UTILIZATION OF BLOCKCHAIN TO INCREASE E-VOTING SECURITY

By

MD. Navid Razzaque

Roll: 1707013

Department of Computer Science and Engineering

Khulna University of Engineering & Technology

Khulna 9203, Bangladesh

February, 2023

Utilization of Blockchain to Enhance E-voting Security

By

MD. Navid Razzaque
Roll: 1707013

A thesis submitted in partial fulfillment of the requirements for the degree of
"Bachelor of Science in Computer Science & Engineering"

Supervisor:

Dr. Kazi Md. Rokibul Alam
Professor
Department of Computer Science and Engineering
Khulna University of Engineering & Technology
Khulna, Bangladesh

_____
Signature

Department of Computer Science and Engineering
Khulna University of Engineering & Technology
Khulna 9203, Bangladesh
February, 2023

Acknowledgment

All praise belongs to the Almighty Allah, whose mercy and blessing enabled me to fairly accomplish **our** thesis work. After that, I humbly acknowledge the recommendations, advice, guidance, and sincere cooperation of Dr. Kazi Md. Rokibul Alam, Professor, Department of Computer Science and Engineering, Khulna University of Engineering & Technology, under whose supervision this study was successfully accomplished. His intellectual guidance, inspiration, and direction inspire trust in us and scientific research demands a lot of learning and application work as well as a comprehensive perspective on issues from numerous aspects. The faculty, administrators and employees of the Department of Computer Science and Engineering have our sincere gratitude for their tireless assistance. Lastly we want to express our gratitude to our friends and family members for their constant support.

Abstract

Elections are the cornerstone of democratic societies, as they provide citizens a way to select and hold responsible their leaders. In recent years, e-voting has emerged as a promising alternative to conventional voting methods, offering advantages such as increased accessibility, convenience, and efficiency. However, e-voting also presents significant security challenges including the possibility of fraud, tampering, and hacking. To address these challenges, the utilization of blockchain technology in e-voting has been proposed as a solution. Our proposed method incorporates the use of smart contracts in an Ethereum based platform to conduct the whole process. This ensures the data kept inside the network will be safe from being tampered or manipulated. Inclusion of multi-authority based decisions with public key cryptography in each major phases provide additional security on top of an already secure platform. This added procedure ensures that every phase is secure all the time and also achieves the trust of the voters and achieves the transparency all the time.

Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

1.1 Background

Elections are a crucial aspect of democratic societies as they provide citizens with the opportunity to choose their leaders and make their voices heard. Through elections, the public has the power to determine the direction and policies of their country and hold elected officials accountable for their actions. Elections also ensure that power is transferred peacefully and in a democratic manner, rather than through violence or authoritarian rule. This helps maintain stability and legitimacy in a country, as well as promoting a culture of respect for the rule of law and human rights. Conventional voting system requires the voters to cast their vote in a ballot paper. The voters have to be present in the voting area and the queue is often long. So it consumes a lot of time and effort. But this does not guarantee that their vote is properly counted. The vote might get destroyed or it might be manipulated. Electronic voting (e-voting) has become a popular alternative to conventional paper-based voting system due to its convenience and ability to speed up the election process. Due to this the voters can cast their vote not being present in the queue. They can even cast their vote outside their residence. Along with the merits there also comes detrimental issues. With the increasing dependency on technology, the security of e-voting systems has become a critical issue. Cyberattacks, hacking, and data breaches have all threatened the integrity of e-voting systems and raised concerns about their reliability. A decentralized system is thought to be a solution to these problems.

Blockchain technology, with its decentralized, secure and transparent nature, can be treated as a solution to enhance the security of e-voting systems. Blockchain could provide a feasible solution in this scenario. Blockchain is a shared database or ledger. In here data are stored in blocks. The blocks are connected through cryptography. All of the blocks are connected with the previous blocks. For this reason, even if an attacker tries to change a block, then the rest of the blocks will be changed too [14]. Hence, the blockchain network

will revert it back to the original form. The blockchain is distributed in a peer to peer network. If an attacker tries and tampers the block of a certain network then all the other networks will see that something is changed and they will revert the chain back to its original form. A blockchain-based e-voting system can provide a secure and tamper-proof record of every vote cast, ensuring the integrity and accuracy of election results. Hence the importance of secure and trustworthy voting systems cannot be overstated. Moreover, the decentralized nature of blockchain eliminates the need for a central authority to manage the voting process, reducing the risk of manipulation.

1.2 Motivation

Elections serve as the cornerstone of democratic societies and their integrity must be maintained to ensure that the will of the people is accurately reflected. In recent years, the widespread adoption of electronic voting (e-voting) has raised concerns about the security and reliability of these systems. Hence the importance of secure and trustworthy voting systems cannot be overstated. In light of these concerns, there is a growing interest in the development of secure and transparent e-voting systems that can guarantee the accuracy and integrity of election results. This is where Blockchain will play a vital role. The utilization of blockchain in e-voting systems has the potential to revolutionize the way elections are conducted. This study aims to examine the feasibility of using blockchain technology in e-voting systems and propose a practical and scalable solution for its implementation. This thesis will focus on addressing the security concerns surrounding e-voting systems and to contribute to the development of secure and transparent voting systems that can be relied upon.

1.3 Problem Statement

Though E-voting systems have provided a really good platform for conducting elections but the chances of it getting hacked/manipulated are huge which cannot be neglected. It mainly occurs due to the centralization of authority/database. The attackers can easily attack and change the contents of the database these days. This is where a decentralized system like Blockchain will come in handy to ensure the integrity of the election. A blockchain-based e-voting system can provide a secure and tamper-proof record of every vote cast, ensuring the integrity and accuracy of election results. However, there is a lack of practical and

scalable solutions for the implementation of blockchain technology in e-voting systems. The main aim is to guarantee the accuracy and integrity of election results. The study will examine the challenges faced by current blockchain base e-voting systems and analyze the potential of blockchain technology in enhancing the security of e-voting systems.

1.4 Objectives

In this thesis, a practical and highly scalable solution for the implementation of blockchain in e-voting systems is proposed. The main objectives are as follows:

- To review the existing e-voting systems and the challenges they face in terms of security and reliability.
- To develop a blockchain based e-voting system for conducting elections
- To develop a multi-authority based administration system
- To include additional cryptography in the current system to increase the security layer of the blockchain system even more
- To evaluate the proposed solution through simulations and experiments to demonstrate its effectiveness in enhancing the security of e-voting systems.

1.5 Scope of the Thesis

As it was mentioned earlier, the primary objective of this thesis is to develop an efficient and highly scalable blockchain based e-voting system. The proposed system allows the inclusion of multiple authorities in the case of permitting the start of the election. The system is optimized in such a way so that it can handle high peers in less Ethereum gas cost. This study will also provide valuable insights into the future of secure and transparent voting systems. The main vision was to research and develop a voting system which will reflect the will of the people.

1.6 Contribution of the Thesis

To counter the drawbacks of e-voting systems, current blockchain based e-voting systems have already been in discussion and developed. But they also have some limitations. The limitations our proposed technique is able to overcome are as follows:

- Exclusion of single authority based e-voting system which removes the partiality and also gains the trust of the voters.
- Introduction of multi-authority permission based system with El Gamal cryptography.
- Build the whole system through a single smart contract. The smart contract will be optimized in such a way so that the computational cost will be as less as possible.

1.7 Organization of the Thesis

The rest of the thesis is organized as follows:

Chapter 2 discusses about the earlier works regarding blockchain based e-voting systems. This includes the works done in other contexts with potentially applicable in our scenario. A brief discussion about their limitations is also given in this chapter.

Chapter 3 discusses about the theoretical knowledge necessary for our development in details. The relevant technologies and tools which have connections with our proposed method is discussed in details.

Chapter 4 describes our proposed technique in details. This includes how different components of the model works with the technique as a whole. This also describes the implementation process of our system and the prototyping we have developed to simulate the e-voting system.

Chapter 5 focuses on the experimental studies of our proposed system. It also analyses our system with other existing systems and explains the improvements we have done over the existing designs.

Chapter 6 draws the conclusion of our proposed system. It also addresses some limitations and future works which can be done for improving the system

CHAPTER 2

Literature Review

2.1 Introduction

The field of electronic voting (E-voting) has been an active area of research for several decades due to the importance it carries in every aspect of a nation. Many decentralized ways have been proposed to eliminate the shortcomings of the e-voting system. But they did not perform as per expectation. On the other hand, the need for secure and transparent voting systems has become increasingly important in recent years as the use of electronic systems has become more widespread. Then comes Blockchain. In this section, several researches related to our work is discussed.

2.2 Previous Related Works

2.2.1 Single Authority Based Approaches

Kshetri et al. [1] conducted a study on the use of blockchain technology for secure and transparent voting systems. Their proposed method found that blockchain technology could be used to create a tamper-proof record of voting data and to ensure the confidentiality and integrity of the voting process. The study concluded that blockchain technology could be a promising solution for enhancing the security of E-voting systems. Though this work was not suitable for high-scale implementation, but its theoretical knowledge laid grounds for future implementation.

Xu et al. [2] conducted a study on the security and privacy of blockchain-based E-voting systems. The authors proposed such a blockchain based system in which it could be used to prevent vote manipulation and to ensure the anonymity and privacy of voters. The study concluded that blockchain technology has the potential to significantly enhance the security and privacy of E-voting systems.

Ali et al. [5] created a small scaled public network for e-voting. The system was developed in Ethereum Blockchain network. They deployed smart contracts to run several operations. They made sure of transparency, eligibility, provability in the platform.Though their work was remarkable but they had some limitations. Their system was designed only for a small scaled networks. They used custom private networks for testing out the system. The users in the network were forced to download and store all of the networks in their system to run the whole process.

Ghayoori et al. [4] proposed a blockchain-based e-voting system that provides security, transparency and accountability in political elections. The proposed system utilizes the characteristics of blockchain such as immutability and decentralization to prevent vote tampering and guarantee the confidentiality of voters. The results of the system are analyzed and compared to traditional e-voting systems, showing its advantages in terms of security and transparency.

Gupta et al. [6] worked on the design and implementation of a blockchain-based E-voting system. The authors found that blockchain technology could be used to create a secure and transparent voting system that could be easily audited and verified.

2.2.2 Encryption Based Approaches

Su et al. [3] proposed a secure and efficient electronic voting system using blockchain technology. The authors aim to address the security and privacy concerns of traditional e-voting systems through the use of blockchain, which offers a tamper-proof and transparent platform for secure online voting. The proposed system includes several security mechanisms such as encrypted voting records, digital signatures, and access control to ensure the confidentiality and authenticity of the voting process. The authors also present a theoretical analysis of the system's security and efficiency.

