the amount of data sent over the network per second. It is measured in Kilobyes/sec. These evaluation metrics are being used in literatures (Dinh et al. 2018) (Tunstad, Khan & Ha 2019)(Thakkar, Nathan & Viswanathan 2018)(Hao et al. 2018)(Seok, Park & Park 2019).

8.3.2 **Summary of result – Second Approach**

The latency between the IoT device and the edge node is calculated and is given in Figure 8.12, and the average throughput is given in Figure 4.10

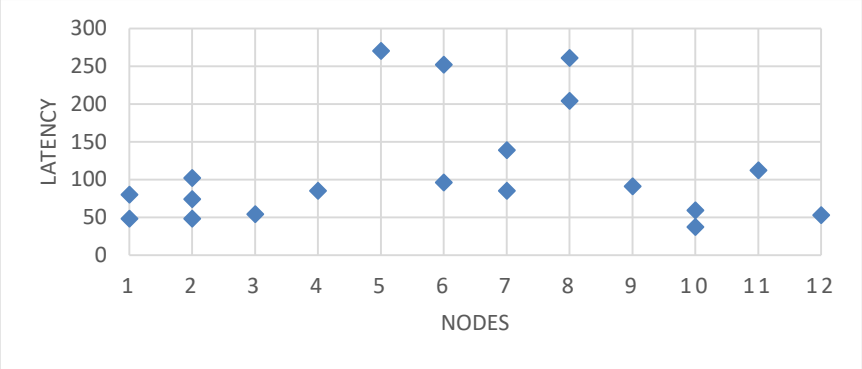


Figure 8.12:Latency

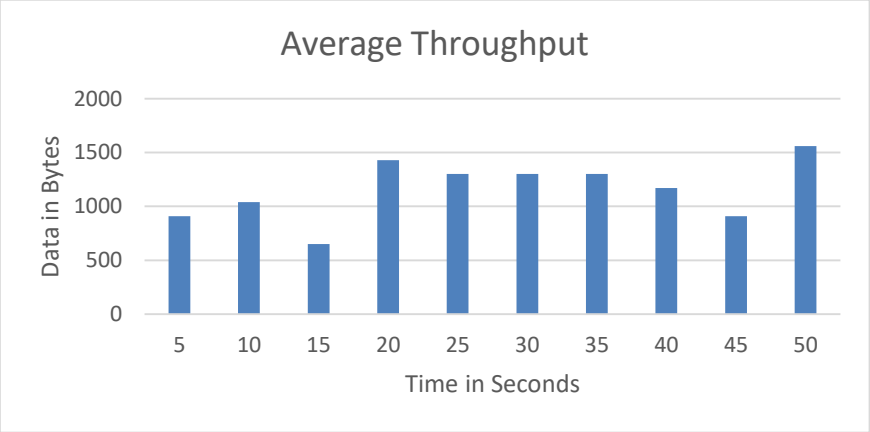


Figure 8.13: Average Throughput

The result show that the average latency is 100 ms and the average throughput is around 1.3 Kb/sec. Anything less than 100 ms is considered as a good latency.

8.4 Security Analysis- Third Approach

Microsoft threat modelling tool is the core element of Microsoft security development life cycle. It helps the security architects to identify the potential threats in their architecture. It uses STRIDE model. Various literatures have conducted security analysis based evaluation on blockchain architectures(Ali, Ali, et al. 2018) (Dorri, Kanhere, et al. 2019b)(Hengartnert & Steenkiste 2005)

S- Spoofing – Identifying threats on Authentication

T- Tampering identifying threats for Integrity

R- Repudiation – Identifying non-reputability threats

I-Information Disclosure – Identifying threats to confidentiality

D-Denial of Service – Identifying threat to Availability

E-Elevation of Privilege- Identifying threats to Authorization

8.4.1 **Evaluation Metrics: Third Approach**

We conducted a security analysis based on the known threats. A theoretical evaluation was conducted to identify how the proposed system is resilient to know threats. The details of the evaluation is provided in Section 7.6.2. We modelled our architecture using Microsoft threat modelling tools as shown in Figure 8.14 and the important findings. The major findings generated by the threat modelling tool includes application vulnerabilities rather than protocol vulnerabilities. Some of them are listed below.

1. Spoofing of the destination data store in edge or cloudlet nodes - Cloudlet Node may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Cloudlet Node. The recommendation provided by the tool was to consider using a standard authentication mechanism to identify the destination data store.
2. Risks from logging- Log readers can come under attack via log files. The recommendation provided by the tool was to consider ways to canonicalize data in all logs, Implement a single reader for the logs, if possible, in order to reduce attack surface area. And to be sure to understand and document log file elements which come from untrusted sources.
3. Potential Excessive Resource Consumption for Blockchain Application or Cloudlet Node- Resource consumption attack on blockchain applications can create a deadlock leading to a denial of service attack.

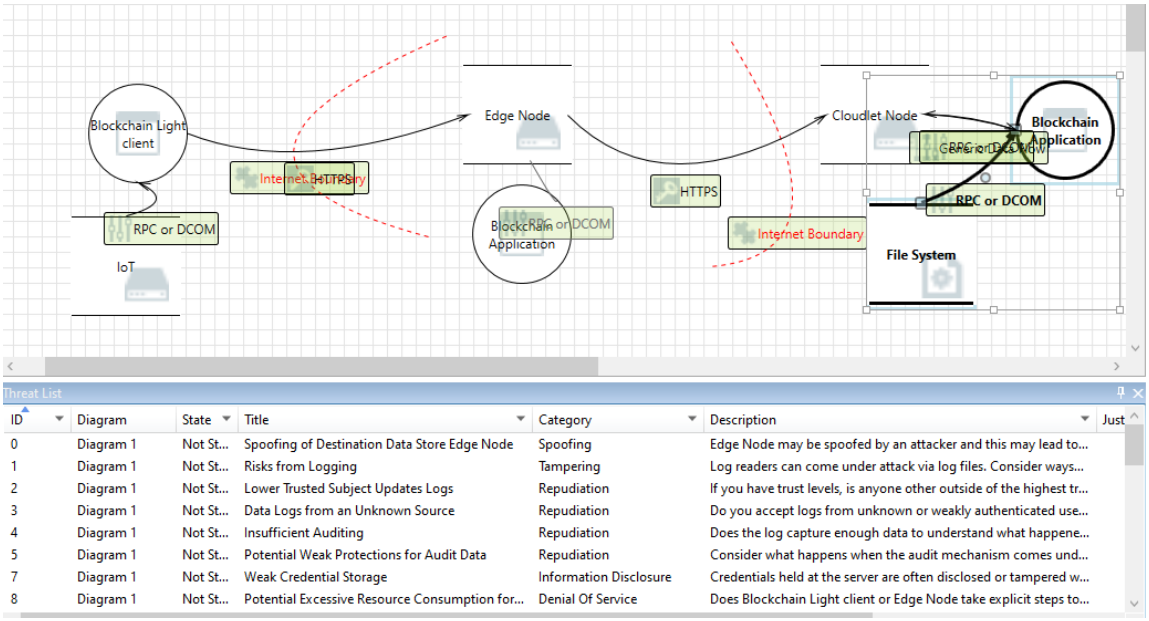


Figure 8.14: Threat Modelling

8.4.2 **Summary of result – Third Approach**

Data confidentiality is achieved using HIBE and the data is end-to-end encrypted. Blockchain provides integrity of data by hashing the transactions and linking blocks with the previous blocks making it harder for an attacker to modify the transactions. Data is distributed to several nodes which provides availability. Each transaction is grouped into to create block and each block is linked to its preceding block through hashes. As the data is not stored in a central location, and is decentralized into various cloudlet nodes, it provides availability. Transactions in blockchain are encrypted, which can be decrypted only by the owner of the vehicle or in case of a dispute the traffic authority can decrypt it and can provide as an evidence in court. Rather, it is worth mentioning that an adequate amount of research on strengthening the network architecture and ensuring an adequate amount of confidentiality, integrity and privacy of transactions are registered into blockchain is being carried out.

8.5 **Chapter Conclusion**

In this chapter, we detailed about the types of experiment we conducted along with the various evaluation metrics we used to measure the performance. Our results indicate that even though when compared with the traditional centralized system, the proposed approach requires additional

computational resources. However the proposed approach was able to provide a secure system for the traffic speed radars when compared with the traditional system. The proposed system was able to meet the CIA objectives of security and acts as a solution to mitigate some attacks on IoT. However, the analysis shows that the proposed system is not completely secure. It still suffers from some minor threats which could be considered as a drawback of the proposed approach.

Chapter 9 : CONCLUSION AND FUTURE WORK

9.1 Introduction

In this Chapter, we briefly summarize the finding of the thesis and test the validity of the hypothesis. We also provide the limitations of the proposed approach and highlight the future works.

9.2 Summary of research

The main contribution of the research is to provide a secure blockchain architecture for event driven IoT device. To reach to a full fledge architecture, we initially studied on blockchain and security in IoT and identified the key components that should be considered while creating a blockchain architecture for IoT. One of the key component we identified was to use device only architecture or an edge based architecture. Even though our experiment result show that device only based architecture has a better throughput when compared with edge based architecture, by considering the storage and security requirement device-only architecture was less feasible for creating IoT blockchain. Hence, we used edge-based architecture for creating blockchain. The traditional blockchain protocol of bitcoin provides integrity of the data but not confidentiality. Confidentiality can be achieved through symmetric or asymmetric encryption. Symmetric encryption uses same key for encryption and decryption. Hence, key management is a challenging tasks. Asymmetric encryption requires a Public Key Infrastructure for authenticating the public key. Hence, we propose how public key authentication can be obtained for IoT blockchain architecture. Our experiment show, how we implemented our design and comparison with the existing system. One of the major problem in a private blockchain is to identify the node that will perform consensus. We proposed a protocol for consensus node selection in IoT blockchain. In addition we proposed a privacy preserving blockchain architecture for traffic speed radars that prevents data tampering by internal or external people and protects privacy of the vehicle which will disclose the vehicle parameters only in case of a dispute.

9.3 **Testing the thesis Hypothesis**

H1: *It is possible to create a blockchain for event-driven IoT :* Through the experiments and results obtained that is discussed in Chapter 8, we were able to create a blockchain for IoT. Hence, the research fulfils the Hypothesis

H2: *HIBE can be used in a blockchain environment to provide data confidentiality for IoT data* – In Chapter 5, we provided with a flow diagram, how IBE can be used to provide data confidentiality in IoT blockchain. We verified it through the experiment and the results discussed in Chapter 8. Hence, the research fulfils the Hypothesis

H3: *The proposed blockchain architecture can improve the security of IoT data*- The security analysis we conducted on Chapter 7 shows the CIA objectives of security are fulfilled. In addition, our architecture is resistant to some types of attacks that are common in IoT. We verified it based in the results we obtained from the threat modelling tool. Hence, the research fulfils the hypothesis.

H4: *The system and network performance of the proposed architecture is better than the traditional centralized architecture of IoT*- Our experiment showed that the system and network performance is high with the traditional centralized approach when compared with the proposed system. Hence, the hypothesis is rejected.

9.4 **Findings**

The following findings are gained based on the experiment results

1. Our experiment showed that the proposed architecture can be implemented using the limited resources available with IoT
2. The system performance we calculated during the mining process in the edge node, for 10, 20 and 30 transactions indicate that the edge nodes will require at least 8 GB RAM with an average disk read/write data of 4 MB per minute.

