



**TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS**

THESIS NO: PUL076MSDSA006

**Blockchain-Based E-Voting With Zero-Knowledge
Proof Using Smart Contracts**

**by
Juned Alam**

A THESIS

**SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND
COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN
COMPUTER ENGINEERING SPECIALIZATION IN DATA SCIENCE
AND ANALYTICS**

**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
LALITPUR, NEPAL**

September, 2022

Blockchain-Based E-Voting With Zero-Knowledge Proof Using Smart Contracts

by

Juned Alam

PUL076MSDSA006

Thesis Supervisor

Prof. Dr. Shashidhar R. Joshi

A thesis submitted in partial fulfillment of the requirements for the
degree of Master of Science in Computer Engineering Specialization in Data
Science and Analytics

Department of Electronics and Computer Engineering
Institute of Engineering, Pulchowk Campus
Tribhuvan University
Lalitpur, Nepal

September, 2022

COPYRIGHT©

The author has agreed that the library, Department of Electronics and Computer Engineering, Institute of Engineering, Pulchowk Campus, may make this thesis freely available for inspection. Moreover the author has agreed that the permission for extensive copying of this thesis work for scholarly purpose may be granted by the professor(s), who supervised the thesis work recorded herein or, in their absence, by the Head of the Department, wherein this thesis was done. It is understood that the recognition will be given to the author of this thesis and to the Department of Electronics and Computer Engineering, Pulchowk Campus in any use of the material of this thesis. Copying of publication or other use of this thesis for financial gain without approval of the Department of Electronics and Computer Engineering, Institute of Engineering, Pulchowk Campus and author's written permission is prohibited.

Request for permission to copy or to make any use of the material in this thesis in whole or part should be addressed to:

Head

Department of Electronics and Computer Engineering

Institute of Engineering, Pulchowk Campus

Pulchowk, Lalitpur, Nepal

DECLARATION

I declare that the work hereby submitted for Master of Science in Computer Engineering Specialization in Data Science and Analytics (MSDSA) at IOE, Pulchowk Campus entitled “**Blockchain-Based E-Voting With Zero-Knowledge Proof Using Smart Contracts**” is my own work and has not been previously submitted by me at any university for any academic award.

I authorize IOE, Pulchowk Campus to lend this thesis to other institution or individuals for the purpose of scholarly research.

Juned Alam

PUL076MSDSA006

Date: September, 2022

RECOMMENDATION

The undersigned certify that they have read and recommended to the Department of Electronics and Computer Engineering for acceptance, a thesis entitled “**Blockchain-Based E-Voting With Zero-Knowledge Proof Using Smart Contracts**”, submitted by **Juned Alam** in partial fulfillment of the requirement for the award of the degree of “**Master of Science in Computer Engineering Specialization in Data Science and Analytics**”.

.....

Supervisor: Prof. Dr. Shashidhar R. Joshi,
Department of Electronics and Computer Engineering,
Institute of Engineering, Tribhuvan University

.....

External Examiner: Mr. Krishna P. Bhandari,
CEO, Nepal Digital Payments Co.

.....

Committee Chairperson: Assoc. Prof. Dr. Arun Kumar Timalina
Program Co-ordinator, MSDSA

Date: September, 2022

DEPARTMENTAL ACCEPTANCE

The thesis entitled “**Blockchain-Based E-Voting With Zero-Knowledge Proof Using Smart Contracts**”, submitted by **Juned Alam** in partial fulfillment of the requirement for the award of the degree of “**Master of Science in Computer Engineering Specialization in Data Science and Analytics**” has been accepted as a bonafide record of work independently carried out by him in the department.

.....
Prof. Dr. Ram Krishna Maharjan
Head of the Department,
Department of Electronics and Computer Engineering,
Pulchowk Campus,
Institute of Engineering,
Tribhuvan University,
Nepal

ACKNOWLEDGEMENT

I would like to extend my appreciation and gratitude to the Department of Electronics and Computer Engineering (DoECE) of the Institute of Engineering, Pulchowk Campus for the opportunities and guidelines throughout my academic career. I want to thank my supervisor **Prof. Dr. Shashidhar R. Joshi** sir, the department faculty members, and our program coordinator **Dr. Arun Kumar Timalina** sir, for providing the opportunity to submit this thesis report for the fulfillment of the requirements for the degree of Master of Science in Computer Engineering Specialization in Data Science and Analytics.

Sincerely,

Juned Alam

(PUL076MSDSA006)

ABSTRACT

The data of the public block chain, being available to all nodes, it is necessary to hide the vote preference of the voter, and preserve the integrity of the casted vote, while at the same time, it is necessary to show that the voter has already voted, to prevent someone from casting multiple votes. This thesis work proposes an e-voting system using block chain and its smart contract as the rule setter. Here, with the help of the Paillier Cryptography system, the zero knowledge proof was accomplished. The zero knowledge proof here was used to show that the voter has already voted while at the same time, hiding the casted vote. The homo-morphic additive property of the Paillier cryptography system was used to perform addition on the encrypted cipher texts without the need to decrypt the cipher text to reveal the votes in the process. In the end, a secure voting mechanism was achieved.

Keywords: block chain, zero knowledge proof, paillier cryptographic system, smart contract

TABLE OF CONTENTS

COPYRIGHT	iii
DECLARATION	iv
RECOMMENDATION	v
DEPARTMENTAL ACCEPTANCE	vi
ACKNOWLEDGEMENT	vii
ABSTRACT	viii
TABLE OF CONTENTS	ix
LIST OF FIGURES	xii
LIST OF TABLES	xiv
LIST OF ABBREVIATIONS	xv
1 INTRODUCTION	1
1.1 Background and Motivation	1
1.2 The motivation behind block chain	2
1.3 Immutable nature of Block Chain	5
1.4 Problem Statement	7
1.5 Objectives	8
1.6 Scope of the thesis work	8
1.7 Probable implementations of this thesis work	9
1.8 The originality of the thesis work	9
1.9 Organization of the thesis work	9
2 LITERATURE REVIEW	10
3 METHODOLOGY	13
3.1 Block chain structure	13

3.1.1	Blocks in a block chain	15
3.2	Mining of a block	16
3.3	Smart Contracts in Block chain	20
3.4	Memory Pools	20
3.5	Elliptic Curve Cryptography (ECC)	21
3.6	Zero Knowledge Proof	22
3.6.1	Key generation	23
3.6.2	Encryption	24
3.6.3	Decryption	24
3.6.4	Homomorphic addition property	24
3.7	The proposed system	24
3.8	State Sequence of the voting system	26
3.8.1	Initialization Phase	26
3.8.2	Voting Phase	27
3.8.3	Result Publishing phase	29
4	RESULTS AND DISCUSSION	31
4.1	Experimental Setup	31
4.2	Paillier cryptography computation time	31
4.3	Hash rate per second using parallelism	32
4.4	Use of SHA-512 in the system	33
4.5	Mining of blocks using a single thread	34
4.6	Mining of blocks using multiple threads	35
4.7	Time taken to generate hash for different number of parties in the ballot	37
4.8	Time taken at each phase of the system	38

4.9	The calculation for the size of the block and block chain	39
4.9.1	Size of the paillier enciphered text	39
4.9.2	Size of parties	39
4.9.3	Size of the map of votes	39
4.9.4	Size of the list of transactions	40
4.9.5	Size of a genesis block	40
4.9.6	Size of a normal block	40
4.9.7	Size of the block chain	41
4.10	Graphics Processing Unit (GPU) and Application-specific integrated circuit for mining (ASIC)	42
4.11	The potential threat to the system with the emerging Quantum Technology	42
4.12	Output	43
5	THESIS TIMELINE	44
6	CONCLUSION AND RECOMMENDATION	45
6.1	Conclusion	45
6.2	Limitation	45
6.3	Recommendation	45
	REFERENCES	48
	APPENDIX A	49
	APPENDIX B	50
	APPENDIX C	52
	APPENDIX D	54
	APPENDIX E	57
	APPENDIX F	65

LIST OF FIGURES

1.1	A trust-less car buy-sell system requiring the third party	2
1.2	Block chain based trusted car buy-sell network	3
1.3	A trust-less money laundering system requiring the third party . . .	4
1.4	Block chain based trusted money laundering network	4
1.5	Traditional voting system vs block chain-based voting system	8
3.1	Blocks in a block chain	15
3.2	Elements of a block	15
3.3	Range of SHA-256 hash function outputs	17
3.4	Memory pool in a block chain	21
3.5	The process of signing a transaction	22
3.6	The proposed system for the E-voting system	25
3.7	State Sequence of the voting system	26
3.8	State Sequence of the initialization phase	27
3.9	State sequence of the voting phase	28
3.10	The process of homomorphic addition in the cipher texts of the ballot of voters; the short cipher texts are only for representational purpose	29
3.11	State Sequence of the Result publishing phase	30
4.1	Duration of paillier operations for different length of key seeds . . .	32
4.2	Hashes per second for different numbers of threads	33
4.3	System's Performance when using a single thread	35
4.4	Performance of mining when using multiple threads	36

4.5	Mining performance using different approaches	37
4.6	Time taken to generate a single hash for different ballot sizes	38
5.1	Thesis Timeline	44
6.1	Miner node	49
6.2	Voter node	49

LIST OF TABLES

3.1	Comparison of RSA vs ECDSA	23
4.1	Specifications of the machine on which the experimentations have been performed	31
4.2	Correlation matrix of seed length, encryption and decryption time of paillier cryptography	32
4.3	Time of each phase of the election	38
6.1	Hash generated per second of SHA-256 and SHA-512 while using given number of threads	50
6.2	Hash generated per second of SHA-256 and SHA-512 while using given number of threads, continued	51
6.3	System Performance Metrics Data	54
6.4	System Performance Metrics Data, continued	55
6.5	System Performance Metrics Data, continued	56
6.6	System Performance Metrics Data with multiple thread	57
6.7	System Performance Metrics Data with multiple thread, continued .	58
6.8	System Performance Metrics Data with multiple thread, continued .	59
6.9	System Performance Metrics Data with multiple thread, continued .	60
6.10	System Performance Metrics Data with multiple thread, continued .	61
6.11	System Performance Metrics Data with multiple thread, continued .	62
6.12	System Performance Metrics Data with multiple thread, continued .	63
6.13	System Performance Metrics Data with multiple thread, continued .	64

LIST OF ABBREVIATIONS

ABBR	ABBREVIATIONS
ASIC	Application-specific integrated circuit
CPU	Central Processing Unit
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EVM	Electronic Voting Machine
GPU	Graphics Processing Unit
IM_RSA	Improved Modification RSA Algorithm
IT	Information Technology
Nonce	Number Only UsedOnce
OTP	One-time password
P2P	Peer to Peer
PBFT	Practical Byzantine Fault Tolerance
PoB	Proof of Burn
PoS	Proof of Stake
PoW	Proof of Work
SHA-256	Secure Hash Algorithm 256
SHA-512	Secure Hash Algorithm 512

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

The trending topic everywhere nowadays is elections and voting. Voting is the process of selection or election of decisions or individuals by a population. The process of voting is done in an election to choose a government, elect a person or a group of persons, has been done since medieval times and is still going on as a way to represent democracy. The voting in the elections is mostly done the old-fashioned way, that is using the ballot papers. There are also a few alternatives to the old-fashioned paper-based ballot-paper voting like voting via Electronic Voting Machine (EVM), which is popular nowadays, the electronic voting, that is voting done online. Although convenient and modern, the legitimacy of EVM is always a concern and there is always a chance of rigging the EVM.

If or when we press the button of the EVM during any future election, we may have a few concerns. Like whether, there is any guarantee that our vote is actually being registered, that our vote will actually be counted, or if any scam might take place. All these concerns are guaranteed by a central agency: The Election Commission. Basically, we have to trust the Election Commission that the commission will work properly for our votes to be counted properly. But what if we want to conduct our own elections on our organizations, area, or even be a part of these elections. We would require a system that is transparent, automated, and trustable. The answer lies in blockchain.

A Block chain is an incrementally growing list of transaction ledgers called blocks which are linked together in a chain-like structure using pointers referred to as hash; which will be explained further in this thesis document. Basically, block chain is the left-linked list of blocks.

In the current scenario, centralized systems are dominating every industry. Block chain technology aims to get rid of these centralized systems (to a certain extent)

and use decentralized ones instead. Blockchain technology has been used to build many crypto-currency systems like the most popular one: Bitcoin.

The block chain technology can revolutionize the entire voting system. Not only voting, block chain has the potential to revolutionize the entire Information Technology (IT) domain.

1.2 The motivation behind block chain

First of all, of we need to talk about the motivation for requiring block chains at all. I will consider an example. For example, a given person, who wants to buy a car, and another person tries to sell the given car. So, this is the car we are talking about. And the problem is that there is no trust between the buyer and the seller. The buyer does not trust the seller and vice versa, the seller does not trust the buyer. So that's why we need a trusted third party such as banks or governments. So, in this case, both the buyer and seller are going to notify the government or a bank. The buyer has to notify the government that now he does own the given car and the seller has to notify the government that he sold his car. For example, the seller bought the car approximately three months ago and it took approximately one week to be able to buy this given car because the buyer had to notify the government that the buyer would like to buy the car. The seller has to notify the bank of the cost of the car, create new identification papers and transfer ownership of the car, and so on. This process requires the expenditure of time and money.

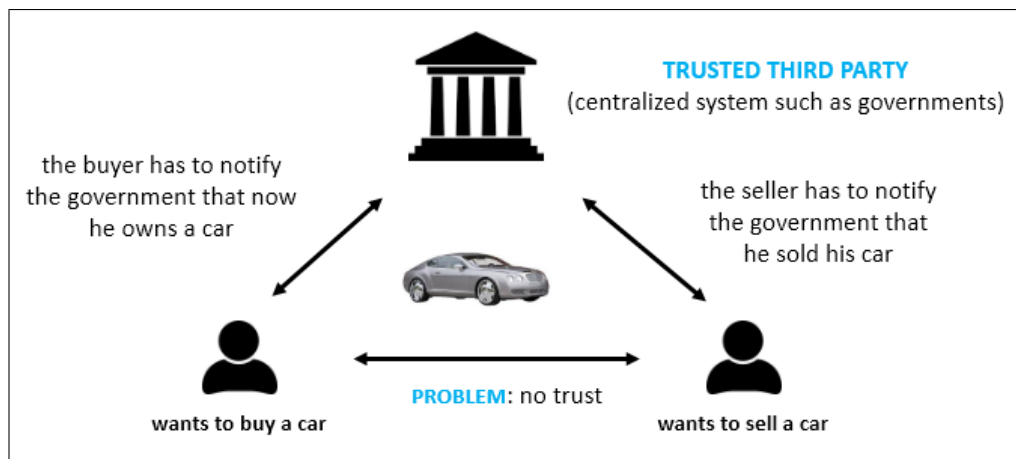


Figure 1.1: A trust-less car buy-sell system requiring the third party

So, there is a very important inference, because there's no trust between the buyers and sellers, and that's why trusted third parties came to be, such as governments or banks. With the help of block chains, we can get rid of these third parties. So instead of notifying the bank, with the help of blockchain technology and a decentralized network of car buyers and sellers we can solve the same problem easier. It is costless because the block chain itself guarantees trust. So, as we can see the main problem is that there's no trust in a centralized system. And that's why these third parties are needed when dealing with block chains and decentralized networks. Here, block chain can guarantee trust by forming a network of buyers and sellers without requiring a third party to establish trust.

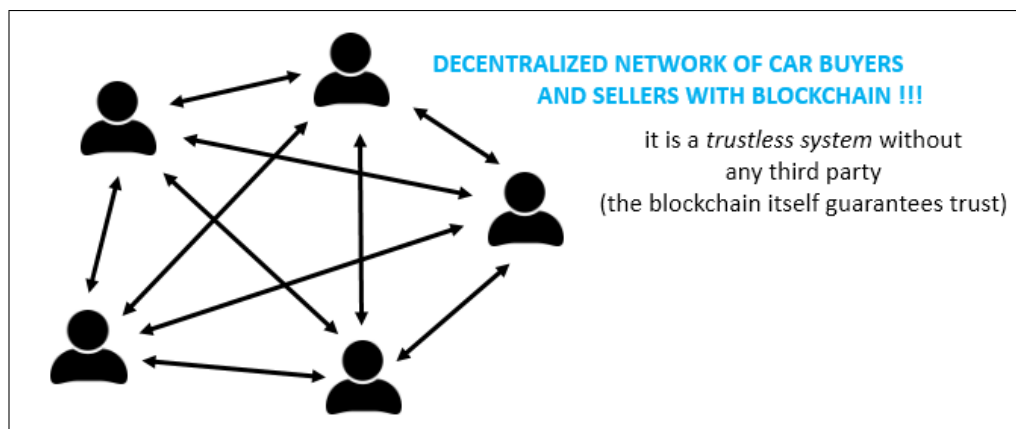


Figure 1.2: Block chain based trusted car buy-sell network

It is the same when a given person 'A' wants to send the money to another person 'B'. 'B' wants to receive money from 'A' and again that there is no trust. They don't trust each other and that's why a trusted third party in this case a bank is going to guarantee that the sender will send the money to the receiver and the receiver is going to get the money. The sender has to notify the bank that he/she wants to send X dollars to B and the bank is going to handle everything. The bank has a centralized database and the bank is going to handle this transaction by updating the database on the transactions accordingly. Now, the bank is going to update the entry for 'A' which, means that the bank is going to decrement the actual balance of 'A' by X dollars and the bank is going to update the database entry for 'B' as well, which means that the balance of 'B' is going to be incremented by X dollars. So again, there is a trusted third party, in this case, a bank, that is

going to handle the transaction(s) and update its central database accordingly.