Chiara et al. [7] created an online auction system based on Ethereum smart contracts. In their work they used smart contracts to provide transparency and avoid cheating auctioneers. Though their work has provided a way for implementing smart contracts in the blockchain but their work is used only for a small number of people.

Patil et al. [8] presents a blockchain-based e-voting system that aims to improve the security and efficiency of the voting process. The authors propose a system that uses blockchain technology to ensure the transparency, integrity, and confidentiality of the voting process. The system also includes features such as digital signature, encryption, and smart contracts to further enhance its security. The authors evaluated the performance of their proposed system and showed that it is capable of providing a secure and efficient e-voting solution.

Fan et al. [9] presents an anonymous blockchain-based e-voting system with privacy protection. The authors focus on the use of blockchain technology to enhance the privacy and security of electronic voting systems. The proposed system is designed to protect the anonymity of voters and to prevent unauthorized access to the voting process. The authors provide a detailed analysis of the system's security and privacy features, and they evaluate the performance of the system using a variety of performance metrics. The results of the study indicate that the system is able to provide high levels of security and privacy while maintaining a high level of efficiency and scalability.

Vikas et al. [10] proposed a better secure system with enhanced security features. The authors proposed a blockchain-based e-voting system that incorporates enhanced security features to prevent tampering and ensure transparency. The system uses a combination of digital signatures and encryption to ensure the authenticity and confidentiality of the voting process. Due to combinations of digital signatures and encryption being used, the system is more secure than most of the system. The authors carried out a series of experiments to evaluate the performance of their proposed system and showed that it provides a high level of security and transparency compared to traditional e-voting systems. But the major drawback of this method is the complexity and computation cost behind the whole operation.

Hsiao et al. [11] proposed a blockchain-based secure e-voting system using homomorphic encryption. The authors aimed to address the security concerns of traditional e-voting systems such as tampering and manipulation of voting results, privacy issues, and lack of transparency. In this study, the authors used homomorphic encryption to encrypt the vote data, which allows for the computation to be performed on encrypted data without decrypting it. This approach offers higher security and privacy compared to traditional encryption methods. The authors used the blockchain technology to create a secure and

transparent voting system where each vote is recorded as a transaction on the blockchain, and the results are made publicly available. The study demonstrated the feasibility of the proposed system, and the results showed improved security, privacy, and transparency compared to traditional e-voting systems.

2.3 Summary

In this chapter, out of many methods, some of them are mentioned and discussed. The reviewed systems mostly emphasized on the theoretical perspective of blockchain being implemented for e-voting system. Some of them laid the foundation for future implementations. Then in those implementations, variety of works were visible. Many authors proposed different kinds of implementation techniques in different kinds of platforms. Some were able to produce high security in trade of high computation cost. Some provided the basic property of security needed for their purpose. Most of the systems are not highly scalable. Some used custom networks which were forced to download in the local storage.

But one thing is noticeable in most cases that in all the systems which are implemented requires an authority to start the whole process and execute the whole process. But that users' account could be hacked or manipulated. This doubt could lead to confusion among the voters. Though the whole system is decentralized, but this sense of doubt could be eliminated if multiple authorities would exist in the system and they would grant permission to that requested person. Then that systems' security would have enhanced even more as it would require multiple layers of permission for the person to achieve authority.

CHAPTER 3

Building Blocks

3.1 Blockchain

Blockchain is a distributed P2P (peer to peer) technology that provides a secure and transparent way to store and transfer data. The ledgers used in this system are immutable. So they cannot be tampered or manipulated. Though it was originally created as a core component of cryptocurrency, but now it has been adapted in various scenarios [21]. In a blockchain network the data are stored in a block. Each block is distinguished from other by its unique hash. Each hash is generated using the SHA 256 algorithm. The blocks are linked together to form a chain. Each block consists of data(transactions), nonce, block number, hash of its previous block. When a new block is added to the network, the previous hash is included in the new block which as a result creates an unalterable link between blocks. When an attacker will try to manipulate, say change a blocks' content, then due to the characteristics of the hashing algorithm, the block will have a new hash. But the next blocks will show error here as the next blocks previous hash value will not match. As a result, the attacker will fail to change the system. Figure 3.1 shows a general structure of the blocks in a blockchain network.

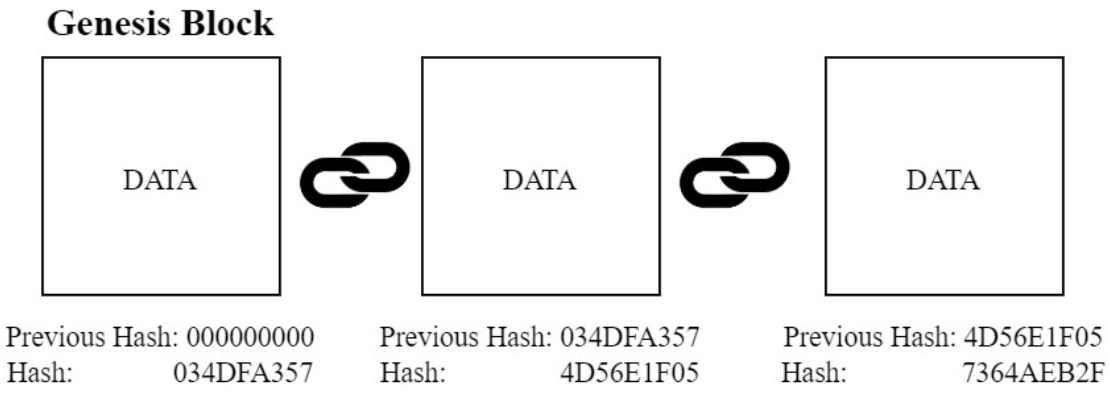

Figure 3.1: Structure of Blockchain

Blockchain mainly consists of 5 pillars. Together they form a structure which turns out to be unalterable, unbreakable. The 5 pillars are as follows:

- **Hash Cryptography:** We human beings have unique fingerprints which are used in various purposes of our life. The fingerprints are different from each other. That's why we can use them as an identifier. Similarly to distinguish each blocks in the blockchain network SHA-256(Secure Hashing Algorithm) is used [15]. It is a hashing algorithm implemented for creating unique hexadecimal numbers. The hash value will be of 256 bits (64 characters). This algorithm is useful because the numbers it creates are "One-Way" which means once we have created the number from the algorithm, we cannot turn it back to the original number. The numbers are also "Deterministic", "Fast Computable". The hashing technique also has the "Avalanche Effect" which means change in digit will result in change in many places in the hashed number [13]. A general overview of a Hash Function is shown in Figure 3.2.

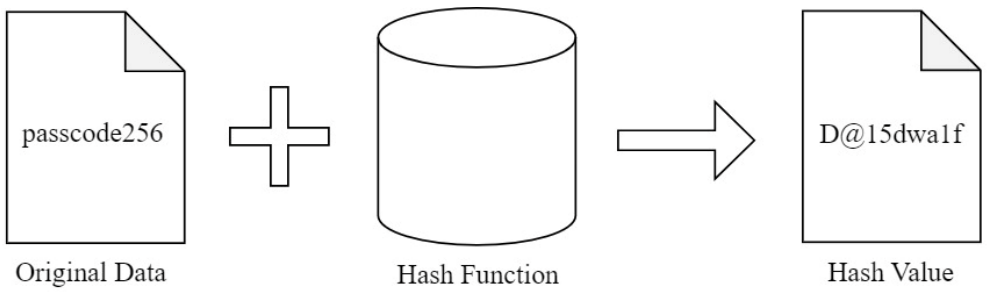

Figure 3.2: Mechanism of a Hash Function

- **Distributed P2P (Peer to Peer) Network:** The blockchain network consists of several nodes(devices). The data are stored in all of the designated nodes in the network. If one block is added to one of the nodes then that block is added to the rest of the nodes in the network too. But adding a block to a network is not that easy as to add a block it has to abide by some strong set of rules. Only if they pass all of them then they can be added to the network.
- **Immutable Ledger:** The blocks' contents are considered as ledgers. The ledgers which will reside in the blockchain network are immutable. The reason is that say a block is added to the network. Now an attacker tries and succeeds to change the

block and also the rest of the blocks in the network. Now the thing which will happen is as the blockchain is a distributed peer to peer network, the other nodes(devices) will see the change in the blocks with their copy. When majority of the nodes in the network will see this change, due to the consensus protocol they will change the corrupted copy of blocks with their own copy of blocks.

- **Mining:** In blockchain, the blocks mainly contains data in the form of transactions. The miners work hard to add these blocks(with transactions) into a network. They require high-powered technologies which is used to solve highly complex mathematical problems. Say a target is set. The miners then try to create a block achieving that target. When they will be able to create a block achieving the target, they will be able to add that block into the network. Thus they will be rewarded. These problems are "Hard to solve, Easy to verify".
- **Consensus Protocol:** In blockchain, there consists many consensus protocols like PoW(Proof of Work), PoS(Proof of Stake) etc. Consensus Protocols are used to maintain the harmony in the blockchain network. Due to it, it protects the network from the attacker. It also provides a solution for the "Competing Chain" problem. Say in one instance, two different blocks are added to two different nodes in the network. The copy of blocks will propagate to its nearby nodes. Now the nodes will have two options to accept. They will now wait until the next block gets added. When the next block will be added by one of nodes. Then the network will accept that nodes' copy of blocks and reject the other nodes' block. That block will then be recognized as "Orphan Block".

Therefore, it can be justified that due to all of the above characteristics, blockchain can be implemented for storing and accessing different kinds of information in a secure way. An overview of Data being encapsulated, transferred and stored is shown in Figure 3.3.