3. The throughput and latency we calculated through experiment indicated that the proposed architecture throughput and latency is less than the original centralized architecture. However considering the advantages in terms of security, this is negligible
4. The Security analysis we conduced indicates that the proposed architecture is resistant to various forms of attacks. However, it is not resistant to all forms of attacks. These are listed in Section 8.4.1

9.5 Limitations of Proposed work

One of the limitation of the research is this architecture can be exactly used as it is, only for use cases where event driven IoT is used and the device is sensing some data that needs privacy. However, for IoTs within a home environment or industrials IoTs will require some modifications in this architecture to adapt with its requirements.

The result of threat modelling tool identified some threats in the proposed system like Risk of logging and spoofing attacks on the edge and cloudlet nodes. This is detailed in Section 8.4.1. The proposed architecture can be enhanced in future to prevent such type of attacks.

The proposed system does not consider the false positive cases, in case of a false positive there is no solution to remove the false positive entry.

9.6 Recommendations for Future work

Firstly, our approach use encryption for data privacy; however, if it is possible to replace the encryption system with a pseudo-anonymous system like the bitcoin, it will be a great achievement in this field. This can create a transparent system with lesser computations.

Secondly, Instead of creating a cloudlet-based storage, if it is possible to store the transactions on the users' computers then we will truly have a distributed architecture for IoT eco-system. Hence, the organizations will not need to store the IoT data. Data related to a particular user, will be stored on

the users device and only a reference for that data will be stored in the organizations device. This will ensure privacy and reduces the computational overhead of the organization.

Thirdly, the proposed architecture can be expanded to integrate a blockchain platform for the IoT device repair and maintenance. If any of the device is not working, this will be logged into a blockchain platform that can provide transparency to the quality of work by maintenance companies.

Finally, the proposed architecture can be expanded to accept the vehicle fines through the same blockchain as cryptocurrencies.

9.7 Summary

From our research, we were able to provide a decision tree including the key components that should be considered while creating blockchain for IoT. From our experiment, we prove that it is possible to create blockchain for IoT ecosystem. We proved that the proposed architecture is better than the traditional architecture by means of security analysis.

We were able to provide a public key authentication framework for event driven IoT using blockchain. Our experiment showed how to create a public key authentication process using blockchain. Our comparison with the existing systems shows that blockchain based systems can provide a decentralized system for authentication. A literature review study also indicated that our approach provides solution for backing up and updating keys, which was not addressed by other literature.

Our approach on using HIBE for IoT blockchain clearly solves the key management problems. When compared with the existing encryption techniques for IoT blockchain, we identified that they suffer from key management problem. Hence using HIBE for this architecture is an efficient solution to key management problem in IoT blockchain. Through our experiment, we show how to integrate HIBE with the blockchain.

The proposed approach solve the problem of tampering the data generated by sensors along with providing privacy. In the traditional approach, when a vehicle get fine for over speeding, the details of the vehicle along with the location, fine amount are transparent. We solved this problem, by providing privacy to the vehicle owners, where the fine details will be visible only to vehicle owner rather than making it transparent.

REFERENCES

2413-2019 - IEEE Approved Draft Standard for an Architectural Framework for the Internet of Things (IoT). (2019). *ISO* [online].Available at: https://standards.ieee.org/content/ieee-standards/en/standard/2413-2019.html.

Abeyratne, Saveen A., and R. P. M. (2016). Blockchain Ready Manufacturing Supply Chain Using Distributed Ledger. *International Journal of Research in Engineering and Technology*, vol. 05(09), pp. 1–10.

Administration, N. H. T. S. & Administration, F. H. (2008). Speed Enforcement Camera Systems Operational Guidelines, p. 91p [online].Available at: https://safety.fhwa.dot.gov/speedmgt/ref_mats/fhwasa09028/resources/Speed Camera Guidelines.pdf%0Ahttp://ntl.bts.gov/lib/30000/30100/30166/810916.pdf%5Cnhttps://trid.trb.org/view/863908.

Agyekum, K. O. B. O., Xia, Q., Sifah, E. B., Gao, J., Xia, H., Du, X. & Guizani, M. (2019). A secured proxy-based data sharing module in IoT environments using blockchain. *Sensors (Switzerland)*, vol. 19(5), pp. 1–20.

Al-Karaki, J. N. & Kamal, A. E. (2004). Routing techniques in wireless sensor networks: A survey. *IEEE Wireless Communications*, vol. 11(6), pp. 6–27.

Ali, J., Ali, T., Musa, S. & Zahrani, A. (2018). Towards secure IoT communication with smart contracts in a Blockchain infrastructure. *International Journal of Advanced Computer Science and Applications*, vol. 9(10), pp. 578–585.

Ali, J., Khalid, A. S., Yafi, E., Musa, S. & Ahmed, W. (2019). Towards a secure behavior modeling for IoT networks using blockchain. *CEUR Workshop Proceedings*, vol. 2486(January 2020), pp.

244–258.

Ali, M., Nelson, J., Shea, R. & Freedman, M. J. (2016). Blockstack : A Global Naming and Storage System Secured by Blockchains. *USENIX Annual Technical Conference*, pp. 181–194 [online].Available at: https://www.usenix.org/conference/atc16/technical-sessions/presentation/ali.

Ali, M. S., Vecchio, M., Pincheira, M., Dolui, K., Antonelli, F. & Rehmani, M. H. (2018). Applications of Blockchains in the Internet of Things : A Comprehensive Survey. *IEEE Communications Surveys & Tutorials*. IEEE, vol. PP(X), p. 1.

Armknecht, F., Boyd, C., Carr, C., Gjosteen, K., Jaschke, A., Reuter, C. A. & Strand, M. (2015). A Guide to Fully Homomorphic Encryption. *Cryptology ePrint Archive*, pp. 1–35 [online].Available at: https://eprint.iacr.org/2015/1192.

Ateniese, G., Fu, K., Green, M. & Hohenberger, S. (2006). Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security*, vol. 9(1), pp. 1–30.

Atzori, L., Iera, A. & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, vol. 54(15), pp. 2787–2805.

Axon, L. & Goldsmith, M. (2017). PB-PKI : a Privacy-Aware Blockchain-Based PKI Conventional Approaches to PKI. *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017)*, pp. 311-- 318.

Babar, S., Stango, A., Prasad, N., Sen, J. & Prasad, R. (2011). Proposed embedded security framework for Internet of Things (IoT). *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace and Electronic Systems Technology, Wireless VITAE 2011*, pp. 1–5.

Back, A. (2002). Hashcash-A Denial of Service Counter-Measure.

Badr, S., Gomaa, I. & Abd-Elrahman, E. (2018). Multi-tier blockchain framework for IoT-EHRs systems. *Procedia Computer Science*. Elsevier B.V., vol. 141, pp. 159–166.

Bagchi, S., Siddiqui, M. B., Wood, P. & Zhang, H. (2020). Dependability in edge computing. *Communications of the ACM*, vol. 63(1), pp. 58–66.

Baliga, A. (2017). Understanding Blockchain Consensus Models. *Whitepaper*, (April), pp. 1–14.

Banerjee, M., Lee, J. & Choo, K. K. R. (2018). A blockchain future for internet of things security: a position paper. *Digital Communications and Networks*. Elsevier Ltd, vol. 4(3), pp. 149–160.

Bao, Z., Shi, W., He, D. & Chood, K.-K. R. (2018). IoTChain: A Three-Tier Blockchain-based IoT Security Architecture, pp. 1–24 [online].Available at: http://arxiv.org/abs/1806.02008.

Bewley, S. Dean, R. Feddida, A. Maxwell, K. (2017). The Internet of Things Is becoming an everyday reality for IT decision makers, (June) [online].Available at: http://www.altvil.com/wp-content/uploads/2017/06/AVCo.-IoT-Security-White-Paper-June-2017.pdf.

Bhattacharya, P., Tanwar, S., Shah, R. & Ladha, A. (2020). Mobile Edge Computing-Enabled Blockchain Framework — A Survey, pp. 797–809.

Biryukov, Alex, Dmitry Khovratovich, and I. P. (2014). Deanonymisation of Clients in Bitcoin P2P Network. *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 15–29 [online].Available at: https://dl.acm.org/doi/pdf/10.1145/2660267.2660379?casa_token=rd2qMQMfkmMAAAAA:hWXVTCm53OVUHo52Ptq_6Y4_mpRo_eg2lOKDERIphZIYeqOztRJ5RJEVlRWfRuTV9VgYPx2v74dt.

bitcoin. (2018). *Developer Guide - Bitcoin* [online]. [Accessed 24 August 2019]. Available at: https://bitcoin.org/en/developer-guide#block-chain.

Bocek, T., Rodrigues, B. B., Strasser, T. & Stiller, B. (2017). Blockchains Everywhere -A Use Case of Blockchains in the Pharma Supply-Chain.

Boeckl, K., Fagan, M., Fisher, W., Lefkovitz, N., Megas, K. N., Nadeau, E., Gabel, D., Rourke, O. ', Piccarreta, B. & Scarfone, K. (2019). NISTIR 8228 Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks, p. 44.

Boneh, D. & Franklin, M. (2003). Identity-based encryption from the weil pairing. *SIAM Journal on Computing*, vol. 32(3), pp. 586–615.

Borah, M. D., Naik, V. B., Patgiri, R., Bhargav, A., Phukan, B. & Basani, S. G. M. (2020). *Supply Chain Management in Agriculture Using Blockchain and IoT*. Springer Singapore.

Buccafurri, F., Lax, G., Nicolazzo, S. & Nocera, A. (2017). Overcoming Limits of Blockchain for IoT Applications, pp. 1–6.

Buntinx, J. (2018). *Glitched Consensus Node Briefly Cripples NEO Network » The Merkle Hash.* [online].Available at: https://themerkle.com/glitched-consensus-node-briefly-cripples-neo-network/.

Byzantine fault. (2020) [online].Available at: https://en.wikipedia.org/wiki/Byzantine_fault.

Carrefour. (2019). *Carrefour launches Europe's first food blockchain* [online].Available at: http://www.carrefour.com/current-news/carrefour-launches-europes-first-food-blockchain.

Cermaq. (2019). *Cermaq contributes to traceability with blockchain* [online].Available at: https://www.cermaq.com/wps/wcm/connect/cermaq/news/mynewsdesk-press-release-2945012/.

Cermeño, J. S. (2016). Blockchain in financial services: Regulatory landscape and future challenges for its commercial application. *BBVA Research Working Paper*, vol. 16/20(December) [online].Available at: https://www.bbvaresearch.com/wp-content/uploads/2016/12/WP_16-20.pdf.

Cha, S. C., Chen, J. F., Su, C. & Yeh, K. H. (2018). A Blockchain Connected Gateway for BLE-Based Devices in the Internet of Things. *IEEE Access*. IEEE, vol. 6, pp. 24639–24649.

Chakraborty, S., Thomas, D., DeHart, J., Saralaya, K., Tadepalli, P. & Narendra, S. G. (2018). Solving internet's weak link for blockchain and IoT applications. *Proceedings of the 1st*

ACM/EIGSCC Symposium on Smart Cities and Communities, SCC 2018, pp. 3–7.