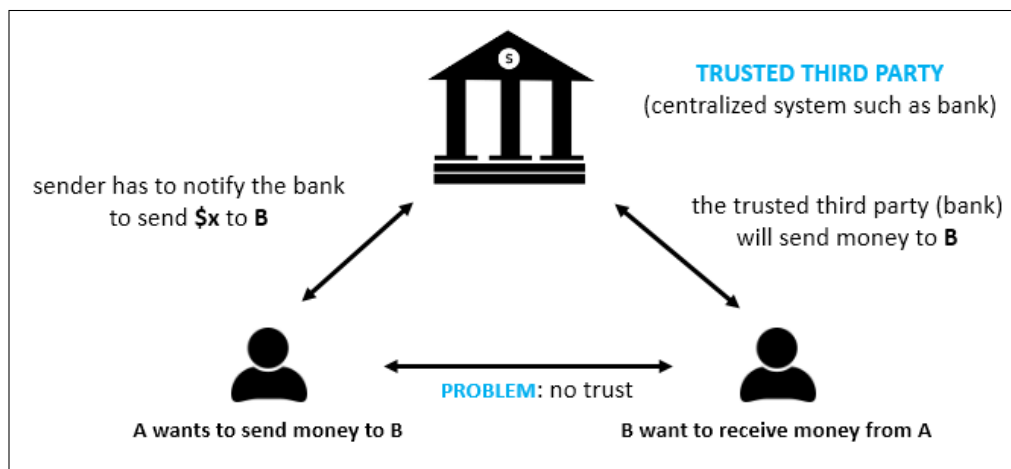


Figure 1.3: A trust-less money laundering system requiring the third party

We also have decentralized networks like block chain networks to guarantee trust. So, clients within this decentralized network can send money without a trusted third party because the block chain itself guarantees trust. So that's the motivation behind block chains, that we would like to get rid of the trusted third parties to some extent and form a decentralized network of clients; clients who need the help of this block change technology, where the block chain itself guarantees trust.

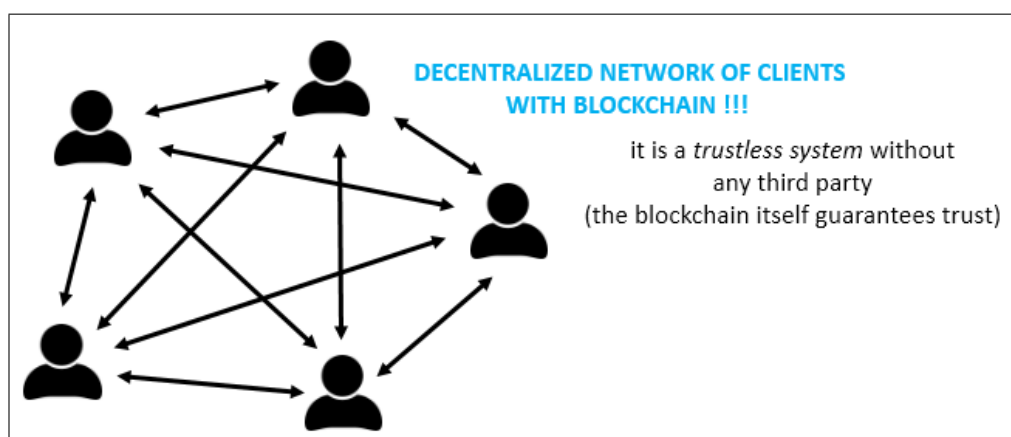


Figure 1.4: Block chain based trusted money laundering network

So, this is the motivation behind my thesis to introduce a voting system using block chain. A voting system, where clients can themselves be a part of, where the clients can act as a mediator of the election process, along with less dependency

on a third party and faster time to results publication. Also, where the clients would be able to vote, and prove to others that they have cast the votes, without actually revealing to whom they have given the vote.

1.3 Immutable nature of Block Chain

In a centralized system, there is a central database or a central server where the data is stored. And this is called a centralized ledger. Let's take a look at an example that we have already explained earlier in this thesis documentation. 'A' wants to send some money to 'B' and 'B' wants to receive money from 'A'. The problem is that they do not trust each other and this is why there are trusted third parties in this case, for example, banks. The bank is going to have a centralized ledger, which is basically a database with all of the given transactions. In this case, the transaction is that a certain sum of money to be this transaction is going to be inserted in the central ledger and a trusted third party. So, in this case, they give them the bank, which owns the given database decentralized ledger, with these transactions. The bank maintains all the information such as the identity of the clients: 'A' and 'B' along with their previous and current transaction(s). So, in this case, that person 'A' sends X dollars to person 'B'. The problem is that if the central database is compromised then the entire system is compromised. This centralized architecture is very similar to the client-server model. The server is the center and all the clients can contact these centralized servers in order to get the given information. The data is present on a single logical server and anyone with the right credentials username and password can access and manipulate these systems.

On the other hand, in decentralized systems, there is no central database. There is no centralized server storing the data. That's why it is called a de-centralized ledger. We are not able to hack the system because we should hack most of the nodes in the network in order to do so. So, it is a peer-to-peer (p2p) decentralized network of nodes and every node in the network has a copy of the block chain. So, every client has a copy of the block chain. In this case the block chain stores all the transactions. So, the block chain is the decentralized ledger and all the clients

have a decentralized ledger. So basically, there's no central database and there's no central server.

Even if someone tampers the data in a given block the cryptographic hash changes as well. So, the pointers are broken, breaking the block chain in turn. That is, if we change something in a given block for example the transaction done because the hash is generated based on the data, based on the previous hash, the value of the hash will be changed. If the hash is changed then the next block's previous hash is not going to match with the hash in the previous block. This means that there's going to be a problem in the block chain. This is how we can detect malfunctions. So even the hash pointers are not available. So, if the previous hash is not the same as the actual hash in the previous block then we know for certain that something has changed in the block because the hash is generated based on the data in the given block.

The hacker may change the hash values in other blocks as well. But it is extremely hard to hack de-centralized systems because instead of hacking just a central database one has to hack every single client. After all, every single client or every single node in the network has a de-centralized Ledger. So, one has to change the ledger in every single node in the decentralized network. So, that's why the data that has been written or recorded to a block in a block chain cannot be changed or erased. This is why block chain are called immutable. So, it is called an immutable decentralized ledger because we are not able to change this ledger if we are dealing with a centralized system such as for a trusted search party.

For the banks, we just have to hack the bank's database and we have to manipulate these transactions. And if we have managed to change these transactions then basically, we have hacked the system. So, this is why decentralized blocks are quite powerful, because every single node has a decentralized Ledger. So, if we hack a given node basically it doesn't matter because all the other nodes in the network have a valid ledger.

For example, we have a block chain and we had four transactions. Let's consider the situation that every single block within the block chain stores just a single transaction and the data and transactions are visible to everyone in the network

because every single node is going to have a copy of the decentralized ledger. If a node added a transaction, it is going to notify every single node in the network in order to update their ledgers. So, in this case, every single node in the network will know that this client added a new transaction. So that's why they update their block chain as well. And if for example, the hacker manages to tamper with this block first of all the hacker has to change the hash pointers in subsequent nodes as well, in order to make sure that the hash references will be valid. Even if we assume that the hacker is going to hack the system.

Because every other node in the network has developed a copy of the ledger. So, the nodes are going to notify this user that the block for the block chain is not valid. So, the user is going to get rid of these tamper blocks and going to have the valid block chain again. And basically, this is the most powerful feature of block chain that we are not able to hack the system. We have to hack at least half of the nodes in the network (also referred to as a 51% attack) in order to be able to manipulate the given block chain with invalid transactions. This makes block chain an immutable ledger.

1.4 Problem Statement

With everything digitizing why should our voting system operate like the old ways by sticking to the ballot papers method. The EVMs used in voting, generate controversy on the ground of being rigged worldwide, especially in our neighboring country India, no matter how secure they are. Alternatively, electronic voting being a viable alternative to traditional voting, still is untrustworthy based on security issues. The software can be undermined, rigged, manipulated, and may be damaged. Elections always need to be secure and trusted irrespective of the scope and size and at the same time must ensure the anonymity of the voters. So, a block chain based e-voting system can be used to secure the voting data along with the transactions. Albeit contrary to some beliefs, block chain provides only security and immutability of the data, not privacy. Here, as will be proposed in the upcoming sections, Zero-Knowledge proof, can be utilized along with smart contracts of block

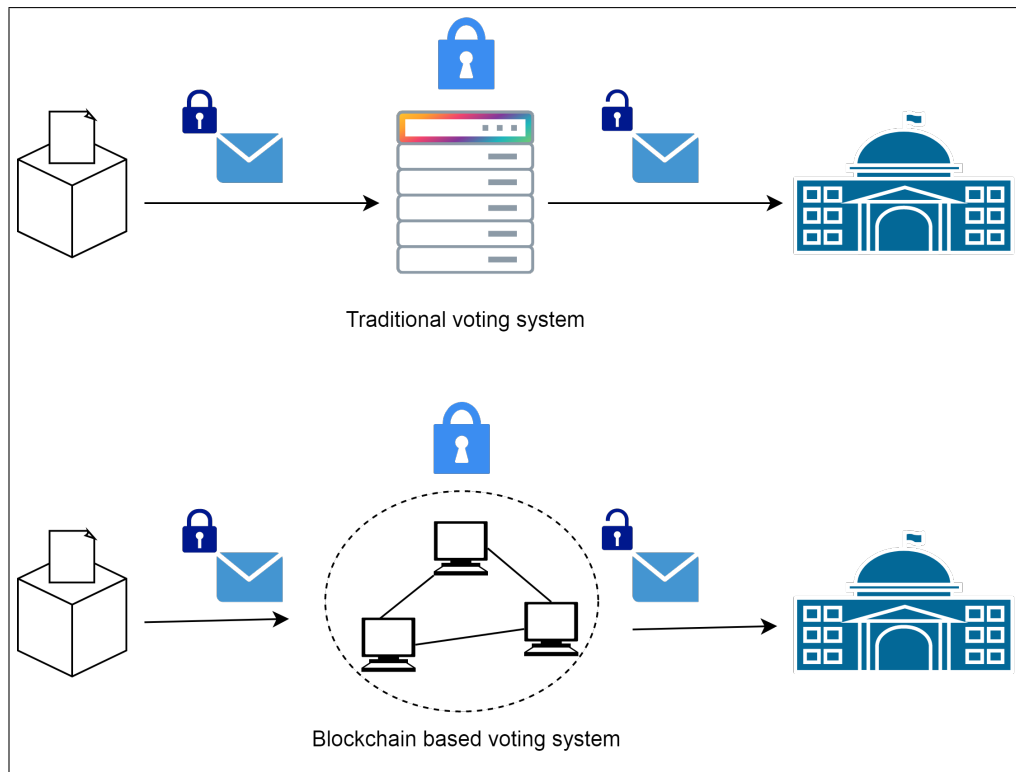


Figure 1.5: Traditional voting system vs block chain-based voting system

chain to preserve the anonymity of the voters and the transactions of the vote itself.

1.5 Objectives

The objective of this research work is:

- Design and create a blockchain-based e-voting system that preserves the anonymity of the voters and their votes.

1.6 Scope of the thesis work

There are many types of voting systems, researched upon. The security and complexity of the systems vary upon the use case. This research work focuses on designing an e-voting system that uses block chain technology to store votes in such a way that even if the blocks are shared by all the nodes, the anonymity of the voter's vote is preserved.

1.7 Probable implementations of this thesis work

The proposed e-voting system in this thesis work can be applied from small organization-wide elections to country-wide elections; the parameters proposed in this thesis work can be adjusted to meet and balance the security requirements and the capacity of the machines at the organizer's side.

1.8 The originality of the thesis work

Many kinds of research work on conducting e-voting and the use of EVMs have already been done before this thesis work, as will be presented in the Section 2. Many have also incorporated block chain as a means to store voting data by using the block chain privately (in other words, within the confined network of an organization). This thesis work proposes a system that utilizes a public block chain to securely and anonymously store voter data.

1.9 Organization of the thesis work

This thesis report has been organized as below:

Section 1 lays out the introduction to the voting system, and e-voting system, along with block chain, the objective of this thesis work, its scope, and the probable implementation areas.

Section 2 lays out the literature review on the previously done research on the domain of e-voting.

Section 3 lays out the process and methodologies applied for this thesis work to create the desired system.

Section 4 lays out and discusses the results obtained and the various observations derived from them.

Section 5 lays out the timeline followed for the accomplishment of this thesis work.

Section 6 provides the conclusion of this thesis work.

CHAPTER 2

LITERATURE REVIEW

Paper [1] presents a modified blockchain technology for the healthcare sector to improve performance and overcome some gaps that appeared in normal blockchain techniques. They proposed building a real and convenient blockchain environment for medical applications, along with an Improved Modification RSA Algorithm (IM_RSA), and developed a Lightweight Secure Hash Function algorithm of high randomly generation keystream sequence.

Paper [2], has explored the application of blockchain in food supply chain management. They investigated the implementation of blockchain in food companies. And how such companies implement the blockchain in food supply chain management.

Paper [3], has implied to make sure that the customers can find the genuineness of the product without relying on the words of middlemen. Here the authors have implemented a block chain-based system of authenticating that manufacturers can ensure that their products are genuine without having to run direct-operated stores, utilizing the tamper-proof nature of data in block chain.

Paper [4], implies the usage of block chain in the health domain for securing patient records, along with clinical trials and the supply chain of drugs and medical devices. Also, various research work has been done on block chain to secure and prove the authenticity of data in various domains [5, 6, 7].

The paper [8] proposed a leader-free algorithm to achieve consensus in a partially synchronous system. A partial synchronous system is an asynchronous system that eventually changes to synchronous mode. The consensus leader-free algorithm for the synchronous system is extended for the partial synchronous system.

In the paper [9], the authors analyzed proof of work in detail. In Proof of Work (PoW), introduced by Bitcoin all the nodes vote by solving a proof of work (a complex calculation operation) and create new blocks with their computation power. The authors have measured the average time to mine a block, the number

of stale blocks, and the average fork length varying problem complexity. It was found that the mining time for new blocks was directly proportional to the problem complexity.

Paper [10], proposes Proof of Vote to be more efficient than Proof of work (PoW). In the Proof of Vote (PoV), the voting mechanism verifies the blocks. The authors have defined the roles: commissioner, butler candidate, butler, and ordinary user in a consortium network model. It was found that the PoV is the most power efficient.

In the paper [11], an analysis and comparison of the parameters related to the performance and security of consensus in block chain algorithms have been done. The paper intends to act as a guide for developers and researchers to evaluate and design a consensus algorithm.

Paper [12], highlights and summarizes the benefits of block chain on personal data protection by using zero-knowledge proof which is Zerocoin and Zerocash.

Paper [13], reviews and presents open research challenges of block chain in e-voting domain.

Paper [14], proposes an e-voting system that allows voters to connect to the system having an easy-to-use user interface, through which they can cast their vote by importing their account and can easily review their vote.

Paper [15], also proposes a similar voting system to paper [14]. The problem with these types of systems is that the blocks if fall into the wrong hand, easily reveal the voter's vote.

In paper [16], a block chain based e-voting system has been proposed. Here it has been proposed to encrypt the voter's vote along with voter's data. Not much detail has been provided on the type of encryption to be used and also, and the process of counting the votes has not been mentioned properly.

Paper [17], proposes a sealed bid auction mechanism, where the participants themselves are part of the block chain, using Pedersen Commitment's slightly modified version bulletproof to achieve zero knowledge proof to compare the auctioned amount's positive differential without revealing the auctioneer's data.

My thesis work has been inspired by this paper to hide the voter's data in such a way that they are visible in the blocks without revealing the actual content of the data.