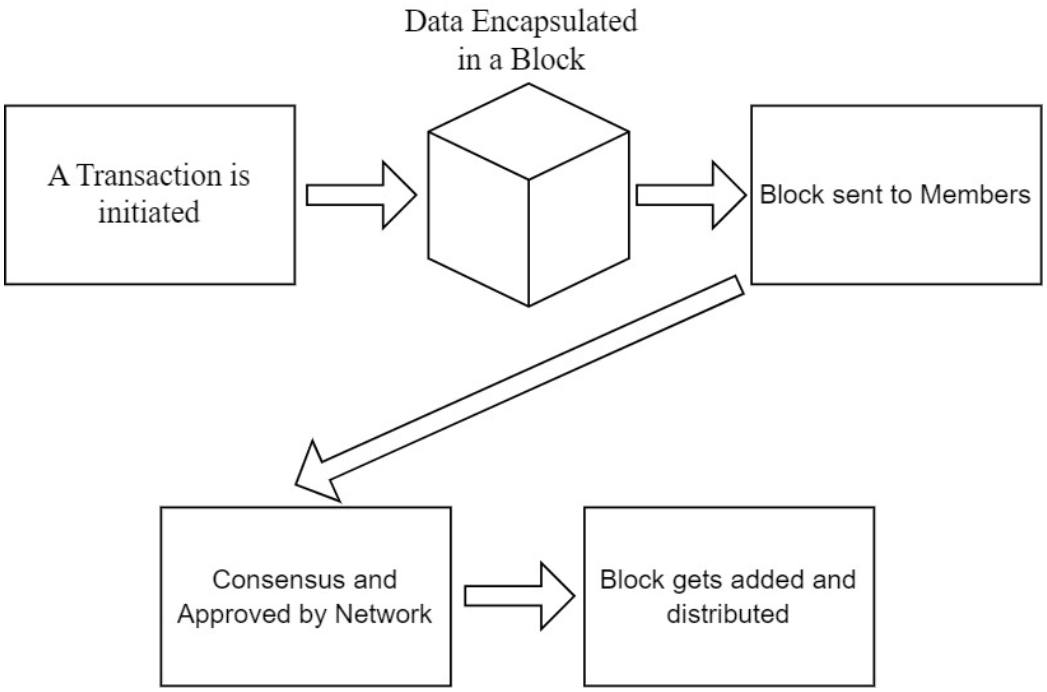

Figure 3.3: Data Flow Diagram of a Blockchain Network

3.2 Ethereum

Ethereum is a decentralized blockchain platform that provides a powerful infrastructure for the creation of decentralized applications (dApps). It was introduced in 2013 by Vitalik Buterin and has since become one of the most popular blockchain platforms, with a wide range of use cases and applications [16]. Ethereum extends the capabilities of blockchain beyond the simple storage and transfer of data, by allowing for the creation of smart contracts. Smart contracts are written on top of this platform which are cryptographic blocks that store value and can only be unlocked if certain conditions are met. So developers are able to create automated, transparent, and tamper-proof applications on the Ethereum network. Ethereum also has its own cryptocurrency, Ether (ETH), which is used to power transactions and pay for the computational resources needed to run dApps on the Ethereum network. This allows developers to create decentralized applications that are not controlled by any central authority, providing greater security, transparency, and decentralization. So

like data(transactions), smart contracts are also added to the blocks. Once it gets added to one block, it gets copied to other blocks too.

Ethereum also has a thriving developer community and a strong ecosystem of tools and resources for developers. This includes the Ethereum Development Network (EDN), which provides developers with a test environment for developing and testing dApps, and the Ethereum Virtual Machine (EVM), which is the runtime environment for executing smart contracts on the Ethereum network. EVM can be considered as a computer inside our computer. More on this is discussed in the later section. The potential of Ethereum to support complex applications and its strong developer community make it an attractive choice for the development of secure and transparent e-voting systems.

3.3 Smart Contract

A smart contract is a self-executing contract that automatically executes when specific conditions are met. It is a piece of code that is stored on the blockchain and runs on the blockchain network, providing a secure and transparent way to enforce the terms of a contract without the need for intermediaries. The process of smart contract's various activities with a blockchain network is shown in Figure 3.4. It is based on a Turing-complete language called Solidity. It is different from bitcoin-script because it contains loops. Smart contracts are basically programs which executes when certain conditions are met. Thus the smart contracts are deterministic. When they are deployed in the network they obtain their own address. With help of smart contract, the data can be stored and transferred.

Smart Contracts are basically programs. So a question might come into mind that programs can be manipulated to harm the device, the program might also contain some coding error which might lead to infinite loop problem. But all of these problems are solved in EVM. As we mentioned earlier, EVM is like a computer inside of a computer. So all of the programs/smart contracts running on the machine is encapsulated. So anything done inside there will not have any effect on the actual computer(device). This allows the devices on the blockchain network to successfully run the smart contracts. Secondly comes the concept of "Ethereum Gas Cost". When we try to drive a car, we drive it using gas. To purchase gas we need to bear some cost. That concept is used here too. As we said Smart contracts are basically programs, so inside a program there are many operations like logical operations,

bitwise operations, mathematical operations etc. All of these operations require some amount of ethers to get executed. This greatly motivates the developers to write code efficiently so that least amount of ether is used. Say if infinite loop problem occurs, then for each loop operation the contract will cut ether. So there will come a time when the ether will run out. That's how the program will not get hung or stuck. This concept of ethereum gas cost also motivates the developers to avoid writing big blocks of codes.

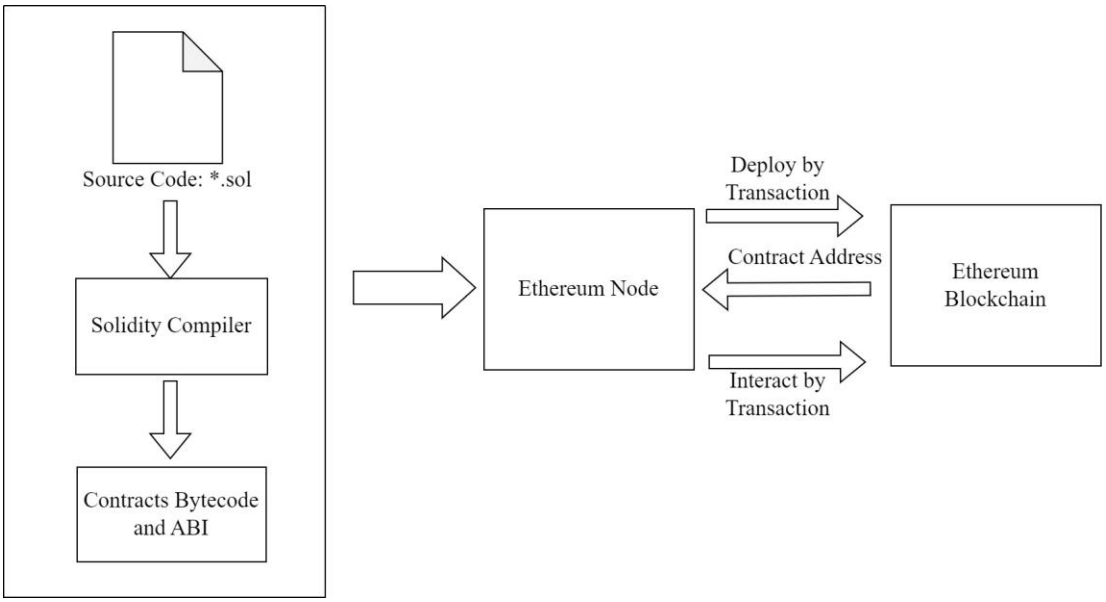

Figure 3.4: The process of smart contract's development, deployment, and interaction with the blockchain

3.4 Solidity

Solidity is a high-level programming language that is used to write smart contracts for the Ethereum blockchain [17]. It was created specifically for the Ethereum platform and is designed to be easy to use and understandable for developers, while still providing a high degree of security and reliability. Solidity is a statically typed, contract-oriented programming language, meaning that developers can create and deploy self-executing contracts on the Ethereum blockchain using Solidity code. The contracts written in Solidity are compiled into machine-readable code, which is then executed by the Ethereum Virtual

Machine (EVM). The contracts are structurally similar to the Classes of object-oriented programming. Solidity defines unique variables like “msg”, “block”, “tx” and others that are always present in the global namespace and have properties that allows users to access information about an invocation-transaction and the blockchain. Modifiers are a feature of Solidity that is very useful. Modifiers are enclosed unit of code that augments functions in order to change the flow of execution of the code. The purpose of this method is to eliminate conditional routes in function bodies using the condition-oriented programming (COP) paradigm. Modifiers are used to quickly change the behavior of functions and are specified after the function name in a whitespace-separated list. The modifiers body is the new function body, with ‘_’ substituted with the original function body. There also consists “Events” which are fired in Solidity. User interfaces and applications can listen for those events without incurring significant costs. Solidity is also designed to be modular and scalable, making it easy for developers to create and deploy complex contracts on the Ethereum network. It includes a number of libraries and tools, such as the Remix development environment, that provide developers with a comprehensive development environment for writing, testing, and deploying smart contracts on the Ethereum network.

3.5 Remix IDE

Remix is a web-based integrated development environment (IDE) for writing, testing, and deploying smart contracts on the Ethereum blockchain. It is a powerful tool for developers that provides a comprehensive development environment for writing, testing, and deploying smart contracts written in Solidity. Remix is designed to be user-friendly, with a simple and intuitive interface that makes it easy for developers to write and test their smart contracts. It includes a number of features that streamline the development process, such as syntax highlighting, code completion, and error checking. One of the key benefits of Remix is its ability to provide a safe and secure environment for testing and deploying smart contracts. Developers can write and test their contracts in a simulated environment, allowing them to identify and fix any errors or vulnerabilities before deploying the contract to the Ethereum network. This helps to ensure the security and reliability of the contract, reducing the risk of bugs or vulnerabilities that could compromise the contract once it is deployed. If the programs run correctly then they can be assured that it will run okay in the main network also. Remix also provides a number of tools and resources that help developers to optimize their contracts for performance and efficiency. For example, it includes a number of

performance profiling and debugging tools that help developers to identify and resolve performance issues, ensuring that the contract runs smoothly and efficiently on the Ethereum network. Remix has its own network like Remix VM (London). Apart from that it allows users to run using their Metamask Account. The metamask account will be connected to the main ethereum network or local blockchain network (ganache). Every user will be attached with an account also known as public/wallet address. Through this accounts certain users will be able to deploy smart contracts and run certain operations on the network. For each deploy, operations certain amount of ether will be cut from their account. Thanks to Remix's really nice user friendly interface, the users will be able to see their account, amount and the functions available for them to execute in the network.

3.6 Metamask

MetaMask is a browser extension and mobile wallet that enables users to interact with decentralized applications (dapps) on the Ethereum blockchain [19]. It provides a secure and user-friendly interface for users to store, manage, and send Ether (ETH) and other Ethereum-based tokens, as well as interact with dapps on the Ethereum network. With the help of it the user will be able to connect to the ethereum network with the browser. It allows users to interact with the dapps and smart contracts without the need of running a full ethereum node on their device. This also lets them engage with the network without understanding the technical details of the underlying technology. Metamask uses a hierarchichal deterministic wallet which generates a unique set of public and private keys for each user. It provides the users with a seed Phase that can be used to recover their funds in case they lose it. It supports not only Ethereum mainnet but also various testnets. It allows users to experiment with different blockchain environments. Of all wallets available till date, metamask is one of the most secure wallets out there. Till date it has no records of getting overridden or getting hacked. Hence it is a favored choice for both individual users and developers. Figure 3.5 shows the process of usage and interaction of Metamask with a blockchain network.