Chen, J., Yao, S., Yuan, Q., He, K., Ji, S. & Du, R. (2018). CertChain: Public and Efficient Certificate Audit Based on Blockchain for TLS Connections. *Proceedings - IEEE INFOCOM*, vol. 2018-April, pp. 2060–2068.

CNN Editorial Research. (2019). *Flint Water Crisis Fast Facts* [online].Available at: https://edition.cnn.com/2016/03/04/us/flint-water-crisis-fast-facts/index.html.

Cocks, C. (2001). An identity based encryption scheme based on quadratic residues. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2260, pp. 360–363.

comodo. (2018). *Bitcoin under attack: Comodo Stops Cunning Spear-Phishing Attack on a Cryptowallet Owner* [online].Available at: https://blog.comodo.com/comodo-news/bitcoin-phishing-attack-on-cryptowallet-owner/.

Conoscenti, M., Vetro, A. & De Martin, J. C. (2017). Blockchain for the Internet of Things: A systematic literature review. *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, (October).

Consensus. (2019). *How will Blockchain Benefit the Energy Industry* [online].Available at: https://consensys.net/enterprise-ethereum/use-cases/energy-and-sustainability/.

D. Foote, K. (2016). *A Brief History of the Internet of Things (Date accessed: 04-12-2019). Dataversity* [online].Available at: https://www.dataversity.net/brief-history-internet-things/#.

Dagher, G. G., Mohler, J., Milojkovic, M. & Marella, P. B. (2018). Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology. *Sustainable Cities and Society*. Elsevier, vol. 39(August 2017), pp. 283–297.

David, P. & Hsu, J. (2018). *Contiki OS* [online].Available at: http://www.contiki-os.org/.

Dedeoglu, V., Jurdak, R., Dorri, A., Lunardi, R. C., Michelin, R. A., Zorzo, A. F. & Kanhere, S. S. (2020). *Blockchain Technologies for IoT*. Springer Singapore.

Deogirikar, J. & Vidhate, A. (2017). Security attacks in IoT: A survey. *Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2017*, pp. 32–37.

Dinh, T. T. A., Liu, R., Zhang, M., Chen, G., Ooi, B. C. & Wang, J. (2018). Untangling Blockchain: A Data Processing View of Blockchain Systems. *IEEE Transactions on Knowledge and Data Engineering*. IEEE, vol. 30(7), pp. 1366–1385.

Dinh, T. T. A., Wang, J., Chen, G., Liu, R., Ooi, B. C. & Tan, K.-L. (2017). BLOCKBENCH: A Framework for Analyzing Private Blockchains [online].Available at: http://arxiv.org/abs/1703.04057.

Dolui, K. & Datta, S. K. (2017). Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. *GIoTS 2017 - Global Internet of Things Summit, Proceedings*.

Dorri, Ali and Kanhere, Salil S and Jurdak, R. (2017). Towards an Optimized BlockChain for IoT Ali. *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, pp. 173--178.

Dorri, A., Kanhere, S. S. & Jurdak, R. (2017). Towards an optimized blockchain for IoT. *Proceedings - 2017 IEEE/ACM 2nd International Conference on Internet-of-Things Design and Implementation, IoTDI 2017 (part of CPS Week)*, (April), pp. 173–178.

Dorri, A., Kanhere, S. S., Jurdak, R. & Gauravaram, P. (2017). Blockchain for IoT security and privacy: The case study of a smart home. *2017 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2017*, (January), pp. 618–623.

Dorri, A., Kanhere, S. S., Jurdak, R. & Gauravaram, P. (2019a). LSB: A Lightweight Scalable Blockchain for IoT security and anonymity. *Journal of Parallel and Distributed Computing*, vol. 134, pp. 180–197.

Dorri, A., Kanhere, S. S., Jurdak, R. & Gauravaram, P. (2019b). LSB: A Lightweight Scalable Blockchain for IoT security and anonymity. *Journal of Parallel and Distributed Computing*, vol. 134(December 2017), pp. 180–197.

Dorri, A., Luo, F., Kanhere, S. S., Jurdak, R. & Dong, Z. Y. (2019). SPB: A secure private blockchain-based solution for distributed energy trading. *IEEE Communications Magazine*, vol. 57(7), pp. 120–126.

Dorri, A., Steger, M., Kanhere, S. S. & Jurdak, R. (2017). BlockChain: A Distributed Solution to Automotive Security and Privacy. *IEEE Communications Magazine*, vol. 55(12), pp. 119–125.

Dwivedi, A. D., Srivastava, G., Dhar, S. & Singh, R. (2019). A decentralized privacy-preserving healthcare blockchain for IoT. *Sensors (Switzerland)*, vol. 19(2), pp. 1–17.

Eltayieb, N., Sun, L., Wang, K. & Li, F. (2019). *A Certificateless Proxy Re-encryption Scheme for Cloud-Based Blockchain*. Springer Singapore.

Ethereum. (2019). *Ethereum* [online].Available at: https://ethereum.org/.

ethereumwiki. (2018). *ethereum/wiki*. [online]. [Accessed 24 August 2019]. Available at: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQs#what-is-the-nothing-at-stake-problem-and-how-can-it-be-fixed.

Di Francesco Maesa, D., Marino, A. & Ricci, L. (2018). Data-driven analysis of Bitcoin properties: exploiting the users graph. *International Journal of Data Science and Analytics*, vol. 6(1), pp. 63–80.

Frimpong, E., Bakas, A., Dang, H.-V. & Michalas, A. (2020). Do Not Tell Me What I Cannot Do! (The Constrained Device Shouted under the Cover of the Fog): Implementing Symmetric Searchable Encryption on Constrained Devices, vol. 825355(826093), pp. 119–129.

Fromknecht, C., Velicanu, D. & Yakoubov, S. (2014). CertCoin: A NameCoin Based Decentralized Authentication System, pp. 1–19.

Garfinkel, S. (1995). *PGP: pretty good privacy*. O'Reilly Media, Inc.

Gartner. (2017). *Gartner Says 8.4 Billion Connected 'Things' Will Be in Use in 2017, Up 31 Percent From 2016* [online].Available at: https://www.gartner.com/newsroom/id/3598917.

Gentry, C., Sahai, A., Waters, B., Gentry, C., Sahai, A. & Waters, B. (2013). LNCS 8042 - Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. *2013. c International Association for Cryptologic Research*, vol. 8042, pp. 75–92 [online].Available at: https://link.springer.com/content/pdf/10.1007/978-3-642-40041-4_5.pdf.

Gentry, C. & Silverberg, A. (2002). Hierarchical id-based cryptography. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2501, pp. 548–566.

Github. (2018). *Hyperledger fabric github* [online]. [Accessed 24 August 2019]. Available at: https://github.com/hyperledger/fabric.

Goyal, V., Pandey, O., Sahai, A. & Waters, B. (2006). Attribute-based encryption for fine-grained access control of encrypted data. *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 89–98.

GSMA. (2015). GSMA: The Impact of the Internet of Things - The Connected Home. *Statista*, pp. 1–19 [online].Available at: https://www-statista-com.liverpool.idm.oclc.org/study/27374/internet-of-things-future-impact/%0Awww.gsma.com/connectedliving.

Gu, C. S. (2015). Fully homomorphic encryption from approximate ideal lattices. *Ruan Jian Xue Bao/Journal of Software*, vol. 26(10), pp. 2696–2719.

Gu, L., Wang, J. & Sun, B. (2014). Trust management mechanism for Internet of Things. *China Communications*, vol. 11(2), pp. 148–156.

Gubbi, J., Buyya, R., Marusic, S. & Palaniswami, M. (2013). Internet of Things (IoT): A vision,

architectural elements, and future directions. *Future Generation Computer Systems*, vol. 29(7), pp. 1645–1660.

Guo, F., Mu, Y., Susilo, W., Hsing, H., Wong, D. S. & Varadharajan, V. (2017). Optimized identity-based encryption from bilinear pairing for lightweight devices. *IEEE Transactions on Dependable and Secure Computing*, vol. 14(2), pp. 211–220.

Hall, R. E., Bowerman, B., Braverman, J., Taylor, J., Todosow, H., & Von Wimmersperg, U. (2000). The vision of a smart city. *BNL-67902; 04042. Brookhaven National Lab., Upton, NY*.

Hang, L. & Kim, D. H. (2019). Design and implementation of an integrated iot blockchain platform for sensing data integrity. *Sensors (Switzerland)*, vol. 19(10).

Hao, Y., Li, Y., Dong, X., Fang, L. & Chen, P. (2018). Performance Analysis of Consensus Algorithm in Private Blockchain. *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2018-June(Iv), pp. 280–285.

Heilman, E., Kendler, A. & Goldberg, S. (2015). Eclipse Attacks on Bitcoin ' s Peer-to-Peer Network. *USENIX Security Symposium 2015*, (August), pp. 129–144.

Hengartnert, U. & Steenkiste, P. (2005). Exploiting hierarchical identity-based encryption for access control to pervasive computing information. *Proceedings - First International Conference on Security and Privacy for Emerging Areas in Communications Networks, SecureComm 2005*, vol. 2005(October), pp. 384–394.

Horrow, S. & Sardana, A. (2013). Identity management framework for cloud based internet of things, pp. 200–203.

Horwitz, J. & Lynn, B. (2002). Toward hierarchical identity-based encryption. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2332, pp. 466–481.

Hyperledger. (2018). *Hyperledger* [online].Available at: https://www.hyperledger.org/.

Imbault, F., Swiatek, M., De Beaufort, R. & Plana, R. (2017). The green blockchain: Managing decentralized energy production and consumption. *Conference Proceedings - 2017 17th IEEE International Conference on Environment and Electrical Engineering and 2017 1st IEEE Industrial and Commercial Power Systems Europe, EEEIC / I and CPS Europe 2017*.

IoT Alliance Australia. (2018). IoT REFERENCE FRAMEWORK, (September).

ISO/IEC 30141:2018 [ISO/IEC 30141:2018] Internet of Things (loT) — Reference Architecture. (2018). *ISO* [online].Available at: https://www.iso.org/standard/65695.html.

ITU. (2012). *Internet of things global standards initiative. Internet of things global standards initiative* [online].Available at: http://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx.

Ivanov, A. (2014). *The internet of things*. *IEEE Design and Test*.

Jain, R. & Dogra, A. (2019). Solar energy distribution using blockchain and IoT integration. *ACM International Conference Proceeding Series*, pp. 118–123.

Jararweh, Y., Doulat, A., Alqudah, O., Ahmed, E., Al-Ayyoub, M. & Benkhelifa, E. (2016). The future of mobile cloud computing: Integrating cloudlets and Mobile Edge Computing. *2016 23rd International Conference on Telecommunications, ICT 2016*, pp. 760–764.

Jita, H. & Pieterse, V. (2018). A framework to apply the internet of things for medical care in a home environment. *ACM International Conference Proceeding Series*, pp. 45–54.

K, P., Babu, S. & Manoj, B. S. (2020). Sliding Window Blockchain Architecture for Internet of Things. *IEEE Internet of Things Journal*, vol. 4662(c), pp. 1–1.

Kafhali, S. El, Chahir, C., Hanini, M. & Salah, K. (2019). Architecture to manage internet of things data using blockchain and fog computing. *ACM International Conference Proceeding Series*.

Kaku, E. (2017). Using Blockchain To Support Provenance in the Internet of Things.

Kalodner, H., Carlsten, M., Ellenbogen, P., Bonneau, J. & Narayanan, A. (2015). An empirical study of Namecoin and lessons for decentralized namespace design. *14th Annual Workshop on the Economics of Information Security (WEIS)*.