CHAPTER 3

METHODOLOGY

3.1 Block chain structure

In this section, we are going to look at the underlying data structure and the technology itself. Block chain is an incrementally growing list of transaction ledgers called blocks which are linked together in a chain-like structure. This block chain technique was first constructed in the early 1990s by Stuart Haber and W. Scott Tormetta to timestamp digital document records in a tamper-proof way. This block chain technique was later used by Satoshi Nakamoto; it is actually considered a pseudo name or an alias for a person or group of persons; used in 2009 to create the most popular digital cryptocurrency “Bitcoin” [18].

In the block chain, multiple transactions are stored in a ledger and each ledger consists of numerous transactions. The blocks are continually growing, each time, adding new sets of transactions. The blocks grow in a continual list structure, consisting of only a single branch. Thus, this single branched structure forms a name, thus its name derived to be: block chain. The chain is formed based on the linking of references. Each block in a chain refers to its previous block using a reference, referred to as a previous hash. It also exposes its own reference in the form of a hash. Thus, the subsequent block can be used to refer to the previous block using the hash. The hash of the current block can be acquired by applying a one-way cryptographic function to the content of the block itself. The hash used in this thesis work is Secure Hash Algorithm 256 or SHA-256 [19] for generating a hash of the block. Alternatively, SHA-512 can also be used [20]. The choice of using a slow hash function instead of a faster one will be explained in the Proof of Work (PoW) section.

The first block in the block chain is called the “Genesis Block”. Since it is the starting block and does not refer to any previous blocks, its previous hash value is generally set as “000..”. The number of ‘0’s is equal to the length of the hash

generated by the hashing function. In my case, the SHA-256 hashing function was used, which produces a hash of length 64, irrespective of the input object, i.e. the output is always 64 characters hexadecimal string.

Being hexadecimal, $[0 : 9]$ and $[A : F]$ are the possible values for any particular character. It means there are 16 possibilities that can be represented on 4 bits ($2^4 = 16$). Because the hash itself takes up 256 bits in the memory and every character's size is 4 bits, that's why the length of the output is 64. An example of a hash generated by the SHA-256 hash function is:

37f47ded94c31186f3a1d6c27fb7d607847ff2a91b4e98d84e1e28ec583cedbd

A hashing function have the properties of being: deterministic, one-way, collision-free, and having an avalanche effect.

- **The deterministic** property of a hash function denotes the production of the exact same hash output for a given input object.
- **One-way** property of a hash function denotes the ease of producing a hash from a given object and the extreme difficulty of restoring the same object from the given hash, denoting the trap-door functionality of the hash function.
- **The collision-free** property of a hash function denotes the very low probability of 2 different objects producing the same hash function.
- **The Avalanche Effect** property of a hash function denotes the vast change in the output of the hash function, even for a slight change of a single bit in the input object.

Let's say that the hash of the GENESIS block is '056FH' as shown in Figure 3.1. Now, the subsequent block: "BLOCK 1" will now refer to this genesis block by storing the hash '056FH' as its previous Hash. The referencing of the previous hash by its subsequent hash follows in this manner.

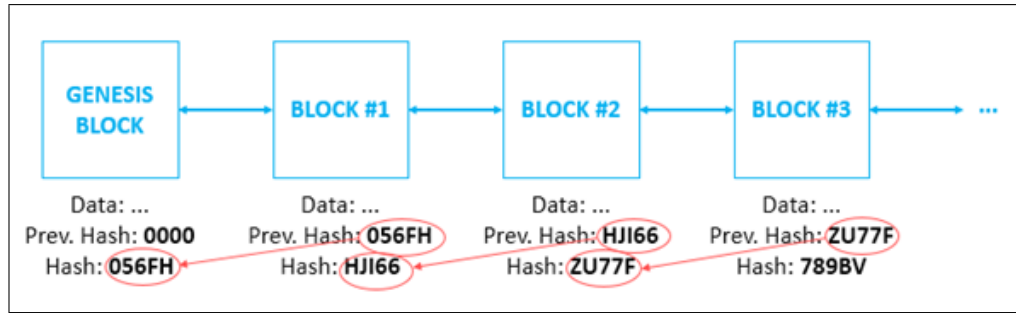


Figure 3.1: Blocks in a block chain

3.1.1 Blocks in a block chain

Each block in a block will have elements as shown in Figure 3.2.

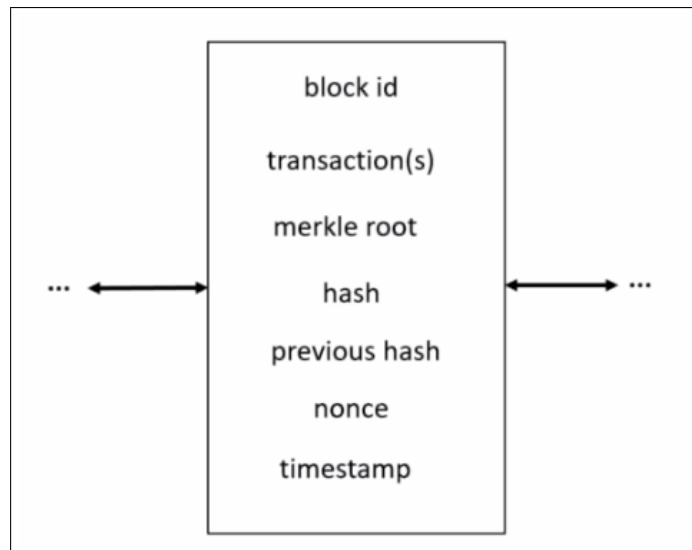


Figure 3.2: Elements of a block

The block id is the identification of a block.

The data is a data structure containing numerous other data structures and lists. The data used in my case for this thesis work is the List of voters, who have voted and the List of Cumulative encrypted votes, achieved by performing homo morphic addition, which I will explain in further sections.

The hash, as previously mentioned is the hash obtained by passing the block to the SHA-256 hash function. The generated hash acts as the fingerprint of the block, denoting the uniqueness of the block.

The previous hash is the hash referring to the previous block as explained earlier. The timestamp, as the name suggests is the timestamp at which the block was created.

A nonce is a number generated only once. It will be further discussed in the upcoming section on the mining of a block.

3.2 Mining of a block

When dealing with decentralized systems trusted the third party like the banks are going to handle the transactions with software and database updates and so on. That's why all the clients have to notify the bank in order to make a given transaction. In this case, the bank, which is a centralized system handles the transactions. But the problem is that in a decentralized system such as block chains which node will handle the transactions. And basically, this is why mining came to be. Miners are special node(s) that will handle and verify the transactions. We must have heard of the miners getting a reward in bitcoin mining. Getting a reward is not the aim of mining, it is just the byproduct. In other words, mining is the mechanism that allows the block chain to be a decentralized system because without a trusted third party no one is to verify the transactions and basically with the help of the mining procedure, miners can verify these transactions.

Mining is about finding the right hash values for the blocks and adding these blocks to the block chain. Firstly, miners will verify transactions, and add the blocks to the block chain. They are going to find hash values for their given block. And this is how they can add the given block to the block chain. Miners reveal verified transactions and add the blocks to the block chain.

But the question arises on how to find these given hash values and how to know that these hash values are generated by miners. So this is why mining is the fundamental concept of block chain because miners will generate these hash values and the miners will make sure that these hash pointers are valid. It is a computationally heavy procedure to find these hash values. And basically, that's why miners need good computers with good CPUs, GPUs, and specifications in order to be able to

find these hash values quite fast. Firstly, let us look at the hash value generated from SHA-256 hash function.

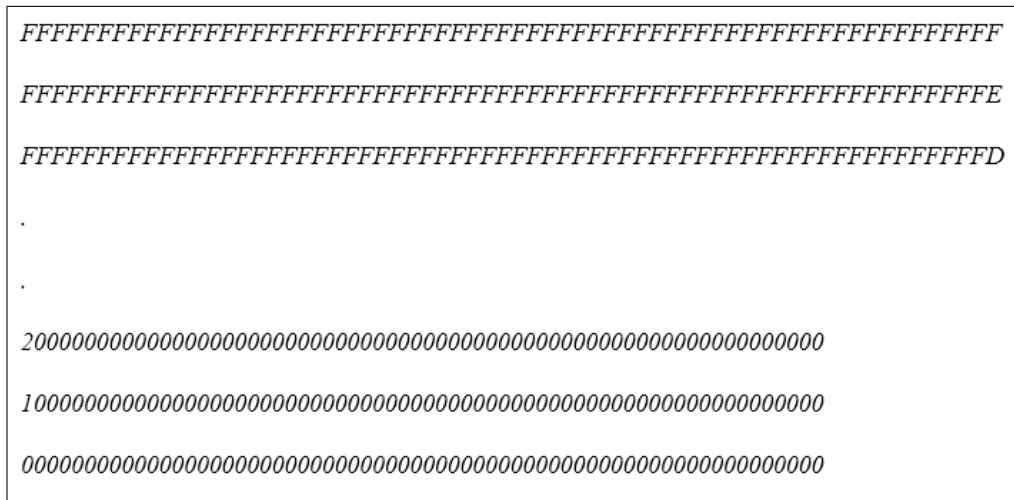


Figure 3.3: Range of SHA-256 hash function outputs

One hash takes up 256 bits in the memory with binary values (0or1): it means the total number of hashes is 2^{256} . Our hash output ranges from all 0's; 64 number of zeros; to all *F*'s; 64 number of *F*. The same result is in hexadecimal format: there are 64 hexadecimal characters (so 16 possible values) which yield 16^{64} possible combinations. It is an extremely huge value.

Mining aims to find the right hash value and there is one more important concept we have to understand it is the difficulty level and it is characterized by leading zeros. Mining aims to generate hashes, but there are some constraints: most of the generated SHA-256 hashes are not allowed. It is extremely important that we would like to make sure that the mining procedure is computationally heavy so as to make sure that the block chain is going to be a secured decentralized network.

It is going to take a while to generate the valid hash values. So the difficulty of mining is defined by the leading zeros. So let's take a look at a given example. The difficulty can be defined by the number of leading zeros. Let's say the difficulty is four which means that the first four characters of the 64 characters long hexadecimal string must be all zeros. In this case, there are four leading zeros in the hash. So the aim of mining is to find an arbitrary hash if it is for leading zeros. We call it difficulty because the more the leading zeros are there the harder to find the given

hash. So let's calculate the probability of finding a hash with one leading zero. We have just a single leading zero then we have to calculate the total number of shares which is 16^{64} . We have to calculate the number of hashes we want leading 0 and all the other characters can be anything. So that's why $(64 - 1)$ which is 63.

`07d38ebf07b0ca1ed92f3cdce825df28d36d8fdc39904060d2c18b13c096edc`

$$\begin{aligned} P(\text{finding hash with 1 leading zero}) &= \frac{\text{hashes with 1 leading zero}}{\text{total number of hashes}} \\ &= \frac{16^{63}}{16^{64}} \\ &= 0.0625 \end{aligned}$$

So it means that there is approximately a 6 percent chance that if we generate a hash at random then the first character will be zero. Let's calculate the probability for two leading zeros. So was the probability of finding a hash with two leading zeros. We have to calculate the number of hashes with two leading zeros and divide them by the total number of hashes. It means that the first two characters are fixed. And we just have to bother about the other characters. So that's why there are 16 to the power of $(64 - 2)$ which is 62. So that's why there is 16^{62} . This is the number of hashes with two leading zeros. So that is going to be the probability is going to be about 0.4 percent if we generate a hash at random it will have two leading zeros.

`00d38ebf07b0ca1ed92f3cdce825df28d36d8fdc39904060d2c18b13c096edc`

$$\begin{aligned} P(\text{finding hash with 1 leading zero}) &= \frac{\text{hashes with 1 leading zero}}{\text{total number of hashes}} \\ &= \frac{16^{62}}{16^{64}} \\ &= 0.0039 \end{aligned}$$

The actual difficulty of Bitcoin is 18 leading 0s. For, the probability of finding the hash with 18 leading zeros. We just have to calculate the total number of hashes with 18 leading zeros which is $16^{(64-18)}$ which is 46 divided by the total number of

hashes and we get an extremely small number.

00000000000000000008f3cdce825df28d36d8fdc39904060d2c18b13c096edc

$$\begin{aligned} P(\text{finding hash with 1 leading zero}) &= \frac{\text{hashes with 1 leading zero}}{\text{total number of hashes}} \\ &= \frac{16^{46}}{16^{64}} \\ &= 2.1 * 10^{-22} \end{aligned}$$

If we make sure that there must be lots of lots of leading zeros in the hash we are able to define how hard it is to find the right hash. To get the right hash, we use all the information present in the block and feed these data to the SHA-256 algorithm to get the output which is a 64-character long character. We are basically going to use all the data within the block in order to generate the hash. Since most of the data is immutable and if we want to generate a new Hash we have to change something away in the block. We are not able to change the block ID. We are not able to change the data of course. We are not able to change the previous hash because then the hash pointers would be invalid. And basically, this is why we have these nonce variables. We can change the value of the nonce and thus we change the output of the SHA-256 hash. So the mining procedure is to change the value of these nonces until we find a given hash will be the right amount of leading zeros. As for now for Bitcoin, the difficulty can be characterized by 18 leading zeros which means that we have to generate lots and lots of hashes, and have to change the value of the nonce lots and lots of times before we end up with the right hash value of the given block.

The number only used once or nonce, as it is asserted to be is usually an unsigned integer. So the range is zero up to 4 billion and during the mining operation, miners change the value of the nonce. In java, the primitive int data type is of 4 bytes, which means that it has a maximum value of 4,294,967,296. For every possible value of the nonce, the miners get a new output hash. If we change something in the block, the output of the algorithm will change so we get a different hash because the input has changed. The very important thing here is that the miners cannot guess the value of the nonce because of the avalanche effect of the hash

function, as mentioned in the earlier section, and will have to increase the nonce value by one without skipping any of the nonce value.

Usually, the miners have to start with zero and keep incrementing the value of the nonce by one until they find the golden hash. The golden hash is the hash with the right amount of leading zeros. So, basically the hash we can use for a given block in the block chain, is called the Golden hash. The process of making mining so difficult is called the proof of work. The aim of mining is to verify the given transactions, make them into a block and add the block to the block chain. In my thesis work, the difficulty level is configurable and ready to be made available to the miner nodes via the smart contract.

3.3 Smart Contracts in Block chain

Smart contracts are static rules for block chain technology that run when predetermined conditions are met and determine the overall working of the block chain. They are used to complete the execution of pre-defined criteria, so that all nodes can be certain of the outcome, without any third party's involvement or time loss. They can also automate a workflow, triggering the next action when conditions are met. They are basically programmed using a high-level language called solidity [21], but for this thesis work, a similar implementation of smart contracts has been done using another high-level language: java [22].

3.4 Memory Pools

The memory pools or shortly referred to as mempools are the local cache of the transactions that each node keeps until miners perform mining on them and are later added into blocks. Every node after generating a transaction broadcasts that transaction to all other nodes in the blockchain network, including the miner nodes. The receiving nodes will take the transaction and store it in their mempools. The miner nodes will take up the transaction, from the mempool, depending upon the limit of transactions per block. Then the miner will perform pow on them and once the golden hash is received, the miner will add these transactions to the block and

broadcast the block to all the other nodes in the block chain network. Here in this thesis work, the transaction limit for each mining is made to be fully configurable via the Smart contract.