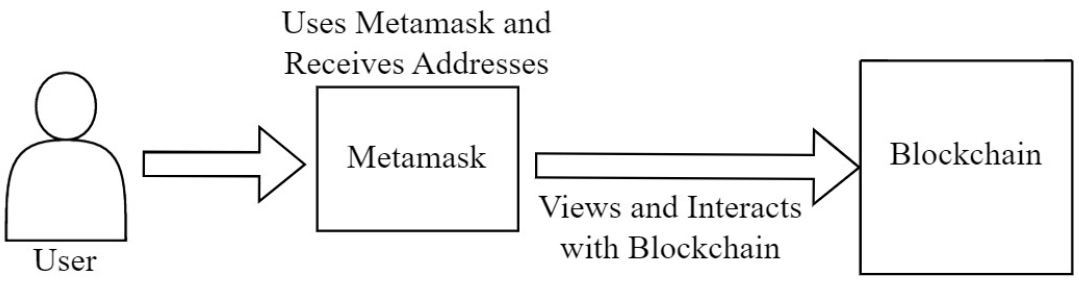

Figure 3.5: The process of using and interacting blockchain via Metamask

3.7 Ganache

Ganache is a personal/local blockchain for Ethereum development that is used to test and deploy smart contracts [18]. This is simulator which gives an essence of real blockchain network that runs on a user's machine. It provides a simulated Ethereum environment for developers to test their smart contracts before deploying them to the main Ethereum network. Ganache offers a multitude of features that make it an indispensable tool for Ethereum development. One such feature is the automatic generation of test accounts with assigned balances, which enables developers to perform simulations and test transactions effortlessly. Additionally, Ganache features a web-based graphical user interface that allows developers to monitor the blockchain's state and keep track of the performance of smart contracts. Through Ganache the users will be able to keep track of the whole network. They will be able to see how many blocks are created and how they are linked, see if the block contains any smart contracts, see which address(account) sent transactions to which address. They will also be able to check other details like the transaction hash, block hash etc. It allows the developers to control the speed and behavior of the blockchain which leads them to simulate different scenarios and test their smart contracts in a controlled environment. This feature helps to identify any issue and fix them before deploying them in the main network. Therefore, it increases the overall reliability and security of the system.

3.8 ElGamal Encryption

Principle of El-Gamal Cryptosystem

El-Gamal encryption is a public-key cryptography. It uses asymmetric key encryption for communicating between two parties and encrypting the message. It is based on a discrete logarithm problem

Public and Private Key Generation:

El-Gamal system consists of two public keys and one private key. Say the public keys are denoted as *y* and *g* and the private key is denoted as *x*. Figure 4.2 helps to show the process of key generation in a simpler fashion. Steps to generate them are as follows:

- Choose a large Prime Number, *p*
- Now generate *g* from *p*. *g* is the generator which is also the first encryption key (public key)
- Choose a private key, *x* where it meets the condition: $1 < x < p$
- Generate second encryption key (public key) using,

$$y = g^x \bmod p$$

- So the public key will be $\{g,y\}$ and private key will be *x*.

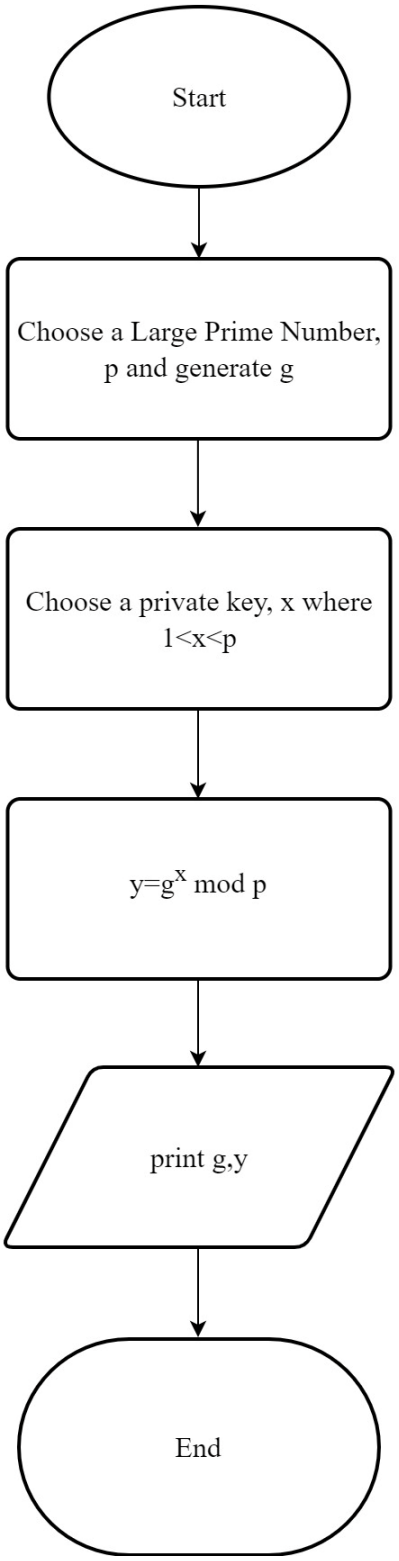

Figure 3.6: Flowchart of Key Generation in El-Gamal

Encryption:

In the process of encryption, two cipher-texts will be generated for one single message. The process is shown below:

$$C_1 = g^r \bmod p$$

$$C_2 = m*y^r \bmod p$$

Here,

r = secret random integer of the sender

m = the message of the sender

C_1 = first cipher-text

C_2 = second cipher-text

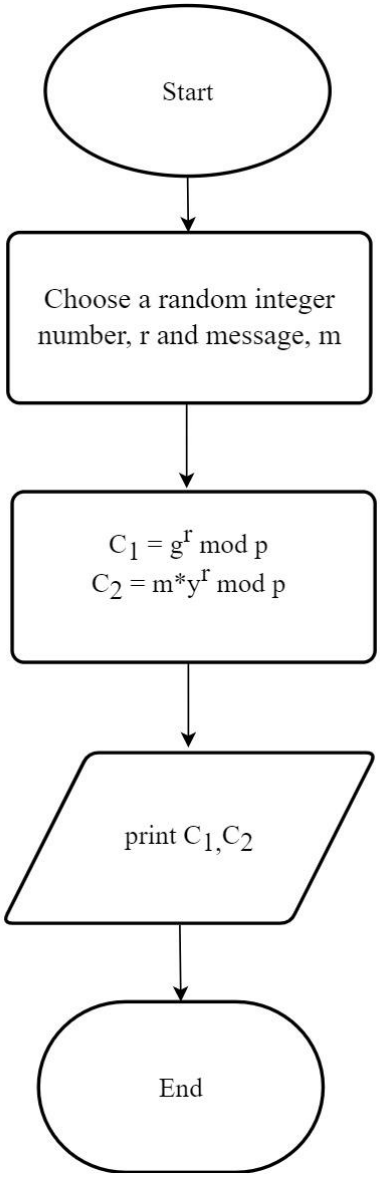

Figure 3.7: Flowchart of Encryption in El-Gamal

Decryption:

In this process, the cipher-texts will be used to a message. The process is shown below:

$$\frac{C_2}{C_1^x} = \frac{m.y^r}{(g^r)^x} = \frac{m.y^r}{y^r} = m$$

This is how the decryption will be done in case of El-Gamal encryption to get the message which was intended for the receiver.

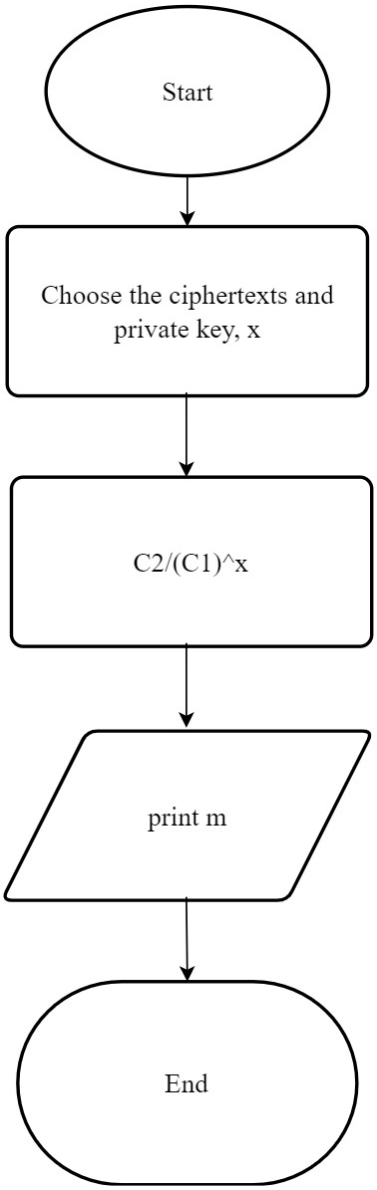

Figure 3.8: Flowchart of Decryption in El-Gamal

CHAPTER 4

Proposed Methodology

4.1 Introduction

In this chapter our proposed method for the blockchain based e-voting system is discussed in details. Our method consists of different components which are essential to run the system in a correct manner. Our proposed system is divided into three phases. The phases are as follows:

- Pre-Election Phase
- Election Phase
- Post-Election Phase

All of the above mentioned phases happens sequentially one after another. The whole process starts with Pre-Election Phase. A certain administrator with certain rights initiates the process. An overview of our system architecture is shown in Figure 4.1