Katz, J. (2007). Introduction to Modern Cryptography. *Introduction to Modern Cryptography*.

Kawamoto, D. (2017). IoT Security Incidents Rampant and Costly.

Kenneth G Paterson, G. P. (2003). A comparison between traditional Public Key Infrastructures and Identity-Based Cryptography. *Information Security Technical Report 8*, vol. 3, pp. 57–72.

Khacef, K., Pujolle, G., Khacef, K., Pujolle, G., Peer-to-peer, S., Khacef, K. & Pujolle, G. (2019). Secure Peer-to-Peer communication based on Blockchain To cite this version : HAL Id : hal-02180329 Secure Peer-to-Peer communication based on Blockchain.

Khan, R., Khan, S. U., Zaheer, R. & Khan, S. (2012). Future internet: The internet of things architecture, possible applications and key challenges. *Proceedings - 10th International Conference on Frontiers of Information Technology, FIT 2012*, pp. 257–260.

Kiayias, A., Russell, A., David, B. & Oliynykov, R. (2017). Ouroboros: A provably secure proof-of-stake blockchain protocol. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10401 LNCS, pp. 357–388.

Konashevych, O. (2015). Emercoin Blockchain Anchoring as a Way of Singing Contracts, (2015).

Korpela, Kari, Jukka Hallikas, and T. D. (2017). Digital supply chain transformation toward blockchain integration. *proceedings of the 50th Hawaii international conference on system sciences*.

Kotb, Y., Al Ridhawi, I., Aloqaily, M., Baker, T., Jararweh, Y. & Tawfik, H. (2019). Cloud-Based Multi-Agent Cooperation for IoT Devices Using Workflow-Nets. *Journal of Grid Computing*. Journal of Grid Computing.

Kroll, J. A., Davey, I. C. & Felten, E. W. (2013). The economics of Bitcoin mining, or Bitcoin in the

presence of adversaries. *Proceedings of WEIS*, pp. 1–21.

Lamport, L., Shostak, R. & Pease, M. (1982). The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, vol. 4(3), pp. 382–401.

Larimer, D. (2014). Delegated proof-of-stake (dpos). *Bitshare whitepaper.*

Lazaroiu, C. & Roscia, M. (2017). Smart district through IoT and blockchain. *2017 6th International Conference on Renewable Energy Research and Applications, ICRERA 2017*, vol. 2017-Janua(June 2018), pp. 454–461.

Liñán, A., Alvaro, C., Bagula, V. A., Zennaro, M. & Pietrosemoli, E. (2015). IoT in Five Days, (March), pp. 1–177.

Lo, S. K., Liu, Y., Chia, S. Y., Xu, X., Lu, Q., Zhu, L. & Ning, H. (2019). Analysis of Blockchain Solutions for IoT: A Systematic Literature Review. *IEEE Access*. IEEE, vol. 7(May), pp. 58822–58835.

Lo3Energy. (2019). *Lo3Energy* [online].Available at: https://lo3energy.com/.

Lowne, A. (2014). *Securing the internet of things. ECN Electronic Component News* [online].Available at: https://www.us-cert.gov/ncas/tips/ST17-001.

Lunardi, R. C., Michelin, R. A., Neu, C. V. & Zorzo, A. F. (2018). Distributed access control on IoT ledger-based architecture. *IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS 2018*, pp. 1–7.

Magnusson, S. (2018). Evaluation of Decentralized Alternatives to PKI for IoT Devices.

Makrushin, Denis, and V. D. (2016). Fooling the'Smart City'. *Technical Report*.

Malan, D. J., Welsh, M. & Smith, M. D. (2004). A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, IEEE SECON 2004*, pp. 71–80.

Manzoor, A., Liyanage, M., Braeken, A., Kanhere, S. S. & Ylianttila, M. (2018). Blockchain based Proxy Re-Encryption Scheme for Secure IoT Data Sharing [online].Available at: http://arxiv.org/abs/1811.02276.

Markmann, T., Schmidt, T. C. & Wählisch, M. (2015). Federated end-to-end authentication for the constrained internet of things using IBC and ECC. *SIGCOMM 2015 - Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 603–604.

Mattern, Friedemann, and C. F. (2010). From Active Data Management to Event-Based Systems and More - Papers in Honor of Alejandro Buchmann on the Occasion of His 60th Birthday. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6462 LNCS, pp. 242–259.

Mazieres, D. & Mazières, D. (2015). The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation*, pp. 1–45.

Meena, D. K. & Dwivedi, R. (2019). Preserving patient ' s privacy using proxy re-encryption in permissioned blockchain.

Mendki, P. (2019). Blockchain enabled IoT edge computing. *ACM International Conference Proceeding Series*, vol. Part F1481, pp. 66–69.

Michelin, R. A., Dorri, A., Steger, M., Lunardi, R. C., Kanhere, S. S., Jurdak, R. & Zorzo, A. F. (2018). SpeedyChain: A framework for decoupling data from blockchain for smart cities. *ACM International Conference Proceeding Series*, (July), pp. 145–154.

Miorandi, D., Sicari, S., Pellegrini, F. De & Chlamtac, I. (2012). Ad Hoc Networks Internet of things : Vision , applications and research challenges. *Ad Hoc Networks*. Elsevier B.V., vol. 10(7), pp. 1497–1516.

Modum. (2018). *Modum* [online].Available at: https://modum.io/.

Mohanty, S. N., Ramya, K. C., Rani, S. S., Gupta, D., Shankar, K., Lakshmanaprabu, S. K. & Khanna, A. (2020). An efficient Lightweight integrated Blockchain (ELIB) model for IoT security and privacy. *Future Generation Computer Systems*. Elsevier B.V., vol. 102, pp. 1027–1037.

Moore, S. (2016). Practical Byzantine Fault Tolerance. *American Mathematical Monthly*, vol. 27(4), pp. 1–24.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. *Www.Bitcoin.Org*.

Nestlé, N. (2019). *Nestlé breaks new ground with open blockchain pilot | Nestlé Global*. *Nestlé* [online].Available at: https://www.nestle.com/media/pressreleases/allpressreleases/nestle-open-blockchain-pilot.

Newsome, J., Shi, E., Song, D. & Perrig, A. (2004). The sybil attack in sensor networks, p. 259.

Novo, O. (2018). Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT. *IEEE Internet of Things Journal*, vol. 5(2), pp. 1184–1195.

Nuñez, D., Agudo, I. & Lopez, J. (2017). Proxy Re-Encryption: Analysis of constructions and its application to secure access delegation. *Journal of Network and Computer Applications*, vol. 87, pp. 193–209.

O'Dwyert, K. J. & Malone, D. (2014). Bitcoin mining and its energy footprint. *IET Conference Publications*, vol. 2014(CP639), pp. 280–285.

Official Go implementation of the Ethereum protocol. (2019). *Github* [online]. [Accessed 12 September 2019]. Available at: https://github.com/ethereum/go-ethereum.

Oliveira, L. B., Aranha, D. F., Gouvêa, C. P. L., Scott, M., Câmara, D. F., López, J. & Dahab, R. (2011). TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. *Computer Communications*. Elsevier B.V., vol. 34(3), pp. 485–493.

Olleros, F., Zhegu, M. & Pilkington, M. (2016). Blockchain technology: principles and applications.

Research Handbook on Digital Transformations, pp. 225–253.

Osanaiye, O., Chen, S., Yan, Z., Lu, R., Choo, K. K. R. & Dlodlo, M. (2017). From Cloud to Fog Computing: A Review and a Conceptual Live VM Migration Framework. *IEEE Access*, vol. 5, pp. 8284–8300.

Ouaddah, A., Abou Elkalam, A. & Ait Ouahman, A. (2016). FairAccess: a new Blockchain-based access control framework for the Internet of Things. *Security and Communication Networks*, vol. 9(18), pp. 5943–5964.

Paganini, P. (2018). *Anonymous Italia hacked speed camera database and took over the police systems in Correggio* [online].Available at: https://securityaffairs.co/wordpress/67378/hacktivism/anonymous-speed-camera-database.html.

Pahl, C., El Ioini, N. & Helmer, S. (2018). A decision framework for blockchain platforms for iot and edge computing. *IoTBDS 2018 - Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security*, vol. 2018-March, pp. 105–113.

Pan, J., Wang, J., Hester, A., Alqerm, I., Liu, Y. & Zhao, Y. (2018). EdgeChain: An Edge-IoT Framework and Prototype Based on Blockchain and Smart Contracts. *IEEE Internet of Things Journal*, pp. 1–14.

Paul, R., Baidya, P., Sau, S., Maity, K., Maity, S. & Mandal, S. B. (2018). IoT Based Secure Smart City Architecture Using Blockchain. *Proceedings - 2nd International Conference on Data Science and Business Analytics, ICDSBA 2018*. IEEE, pp. 215–220.

Pavithran, D. & Shaalan, K. (2020). An Optimal Consensus Node Selection Process for IoT Blockchain. *2019 Sixth HCT Information Technology Trends (ITT)*. IEEE, pp. 115–119.

Pinto, G., Dias, J. P. & Ferreira, H. S. (2018). Blockchain-based PKI for Crowdsourced IoT Sensor Information, pp. 1–16 [online].Available at: http://arxiv.org/abs/1807.03863.

Polyzos, G. C. & Fotiou, N. (n.d.). Blockchain-assisted Information Distribution for the Internet of Things Telegram : @ Computer _ IT _ Engineering Telegram : @ Computer _ IT _ Engineering.

Popov, S. (2018). IOTA whitepaper v1.4.3, pp. 1–28 [online].Available at: https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf.

Radar Gun Wiki. (n.d.). *Radar Gun Wiki* [online].Available at: https://en.wikipedia.org/wiki/Radar_gun.

Rahulamathavan, Y., Phan, R. C. W., Rajarajan, M., Misra, S. & Kondoz, A. (2018). Privacy-preserving blockchain based IoT ecosystem using attribute-based encryption. *11th IEEE International Conference on Advanced Networks and Telecommunications Systems, ANTS 2017*, pp. 1–6.

Ramachandran, A. & Kantarcioglu, M. (2018). Smartprovenance: A distributed, blockchain based data provenance system. *CODASPY 2018 - Proceedings of the 8th ACM Conference on Data and Application Security and Privacy*, vol. 2018-Janua, pp. 35–42.

Ramachandran, G. S. & Krishnamachari, B. (2018). Blockchain for the IoT: Opportunities and Challenges [online].Available at: http://arxiv.org/abs/1805.02818.

Rashid, F. Y. (2019). Attackers are signing Malware with valid certificates. *Duo Security*.

Ray, P. P. (2018). A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*. King Saud University, vol. 30(3), pp. 291–319.

Research, J. (2019). *Blockchain to Save the Food Industry $31 Billion by 2024, Driven by IoT Partnerships* [online].Available at: https://www.juniperresearch.com/press/press-releases/blockchain-to-save-the-food-industry-%2431-billion-b.

RSA Data Security. (1999). Understanding Public Key Infrastructure (PKI), pp. 1–7.

Sagirlar, G., Carminati, B., Ferrari, E., Sheehan, J. D. & Ragnoli, E. (2018). Hybrid-IoT: Hybrid Blockchain Architecture for Internet of Things - PoW Sub-blockchains, pp. 1–10.