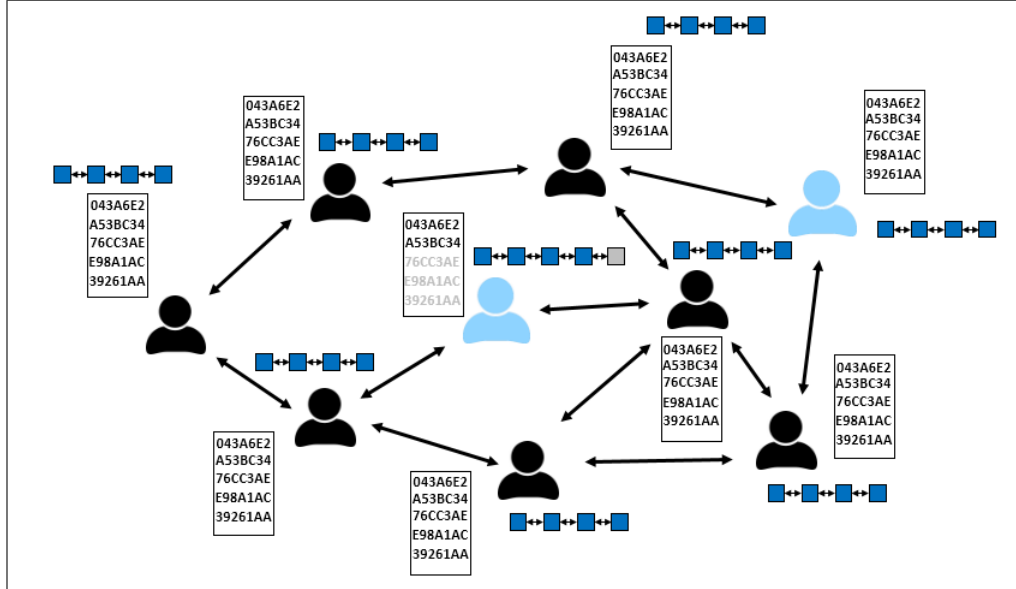


Figure 3.4: Memory pool in a block chain

3.5 Elliptic Curve Cryptography (ECC)

There's a huge problem with the public block chain; all the data is public. So, somehow, we have to encrypt the transactions and have to make sure that other nodes in the network can verify these transactions. The problem is that we can create multiple new vote transactions for a given party of my choice, then one of the miners will take that given transaction and put that transaction within a given block. And now the block is in the block chain with a valid transaction. So, it means that we are able to send numerous votes to our favorite party. And that's why for my thesis work, the voting system of block chain network uses Elliptic Curve Digital Signature Algorithm (ECDSA)[23] to ensure that votes can only be cast by the rightful voter to a valid party one time.

For this, a private key and a public key together forming a keypair are going to be generated. The private key is a secret number 256-bit integer known only to the person that generated it. It is then used to sign the transaction; the private

key is secret. Other nodes of the network do not know about this private key and we can generate a public key based on the private key and not the other way around. And there's no need to keep the public key secret because it is extremely hard to get the private key from the public key. The public key is a point on a two-dimensional plane. Basically, the coordinates are on an elliptic curve. When we create a transaction, we can sign it with the help of the private key, and we kind broadcast that transaction with the given signature, and anyone else can verify that the transaction belongs to us. With the help of the public key which is easily accessible, we can verify the message that has the signature. And this is how the nodes of the voting system I proposed ensure that votes can only be cast by their rightful voters.

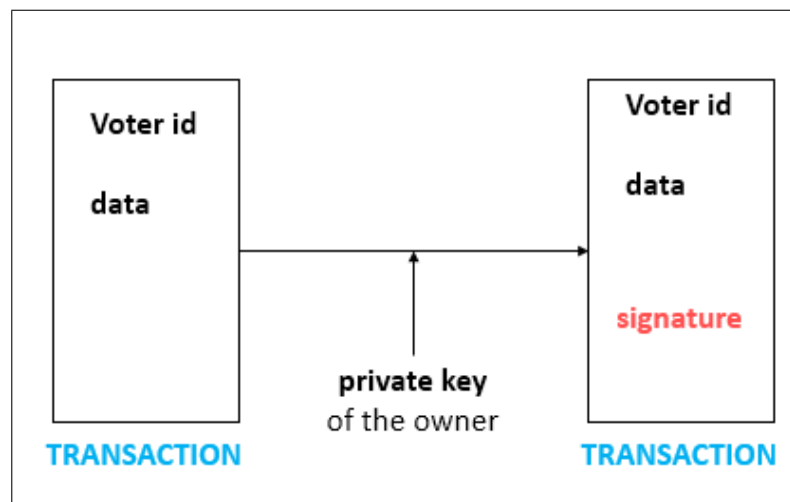


Figure 3.5: The process of signing a transaction

The choice for signing the blocks with an ECDSA key comes from the fact that it has a shorter key length compared to Rivest–Shamir–Adleman (RSA) cryptographic algorithm to provide the same level of security. Table 3.1 illustrates the difference in key length between ECDSA and RSA to provide the same level of security in bits.

3.6 Zero Knowledge Proof

In this thesis scope, Zero Knowledge Protocol (ZPF) can be defined as a way for the voter to prove that they have actually voted, without actually revealing their

Table 3.1: Comparison of RSA vs ECDSA

Security (In Bits)	RSA Key Length Required (In Bits)	ECDSA Key Length Required (In Bits)
80	1024	160-223
112	2048	224-255
128	3072	256-383
192	7680	384-511
256	15360	512+

vote. For this thesis work, I have chosen the Paillier cryptosystem[24], named after and invented by Pascal Paillier, which is a probabilistic asymmetric algorithm for public key cryptography. The probabilistic mentioned in the previous sentence means that for the same key pair and for the same input, the encrypted results are vastly unrelated, due to the introduction of some random factors. I chose the Paillier cryptosystem, as it supports the homomorphic sum of plaintexts. That is with the help of the Paillier cryptosystem, we can easily perform the addition of two cipher texts, obtained from encrypting a number with the same key pair without actually needing to decrypt the cipher texts during the process. Later using the key-pair, the homomorphic sum can be decrypted to a value, which is the result of the sum of the previous two numbers.

This makes our communication so secure and protected that nobody else can find out which voter has voted for which party and how much a party has achieved in votes, until the end of election time as governed by the smart contract.

The Paillier algorithm as taken from[24] is:

3.6.1 Key generation

- Choose two large prime numbers p and q randomly and independently of each other and of equal length; here p and q act as a seed for the creation of the key pair.
- calculate $n = p * q$ and $\lambda = lcm(p - 1, q - 1)$
- Select random integer g where where $g \in Z_{n^2}^*$
- Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse: $\mu = (L(g^\lambda \text{mod } n^2))^{-1} \text{mod } n$, where function L is defined as where, $L(\mu) = \frac{\mu-1}{n}$
- The public encryption key is (n, g) and the private decryption key is (λ, μ)

3.6.2 Encryption

- Let m be a message to be encrypted, where $0 \leq m \leq n$
- Select random r where $0 \leq r \leq n$
- Compute ciphertext as: $c = g^m \cdot r^n \text{ mod } n^2$

3.6.3 Decryption

- Let c be the ciphertext to decrypt, where $c \in Z_{n^2}^*$
- Compute the plaintext message as: $m = L(c^\lambda \text{ mod } n^2) * \mu * \text{ mod } n$

3.6.4 Homomorphic addition property

- The product of two ciphertexts will decrypt to the sum of their corresponding plaintexts such as:

$$D(E(m_1, r_1) * E(m_2, r_2) \text{ mod } n^2) = (m_1 + m_2) \text{ mod } n$$

A simple implementation of the working of the Paillier algorithm can be found in the APPENDIX C.

3.7 The proposed system

The key elements of the proposed system are the voters, the miners, and the system consisting of both the voters and miners.

The voters will have to pre-register themselves from the election committee by showing their eligibility by providing valid identification, along with an at least 8-character password of their choice, their mobile number; where the One Time Password (OTP) will come during the start of election phase and which would correspond to only a single vote; all of which will be used for logging-in in the election. The election committee after registering the voter would provide them a unique 12-character string for uniquely identifying a voter.

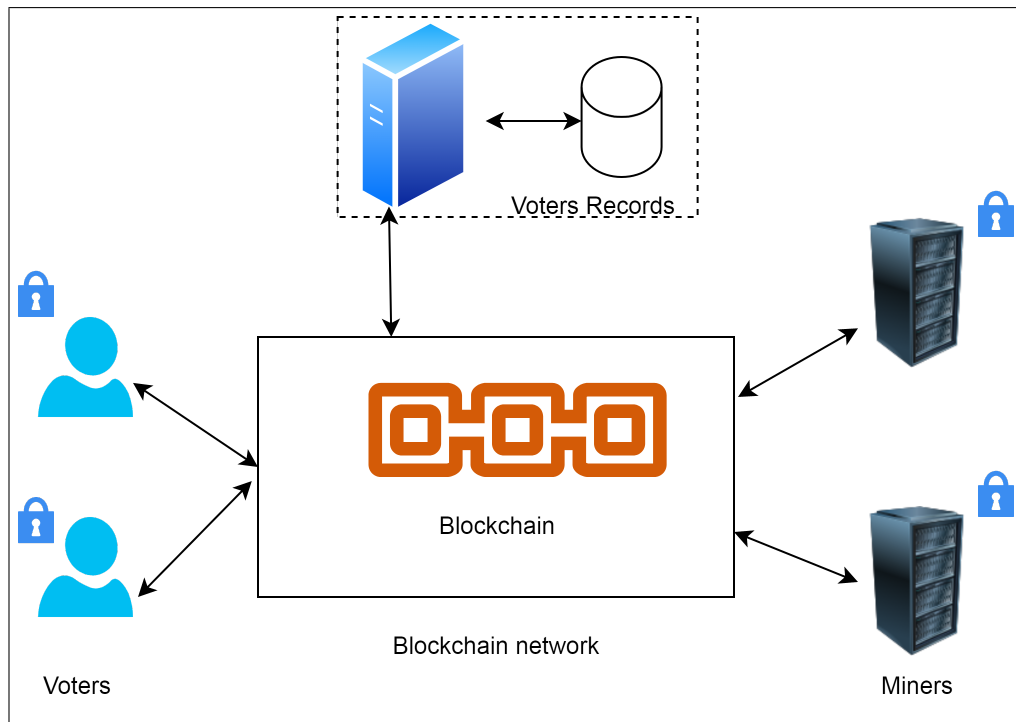


Figure 3.6: The proposed system for the E-voting system

During the election phase, the voters after entering their 12-character string along with their password to a Voting User Interface will receive an OTP. The combination of these 3 things will help to identify a valid unique voter. Once the voter has voted, the voter's side of the application would encrypt the ballot object with the inserted vote by using a key provided by the miner's side, which is further explained in section 3.8. Later he/she may remain connected to the system to store the block chain on their device until the elections end to get the election results once published.

Another element to the system is the miners, which would be the powerful computers performing proof of work at the election's organizing side. They would do the computation work required to create a new block to the block chain. The creation of the block would require the tallying of the maximum allowed votes per transaction, without decrypting the ballot object by performing homomorphic addition, and finding the appropriate hash of the block in accordance with the difficulty level.

More details of the working of the system is explained in section 3.8.

3.8 State Sequence of the voting system

The voting process in my thesis work has been divided into three phases: the initialization phase, the voting phase, and the election end phase. The explanation of each phase can be found in the upcoming sections.

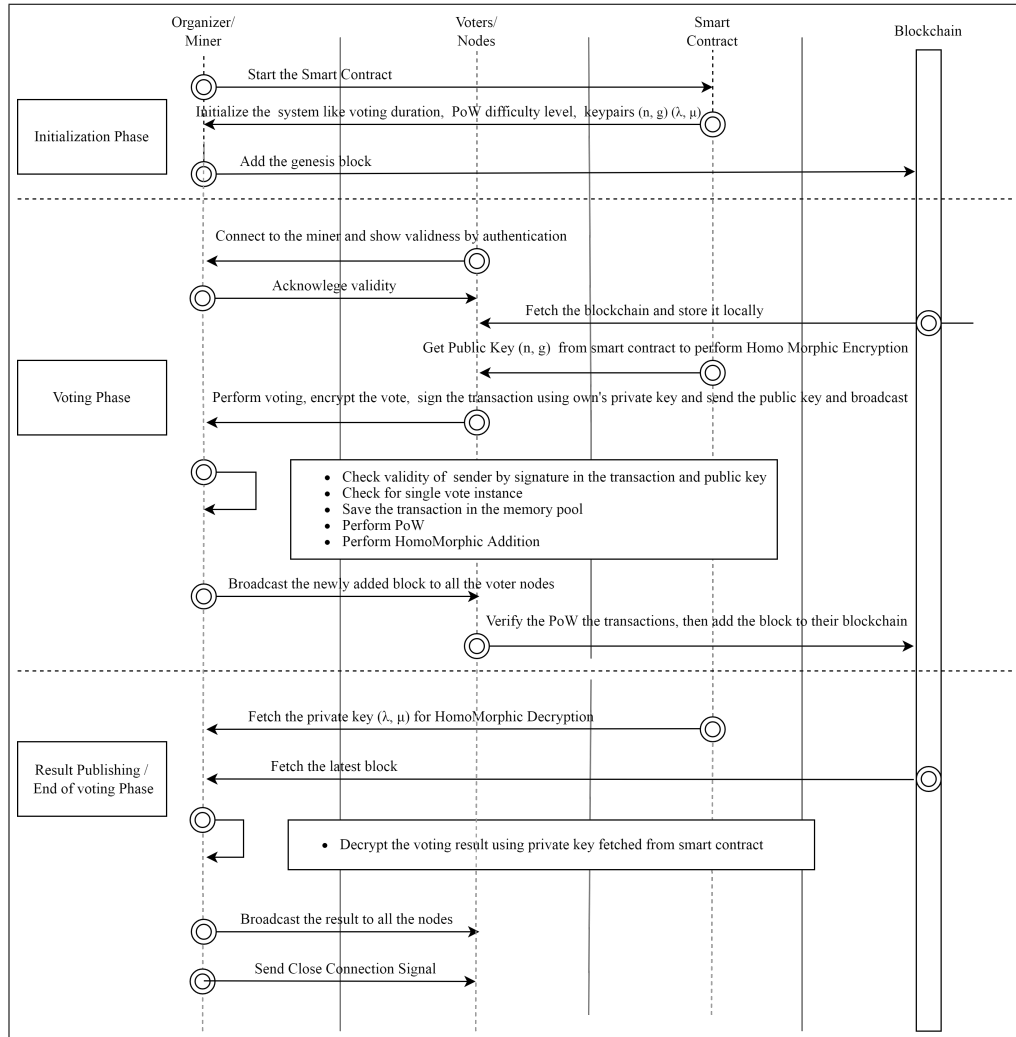


Figure 3.7: State Sequence of the voting system

3.8.1 Initialization Phase

This phase will be driven by the miner and the smart contract. The organizer of the election will first set the configurations of the system like the election period, the difficulty level for mining, and the maximum limit of transactions in the block.

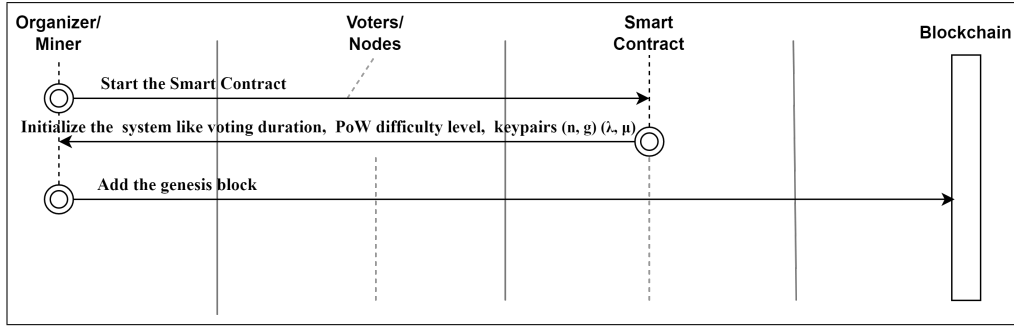


Figure 3.8: State Sequence of the initialization phase

The organizer will then start the miner node and first the smart contract will be instantiated. The instantiated smart contract will then convey to the miners the pre-configured parameters like the election period, the maximum limit of transactions in the block, and the difficulty level for mining. Also, the smart contract will generate a Paillier cryptography key pair; (n, g) as the public key and (λ, μ) as the private key. The smart contract will provide the Paillier public key (n, g) to the miner and will not release the Paillier private key (λ, μ) until the end of the election period.

Based on these configurations, the miner will mine a genesis block and add its local copy of block chain.

3.8.2 Voting Phase

In the voting phase, a voting node with the proper credential will connect to the miner node via passing the 12-character identifier and the password. In turn, the voters would receive an OTP via SMS. Now, after passing OTP to the miner by the voter, the miner will acknowledge the voter node to vote based on the credentials. The unique voter would only be able to vote once during the election phase.