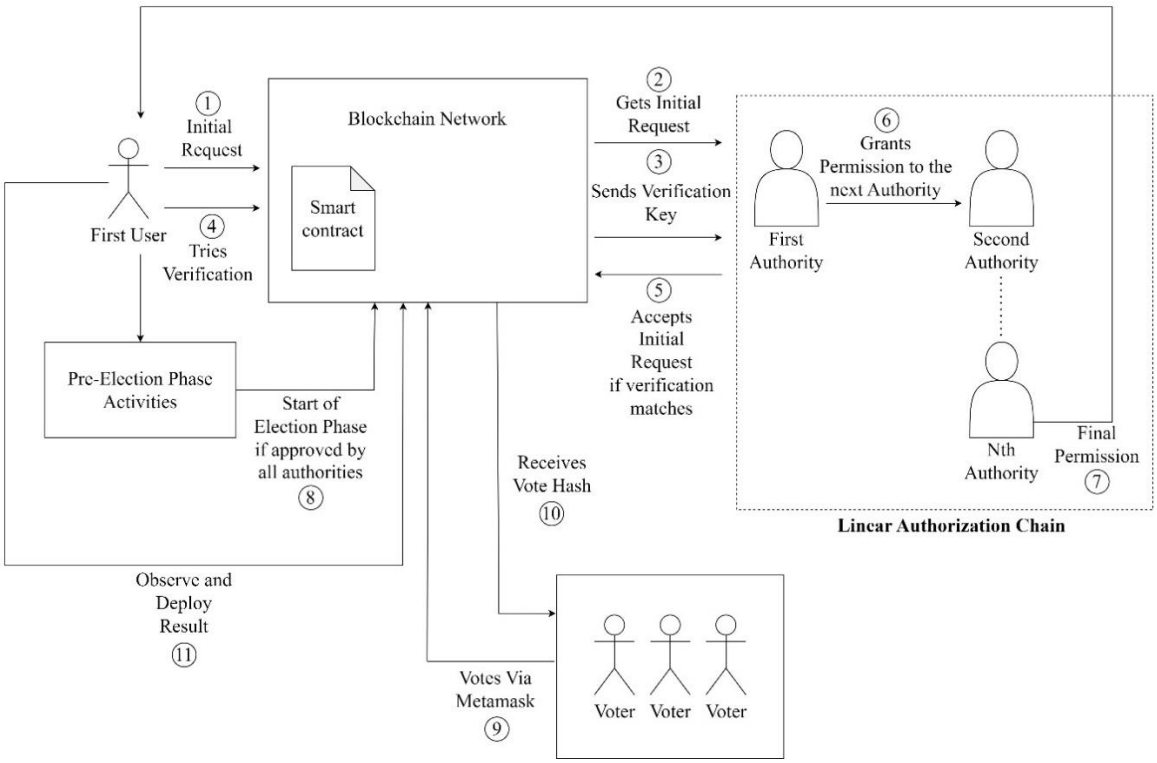

Figure 4.1: System Architecture of our proposed system

4.2 Pre-Election Phase

During this phase different works like proposing an election, adding candidates, authorizing voters, setting time-limit for election etc. are done. All of these are completed by a certain person who plays the role of the authority in the system. But that certain authority does not get the rights/permission right away. Our proposed system consists of multiple authority who propagates the permissions linearly among them until the final targeted authority gets the right. This phase is explained in later sub-sections.

4.2.1 Linear Authorization Chain for Permission Management and Resource Access

Initially an Ethereum Based Blockchain network is deployed. We used REMIX IDE to write, test and deploy our smart contracts. The process for a linear authorization chain for permission management would involve the following steps:

- **Request for Access:** An user initiates a request for obtaining the permission to be the main authority to start the process of the election.
- **Initial Approval:** The first authority in the authorization chain would receive the request and review it to determine whether it is valid. If the request is deemed valid, the authority would then grant initial approval.
- **Forwarding to Next Authority:** The approved request would then be forwarded to the next authority in the chain for further review.
- **Review and Approval:** Each subsequent authority in the chain would review the request and either approve or reject it based on their level of authority.
- **Final Approval:** The final authority in the chain would have the final say on whether to approve or reject the request.
- **Access Granted:** If the request is approved by all authorities in the chain, access would be granted to the user.
- **Access Denied:** If the request is rejected by any of the authorities in the chain, access would be denied to the user.

The requests which are passed onto the authorities will be in an encrypted way so that the authorities will have to earn the ownership. We used El-Gamal encryption in our system for encryption and decryption purposes. Figure 4.1 presents the flow of permission in Linear Authorization Chain.

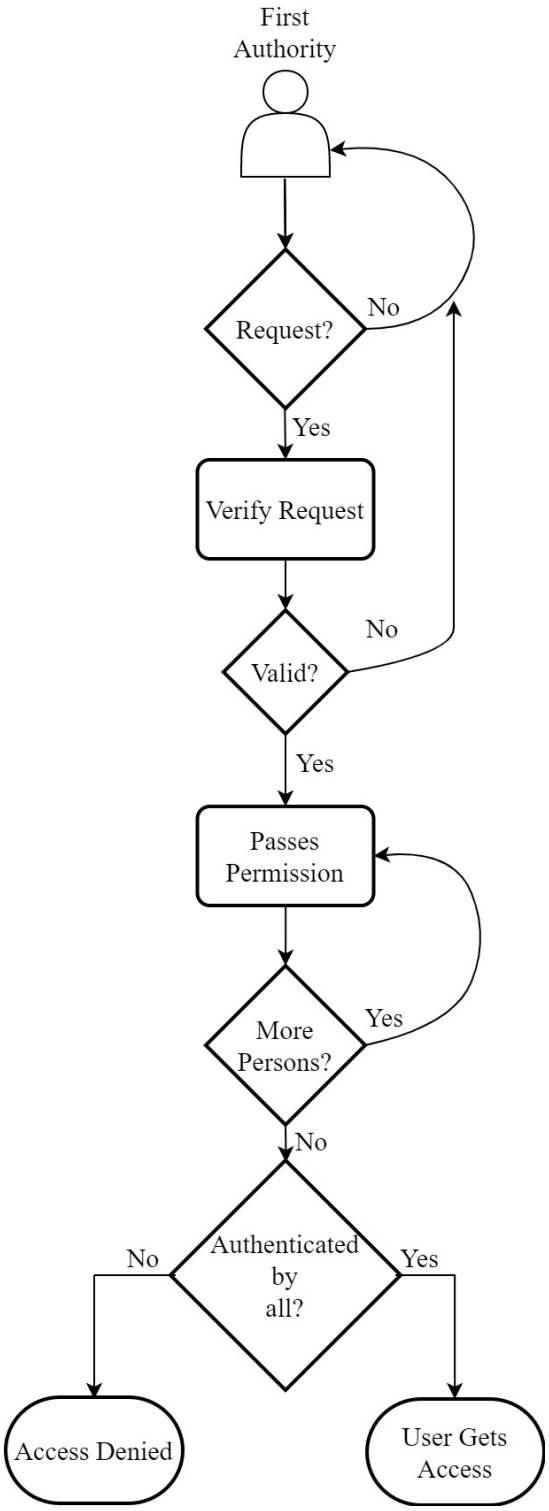

Figure 4.2: Flowchart of Linear Authorization Chain for Permission Management

Algorithm 1: Algorithm for initiating and requesting for access

1. procedure **openRequestAccess**(*_req_owner, _req_key*)
2. **If** (*open_req = false, request = false*) **then**
3. *req_owner = _req_owner*
4. *req_key = _req_key*
5. *open_req = true*
6. **End If**
7. End procedure
8. procedure **reqAccess**(*_key*)
9. **If** (*request = false, open_req = true, req_owner = msg.sender*) **then**
10. *request = true*
11. *open_req = false*
12. **End If**
13. End procedure

4.2.2 Implementation of El-Gamal in the system

The first authority will pass the ownership to the next authority and along with that he will also pass down a secret token. The token will be converted into two cipher-texts. Though the cipher-texts will be available publicly but only the second authority in line would be able to decrypt it as the token will be encrypted with the public key of the second authority. Only the second authority will be able to decrypt it using his private key. This is the principle of public key cryptography. So the steps are as follows:

- After receiving the request, the first authority will pass the ownership to the second authority. Along with it the first authority will also pass down a token which will be encrypted into two cipher-texts.
- The second authority will decrypt the cipher-texts and get a token. To gain ownership the second authority will then match the token with the original token of the first authority.
- If it matches, then the second authority will be granted the ownership of the smart contract. Then the second authority will do the same previous processes for transferring the ownership to the third authority.

- The transfer goes on till final authority.
- Ultimately when the user who requested in the first place will earn the ownership then he will start the pre-election Phase of our proposed system.

This encrypted, linear authorization process enhances the security of the permission management system, as the confidentiality of the request and each step of the approval process would be protected by encryption. This ensures that sensitive information is protected even as it is transmitted between different authorities in the chain.

Algorithm 2: Linear Authorization Chain for Permission Management

---

1. procedure **transfer_ownership** (*_newOwner, _h1, _h2, _message*)
2. **If** (*_newOwner != address(0), _newOwner != owner, freq_of_owner=0*) **then**
3. *nextOwner = _newOwner*
4. *hidden_number1 = _h1*
5. *hidden_number2 = _h2*
6. *message = _message*
7. *Inclusion of that owner in the ownerlist*
8. *owner_count++*
9. **End If**
10. End Procedure
11. procedure **accept_ownership**(*_checkMsg, _owner_count*)
12. **If**(*msg.sender=nextOwner, _owner_count=owner_count, message=_checkMsg*) **then**
13. *owner=nextOwner*
14. *Inclusion in ownerlist*
15. *nextowner=address(0)*
16. *hidden_number1=0*
17. *hidden_number2=0*
18. **End If**
19. End Procedure
20. procedure **transferOwnershiptoAnotherParty**(*_address, _h1, _h2, _msg*)
21. **If**(*_address!=address(0), address!=owner, freq_of_owner=1*) **then**
22. *Inclusion in ownerlist*
23. *nextowner=_address*

24. *hidden_number1=_h1*

25. *hidden_number2=_h2*

26. *msg=_msg*

27. *ownercount++*

28. End If

29. End Procedure

4.2.3 Proposal of Election and Adding Candidates

Following the previous event when the user will finally obtain the permission to start the pre-election phase, he will be considered as the main authority now. He will then express a proposal for election. He will give an appropriate name for the election. Then comes the process of adding verified candidates. He will be provided a list of valid candidates forehand to eliminate the need of verifying them. The process of adding candidates during the pre-election phase of our proposed system consists of following steps:

- The current authority will set the number of candidates.
- After receiving the list, the current authority will add the candidate with an ID. This ID is used to distinguish a candidate with other candidates.
- The smart contract will keep checking if candidates with same ID occurs or not. Adding candidates with same ID will result in penalty.
- The smart contract will also keep checking if the number of candidates the authority wants to add, exceeds the number of candidates set previously in the first step.

After following the above processes, it can be confirmed that the candidates who participate are well-informed and eligible to compete in the election. Figure 4.5 can be observed to visualize the whole process.