Sahai, A. & Waters, B. (2005). Fuzzy identity-based encryption. *Lecture Notes in Computer Science*, vol. 3494, pp. 457–473.

Salimitari, M. & Chatterjee, M. (2018). An Overview of Blockchain and Consensus Protocols for IoT Networks.

Samaniego, M. & Deters, R. (2019). Pushing Software-Defined Blockchain Components onto Edge Hosts. *Proceedings of the 52nd Hawaii International Conference on System Sciences*, pp. 7079–7086.

Sankar, L. S. S., Sindhu, M. & Sethumadhavan, M. (2017). Survey of Consensus Protocols on Blockchain Applications. *2017 International Conference on Advanced Computing and Communication Systems (ICACCS -2017), Jan. 06 – 07, 2017, Coimbatore, INDIA.*

Sankaran, S. (2016). Lightweight security framework for IoTs using identity based cryptography. *2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016*, pp. 880–886.

Sankaran, S., Sanju, S. & Achuthan, K. (2018). Towards realistic energy profiling of blockchains for securing internet of things. *Proceedings - International Conference on Distributed Computing Systems*. IEEE, vol. 2018-July, pp. 1454–1459.

Satyanarayanan, M., Bahl, P., Caceres, R. & Davies, N. (2006). The Case for VM-based Cloudlets in Mobile Computing. *IEEE pervasive Computing*, vol. 1, pp. 1–9.

Schwartz, D., Youngs, N. & Britto, A. (2014). The Ripple protocol consensus algorithm. *Ripple Labs Inc White Paper*, pp. 1–8 [online].Available at: http://www.naation.com/ripple-consensus-whitepaper.pdf.

Seibold S and Samman G. (2018). *Immutable agreement for the Internet of value. KPMG*.

Seok, B., Park, J. & Park, J. H. (2019). A lightweight hash-based blockchain architecture for industrial IoT. *Applied Sciences (Switzerland)*, vol. 9(18).

Seyed, F. M., Mirghasemi, A., Dorri, A. & Ahmadifar, H. (2020). DMap: A Distributed Blockchain-based Framework for Online Mapping in Smart City, (August).

Shamir, A. (1985). Identity-Based Cryptosystems and Signature Schemes. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 196 LNCS, pp. 47–53.

Sharifinejad, M., Dorri, A. & Rezazadeh, J. (2020). BIS- A Blockchain-based Solution for the Insurance Industry in Smart Cities, (January).

Sharma, P. K. & Park, J. H. (2018). Blockchain based hybrid network architecture for the smart city. *Future Generation Computer Systems*. Elsevier B.V., vol. 86, pp. 650–655.

Shrestha, R. & Kim, S. (2019). Integration of IoT with blockchain and homomorphic encryption : Challenging issues and opportunities, vol. 115.

Sianturi, E. T. (2019). *No TitleΕΛΕΝΗ. Αγαη* [online]. [Accessed 12 September 2019]. Available at: https://github.com/ethereum/go-ethereum.

Silva, F. C., Ahmed, M. A., Mart, M. & Kim, Y. (2019). Design and Implementation of a Blockchain-Based Energy Trading Platform for Electric Vehicles in Smart Campus Parking Lots.

Simic, M., Sladic, G. & Milosavljević, B. (2017). A Case Study IoT and Blockchain powered A Case Study IoT and Blockchain powered Healthcare. *The 8th PSU-UNS International Conference on Engineering and Technology (ICET-2017)*, (June).

Singh, B., Dhawan, S., Arora, A. & Patail, A. (2018). A View of Cloud Computing. *International Journal of Computers & Technology*, vol. 4(2b1), pp. 387–392.

Sklavos, N. & Ioannis, D. Z. (2016). Cryptography and security in internet of things (iots): Models, schemes, and implementations. *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 2–3.

Solidity. (2016). *Introduction To Smart Contracts Introduction To Smart Contracts* [online].Available at: https://solidity.readthedocs.io/en/v0.6.10/introduction-to-smart-contracts.html.

Sousa, J., Bessani, A. & Vukolic, M. (2018). A byzantine Fault-Tolerant ordering service for the hyperledger fabric blockchain platform. *Proceedings - 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2018*, (Section 4), pp. 51–58.

Srinivasan, C. R., Rajesh, B., Saikalyan, P., Premsagar, K. & Yadav, E. S. (2019). A review on the different types of internet of things (IoT). *Journal of Advanced Research in Dynamical and Control Systems*, vol. 11(1), pp. 154–158.

Stallings, W. (2006). *Cryptography and Network Security*. Pearson Education.

Stanciu, A. (2017). Blockchain Based Distributed Control System for Edge Computing. *Proceedings - 2017 21st International Conference on Control Systems and Computer, CSCS 2017*, pp. 667–671.

Stavrou, A., Voas, J. & Fellow, I. (2016). Ddos in Iot, (August).

Sun, J., Yan, J. & Zhang, K. Z. K. (2016). Blockchain-based sharing services: What blockchain technology can contribute to smart cities. *Financial Innovation*. Financial Innovation, vol. 2(1).

Swan, M. (2015). *Blockchain : Blueprint para uma nova economia* [online].Available at: https://books.google.com.br/books?hl=en&lr=&id=RHJmBgAAQBAJ&oi=fnd&pg=PR3&dq=blockchain&ots=XQvHE--Tj5&sig=976PvT8nIJ2rtMmgU_dzNeKKoY4.

Tahir, S. & Rajarajan, M. (2018). Privacy-Preserving Searchable Encryption Framework for Permissioned Blockchain Networks. *Proceedings - IEEE 2018 International Congress on Cybermatics: 2018 IEEE Conferences on Internet of Things, Green Computing and*

Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, iThings/Gree, pp. 1628–1633.

Teslya, N. & Ryabchikov, I. (2018). Blockchain Platforms Overview for Industrial IoT Purposes. *2018 22nd Conference of Open Innovations Association (FRUCT)*, pp. 250–256.

Thakkar, P., Nathan, S. & Viswanathan, B. (2018). Performance benchmarking and optimizing hyperledger fabric blockchain platform. *Proceedings - 26th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS 2018*, pp. 264–276.

The Industrial Internet of Things Volume G1: Reference Architecture. (2019) [online].Available at: https://www.mendeley.com/catalogue/industrial-internet-things-volume-g1-reference-architecture-1/.

Tian. (2018). ASupplyChainTraceabilitySystemforFoodSafetyBasedonHACCP,Blockchain&InternetofThings, (March) [online].Available at: http://epub.wu.ac.at/.

Tran, A. B., Lu, Q. & Weber, I. (2018). Lorikeet: A model-driven engineering tool for blockchain-based business process execution and asset management. *CEUR Workshop Proceedings*, vol. 2196, pp. 56–60.

Tuli, S., Mahmud, R., Tuli, S. & Buyya, R. (2018). FogBus: A Blockchain-based Lightweight Framework for Edge and Fog Computing [online].Available at: http://arxiv.org/abs/1811.11978.

Tunstad, P., Khan, A. M. & Ha, P. H. (2019). HyperProv: Decentralized Resilient Data Provenance at the Edge with Blockchains, pp. 2–4 [online].Available at: http://arxiv.org/abs/1910.05779.

Uckelmann, D., Harrison, M. & Michahelles, F. (2011). Architecting the Internet of Things. *Architecting the Internet of Things*, pp. 1–24.

Uddin, M. A., Stranieri, A., Gondal, I. & Balasubramanian, V. (2020). Blockchain leveraged

decentralized IoT eHealth framework. *Internet of Things*, vol. 9, p. 100159.

Vasek, M., Thornton, M. & Moore, T. (2014). Empirical analysis of denial-of-service attacks in the bitcoin ecosystem. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8438, pp. 57–71.

Verma, D., Desai, N., Preece, A., Taylor, I. & Heights, Y. (2017). A blockchain based architecture for asset management in coalition operations Application Blockchain Data Structure e . g . Digital Currency e . g . Hyperledger, vol. 10190, pp. 1–9.

Vossen, G. (n.d.). Distributed Computing with Permissioned Blockchains and Databases Edited by, vol. 9(6), pp. 69–94.

Vukolić, M. (2017). Rethinking Permissioned Blockchains, pp. 3–7.

Weber, R. H. (2010). Internet of Things - New security and privacy challenges. *Computer Law and Security Review*. Elsevier Ltd, vol. 26(1), pp. 23–30.

Wood, G. (2018). Ethereum: a secure decentralized generalized distributed ledger.

Wörner, Dominic, and T. von B. (2014). When your sensor earns money: exchanging data for cash with Bitcoin. *2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*.

www.blockchain.com/charts/hash-rate. (2019). *blockchain.com* [online].Available at: https://www.blockchain.com/charts/hash-rate.

Xiong, Z., Zhang, Y., Niyato, D., Wang, P. & Han, Z. (2018). When mobile blockchain meets edge computing. *IEEE Communications Magazine*, vol. 56(8), pp. 33–39.

Xu, L. Da, He, W. & Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, vol. 10(4), pp. 2233–2243.

Xu, R., Chen, Y., Blasch, E. & Chen, G. (2018). BlendCAC: A BLockchain-ENabled Decentralized

Capability-based Access Control for IoTs [online].Available at: http://arxiv.org/abs/1804.09267.

Yakubov, A., Shbair, W. M., Wallbom, A., Sanda, D. & State, R. (2018). A blockchain-based PKI management framework. *IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS 2018*, pp. 1–6.

Yang, L., Yu, P., Bailing, W., Xuefeng, B., Xinling, Y. & Geng, L. (2013). IOT secure transmission based on integration of IBE and PKI/CA. *International Journal of Control and Automation*, vol. 6(2), pp. 245–254.

Yang, Y., Wu, L., Yin, G., Li, L. & Zhao, H. (2017). A Survey on Security and Privacy Issues in Internet-of-Things. *IEEE Internet of Things Journal*, vol. 4(5), pp. 1250–1258.

Yunusov, R. R., Kurochkin, Y. V, Lvovsky, A. I., Fedorov, A. K. & Block, N. (2018). Quantum-secured blockchain. *Quantum Science and Technology 3*, vol. 3.

Zhang, Yu, and J. W. (2015). An IoT electric business model based on the protocol of bitcoin. *2015 18th international conference on intelligence in next generation networks*, pp. 184–191.

Zhou, L., Wang, L., Ai, T. & Sun, Y. (2018). BeeKeeper 2.0: Confidential blockchain-enabled IoT system with fully homomorphic computation. *Sensors (Switzerland)*, vol. 18(11).

Zhou, L., Wang, L., Sun, Y. & Lv, P. (2018). BeeKeeper: A Blockchain-Based IoT System with Secure Storage and Homomorphic Computation. *IEEE Access*. IEEE, vol. 6, pp. 43472–43488.

Zhu, X. & Fan, T. (2019). Research on Application of Blockchain and Identity-Based Cryptography. *IOP Conference Series: Earth and Environmental Science*, vol. 252(4).

Zoican, S., Zoican, R., Vochin, M. & Galatchi, D. (2018). Blockchain and Consensus Algorithms in Internet of Things. *2018 13th International Symposium on Electronics and Telecommunications, ISETC 2018 - Conference Proceedings*, pp. 1–4.

Zyskind, G., Nathan, O. & Pentland, A. S. (2015). Decentralizing privacy: Using blockchain to

protect personal data. *Proceedings - 2015 IEEE Security and Privacy Workshops, SPW 2015*, pp. 180–184.