For a valid voter node, the voter node will fetch the blockchain from the miner node, and it will store it locally. Next from the smart contract, the voter node will fetch the Paillier public key (n, g) and using this paillier public key, perform encryption on its ballot object. Since the paillier cryptographic system is probabilistic, the same votes of either 1s and 0s, even though being the same in number, will yield different enciphered text even if encrypted with the same public key (n, g) , one

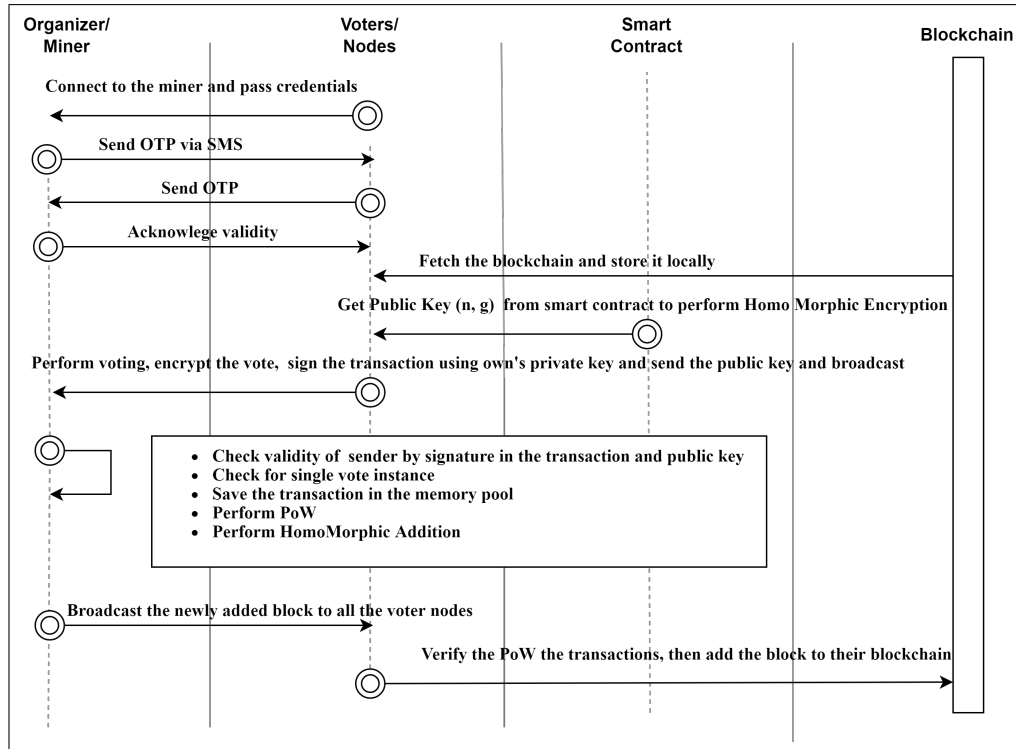


Figure 3.9: State sequence of the voting phase

after another, thus it prevents the exposure of the voter's vote in the ballot object.

Next, the voter node will generate an ECDSA keypair and sign its id and ballot object with the ECDSA private key and add the signature along with ECDSA public key to its transaction. Next, it will broadcast the transaction to all the nodes, where the receivers will be able to verify the validity of the transaction with the help of the signature and the provided public key.

The miner, which continuously runs a loop of mining will verify and store the transaction in its mempool. When the miner has finished mining (if there are any transactions to be mined), it will pop up the transactions from its mempool queue based on the number of maximum transactions allowed in the smart contract. Then based on the Paillier public key (n, g) , the miner will perform a homomorphic sum on the encrypted ballot object. The idea is to store a cumulation of votes in each block so that, during the publishing of the result, the miner will only have to look at the latest block to get the result of the election.

Now, the miner will perform pow, that is it has to obtain the golden hash based on the difficulty level set on the election. Once the desired hash is obtained, the

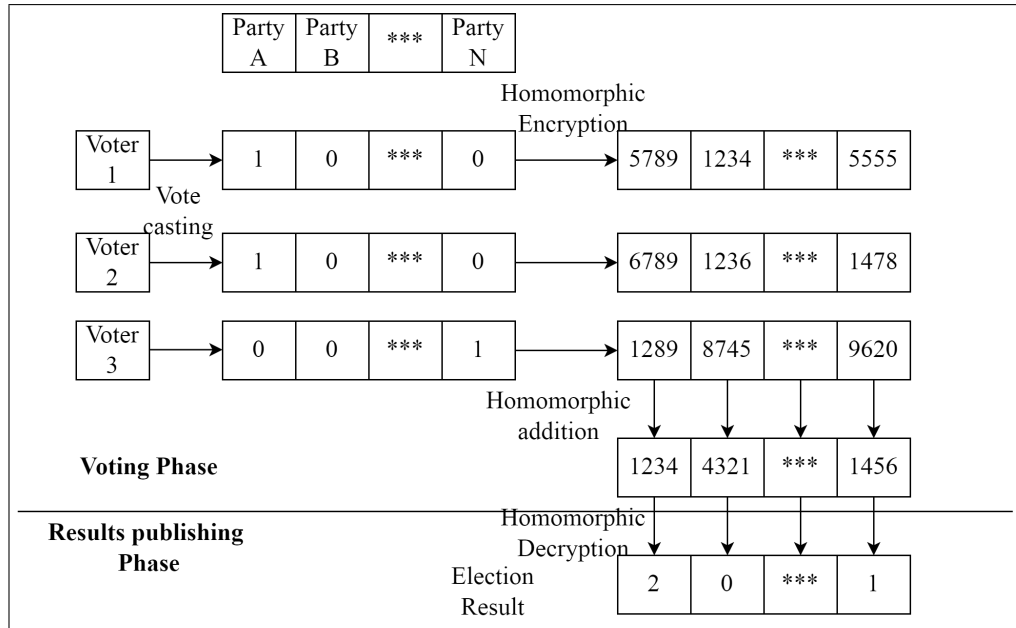


Figure 3.10: The process of homomorphic addition in the cipher texts of the ballot of voters; the short cipher texts are only for representational purpose

miner will broadcast the block to the nodes and add the block to its local copy of block chain.

The voter nodes upon receiving the block will verify if the mining has been done, using the hash value. When everything is right, the voter node will add the block to its local copy of block chain.

This process will be repeated until all the votes received within the voting period have been tallied and mined; even after the election period may have ended.

3.8.3 Result Publishing phase

This is the last and shortest phase of the voting system. Here the miner will fetch the Paillier private key (λ, μ) from the smart contract. The smart contract will only provide the Paillier private key once the duration of the election has ended and once all the mining tasks for the votes within the election duration have been mined.

Now, the miner will fetch the latest block of the block chain from its local copy of block chain.

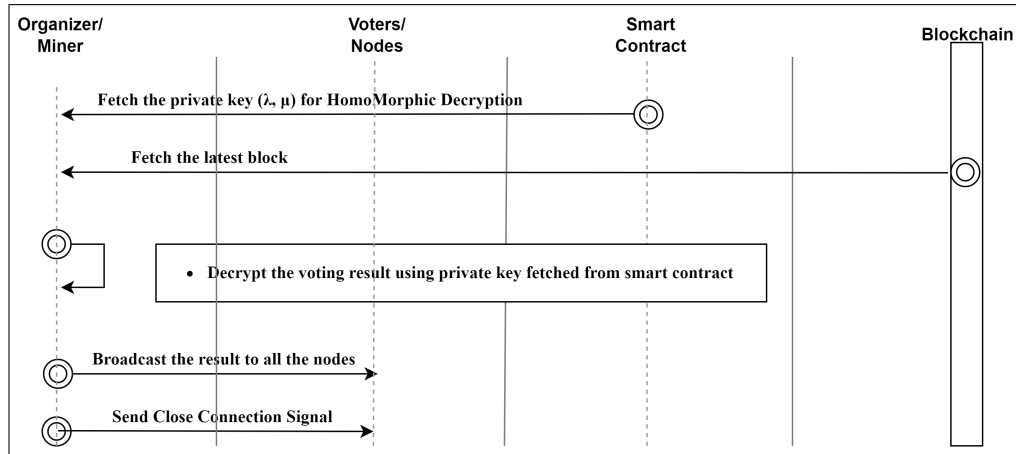


Figure 3.11: State Sequence of the Result publishing phase

Next, the miner will decrypt the cumulative transaction from the latest block using Paillier private key (λ, μ) . Here since before the addition of each block, the results from the previous block and the current transactions are added homomorphically, the result would be in the latest block.

The miner will broadcast all the results to all the other nodes.

And at last, the miner will send a signal to all the other nodes to disconnect from the network as the election would have been successfully organized.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Experimental Setup

This system is built using java language, with java JDK version 1.8 and IntelliJ IDE 2019 version, and also used for the development and testing process of the system.

The specifications of the machine where the experiments have been conducted is tabulated in the Table 4.1.

Table 4.1: Specifications of the machine on which the experimentations have been performed

Specifications	
Processor	2 GHz Quad-core Intel Core i5
Memory	16 GB
Solid-state drive	1 TB

4.2 Paillier cryptography computation time

As mentioned in Section3.6.1, the keypair generation for the paillier operation depends upon the choice of two prime numbers ‘p’ and ‘q’ of equal length, which are the key seeds. The length of these prime numbers will dictate the size of the key pair and in turn affect, the size of the enciphered text generated and the time it takes to perform encryption, decryption, and homomorphic operation. Figure 4.1 shows the different times it takes to perform encryption of votes, decryption of enciphered text of votes, and the homomorphic operations on those enciphered text.

Table 4.2 shows that the higher the seed length, the longer it takes to encrypt and decrypt the paillier cryptography; also, it shows that the encryption and decryption time are very highly positively correlated. Figure 4.1 shows that despite the seed

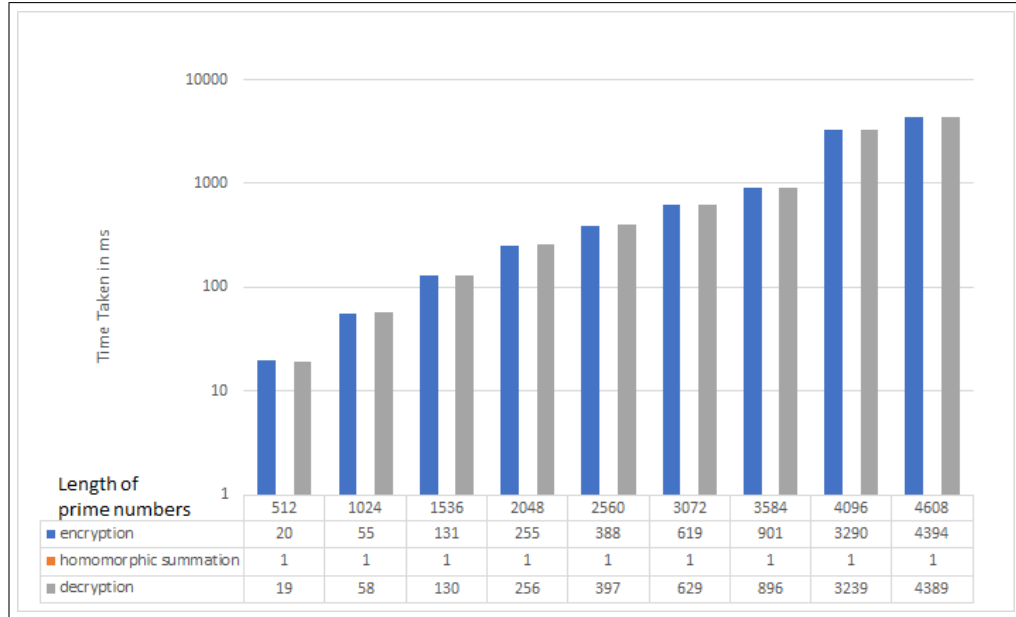


Figure 4.1: Duration of paillier operations for different length of key seeds

Table 4.2: Correlation matrix of seed length, encryption and decryption time of paillier cryptography

	Seed length	Encryption time	Decryption time
Seed length	1	0.811038443	0.811845174
Encryption time	0.81103844	1	0.999959312
Decryption time	0.81184517	0.999959312	1

length, the time taken to perform homomorphic summation is always constant.

4.3 Hash rate per second using parallelism

The hash rate or the number of hashes per second plays a major factor in generating the golden hash in a given range of time. A single thread can only generate a limited amount of hashes per second. Nowadays, we have multi-cored Central Processing Units (CPU). There are two parameters involved with the CPU's cores. They are physical cores and logical cores. Physical cores are the actual hardware cores of the CPU, whereas an actual core is divided into two logical cores for hyperthreading to allow multiple instructions (threads) to be processed on each core simultaneously. In other words, the processor has four cores that are acting as eight cores. Normally a physical core is only divided into 2 different logical cores. As per [25], although the normal approach for designing an application is to use 1

thread per processor, but for high-performance applications deployed in a quad-core system, the optimal number of threads is 3-5 threads per processor, which may differ according to different loads, so testing of different configurations is required to determine the optimal amount of threads for an application.

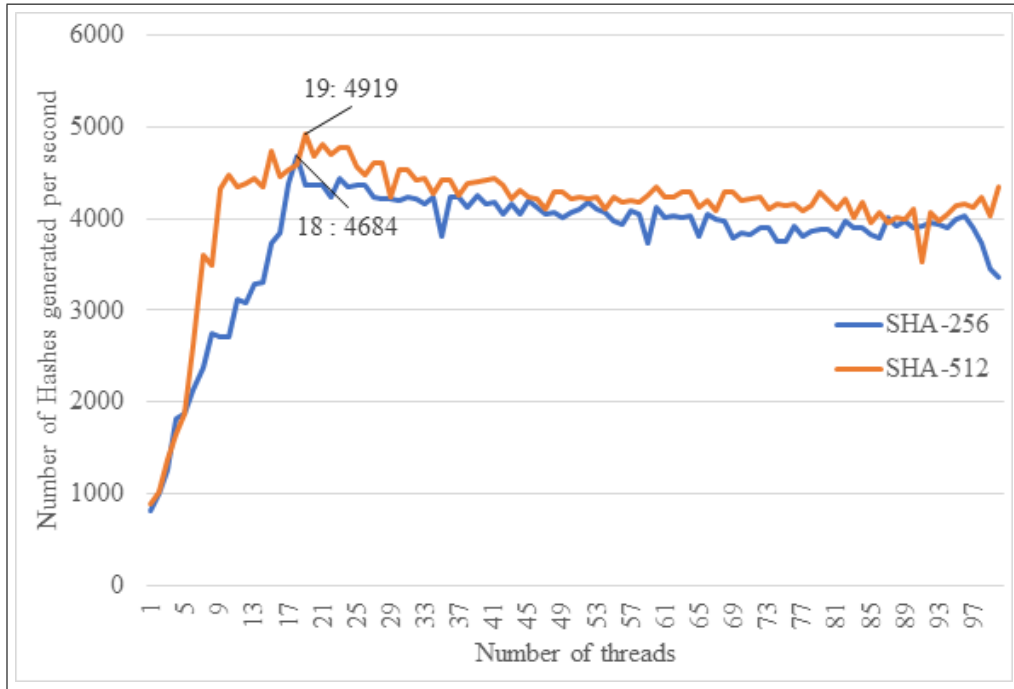


Figure 4.2: Hashes per second for different numbers of threads

The results seen in Figure 4.2, are of experiments conducted on a quad-core CPU for a ballot object size of 5. Each data on result is an average of five different conducted experiments. It can be observed that the peak hash output of SHA-256 is observed while using 18 threads and for SHA-512, it is observed while using 19 threads. The values may vary, depending upon the state of other processes running on the same CPU as well. So for up to 100 threads, we can observe that the peak hash rate was observed only for threads that are multiples of 3-5 of the physical core; the data can be found in APPENDIX B.

4.4 Use of SHA-512 in the system

SHA-512, like the SHA-256, is a hashing algorithm that takes in a variable length of plain text and outputs a fixed length of ciphertext in the hexadecimal format; in

the case of SHA-256, the output is a 64 length of the hexadecimal hash value and in case of SHA-512, the output is a 128 length of hexadecimal hash value; twice in the length of the SHA-256. The total possible combinations of hashes of the SHA-256 hashing algorithm is 2^{256} , and that of the SHA-512 hashing algorithm is 2^{512} .

Currently, bitcoin-like cryptocurrency is self-sufficient with the SHA-256 hashing algorithm. SHA-512 is more secure than and longer in length, it can be applied to the voting system proposed in this report. The probability of finding the golden hash for the difficulty level of 1 in this hashing algorithm is $16^{127}/16^{128}$, which is 0.0625, for difficulty level 2 is 0.00390, and so on like that of the SHA-256. The advantage here is the difficulty level can be extended far beyond that of SHA-256 for highly secured and difficult mining. The hash rate as observed in Figure 4.2, shows that the SHA-512 hashes, are faster to compute compared to SHA-256 hashes, and provide better security with a slight increase in the size of the block; they add an extra overhead of 256 bytes per block.

4.5 Mining of blocks using a single thread

For testing the system, the time in milliseconds (ms), it took the system to homomorphically add, and different numbers of transactions were measured. Considering only a single thread was to be used, the total number of parties was selected to be 3 and the number of transactions was increased. 50 to 1000 transactions in multiple of 50 were repeated for different difficulty levels, ranging from 1 to 4 and only the SHA-256 hashing algorithm was considered for this case. Each of the tasks was repeated 20 times (because the mining time can vary depending upon the block) and the average time in ms was measured. This evaluation was run in a 2 GHz Quad-Core Intel Core I5 processor, with 16 GB of memory. The data obtained is as shown in Figure 4.3. Despite the varying number of transactions, ultimately, the resulting block only stores the tallied results, thus the mining time of the block also remains similar, and thus curve remains almost flat for a particular difficulty level.