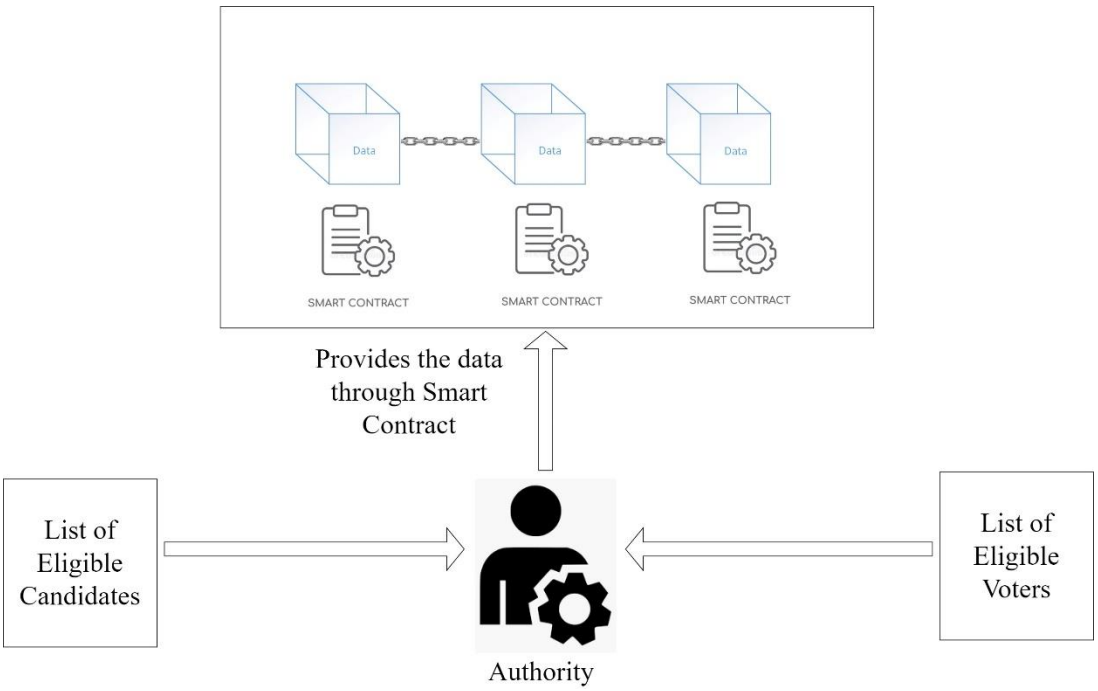

Figure 4.3: Block Diagram of Pre-Election Phase

Algorithm 3: Algorithm for Adding Eligible Candidates

Here next_candidate is set to zero initially which will act as the index pointer for the array.

1. procedure **addCandidates**(*_name, _id*)
2. **If** (*next_candidate >= candidateSize*) **then** return
3. **If** (*_id=candidate.id*) **then** return
4. *candidate.name=_name*
5. *candidate.voteCount=0*
6. *candidate.id=0*
7. **If** (*next_candidate+1 <= candidateSize*) **then** *next_candidate++*

4.2.4 Addition of Eligible Voters

After adding eligible candidates, the authority will now move on to adding eligible voters. The authority will be presented with a list of eligible voters with their public address. Using the Remix IDE the authority will authorize them. The authority will insert the addresses into the smart contract's function for authorizing voters. After the execution of the function the voter will be authorized for voting. But the voter will not be able to vote until the authority starts and changes the state of system.

4.2.5 Setting Time-limit and Starting the Election

Following the event of adding candidates and voters, the authority will now have to set a time-limit for the system. Otherwise the system would lose its importance. Setting time is a bit different in Smart Contracts than regular programs. After setting the time-limit the election process will start and the state will change to "Election" state in the smart contract.

4.3 Election Phase

After starting the election, the authority will now observe the Election Phase. During this phase the registered eligible voters will be able to interact with the blockchain network with their metamask wallet as they are registered with their public address of the metamask. Figure 4.6 visualizes the whole Phase in a minimalistic form. This Phase consists of the following steps:

- **Use of Metamask Wallet:** The voters in our system will need to have the access of Metamask wallet. Through this wallet the eligible voters will cast the vote against their chosen candidate. To perform different operations in the blockchain network using the smart contracts, a fee is required. Normally a very small amount of ETH(Ethereum) will be cut from the voters metamask wallet. By using the wallet each user will get his own unique address which will not be recognizable by the outsiders. So the voters' identity will not get revealed.
- **Hash Generation of the Vote:** When the voter is casting the vote, he will be asked to provide a secret key along with the candidate index. Later, the address of the voter, given candidate index and the provided secret key will be passed down into the SHA-256 algorithm to create a "Vote Hash". This will be used later in the case of vote verification [12].
- **Increment of Vote Counts of Candidates:** When the voters cast their vote against their chosen candidate, the vote count of that candidate is incremented. It will be updated over the whole blockchain network.
- **Verification of Vote:** When the voters will cast their vote, they might want to verify if their vote is correctly cast or not. So they will call the function of the smart contract for it. They will provide their address, their chosen candidate's index and the secret key they passed during their vote. These inputs will be passed down into the SHA-

256 algorithm to create a hash and then will be matched with the “Vote Hash” of the voter which was created before.

- **Elimination of Double Voting:** Double or multiple voting of a voter is also eliminated from the system. The smart contract in our system is developed in such a way that once a voter has cast their vote, they will not be able to cast another vote. Instead if they try to cast a vote, they will receive penalty.

After completion of this Phase, the voters will have cast their vote against their chosen candidate. But they would have to do it in a certain time-limit. Otherwise they will not be able to cast their even if they were eligible.

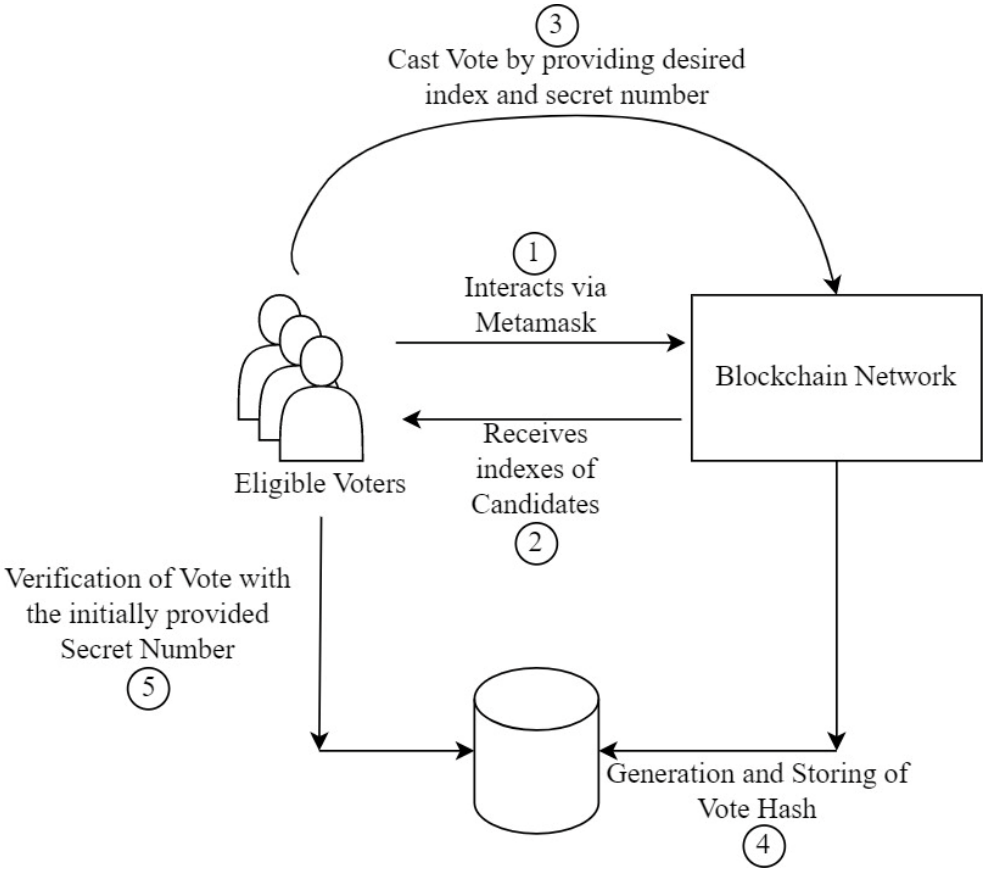

Figure 4.4: Block Diagram of Election Phase

Algorithm 4: Algorithm for Voting and Verifying Vote

1. procedure **vote**(*_candidateIndex, _snum*)
2. **If** (*timeExpired()*) **then** *show_result()*
3. **Else If** (*voter.voted=false, voter.authorized=true*) **then**
4. *candidates.votecount += 1*
5. *totalVotes += 1*
6. *set voter.voted = true*
7. *voter.voteHash = sha256(msg.sender, _candidateIndex, _snum)*
8. **End Else If**
9. **End** Procedure
10. procedure **verify**(*_candidateIndex, _snum*)
11. **If** (*voter.voted=true*) **then**
12. *_voteHash = sha256(msg.sender, _candidateIndex, _snum)*
13. **Check If** (*voter.voteHash=_voteHash*) return if true or not
14. **End If**
15. **End** Procedure

4.4 Post-Election Phase

When this phase will start, the state of the smart contract will change to "End" state. Now this phase can be obtained in two ways. One and the normal way is due to expiration of time. When the Election Phase was being initiated, the authority had to set a certain time. Now time will pass by and when it reaches the end, the smart contract will immediately change its state to "Post-Election Phase". Then no further voting will be allowed.

Another way this phase can be reached is due to authority. If the authority detects something unusual or sees something which might lead to some disaster, then he can terminate the election. But permission of other authorities is required for this operation to execute. This is a very unusual way to end the election. But this is for emergency case only.

Finally, at this Phase, the voters will be able to see the results. They would be able to check out which candidate won, which candidate got how many votes. At this phase, our proposed system will wrap up.

4.5 System Components

4.5.1 Smart contracts

Smart contracts play the most crucial role at any Ethereum-based blockchain system. It does the same in our proposed system too. Due to its non-deterministic, verifiable and self-executable pieces of code, the blockchain is kept running and functional. Our proposed model relies heavily on the functionalities of smart contracts as it does most of the work in the blockchain. As stated earlier, from start of the process to end of the process everything is carried out in the smart contract. Our proposed system is able to do all of the works in single contract in an efficient manner.

At first when the contract is deployed, the first authority waits for the request of the user who wants to get the permission to start the election. When request is received, the other works start. Our proposed method which introduced “Linear Authorization Chain”, then gets happened. That too in Encrypted way. We have used El-Gamal encryption for this purpose. The encryption is done in a separate python file to lessen the computation cost of the smart contract. As a result, less ethereum gas fee will be required. Later when the user ultimately gets the permission to initiate and conduct the election, then he will do all of the requested works as stated earlier to run an election. In the same contract, when the Election Phase will start, only the eligible voters will be able to operate also. They will have the opportunity to cast and verify their votes. Later they will be able to see the result.

All of the above works are done in a single smart contract in our system. Hence it is optimized and programmed in such way that minimum amount of gas cost (eth) is required.

4.5.2 Blockchain Network Nodes

The system we have proposed on the Ethereum platform uses proof-of-work mechanism to verify and maintain the validity of the blockchain. As stated earlier, this requires a network of nodes(devices) or miners to keep the system running. The miners participate and try to solve the problem which is stated. In exchange of finding the proof-of-work consensus, the miners accept Ethereum cryptocurrencies which is commonly referred to as Ethereum Gas cost in the blockchain.

Blockchain provides highly secure and scalable system. But this comes at a cost. In ethereum based blockchain system, GAS cost is required to run. The computational cost is determined by the gas cost. When a system is in development stage, it is important to keep in mind that the cost should be as low as possible by optimizing the program. While developing our proposed system we had to keep this matter in consideration.