APPENDIX 1

We followed the below steps to create a private Ethereum Node in Ubuntu. We used geth version of Ethereum. Geth is a command line interface, used to run a full ethereum node. Initailly, we installed the essential packages required for geth.

```
root@gp:~# sudo apt-get install -y build-essential
E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporari
lable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is anc
ess using it?
root@gp:~# sudo rm /var/lib/apt/lists/lock
root@gp:~# sudo rm /var/cache/apt/archives/lock
root@gp:~# sudo rm /var/lib/dpkg/lock
root@gp:~# sudo rm /var/lib/dpkg/lock
rm: cannot remove '/var/lib/dpkg/lock': No such file or directory
root@gp:~# sudo apt-get install -y build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.1ubuntu2).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
root@gp:~#
```

Then we installed ‘git’ , as we have to download the go-ethereum from Github. We then installed the geth and the related packages.

```
root@gp:~# sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gi
  gitweb git-arch git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
```

```
root@gp:~# git clone https://github.com/ethereum/go-ethereum.git
Cloning into 'go-ethereum'...
^C
root@gp:~# ls
root@gp:~# git clone https://github.com/ethereum/go-ethereum.git
Cloning into 'go-ethereum'...
^C
root@gp:~# git clone https://github.com/ethereum/go-ethereum.git
Cloning into 'go-ethereum'...
remote: Enumerating objects: 2, done.
remote: Counting objects: 100% (2/2), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 75171 (delta 0), reused 0 (delta 0), pack-reused 75169
Receiving objects: 100% (75171/75171), 104.55 MiB | 243.00 KiB/s, done.
Resolving deltas: 100% (50092/50092), done.
Checking connectivity... done.
root@gp:~# ls
go-ethereum
root@gp:~#
```

```
root@gp:~# wget https://dl.google.com/go/go1.11.2.linux-amd64.tar.gz
--2018-11-08 21:41:07--  https://dl.google.com/go/go1.11.2.linux-amd64.tar.gz
Resolving dl.google.com (dl.google.com)... 172.217.18.142, 2a00:1450:4018:800:
00e
Connecting to dl.google.com (dl.google.com)|172.217.18.142|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 126520483 (121M) [application/octet-stream]
Saving to: ‘go1.11.2.linux-amd64.tar.gz’

go1.11.2.linux-amd6 100%[===================>] 120.66M  1.93MB/s    in 90s

2018-11-08 21:42:38 (1.34 MB/s) - ‘go1.11.2.linux-amd64.tar.gz’ saved [1265204
/126520483]

root@gp:~#
```

```
root@gp:~# sudo tar -C /usr/local -xzf go1.11.2.linux-amd64.tar.gz
root@gp:~#
```

```
root@gp:~# locate .profile
/etc/skel/.profile
/home/gp/.profile
/root/.profile
/usr/share/base-files/dot.profile
/usr/share/doc/lintian/examples/profiles/my-vendor/main.profile
/usr/share/lintian/profiles/debian/aptdaemon.profile
/usr/share/lintian/profiles/debian/extra-hardening.profile
/usr/share/lintian/profiles/debian/ftp-master-auto-reject.profile
/usr/share/lintian/profiles/debian/main.profile
/usr/share/lintian/profiles/ubuntu/aptdaemon.profile
/usr/share/lintian/profiles/ubuntu/main.profile
root@gp:~# gedit /root/.profile
```

```
root@gp:~# gedit .bashrc
```

```
if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi
export PATH="$PATH:/usr/local/go/bin"

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
```

```
root@gp:~# which go
/usr/local/go/bin/go
root@gp:~# go version
go version go1.11.2 linux/amd64
root@gp:~#
```

```
root@gp:~# gedit go-ethereum/consensus/ethash/consensus.go
```

```
// CalcDifficulty is the difficulty adjustment algorithm. It returns
// the difficulty that a new block should have when created at time
// given the parent block's time and difficulty.
func (ethash *Ethash) CalcDifficulty(chain consensus.ChainReader, time ui
types.Header) *big.Int {
        return CalcDifficulty(chain.Config(), time, parent)
}

// CalcDifficulty is the difficulty adjustment algorithm. It returns
// the difficulty that a new block should have when created at time
// given the parent block's time and difficulty.
func CalcDifficulty(config *params.ChainConfig, time uint64, parent *type
        next := new(big.Int).Add(parent.Number, big1)
        switch {
        case config.IsConstantinople(next):
                return calcDifficultyConstantinople(time, parent)
        case config.IsByzantium(next):
                return calcDifficultyByzantium(time, parent)
        case config.IsHomestead(next):
                return calcDifficultyHomestead(time, parent)
        default:
                return calcDifficultyFrontier(time, parent)
        }
}

// Some weird constants to avoid constant memory allocs for them.
var (
        expDiffPeriod = big.NewInt(100000)
```

```
        return CalcDifficulty(chain.Config(), time, parent)
}

// CalcDifficulty is the difficulty adjustment algorithm. It returns
// the difficulty that a new block should have when created at time
// given the parent block's time and difficulty.
func CalcDifficulty(config *params.ChainConfig, time uint64, parent *types.Header) *big.Int {
        return big.NewInt(1)
}

// Some weird constants to avoid constant memory allocs for them.
var (
        expDiffPeriod = big.NewInt(100000)
        big1          = big.NewInt(1)
        big2          = big.NewInt(2)
```

```
root@gp:~# cd go-ethereum
root@gp:~/go-ethereum# make geth
```

```
github.com/ethereum/go-ethereum/vendor/github.com/naoina/toml/
github.com/ethereum/go-ethereum/vendor/github.com/naoina/toml
github.com/ethereum/go-ethereum/cmd/geth
Done building.
Run "/root/go-ethereum/build/bin/geth" to launch geth.
root@gp:~/go-ethereum#
```

```
Run "/root/go-ethereum/build/bin/geth" to launch geth.
root@gp:~/go-ethereum# cd ~
root@gp:~# geth account new --datadir ~/gethDataDir
geth: command not found
root@gp:~#
```

```
root@gp:~# gedit .bashrc

(gedit:3142): Gtk-WARNING **: Calling Inhibit
p.DBus.Error.ServiceUnknown: The name org.gno
by any .service files
```

```
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi
export PATH="$PATH:/usr/local/go/bin"
export PATH="$PATH:/root/go-ethereum/build/bin/geth"

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
```

```
if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi
export PATH="$PATH:/root/go-ethereum/build/bin"
export PATH="$PATH:/usr/local/go/bin"
```

```
root@gp:~# geth account new --datadir ~/gethDataDir
WARN [11-08|22:42:37.826] Sanitizing cache to Go's GC limits       provided=1024
 updated=324
INFO [11-08|22:42:37.828] Maximum peer count                       ETH=25 LES=0
total=25
Your new account is locked with a password. Please give a password. Do not forge
t this password.
Passphrase:
Repeat passphrase:
Address: {fa942ade575cd2b1c10a31aa11ae80fb4ad14d95}
root@gp:~#
```

```
Your new account is locked with a password. Please give a password. Do not forge
t this password.
Passphrase:
Repeat passphrase:
Address: {231b3c4ec0b525e5b250923cc6e109bd8f26ecb5}
root@gp:~# cd gethDataDir/
root@gp:~/gethDataDir# gedit genesis.json

(gedit:2302): Gtk-WARNING **: Calling Inhibit failed:
p.DBus.Error.ServiceUnknown: The name org.gnome.Sessio
by any .service files

** (gedit:2302): WARNING **: Set document metadata fai
adata::gedit-spell-enabled not supported

** (gedit:2302): WARNING **: Set document metadata fai
adata::gedit-encoding not supported
```

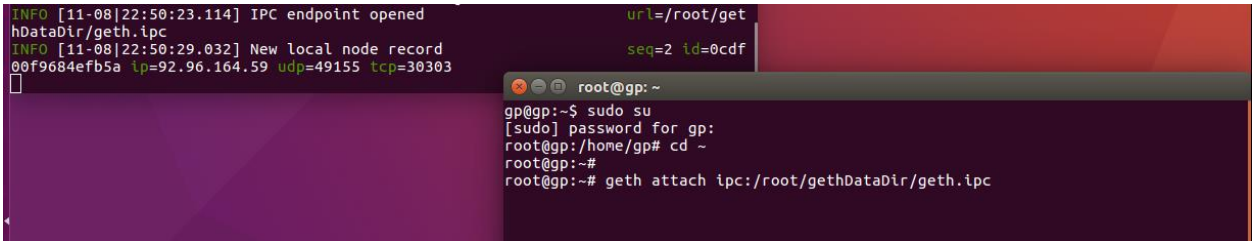
Open ▾ genesis.json ~/gethDataDir

```
{
        "config": {
                "chainId": 15,
                "homesteadBlock": 0,
                "eip155Block": 0,
                "eip158Block": 0
        },
        "difficulty": "1",
        "gasLimit": "2100000",
        "alloc":{
                "fa942ade575cd2b1c10a31aa11ae80fb4ad14d95": {
                        "balance": "300000"
                },

                "231b3c4ec0b525e5b250923cc6e109bd8f26ecb5": {
                        "balance": "500000"
                }
        }
```

```
root@gp:~/gethDataDir# cd ~/gethDataDir
root@gp:~/gethDataDir# geth --datadir ~/gethDataDir/ init genesis.json
WARN [11-08|22:49:30.225] Sanitizing cache to Go's GC limits       provided=1024
 updated=324
INFO [11-08|22:49:30.228] Maximum peer count                       ETH=25 LES=0
```

```
root@gp:~# geth --datadir ~/gethDataDir --networkid 15
WARN [11-08|22:50:22.964] Sanitizing cache to Go's GC limits       provided=1024
 updated=324
INFO [11-08|22:50:22.965] Maximum peer count                       ETH=25 LES=0
total=25
INFO [11-08|22:50:22.966] Starting peer-to-peer node               instance=Geth
/v1.8.18-unstable-144c1c6c/linux-amd64/go1.11.2
INFO [11-08|22:50:22.966] Allocated cache and file handles         database=/roo
```

```
INFO [11-08|22:50:23.114] IPC endpoint opened                      url=/root/get
hDataDir/geth.ipc
INFO [11-08|22:50:29.032] New local node record                    seq=2 id=0cdf
00f9684efb5a ip=92.96.164.59 udp=49155 tcp=30303
```




```
root@gp:~# geth attach ipc:/root/gethDataDir/geth.ipc
WARN [11-08|22:52:41.441] Sanitizing cache to Go's GC limits       provided=1024
 updated=324
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.18-unstable-144c1c6c/linux-amd64/go1.11.2
coinbase: 0xfa942ade575cd2b1c10a31aa11ae80fb4ad14d95
at block: 0 (Thu, 01 Jan 1970 04:00:00 +04)
 datadir: /root/gethDataDir
 modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0
rpc:1.0 txpool:1.0 web3:1.0

> web3.eth.accounts
["0xfa942ade575cd2b1c10a31aa11ae80fb4ad14d95", "0x231b3c4ec0b525e5b250923cc6e109
bd8f26ecb5"]
> web3.eth.getBalance(web3.eth.accounts[0])
300000
> web3.eth.getBalance(web3.eth.accounts[1])
500000
>
```

```
root@gp:~# geth --mine --minerthreads=1 --datadir ~/gethDataDir --networkid 15
```

```
INFO [11-08|23:10:22.847] Successfully sealed new block
alhash=621160…35d48b hash=ddb4c9…9ca1b0 elapsed=77.141ms
INFO [11-08|23:10:22.847] 🔗block reached canonical chain
sh=cb66da…3651b3
INFO [11-08|23:10:22.847] 🔨mined potential block
sh=ddb4c9…9ca1b0
INFO [11-08|23:10:22.848] Commit new mining work
alhash=04c37a…330aec uncles=0 txs=0 gas=0 fees=0 elapsed=104.50
INFO [11-08|23:10:22.913] Successfully sealed new block
alhash=04c37a…330aec hash=3a1455…529fef elapsed=65.619ms
INFO [11-08|23:10:22.913] 🔗block reached canonical chain
sh=2a1d5a…3fc20c
INFO [11-08|23:10:22.913] 🔨mined potential block
sh=3a1455…529fef
INFO [11-08|23:10:22.913] Mining too far in the future
```

```
root@gp:~# geth --networkid 15 --nodiscover --datadir="gethdatadir" --maxpeers=0
 --ipcpath $HOME/.ethereum/geth.ipc
WARN [11-09|09:47:00.426] Sanitizing cache to Go's GC limits       provided=1024
 updated=324
INFO [11-09|09:47:00.438] Maximum peer count                       ETH=0 LES=0 t
otal=0
INFO [11-09|09:47:00.497] Starting peer-to-peer node               instance=Geth
/v1.8.18-unstable-144c1c6c/linux-amd64/go1.11.2
```