The raw data can be found in APPENDIX D and the visualization can be found

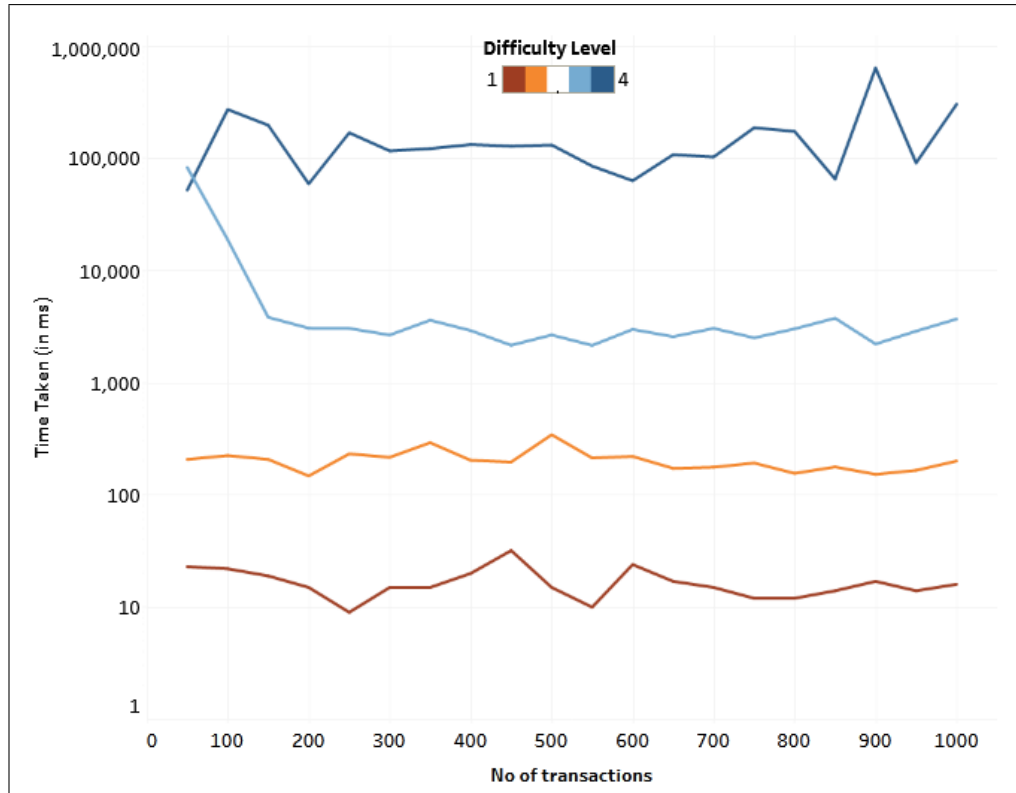


Figure 4.3: System’s Performance when using a single thread

in [26].

4.6 Mining of blocks using multiple threads

Another experiment was conducted where the mining of blocks containing randomized ballots of 100 parties. The aim was to note the time of mining while using multithreading. For this experiment, the seed length of paillier cryptographic system was chosen to be 512. Like in section 4.5, difficulty levels were only chosen from 1 to 4. In this experiment, threads from 1 to 20 were chosen to mine and the time taken to generate a golden hash was noted for both the SHA-256 and SHA-512 hashing algorithms. Also, an average of 20 different test values are selected for each individual data point.

Here, two different approaches were taken for the purpose of mining.

In the first approach, the nonce was divided into blocks, where each thread would perform mining on a specific block only. To simplify, say two threads “T1” and “T2” are taken to mine for a block, which has nonce ranging from 1 to 10. Here,

“T1” will only mine using nonce from 1 to 5 and “T2” will mine from 6 to 10 for the golden hash. And once a thread finds out a golden hash, it would send a signal to all the other threads to stop. This type of mining has been referred to as “block wise” mining in this report.

Another approach was chosen such that the threads would perform mining on continuous values of the nonce. To simplify, say two threads “T1” and “T2” are taken to mine for a block, which has nonce ranging from 1 to 10. Here, “T1” will mine nonce from 1 and lock it, if locked, “T2” will try to lock 2 and mine on it, next after a while, “T1” would try to mine with a value of 2 as a nonce, but since it would be locked, it would try to mine with the nonce value of 3 after acquiring a lock on it. This process would be repeated until the golden hash would have been reached and the thread which would have mined the golden hash would notify all other threads to stop. This type of mining has been referred to as “continuous” mining in this report.

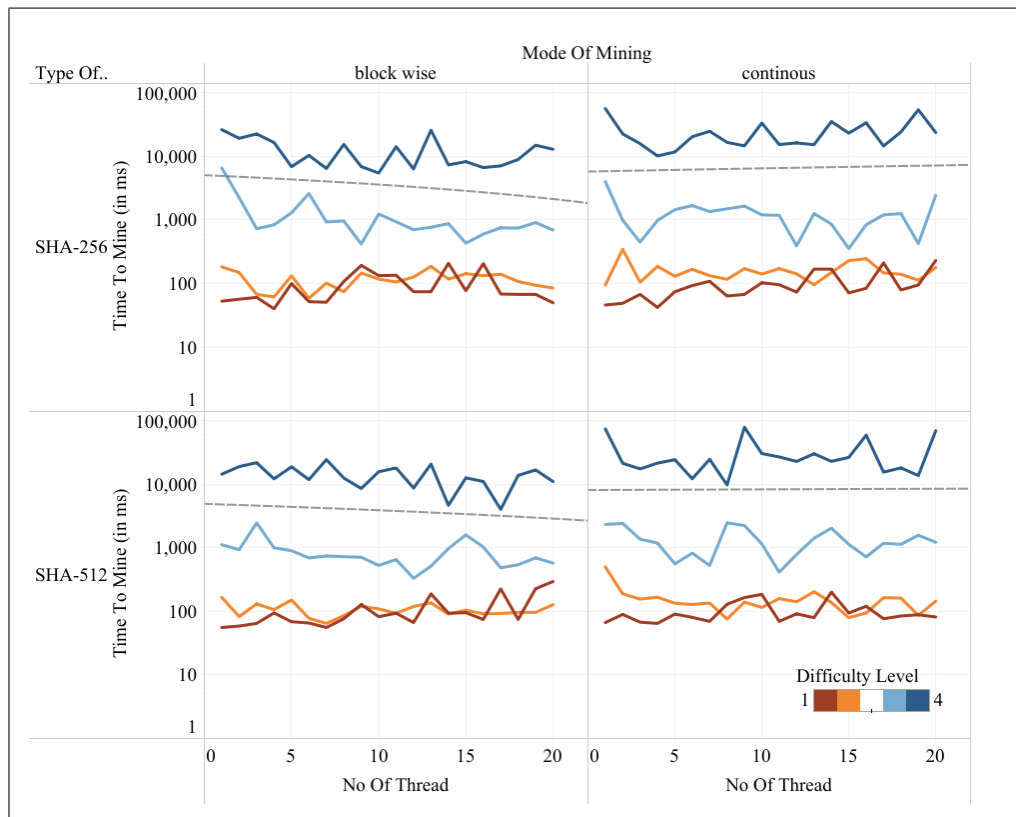


Figure 4.4: Performance of mining when using multiple threads

From Figure 4.4, it can be seen that for both the SHA-256 and SHA-512 hashing,

the block-wise mining on average performs better than the continuous mining approach; since the latter mining approach involved the continuous locking and releasing of nonce resources. Also, needless to say, the multithreaded approach far outperformed the single-threaded approach even for generating hashes for a block with 100 parties, compared to the 5 parties for the single-threaded approach. The raw data can be found in APPENDIX E.

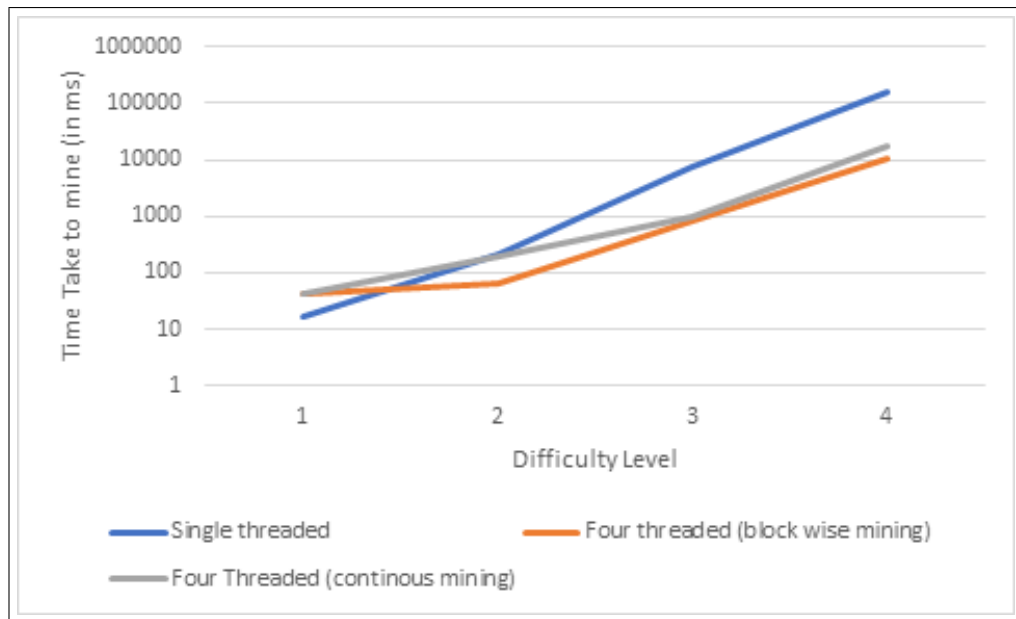


Figure 4.5: Mining performance using different approaches

4.7 Time taken to generate hash for different number of parties in the ballot

The time taken to generate a single hash for different ballot sizes; the number of parties; is as in Figure 4.6.

As seen from the results, as the number of parties increases, for up to 200 parties, the hash generation time remains constant, which will not affect the hash rate and thus the mining time of the system would remain almost similar for a ballot size of up to 200.

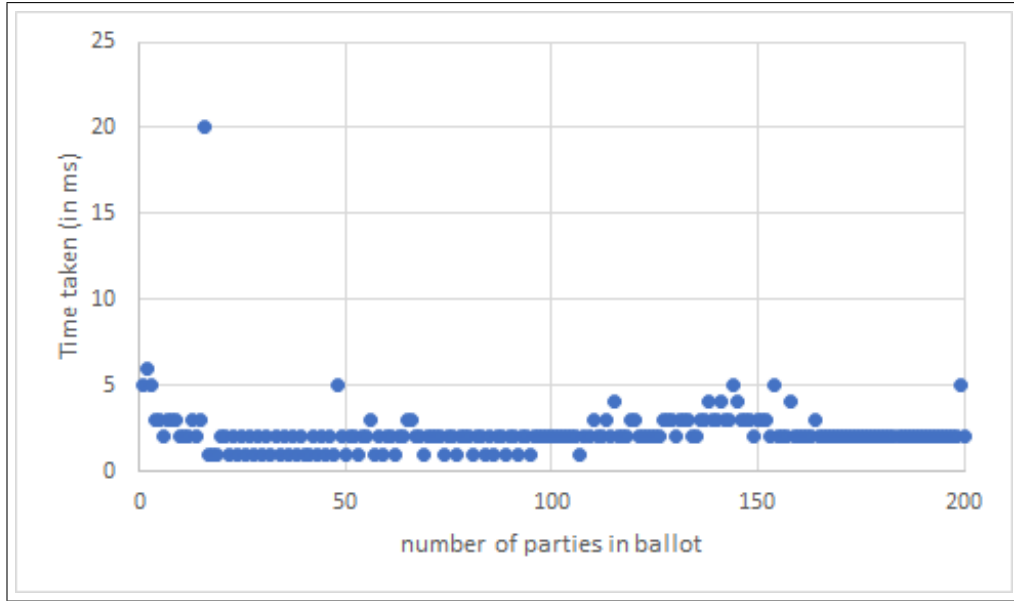


Figure 4.6: Time taken to generate a single hash for different ballot sizes

4.8 Time taken at each phase of the system

For observing the time taken by each phase of the election, the system was configured to allocate 5 parties; the ballot size of 5; seed length of paillier cryptography was set to 512, the difficulty of mining was set to 1, and the maximum transaction limit was set to 50, election duration of 2 minutes, and a random delay ranging from 1 to 10 ms was introduced at the organizer’s node before sending the data to the voter’s end, to simulate a delayed system. The observations are as shown in Table 4.3, for the different number of nodes.

Table 4.3: Time of each phase of the election

number of voter nodes	initialization phase (in ms)	voting phase (in ms)	ending phase (in ms)
10	1356	121740	31
100	1525	120568	36
1000	1524	120816	6567
10000	1530	120182	13685

From the previous table, it can be observed that choosing a difficulty level of 1 and the voting phase was kept under 2 minutes, which was well under the mining capacity of 200 blocks thus, the voting phase was conducted under 2 minutes. The initialization phase took around 1.5 seconds, and the ending phase, where the result was to be broadcasted to different voter nodes by the miners, increased along with the increase in the number of nodes.

4.9 The calculation for the size of the block and block chain

This section proposes a formula for the size of the block when and the block chain when stored as a serialized object.

4.9.1 Size of the paillier enciphered text

For any seed length for paillier cryptographic system and for any value of an input, the length of the enciphered text is 617, which is stored in java's BigInteger data type. The space occupied by BigInteger is given by the formula: $4N + 64$ bytes; where N is the length of the BigInteger and 64 bytes are occupied by the object overhead of the BigInteger object.

By replacing N , with 617, we get 2532 bytes.

4.9.2 Size of parties

The parties are represented as an enum in the proposed system. The size of each enum is 4 bytes. So, the formula can be calculated as: $4 * N_p$ bytes; where N_p is the number of parties to whom the users can vote to.

4.9.3 Size of the map of votes

The map of the votes contains the consists of enum of parties as the key and the corresponding enciphered text of the vote as values. In the proposed system, the block only stores the tallied result for each party and not every ballot object. So, the for a N_p number of parties, the size of the object can be calculated as:

$(4 * Np + Np * 2532)$ bytes;

which becomes:

$N_p * 2536$ bytes; where N_p is the number of the parties.

4.9.4 Size of the list of transactions

The transaction object after the formation of a block only contains the ids of voters who have voted for a given transaction, where the ids are of type integer; which are of 4 bytes. So, the size of the list of transactions can be calculated as:

$4 * N_T$ bytes; where N_T is the number of transactions, or in other words, the number of votes cast per transaction.

4.9.5 Size of a genesis block

The genesis block of the proposed system contains an id of type integer, nonce of type integer, timestamp of time long, hash of type string, previous hash of type string, an empty list of transaction objects, and an empty map of total votes; linking parties to their corresponding votes.

The integer types are of size 4 bytes.

The long types are of size 8 bytes.

The hash size being of string type can be calculated as:

$2 * L_H$ bytes; where L_H is the length of the hash, 64 for SHA-256 hash, and 128 for SHA-512 hash.

Thus, the size of the genesis block proposed by the system is:

$(4 + 4 + 8 + 2 * 2 * L_H)$ bytes,

Which is:

272 bytes; while using SHA-256 hashing and 528 bytes; while using SHA-512 hashing.

4.9.6 Size of a normal block

Like the genesis block, a normal block contains the id of type integer, nonce of type integer, timestamp of time long, a hash of type string, the previous hash of type string, a non-empty list of transaction objects, and a non-empty map of total votes; linking parties to their corresponding votes. The size of the block can be

calculated as:

$(272 + N_p * 2536 + 4 * N_T)$ bytes; where N_T is the number of transactions, N_p is the number of the parties; while using SHA-256 hashing and

$(528 + N_p * 2536 + 4 * N_T)$ bytes; where N_T is the number of transactions, N_p is the number of the parties; while using SHA-512 hashing.

4.9.7 Size of the block chain

A block chain as proposed by the system would consist of a genesis block linked with other normal blocks. The total number of such normal blocks can be calculated as:

Total no of normal blocks = $\frac{\text{Totalnoofvoters}(T_v)}{N_T}$; where N_T is the number of transactions.