4.6 Summary

After thoroughly evaluating our proposed model, we have created a functional e-voting system prototype that is capable of fulfilling all the necessary requirements of an e-voting system. Our system also incorporates a "Linear Authorization" method to enhance security in the permission management process, as all requests and approval steps are protected by encryption. This Ethereum-based prototype operates transparently and has been tested on the Ganache local blockchain runtime environment. The entire system is designed to run within a single smart contract written in Solidity and optimized for minimal Ethereum gas cost. The Metamask plugin is utilized to link the Ganache nodes to the Remix IDE, where different accounts serve various roles in the operation of the system.

CHAPTER 5

Experimental Analysis

5.1 Introduction

The results from all the methods that were evaluated along the course of this thesis work are represented in this chapter. The experimental analysis of this research work is divided into four sections: Experimental Setup, Result Analysis, Comparison and Discussion. These are outlined shortly as follows.

5.2 Experimental Setup

Useful information about the hardware used for executing and completing the process as well as the software information regarding the IDE, Simulator, OS information etc. are discussed in Experimental Setup section.

5.2.1 Hardware

Many hardware were required for the successful completion of the work. Without the correct hardware, the software may not run efficiently or at all. Useful information about the hardware are provided below.

- Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz
- 16 GB RAM

5.2.2 Software

Various types of software were used to make our proposed system functional. Useful information regarding the OS information, programming language information, IDE versions etc. are provided below.

- Operating System: Windows 10 (64 bit)
- Programming Language: Solidity 0.8.0 or above, python
- IDE: Remix IDE v0.29.2

- Wallet: Metamask Wallet
- Simulator: Ganache v2.5.4

5.3 Result Analysis

5.3.1 Trust Distribution

Through our proposed system we were able to achieve a distributed trust system. That method is “Linear Authorization Chain for Permission Management and Resource Access”. In our system, the initialization will not be depending on a single authority like most other previously related system. From initializing the whole process to initializing each sub processes, every step will be verified by multiple authorities. Through these operations the system will be even more secure as extra layer of security is added on top of the blockchain platform. Therefore, it will also eliminate the doubt of voters in some extent and gain their trust.

As per our method, initially a user will be given a chance to start the process. The user will request for the access. Upon the request, the first authority will be notified and he will pass the address of the user and a token. Say the address of the user is “0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db” and the value passed down is “16262”. Now the user will have to insert that exact value to gain the first approval. Storage of data is presented in Table 5.1 and Table 5.2.

Table 5.1: An Example of First Authority opening request access for the user

Function	Value to Store
openRequestAccess()	0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db, 16262

Table 5.2: The input value of the user to get approval of initial request

Function	Value to Insert
RequestAccess()	16262

When the previous process is all said and done, the linear authorization chain will start to take place. Say the first authority will transfer the ownership and grant permission to the second authority. To do this communication in a secure way, El-Gamal is used. Say the address of the second authority is "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2" and the token he wants to pass is "15". Then data are passed in the manner of Table 5.3.

Table 5.3: First Authority granting permission to Second Authority through encryption

Function	transferOwnership()
Address	0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
Token	15
Cipher-Text 1 (C1)	2187
Cipher-Text 2 (C2)	33495

Now only the second authority will be able decrypt the message to earn the ownership. This additional encryption is done so that extra security is ensured. Due to the characteristics of ElGamal encryption only second authority will be able to decrypt the message as for decryption the private key of the second authority will be required. Table 5.4 shows the result after decryption

Table 5.4: Second authority's generated value corresponding to Table 5.3

Function	acceptOwnership()
Decrypted Token	15

The total "Linear Authorization Chain" system increases with the number of authority being involved in a system. The average estimated time for increasing number of authorities. It is a trade-off between time and security.

5.3.2 Relation of Time with Number of Authority

Time for giving permission to vote or time for giving permission to terminate the election also increases with the increasing number of authority. But the time in both cases are same for same no. of authority. Here the propagation of permission granting is not encrypted like

before as it would become redundant. Figure 5.1 visualizes the relation between the average time required to complete the linear authorization with respect to number of authority.

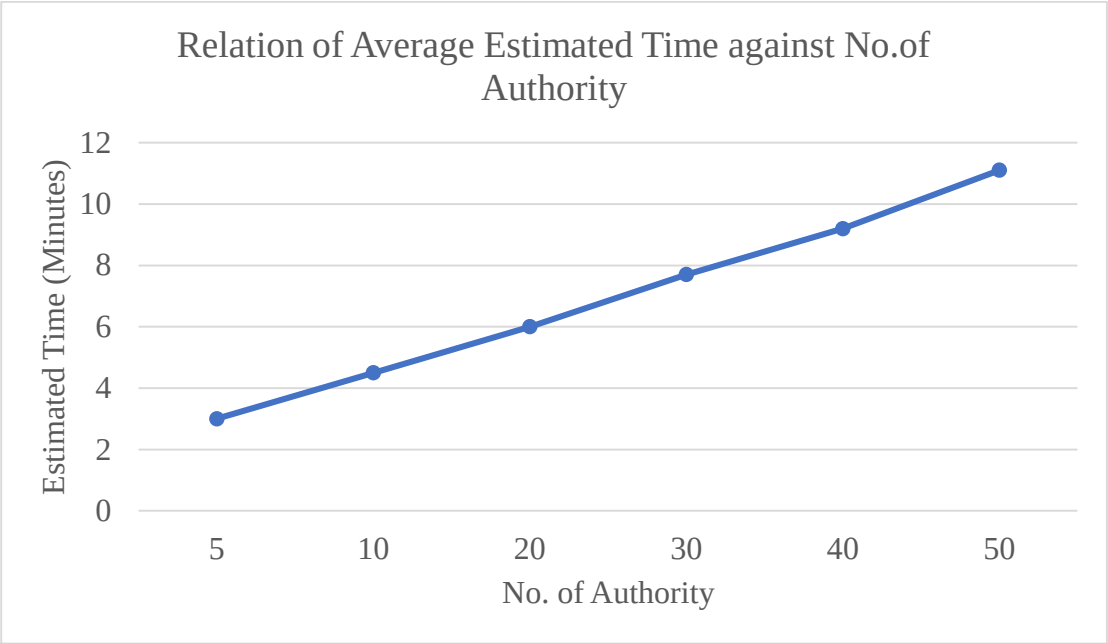

Figure 5.1: Average Estimated Time for Linear Authorization Chain with respect to No. of Authority

5.3.3 Use of ElGamal

Due to the addition of ElGamal encryption, the security of the entire system has increased drastically. Blockchain itself is almost impossible to break. With the addition of ElGamal makes it even stronger to crack. The entire trust distribution phase is completed through this.

Figure 5.2 shows the encryption time of ElGamal compared with other encryption systems like RSA and Paillier [20]. Though the later two shows proportional increase with the increase of File Size, ElGamal is inconsistent in this due to its discrete logarithm feature.

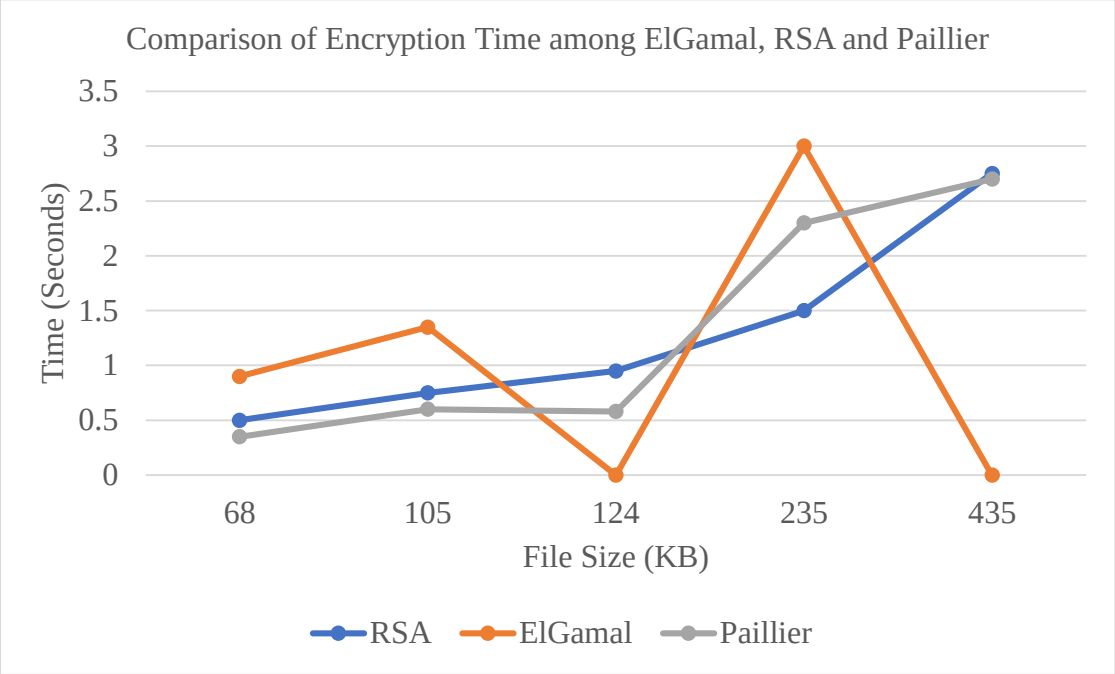

Figure 5.2: Comparison of Encryption Time among ElGamal, RSA and Paillier

Figure 5.3 shows the decryption time between ElGamal and other previously mentioned crypto-systems [20]. Here we can see that no matter how large the size of the cipher-text is, it tends to show similar time. A really small percentage of increment is seen with the increment of file size. In our proposed system minimum decryption time is really important for it to be fruitful.