Installing wallet

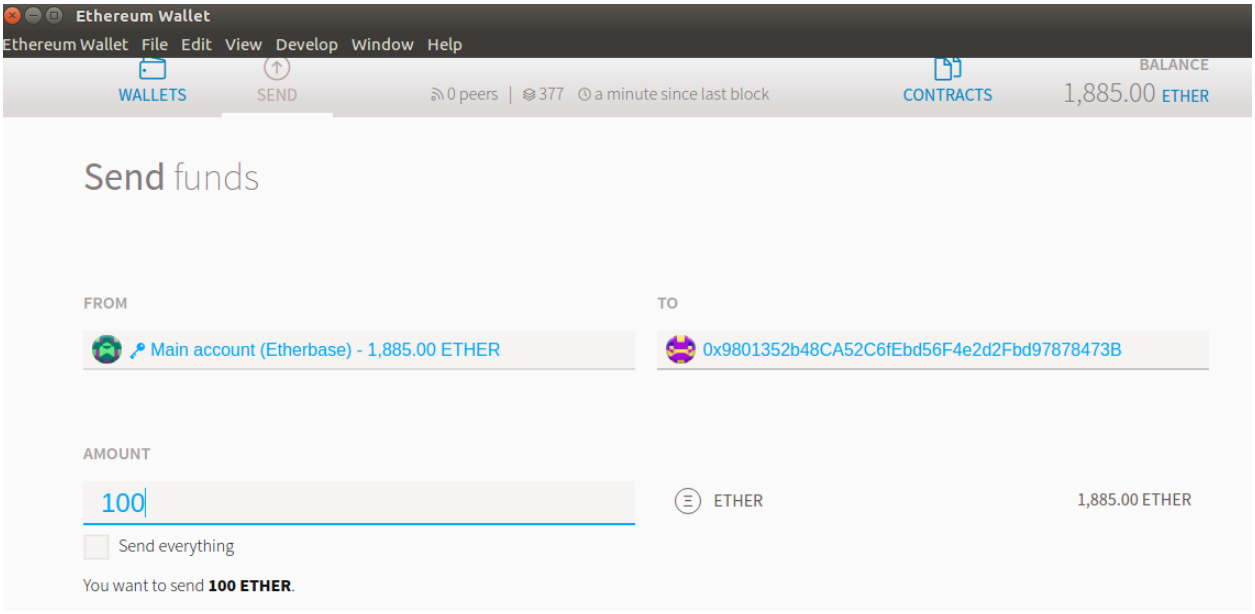
```
root@gp:~# sudo apt-get install software-properties-common
Reading package lists... Done
Building dependency tree
Reading state information... Done
software-properties-common is already the newest version (0.96.20.7).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
root@gp:~#
```

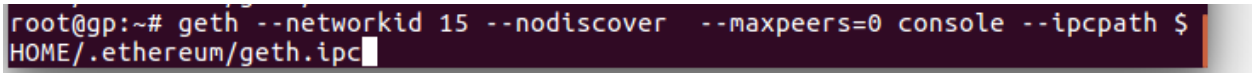
```
root@gp:~# sudo add-apt-repository -y ppa:ethereum/ethereum
gpg: keyring `/tmp/tmpauudv6ni/secring.gpg' created
gpg: keyring `/tmp/tmpauudv6ni/pubring.gpg' created
gpg: requesting key 923F6CA9 from hkp server keyserver.ubuntu.com
gpg: /tmp/tmpauudv6ni/trustdb.gpg: trustdb created
gpg: key 923F6CA9: public key "Launchpad PPA for Ethereum" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
OK
root@gp:~#
```

```
root@gp:~# geth --networkid 15  --datadir="gethDatadir" --maxpeers=0 --ipcpat
h $HOME/.ethereum/geth.ipc
WARN [11-09|10:46:19.315] Sanitizing cache to Go's GC limits       provided=1
024 updated=324
INFO [11-09|10:46:19.316] Maximum peer count                       ETH=0 LES=
0 total=0
INFO [11-09|10:46:19.319] Starting peer-to-peer node               instance=G
eth/v1.8.18-unstable-144c1c6c/linux-amd64/go1.11.2
INFO [11-09|10:46:19.319] Allocated cache and file handles         database=/
root/gethDatadir/geth/chaindata cache=243 handles=512
INFO [11-09|10:46:19.362] Initialised chain configuration          config="{C
```

```
root@gp:~# geth attach ipc:/root/.ethereum/geth.ipc
WARN [11-09|10:49:25.601] Sanitizing cache to Go's GC limits       provided=1024
 updated=324
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.18-unstable-144c1c6c/linux-amd64/go1.11.2
coinbase: 0x8584de806ad292592c84cce8f9e625361428b247
at block: 21 (Fri, 09 Nov 2018 10:46:04 +04)
 datadir: /root/gethDatadir
 modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0
rpc:1.0 txpool:1.0 web3:1.0

> web3.eth.accounts
["0x8584de806ad292592c84cce8f9e625361428b247"]
> web3.eth.getBalance(web3.eth.accounts[0])
10500000000000000000
> web3.eth.getBalance(web3.eth.accounts[0])
10500000000000000000
> web3.eth.accounts
["0x8584de806ad292592c84cce8f9e625361428b247", "0x4c5bab4d55d5d44beb1736218c7c8e
2a3d3d9f50"]
> web3.eth.getBalance(web3.eth.accounts[1])
0
```

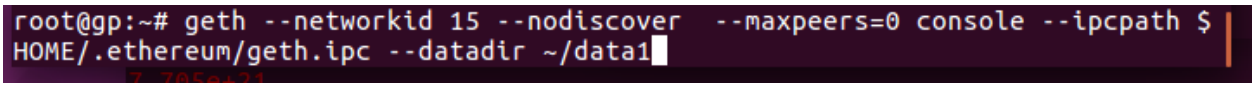
```
root@gp:~# geth attach ipc:/root/.ethereum/geth.ipc
]WARN [11-09|14:28:50.236] Sanitizing cache to Go's GC limits       provided=
  updated=324
]Welcome to the Geth JavaScript console!

]instance: Geth/v1.8.18-unstable-144c1c6c/linux-amd64/go1.11.2
 coinbase: 0x8584de806ad292592c84cce8f9e625361428b247
]at block: 1426 (Fri, 09 Nov 2018 14:28:34 +04)
  datadir: /root/gethDatadir
] modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:
 rpc:1.0 txpool:1.0 web3:1.0
]
 > miner.start(1)
]null
 > miner.stop()
]null
 > miner.start(2)
null
 >
```

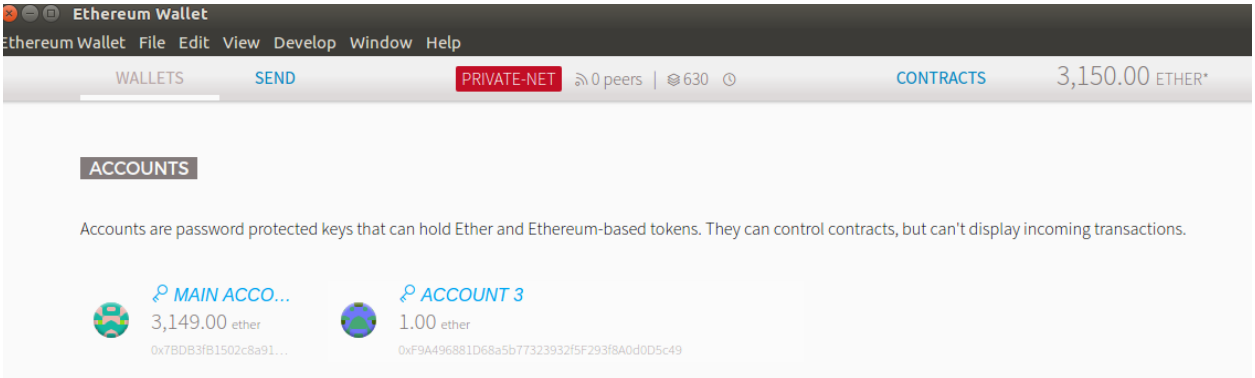


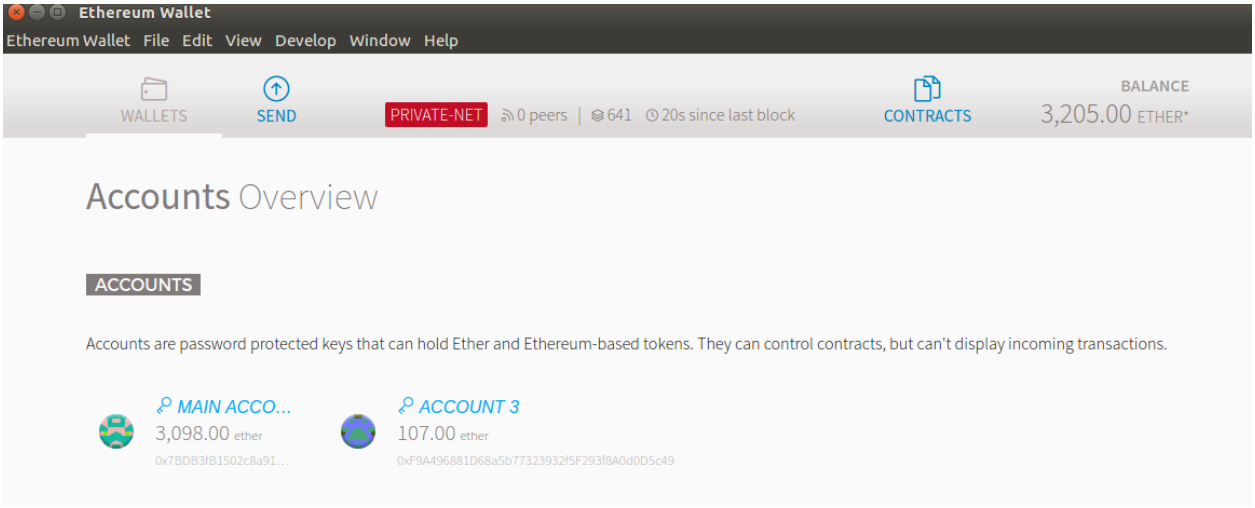


geth --networkid 15 --nodiscover --datadir ~/gethDatadir --maxpeers=0 --ipcpath $HOME/.ethereum/geth.ipc

geth --mine --minerthreads=1 --datadir ~/gethDatadir --networkid 15 --ipcpath $HOME/.ethereum/geth.ipc