Total size of the block chain = Size of genesis block + Total no of normal blocks * Size of normal blocks

Total size of the block chain (in bytes while using SHA-256 hashing) = $272 + T_v/N_T * (272 + N_p * 2536 + 4 * N_T)$

Total size of the block chain (in bytes while using SHA-512 hashing) = $528 + T_v/N_T * (528 + N_p * 2536 + 4 * N_T)$

Assuming a case for 18 million voters in the context of Nepal, by applying the above formula, and taking 1000 votes per transaction, the total size of the block chain becomes:

The total size of the block chain (in bytes while using SHA-256 hashing) = 4.64 GB, and

The total size of the block chain (in bytes while using SHA-512 hashing) = 4.65 GB.

With a total of 18000 blocks. To manage the number of blocks, N_T can be adjusted accordingly. Also, the memory capacity of the machine should be at least three times the size of the block chain, this is taken into account excluding the mining requirements, as mining can and should be run on either a GPU or ASIC.

4.10 Graphics Processing Unit (GPU) and Application-specific integrated circuit for mining (ASIC)

Central Process Unit (CPU), the handful number of physical cores can only generate a few hundred hashes per second, which is only suitable for performing proof of work up to a difficulty level of 4. As the difficulty level for the proof of work increases; for providing security, the few hundred hashes per second is simply not enough to generate a valid block within a time frame of a few minutes (for bitcoin the average block generation speed is 10 minutes, to maintain this almost constant block generation time, they adjust the difficulty level every 2 weeks).

GPUs can generate hundreds of thousands, if not millions of hashes per second, which would also be not enough for higher difficulty levels. For this, a special type of Integrated Circuit (IC) chip is used; this IC is optimized for a particular use case. There are some ASICs available, which are designed especially for the purpose of mining and can output up to 200 Tera hash per second which can only consume 27.5 joules of energy per tera-hash [27]. For the purpose of small-scale elections, a simple GPU would be sufficient, but for large-scale elections like the national election, a specially designed ASIC can be used.

4.11 The potential threat to the system with the emerging Quantum Technology

Quantum computers are the modern emerging computers that can solve complex mathematical problems, which require a lot of time for the current computers; also called the classical computers; to achieve by utilizing the principles of quantum mechanisms. There is currently a handful of quantum computers in the world.

As per [28], quantum computers can exploit two portions of the block chain based cryptocurrency: the ECDSA algorithm and the proof of work consensus protocol. About 31710^6 physical qubits will be required to break the ECDSA encryption algorithm within one hour with a code cycle time of 1 s and 13×10^6 physical qubits will be required to break it within 1 day. The best quantum computer as of November 2021 is IBM's eagle quantum computer with a 127 qubit processor. The

current factor preventing the creation of computers with more qubit processors is the “quantum noise” phenomenon.

As per [29], without considering the advancement of ASIC chips, quantum computers would unlikely be able to outperform ASIC chips for proof of work up until 2028. That too is considered without taking into account the advancement of ASIC chips which is expected to push the date even further. To put it into perspective, a quantum computer with 13×10^6 qubits would be required to break a hash within a day.

So the block chain technology is estimated to last at least around a decade more until quantum computers would supposedly make this technology unreliable.

4.12 Output

The screenshots of the system can be found in APPENDIX A. Also, a typical mined block of difficulty level 5 with ballot object of 100 parties when represented as a text can be found in APPENDIX F.

CHAPTER 5
THESIS TIMELINE

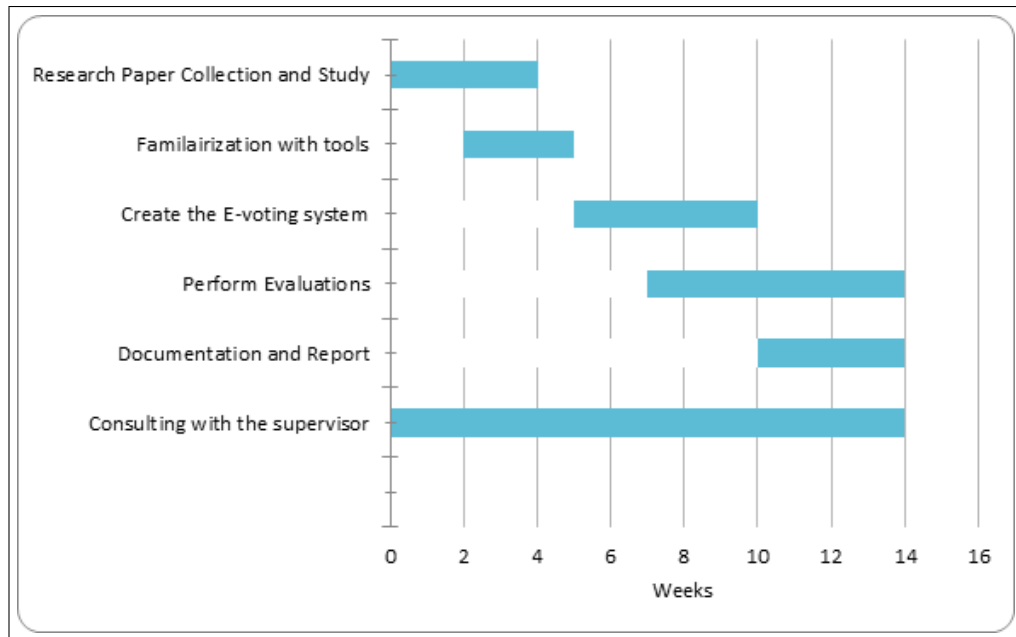


Figure 5.1: Thesis Timeline

CHAPTER 6

CONCLUSION AND RECOMMENDATION

6.1 Conclusion

Thus, a block chain based e-voting system was designed that preserves the anonymity of the voters' vote by using paillier cryptography, which enabled the system to tally the votes of the voters without the need to decrypt the votes for tallying. Smart contracts dictated the entire election process like setting the duration of the entire election, the difficulty level of proof of work, the limit of the transactions per block, and when and whom to provide the paillier cryptography keys.

6.2 Limitation

Although the system proposed in this thesis work is secure and uses multiple threads for the mining process, the thesis work does not experiment with the mining process using GPU.

6.3 Recommendation

This thesis work proposes a voting system using paillier cryptography for anonymity and proof of work as a consensus mechanism. Another more reliable zero-knowledge proof system can be used in another research along with a different consensus mechanism.

REFERENCES

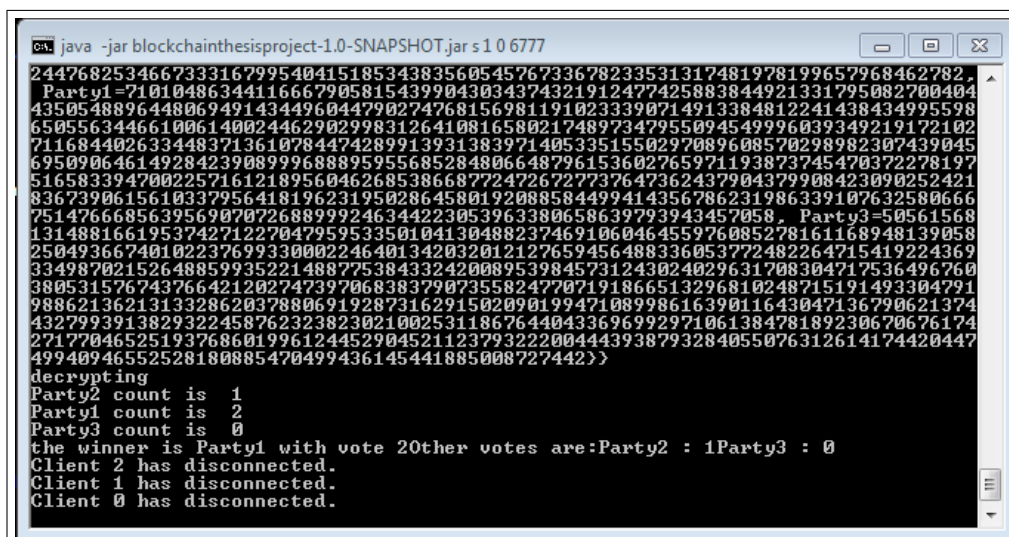
- [1] Dr Kashmar and Awn Jasim. *An Enhancement of Blockchain Technology within Health Care*. PhD thesis, 05 2022.
- [2] Chen Zhang. *The applications of Blockchain in food supply chain management*. PhD thesis, 03 2022.
- [3] Md. Rakibul Hassan Robin. *Product Authentication Using Blockchain*. PhD thesis, 07 2021.
- [4] Cedric Strub. *CONTRIBUTION OF BLOCKCHAIN TO HEALTH DATA MANAGEMENT*. PhD thesis, 04 2021.
- [5] Irene Gelyk. *Applicability of Blockchain for long-term digital preservation of the Canadian nuclear waste management deep geological repository information assets: a literature review*. PhD thesis, 05 2022.
- [6] Niru Raj. *How Blockchain transforms the Future of Retail Shopping*. PhD thesis, 10 2020.
- [7] Saipavan Vallabhaneni. *Leveraging Blockchain for Plasma Fractionation Supply Chains*. PhD thesis, 04 2020.
- [8] Fatemeh Borran and André Schiper. A leader-free byzantine consensus algorithm. pages 67–78, 01 2010.
- [9] Arthur Gervais, Ghassan Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. pages 3–16, 10 2016.
- [10] Kejiao Li, Li Hui, Hanxu Hou, Kedan Li, and Yongle Chen. Proof of vote: A high-performance consensus protocol based on vote mechanism consortium blockchain. pages 466–473, 12 2017.

- [11] Natalia Chaudhry and Muhammad Yousaf. Consensus algorithms in blockchain: Comparative analysis, challenges and opportunities. pages 54–63, 12 2018.
- [12] Seval Çapraz and Adnan Ozsoy. *Personal Data Protection in Blockchain with Zero-Knowledge Proof*, pages 109–124. 03 2021.
- [13] Uzma Jafar, Mohd Aziz, and Zarina Shukur. Blockchain for electronic voting system—review and open research challenges. *Sensors*, 21:5874, 08 2021.
- [14] Saad Khan, Aansa Arshad, Gazala Mushtaq, Aqeel Khalique, and Tarek Husein. Implementation of decentralized blockchain e-voting. *EAI Endorsed Transactions on Smart Cities*, 4:164859, 07 2018.
- [15] Subashka Ramesh. E-voting based on block chain technology. 02 2022.
- [16] Ahmed Ben Ayed. A conceptual secure blockchain-based electronic voting system. 05 2017.
- [17] Honglei Li and Weilian Xue. A blockchain-based sealed-bid e-auction scheme with smart contract and zero-knowledge proof. *Security and Communication Networks*, 2021:1–10, 05 2021.
- [18] Bitcoin: A peer-to-peer electronic cash system. <https://bitcoin.org/bitcoin.pdf>. Accessed: 06 07 2022.
- [19] The cryptographic hash function sha-256. <https://helix.stormhub.org/papers/SHA-256.pdf>. Accessed: 06 07 2022.
- [20] Shay Gueron, Simon Johnson, and Jesse Walker. Sha-512/256. In *Proceedings of the 2011 Eighth International Conference on Information Technology: New Generations*, ITNG '11, page 354–358, USA, 2011. IEEE Computer Society.
- [21] Solidity. <https://docs.soliditylang.org/en/v0.8.15>. Accessed: 06 07 2022.
- [22] Java — oracle. <https://www.oracle.com/java>. Accessed: 06 07 2022.
- [23] Elliptic curve digital signature algorithm. <https://www.encryptionconsulting.com/education-center/what-is-ecdsa>. Accessed: 06 07 2022.

- [24] Public-key cryptosystems based on composite.
https://link.springer.com/content/pdf/10.1007%2F3-540-48910-X_16.pdf.
Accessed: 06 07 2022.
- [25] Recommended threading strategies and os platform.
https://docs.oracle.com/cd/E29584_01/webhelp/PerfTuning/platform.html.
Accessed: 06 07 2022.
- [26] Masters thesis viz,” tableau public. <https://public.tableau.com/app/profile/juned.alam7696>. Accessed: 06 07 2022.
- [27] Bitmain. <https://shop.bitmain.com/product/detail?pid=00020220105112318868myo6YbOL06D3>. Accessed: 06 07 2022.
- [28] Mark Webber, Vincent Elfving, Sebastian Weidt, and Winfried K. Hensinger. The impact of hardware specifications on reaching quantum advantage in the fault tolerant regime. *AVS Quantum Science*, 4(1):013801, 2022.
- [29] Joseph J. Kearney and Carlos A. Perez-Delgado. Vulnerability of blockchain technologies to quantum attacks. *Array*, 10:100065, 2021.

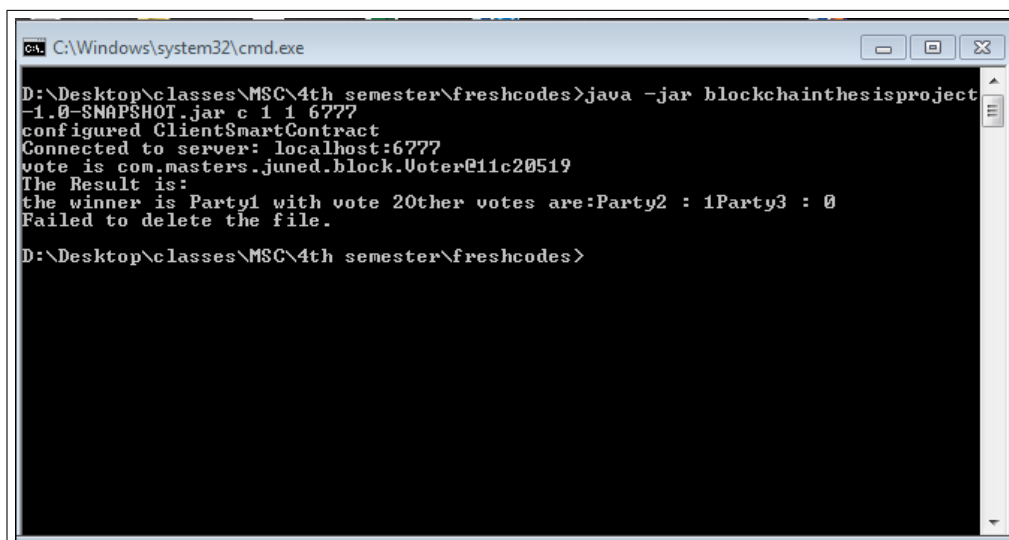
APPENDIX A

The screenshot of the output of the system has been attached below:



```
ca: java -jar blockchainthisproject-1.0-SNAPSHOT.jar s 1 0 6777
2447682534667333167995404151853438356054576733678233531317481978199657968462782.
Party1 =710104863441166679058154399043034374321912477425883844921331795082700404
43505488964480694914344960447902747681569811910233390714913384812241438434995598
65055634466100614002446290299831264108165802174897347955094549996039349219172102
71168440263344837136107844742899139313839714053351550297089608570298982307439045
69509064614928423908999688895955685284806648796153602765971193873745470372278197
51658339470022571612189560462685386687724726727737647362437904379908423090252421
83673906156103379564181962319502864580192088584499414356786231986339107632580666
751476668563956907072688999246344223053963380658639793943457058. Party3 =50561568
13148816619537427122704795953350104130488237469106046455976085278161168948139058
25049366740102237699330002246401342032012127659456488336053772482264715419224369
33498702152648859935221488775384332420089539845731243024029631708304717536496760
38053157674376642120274739706838379073558247707191866513296810248715191493304791
98862136213133286203788069192873162915020901994710899861639011643047136790621374
43279939138293224587623238230210025311867644043369699297106138478189230670676174
27177046525193768601996124452904521123793222004443938793284055076312614174420447
49940946552528180885470499436145441885008727442}>
decrypting
Party2 count is 1
Party1 count is 2
Party3 count is 0
the winner is Party1 with vote 20other votes are:Party2 : 1Party3 : 0
Client 2 has disconnected.
Client 1 has disconnected.
Client 0 has disconnected.
```

Figure 6.1: Miner node



```
ca: C:\Windows\system32\cmd.exe
D:\Desktop\classes\MSC\4th semester\freshcodes>java -jar blockchainthisproject
-1.0-SNAPSHOT.jar c i 1 6777
configured ClientSmartContract
Connected to server: localhost:6777
vote is com.masters.juned.block.Voter@11c20519
The Result is:
the winner is Party1 with vote 20other votes are:Party2 : 1Party3 : 0
Failed to delete the file.
D:\Desktop\classes\MSC\4th semester\freshcodes>
```

Figure 6.2: Voter node

APPENDIX B

Table 6.1: Hash generated per second of SHA-256 and SHA-512 while using given number of threads

No of threads	SHA-256 hashes generated per second	SHA-512 hashes generated per second
1	811	885
2	1003	1021
3	1253	1378
4	1821	1651
5	1863	1877
6	2131	2616
7	2369	3610
8	2752	3495
9	2712	4318
10	2706	4469
11	3113	4350
12	3076	4376
13	3284	4442
14	3297	4354
15	3734	4728
16	3844	4466
17	4384	4525
18	4684	4583
19	4363	4919
20	4368	4684
21	4372	4809
22	4229	4699
23	4438	4768
24	4346	4777
25	4359	4575
26	4365	4470
27	4237	4612
28	4210	4606
29	4224	4227
30	4203	4533
31	4242	4524
32	4217	4412
33	4164	4442
34	4233	4264
35	3809	4421
36	4226	4427
37	4233	4247
38	4130	4390
39	4257	4400
40	4153	4411
41	4179	4440
42	4046	4367
43	4155	4216
44	4056	4311

Table 6.2: Hash generated per second of SHA-256 and SHA-512 while using given number of threads, continued

No of threads	SHA-256 hashes generated per second	SHA-512 hashes generated per second
45	4196	4239
46	4122	4213
47	4048	4095
48	4064	4282
49	4009	4288
50	4062	4207
51	4112	4242
52	4186	4207
53	4095	4236
54	4068	4111
55	3970	4237
56	3944	4174
57	4080	4206
58	4050	4176
59	3727	4254
60	4115	4338
61	4009	4239
62	4031	4234
63	4002	4285
64	4030	4287
65	3813	4125
66	4040	4194
67	3991	4085
68	3971	4297
69	3792	4295
70	3843	4199
71	3826	4224
72	3907	4230
73	3898	4112
74	3747	4160
75	3747	4136
76	3927	4155
77	3815	4079
78	3859	4143
79	3877	4297
80	3873	4188
81	3808	4105
82	3972	4224
83	3896	4019
84	3907	4175
85	3831	3946
86	3786	4075
87	4007	3957
88	3918	4008
89	3974	3996
90	3899	4113
91	3916	3532
92	3949	4069
93	3929	3975
94	3893	4050
95	3988	4140
96	4022	4164
97	3893	4120
98	3736	4237
99	3449	4029
100	3365	4355

APPENDIX C

An example of Paillier cryptography with small parameters is as follows.