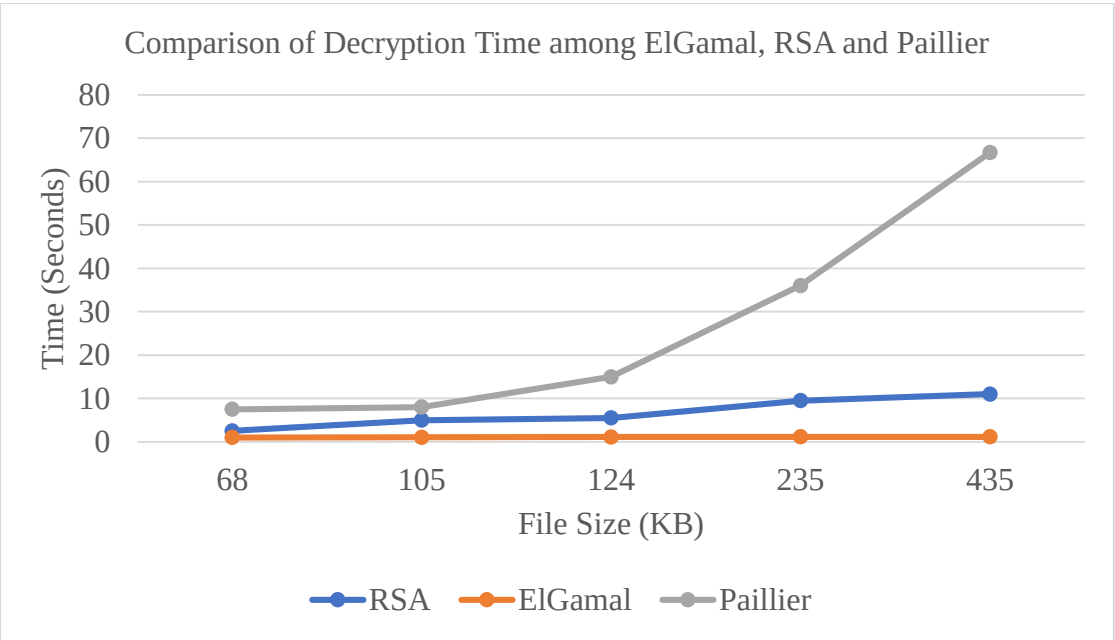

Figure 5.3: Comparison of Decryption Time among ElGamal, RSA and Paillier

Time for granting permission to vote or time for granting permission to terminate the election also increases with the increasing number of authority. But the time in both cases are same for same no. of authority. Here the propagation of permission granting is not encrypted like before as it would become redundant.

5.3.4 Generation of Vote Hash

During the voting, the voters are asked to insert a secret number along with the candidate index. This is done to keep a hash of their vote. The hash value in this case is generated by SHA-256. This value is kept in the blockchain network along with the other records of the voter. This value can be used later if a voter wishes to check if his voting is done correctly or not. Table 5.5 shows an example of generating a voters' vote hash.

Table 5.5: Hash generation of a vote using SHA-256

Voter	Candidate Index	Secret Number	Vote Hash (SHA-256)
0x5B38Da6a701c568545d CfcB03FcB875f56beddC4	0	13	145acfd8b86633dbe09de403 eed4ab633e07d0dc 00c036ee23e7312aea7e2d65

5.4 Comparative Analysis

Our proposed method of enhancing the security of e-voting is inspired from approaches on relevant works in the domain. Our main contribution was to develop such system that would have extra layers of security in every phase of the system including the initialization phase and at the same time improve existing relevant systems. The performance metrics of a blockchain-based system generally depends on the computation cost of hosting the blockchain. But this does not represent the complete image as it is not actually a performance measurement rather a cost analysis. Our proposed system can be compared with the other existing systems on the following sectors:

- Existing e-voting systems like most other blockchain based systems runs on the will of a single authority. Generally, a single authority is available to initiate and conduct the whole voting process. On the contrary, our proposed system introduces a

technique by which multiple authority will exist in a system. They will linearly grant permissions until the requested user achieves it to earn the right of starting the election. Then just before starting the process of actual voting, examination of multiple authority is done again. This checking by multiple authority in start of each major check-points greatly increase the amount of security.

- Another major difference is the entry of encrypted communication between the authorities. The step which is discussed previously occurs in an encrypted way as we included ElGamal encryption in our system. The authorities taking help of ElGamal encryption greatly enhances the security as it is implemented on top of already secure platform, blockchain
- Our proposed system is built using only single smart contract. Inclusion of many smart contracts increases the chance of getting the blockchain network heavier.
- Our proposed system also included a way for the voters to verify their vote.

5.5 Summary

While other existing blockchain based e-voting systems have already shown improvements, but our proposed method will be able to do even more. Though blockchain itself is a decentralized platform, but the idea of having a single authority to initiate and conduct the whole process might raise trust issues between voters. The account of authority itself might get hacked or manipulated. Addition of our proposed technique, “Linear Authorization Chain” can be a solution to this problem. As multiple authorities will exist in the system, they will monitor each major phases. From giving the requested user the permission to initiate the process to granting permission before each major phases, each step will be monitored by the authorities existing in the system. This verification at each phase enhances the systems security immensely. With that being said, the initial permission propagation is done in an encrypted way which enhances the security. Our heart of the entire system, smart contract is built in a really optimized way as our whole system will run on a single smart contract. Thus, computational complexity of interacting with multiple smart contracts is reduced by great extent.

CHAPTER 6

Conclusion

6.1 Concluding Discussions

Elections play a crucial role in democratic societies by providing a mechanism for citizens to select their representatives and hold them accountable. Through this, the will of the majority is reflected in the outcome. Hence with growth of technology, different activities should be implemented to ease out workload and protect the system. In this thesis work, a blockchain based e-voting system is developed with additional security which focuses on adding more layers of security to enhance the overall security of the system. Our system is able to conduct the election process and also able to solidify the security structure even more than existing systems. As the technology continues to evolve and mature, the future of electronic voting systems looks promising. It is expected to provide more advanced and secure e-voting systems in the future. In this chapter the limitations and the future work for the system has been discussed.

6.2 Limitations

The limitations of developing our proposed system are as follows:

- The total completion time of our system increases with the number of authority being in our system. It mainly affects in the first-half portion of our system.
- To reduce the computation cost (Gas Cost) of our smart contract, we provided the encryption and decryption outside the contract. All the authorities will be given separate files to generate cipher-texts and decrypt them. They will get the values stored in different files. They will have to retrieve the values from there in order to proceed.
- Verification by authorities before each major Phase will result in more time to conduct the process. But this comes with added security.

6.3 Future Work

The following tasks can be performed in the future:

- Linearity of time with number of authority can be reduced so that overall time for the system to complete can be faster.
- The prototype of our model has been proved to be functional. So this could be implemented in a large-scale, real time environment.
- More efficient way could be found to do the encryption and decryption all at once in a single smart contract.

REFERENCES

[1] Kshetri, Nir, and Jeffrey Voas. “Blockchain-Enabled E-Voting.” *IEEE Software*, vol. 35, no. 4, Institute of Electrical and Electronics Engineers (IEEE), July 2018, pp. 95–99. *Crossref*, https://doi.org/10.1109/ms.2018.2801546.

[2] Desai, Shreya, Meng Han, Lei Li, Zhigang Li, Jing He, and Xiaohua Xu, "Untampered Electronic Voting in Entertainment Industry: A Blockchain-based Implementation." In *Proceedings of the 20th Annual SIG Conference on Information Technology Education*, pp. 166-166. 2019.

[3] He, O., and Zhongmin Su, “*A new practical secure e-voting scheme*.” na, 1998.

[4] Ghayoori, Majid, and Hamzeh Sezavar. “A Consensus Algorithm for Use in Blockchain Based Voting System.” *Academia Letters*, Academia.edu, Feb. 2022. *Crossref*, https://doi.org/10.20935/al4834.

[5] Yavuz, Emre, Ali Kaan Koç, Umut Can Çabuk, and Gökhan Dalkılıç. "Towards secure e-voting using ethereum blockchain." In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, pp. 1-7. IEEE, 2018.

[6] Agrawal, Komal, Shivani Gupta, and M. Dhiman. "Secure E-Voting System using Blockchain Technology." *International Journal for Research in Applied Science and Engineering Technology-IJRASET* 7.XII (2019).

[7] Braghin, Chiara, Stelvio Cimato, Simone Raimondi Cominesi, Ernesto Damiani, and Lara Mauri, "Towards blockchain-based e-voting systems." In *Business Information Systems Workshops: BIS 2019 International Workshops, Seville, Spain, June 26–28, 2019, Revised Papers 22*, pp. 274-286. Springer International Publishing, 2019.

[8] Patil, Harsha V., Kanchan G. Rathi, and Malati V. Tribhuwan. "A study on decentralized e-voting system using blockchain technology." *Int. Res. J. Eng. Technol* 5.11 (2018): 48-53.

[9] Fan, Wenjun, et al. "A Privacy Preserving E-Voting System Based on Blockchain." *Silicon Valley Cybersecurity Conference: First Conference, SVCC 2020, San Jose, CA, USA, December 17–19, 2020, Revised Selected Papers 1*. Springer International Publishing, 2021.

[10] Chouhan, Vikas, and Anshul Arora, "Blockchain-based secure and transparent election and vote counting mechanism using secret sharing scheme." *Journal of Ambient Intelligence and Humanized Computing* (2022): 1-19.

[11] Hsiao, Jen-Ho, Raylin Tso, Chien-Ming Chen, and Mu-En Wu. "Decentralized E-voting systems based on the blockchain technology." In *Advances in Computer Science and Ubiquitous Computing: CSA-CUTE 17*, pp. 305-309. Springer Singapore, 2018.

[12] Guasch, Sandra, and Paz Morillo, "How to challenge and cast your e-vote." In *Financial Cryptography and Data Security: 20th International Conference, FC 2016, Christ Church, Barbados, February 22–26, 2016, Revised Selected Papers 20*, pp. 130-145. Springer Berlin Heidelberg, 2017.

[13] Khalili, Mojtaba, Mohammad Dakhilalian, and Willy Susilo. "Efficient chameleon hash functions in the enhanced collision resistant model." *Information Sciences* 510 (2020): 155-164.

[14] Braghin, Chiara, Stelvio Cimato, Ernesto Damiani, and Michael Baronchelli, "Designing smart-contract based auctions." In *Security with Intelligent Computing and Big-data Services: Proceedings of the Second International Conference on Security with Intelligent Computing and Big Data Services (SICBS-2018) 2*, pp. 54-64. Springer International Publishing, 2020.

[15] "What is SHA-256 Algorithm — Simplilearn", *Simplilearn*, 2022, [online] Available: https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm

[16] "Gas(Ethereum) — How Gas Fees Work on the Ethereum Blockchain", *Investopedia*, 2022, [online] Available: https://www.investopedia.com/terms/g/gas-ethereum.asp

[17] "Solidity — Solidity 0.8.18 documentation", *Investopedia*, 2023, [online] Available: https://docs.soliditylang.org/en/v0.8.18/

[18] "Ganache — TruffleSuite documentation for ganache", *Trufflesuite,*2022, [online] Available: https://docs.soliditylang.org/en/v0.8.18

[19] "Metamask — Wikipedia Overview for ganache", *Wikipedia,*2022, [online] Available: https://en.wikipedia.org/wiki/MetaMask

[20] "Comparison of RSA, ElGamal and Paillier — Researchgate", *Researchgate,*2012, [online] Available: https://hackernoon.com/blockchain-a-short-and-simple-explanation-with-pictures-d60d652f207f

[21] "Blockchain — A Short Explanation with Pictures", *Hackernoon,*2019, [online] Available: https://hackernoon.com/blockchain-a-short-and-simple-explanation-with-pictures-d60d652f207f