Adding additional node

In Node1 where extip is the ip address of node 1

```
root@gp:~# geth --networkid 15 --nodiscover  console --ipcpath $HOME/.ethereum/g
eth.ipc --datadir ~/data1 --nat=extip:192.168.253.142
```

In Node 2

Create a new directory called data21

```
root@gp:~# mkdir data21
root@gp:~# cd ~/data21
```

Copy the genesis file from data1 to data21

```
root@gp:~/data21# cp /root/data1/genesis.json /root/data21
root@gp:~/data21# ls
genesis.json
```

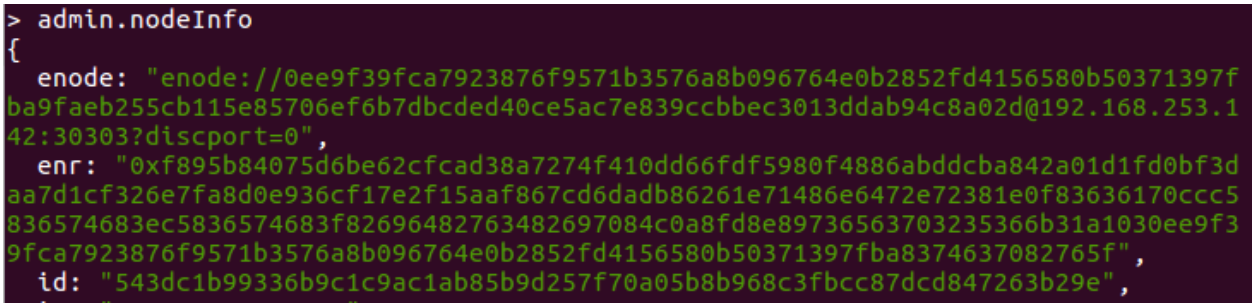
Initialize the genesis file

```
root@gp:~/data21# geth --datadir ~/data21/ init genesis.json
```

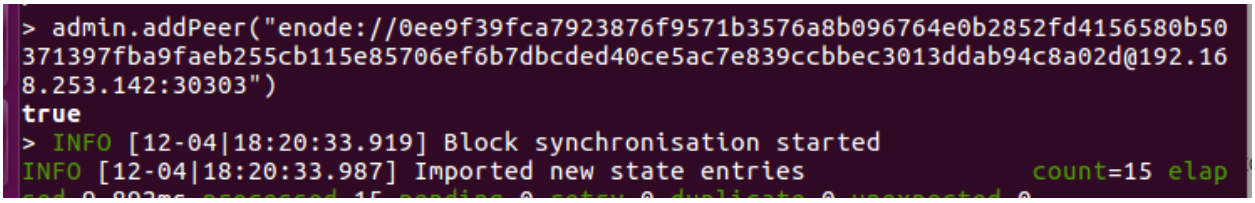
Run geth

```
root@gp:~# geth --datadir ~/data21 --networkid 15 --port 30306 --nodiscover cons
ole --ipcpath $HOME/.ethereum/geth.ipc
```

Now in Node 1, find the enode address



Copy the enode address and add it in node2 as shown below

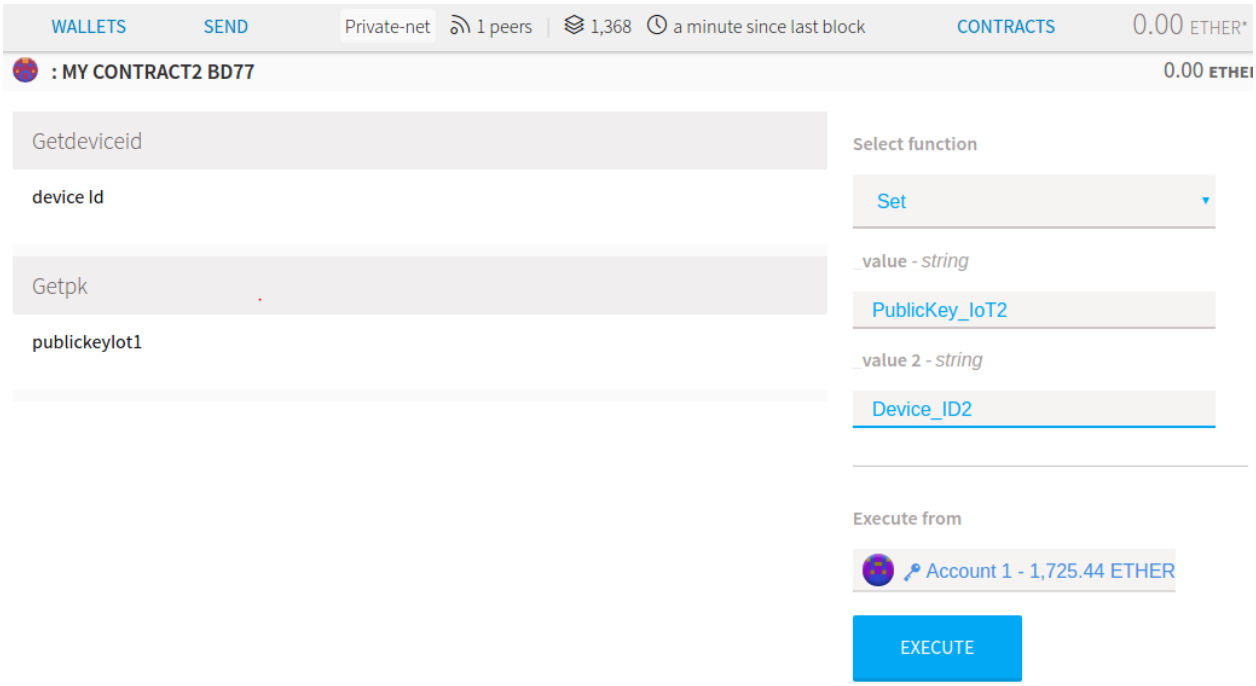


Open ethereum wallet in both the nodes

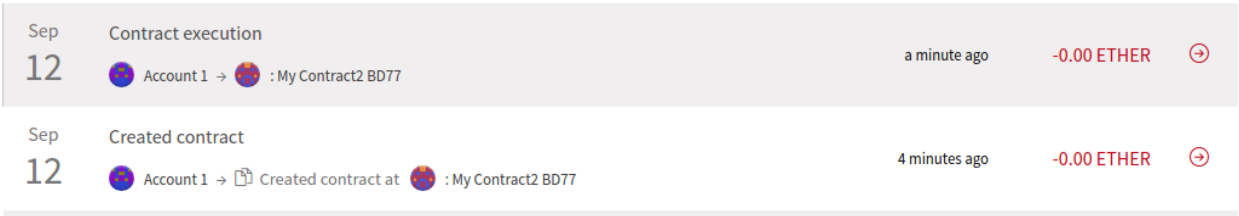
For the smart contract, we deployed, we created two strings value1 and value2 which will store the publickey of IoT and device ID. A node that execute this transaction will be able to store their publickey and device ID.

```
pragma solidity ^0.4.24;

contract MyContract2 {

string value1;

string value2;

constructor() public{

value1 = "publickeyIot1";

value2="device Id";

}

function getpk() public view returns(string){

return value1;

}

function getdeviceid() public view returns(string){

return value2;
```

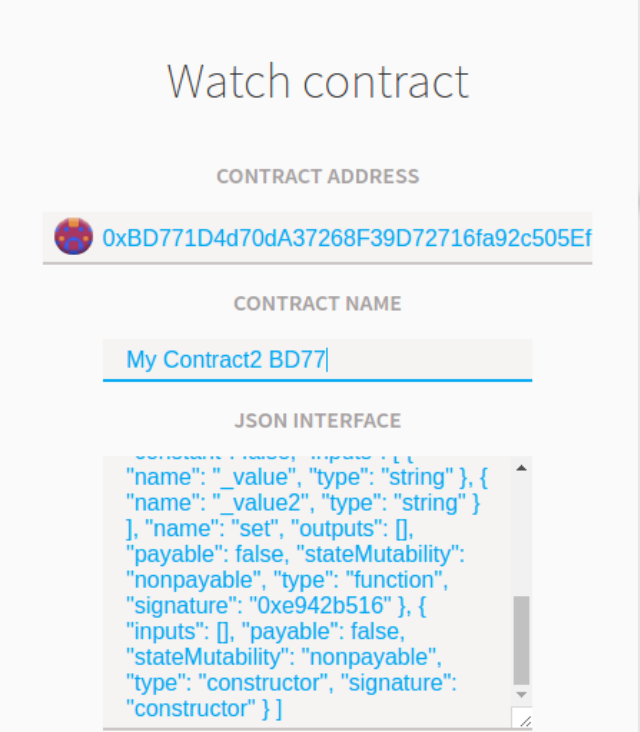
```
}

function set(string _EdgeID, string _ENCtrans) public {

value1 = _EdgeID;

value2=_ENCtrans;

}

}
```



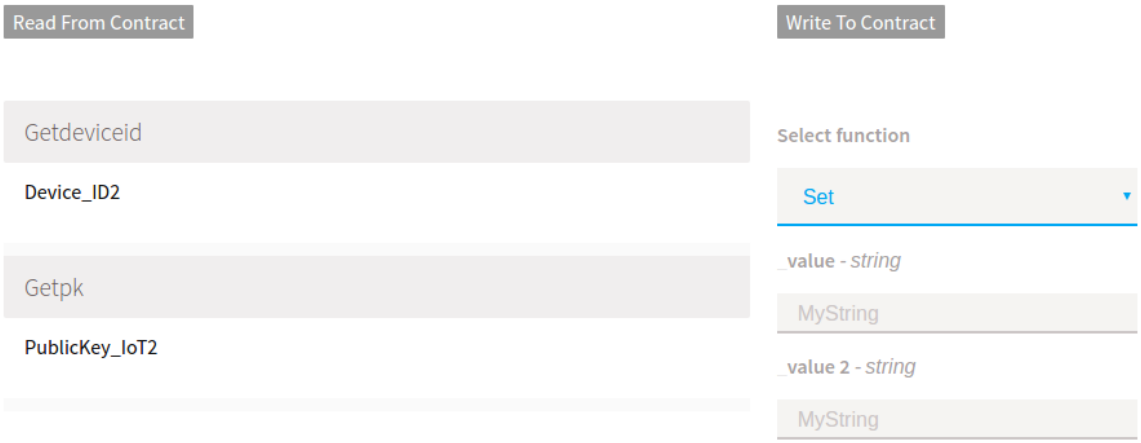
Once the contract is executed but Node1. The contract will be replicated to other nodes.



We then add the contract address to other nodes.



Hence the other nodes were able to retrieve the data in the contract. In addition they can set their public key and device Id to the database.



When the contract is executed, mining process was initiate. It verifies the transactions, and all transactions were added to the block and created a new block.