Let us choose small prime numbers as $P=7$, $q=11$

Then, $n = p \cdot q = 7 \cdot 11 = 77$

Now, let's select an integer g from $Z_{n^2}^*$, such that the order of g is a multiple of n in $Z_{n^2}^*$. If we randomly choose the integer $g = 5652$

then all necessary conditions are met as the order of g is $2310 = 30 \cdot 77$ in $Z_{n^2}^*$.

The public key for the example will be as: $(n, g) = (77, 5652)$

To encrypt a message, let's say 42 such that $m = 42$

where m belongs to Z_n , let us choose a random say 23 such that $r = 23$

where r is a nonzero integer and r belongs to Z_n .

Let us compute the cipher text c as: $c = g^m \cdot r^n \pmod{n^2} = 5652^{42} \cdot 23^{77} \pmod{5929} = 4624 \pmod{5929}$

To decrypt the ciphertext c , let us compute λ as $\lambda = \text{lcm}(6, 10) = 30$

Let us define $L(x) = (x-1)/n$, compute

$$\begin{aligned} u &= (L(g^\lambda \pmod{n^2}))^{-1} \pmod{n} \\ &= L(5652^{30} \pmod{5929}) \\ &= L(3928) \\ &= (3928 - 1)/77 \\ &= 3927/77 \\ &= 51 \end{aligned}$$

Let us compute the inverse of k as

$$\begin{aligned}u &= k^{-1}(\text{mod } n) \\ &= 51^{-1} \\ &= 74(\text{mod } 77)\end{aligned}$$

Thus,

$$\begin{aligned}m &= L(c^\lambda \text{mod } n^2) * u \text{mod } n \\ &= L(462430(\text{mod } 5959)) * 74(\text{mod } 77) \\ &= L(4852) * 74(\text{mod } 77) \\ &= 42\end{aligned}$$

APPENDIX D

Table 6.3: System Performance Metrics Data

Difficulty Level	Time Taken (in ms)	No of transactions
1	23	50
1	22	100
1	19	150
1	15	200
1	9	250
1	15	300
1	15	350
1	20	400
1	32	450
1	15	500
1	10	550
1	24	600
1	17	650
1	15	700
1	12	750
1	12	800
1	14	850
1	17	900
1	14	950
1	16	1000
2	208	50
2	225	100
2	208	150
2	148	200
2	233	250
2	217	300

Table 6.4: System Performance Metrics Data, continued

Difficulty Level	Time Taken (in ms)	No of transactions
2	293	350
2	204	400
2	197	450
2	345	500
2	214	550
2	221	600
2	173	650
2	177	700
2	193	750
2	156	800
2	178	850
2	153	900
2	166	950
2	201	1000
3	83170	50
3	18880	100
3	3846	150
3	3076	200
3	3061	250
3	2673	300
3	3623	350
3	2929	400
3	2166	450
3	2677	500
3	2160	550
3	3001	600
3	2583	650
3	3065	700
3	2522	750
3	3035	800
3	3777	850
3	2221	900
3	2894	950
3	3711	1000

Table 6.5: System Performance Metrics Data, continued

Difficulty Level	Time Taken (in ms)	No of transactions
4	52440	50
4	274111	100
4	198540	150
4	59668	200
4	169964	250
4	117144	300
4	122808	350
4	133503	400
4	129307	450
4	132295	500
4	85739	550
4	63507	600
4	108375	650
4	103461	700
4	188275	750
4	174948	800
4	65728	850
4	642173	900
4	91449	950
4	305006	1000

APPENDIX E

Table 6.6: System Performance Metrics Data with multiple thread

difficulty level	no of thread	average time to mine (in ms)	type of hash	mode of mining
1	1	67	SHA-512	continous
1	2	90	SHA-512	continous
1	3	68	SHA-512	continous
1	4	65	SHA-512	continous
1	5	91	SHA-512	continous
1	6	81	SHA-512	continous
1	7	70	SHA-512	continous
1	8	130	SHA-512	continous
1	9	166	SHA-512	continous
1	10	186	SHA-512	continous
1	11	70	SHA-512	continous
1	12	92	SHA-512	continous
1	13	80	SHA-512	continous
1	14	202	SHA-512	continous
1	15	95	SHA-512	continous
1	16	121	SHA-512	continous
1	17	77	SHA-512	continous
1	18	85	SHA-512	continous
1	19	89	SHA-512	continous
1	20	82	SHA-512	continous
2	1	506	SHA-512	continous
2	2	190	SHA-512	continous
2	3	157	SHA-512	continous
2	4	168	SHA-512	continous
2	5	135	SHA-512	continous
2	6	129	SHA-512	continous
2	7	136	SHA-512	continous
2	8	76	SHA-512	continous
2	9	141	SHA-512	continous
2	10	116	SHA-512	continous
2	11	160	SHA-512	continous
2	12	142	SHA-512	continous
2	13	205	SHA-512	continous
2	14	139	SHA-512	continous
2	15	80	SHA-512	continous
2	16	95	SHA-512	continous
2	17	165	SHA-512	continous
2	18	163	SHA-512	continous
2	19	87	SHA-512	continous

Table 6.7: System Performance Metrics Data with multiple thread, continued

difficulty level	no of thread	average time to mine (in ms)	type of hash	mode of mining
2	20	146	SHA-512	continous
3	1	2354	SHA-512	continous
3	2	2441	SHA-512	continous
3	3	1379	SHA-512	continous
3	4	1199	SHA-512	continous
3	5	564	SHA-512	continous
3	6	831	SHA-512	continous
3	7	531	SHA-512	continous
3	8	2493	SHA-512	continous
3	9	2260	SHA-512	continous
3	10	1164	SHA-512	continous
3	11	416	SHA-512	continous
3	12	792	SHA-512	continous
3	13	1427	SHA-512	continous
3	14	2057	SHA-512	continous
3	15	1142	SHA-512	continous
3	16	726	SHA-512	continous
3	17	1188	SHA-512	continous
3	18	1144	SHA-512	continous
3	19	1590	SHA-512	continous
3	20	1234	SHA-512	continous
4	1	75328	SHA-512	continous
4	2	21587	SHA-512	continous
4	3	17659	SHA-512	continous
4	4	21740	SHA-512	continous
4	5	24670	SHA-512	continous
4	6	12326	SHA-512	continous
4	7	24998	SHA-512	continous
4	8	9859	SHA-512	continous
4	9	80075	SHA-512	continous
4	10	30772	SHA-512	continous
4	11	27252	SHA-512	continous
4	12	23135	SHA-512	continous
4	13	30571	SHA-512	continous
4	14	23213	SHA-512	continous
4	15	26724	SHA-512	continous
4	16	60063	SHA-512	continous
4	17	15694	SHA-512	continous
4	18	18286	SHA-512	continous

Table 6.8: System Performance Metrics Data with multiple thread, continued

difficulty level	no of thread	average time to mine (in ms)	type of hash	mode of mining
4	19	13837	SHA-512	continous
4	20	70547	SHA-512	continous
1	1	56	SHA-512	block wise
1	2	59	SHA-512	block wise
1	3	65	SHA-512	block wise
1	4	95	SHA-512	block wise
1	5	69	SHA-512	block wise
1	6	66	SHA-512	block wise
1	7	56	SHA-512	block wise
1	8	77	SHA-512	block wise
1	9	129	SHA-512	block wise
1	10	83	SHA-512	block wise
1	11	94	SHA-512	block wise
1	12	67	SHA-512	block wise
1	13	190	SHA-512	block wise
1	14	94	SHA-512	block wise
1	15	96	SHA-512	block wise
1	16	75	SHA-512	block wise
1	17	227	SHA-512	block wise
1	18	75	SHA-512	block wise
1	19	229	SHA-512	block wise
1	20	297	SHA-512	block wise
2	1	167	SHA-512	block wise
2	2	83	SHA-512	block wise
2	3	132	SHA-512	block wise
2	4	107	SHA-512	block wise
2	5	152	SHA-512	block wise
2	6	78	SHA-512	block wise
2	7	65	SHA-512	block wise
2	8	86	SHA-512	block wise
2	9	121	SHA-512	block wise
2	10	110	SHA-512	block wise
2	11	94	SHA-512	block wise
2	12	120	SHA-512	block wise
2	13	137	SHA-512	block wise
2	14	91	SHA-512	block wise
2	15	105	SHA-512	block wise
2	16	92	SHA-512	block wise
2	17	93	SHA-512	block wise

Table 6.9: System Performance Metrics Data with multiple thread, continued

difficulty level	no of thread	average time to mine (in ms)	type of hash	mode of mining
2	18	97	SHA-512	block wise
2	19	97	SHA-512	block wise
2	20	128	SHA-512	block wise
3	1	1128	SHA-512	block wise
3	2	942	SHA-512	block wise
3	3	2488	SHA-512	block wise
3	4	1008	SHA-512	block wise
3	5	909	SHA-512	block wise
3	6	698	SHA-512	block wise
3	7	750	SHA-512	block wise
3	8	731	SHA-512	block wise
3	9	716	SHA-512	block wise
3	10	532	SHA-512	block wise
3	11	658	SHA-512	block wise
3	12	334	SHA-512	block wise
3	13	517	SHA-512	block wise
3	14	982	SHA-512	block wise
3	15	1611	SHA-512	block wise
3	16	1038	SHA-512	block wise
3	17	488	SHA-512	block wise
3	18	545	SHA-512	block wise
3	19	701	SHA-512	block wise
3	20	578	SHA-512	block wise
4	1	14567	SHA-512	block wise
4	2	19177	SHA-512	block wise
4	3	22088	SHA-512	block wise
4	4	12288	SHA-512	block wise
4	5	19059	SHA-512	block wise
4	6	11943	SHA-512	block wise
4	7	24710	SHA-512	block wise
4	8	12669	SHA-512	block wise
4	9	8660	SHA-512	block wise
4	10	15905	SHA-512	block wise
4	11	18257	SHA-512	block wise
4	12	8795	SHA-512	block wise
4	13	21013	SHA-512	block wise
4	14	4709	SHA-512	block wise
4	15	12744	SHA-512	block wise
4	16	11177	SHA-512	block wise

Table 6.10: System Performance Metrics Data with multiple thread, continued

difficulty level	no of thread	average time to mine (in ms)	type of hash	mode of mining
4	17	4059	SHA-512	block wise
4	18	13942	SHA-512	block wise
4	19	16999	SHA-512	block wise
4	20	11152	SHA-512	block wise
1	1	47	SHA-256	continous
1	2	50	SHA-256	continous
1	3	69	SHA-256	continous
1	4	43	SHA-256	continous
1	5	76	SHA-256	continous
1	6	95	SHA-256	continous
1	7	111	SHA-256	continous
1	8	65	SHA-256	continous
1	9	69	SHA-256	continous
1	10	105	SHA-256	continous
1	11	98	SHA-256	continous
1	12	75	SHA-256	continous
1	13	172	SHA-256	continous
1	14	172	SHA-256	continous
1	15	73	SHA-256	continous
1	16	86	SHA-256	continous
1	17	216	SHA-256	continous
1	18	81	SHA-256	continous
1	19	97	SHA-256	continous
1	20	235	SHA-256	continous
2	1	97	SHA-256	continous
2	2	353	SHA-256	continous
2	3	107	SHA-256	continous
2	4	190	SHA-256	continous
2	5	133	SHA-256	continous
2	6	171	SHA-256	continous
2	7	136	SHA-256	continous
2	8	119	SHA-256	continous
2	9	175	SHA-256	continous
2	10	144	SHA-256	continous
2	11	175	SHA-256	continous
2	12	145	SHA-256	continous
2	13	98	SHA-256	continous
2	14	152	SHA-256	continous
2	15	234	SHA-256	continous

Table 6.11: System Performance Metrics Data with multiple thread, continued

difficulty level	no of thread	average time to mine (in ms)	type of hash	mode of mining
2	16	251	SHA-256	continous
2	17	150	SHA-256	continous
2	18	143	SHA-256	continous
2	19	115	SHA-256	continous
2	20	182	SHA-256	continous
3	1	4062	SHA-256	continous
3	2	1010	SHA-256	continous
3	3	457	SHA-256	continous
3	4	1000	SHA-256	continous
3	5	1478	SHA-256	continous
3	6	1708	SHA-256	continous
3	7	1379	SHA-256	continous
3	8	1522	SHA-256	continous
3	9	1679	SHA-256	continous
3	10	1223	SHA-256	continous
3	11	1204	SHA-256	continous
3	12	399	SHA-256	continous
3	13	1284	SHA-256	continous
3	14	875	SHA-256	continous
3	15	362	SHA-256	continous
3	16	857	SHA-256	continous
3	17	1225	SHA-256	continous
3	18	1282	SHA-256	continous
3	19	433	SHA-256	continous
3	20	2472	SHA-256	continous
4	1	57142	SHA-256	continous
4	2	22871	SHA-256	continous
4	3	16074	SHA-256	continous
4	4	10304	SHA-256	continous
4	5	11889	SHA-256	continous
4	6	20666	SHA-256	continous
4	7	25144	SHA-256	continous
4	8	16834	SHA-256	continous
4	9	14837	SHA-256	continous
4	10	33827	SHA-256	continous
4	11	15560	SHA-256	continous
4	12	16507	SHA-256	continous
4	13	15475	SHA-256	continous
4	14	35691	SHA-256	continous

Table 6.12: System Performance Metrics Data with multiple thread, continued

difficulty level	no of thread	average time to mine (in ms)	type of hash	mode of mining
4	15	23509	SHA-256	continous
4	16	34236	SHA-256	continous
4	17	14753	SHA-256	continous
4	18	24560	SHA-256	continous
4	19	54737	SHA-256	continous
4	20	23991	SHA-256	continous
1	1	54	SHA-256	block wise
1	2	58	SHA-256	block wise
1	3	62	SHA-256	block wise
1	4	41	SHA-256	block wise
1	5	102	SHA-256	block wise
1	6	53	SHA-256	block wise
1	7	52	SHA-256	block wise
1	8	110	SHA-256	block wise
1	9	197	SHA-256	block wise
1	10	136	SHA-256	block wise
1	11	138	SHA-256	block wise
1	12	76	SHA-256	block wise
1	13	76	SHA-256	block wise
1	14	212	SHA-256	block wise
1	15	79	SHA-256	block wise
1	16	210	SHA-256	block wise
1	17	70	SHA-256	block wise
1	18	69	SHA-256	block wise
1	19	69	SHA-256	block wise
1	20	51	SHA-256	block wise
2	1	187	SHA-256	block wise
2	2	152	SHA-256	block wise
2	3	69	SHA-256	block wise
2	4	63	SHA-256	block wise
2	5	135	SHA-256	block wise
2	6	60	SHA-256	block wise
2	7	104	SHA-256	block wise
2	8	76	SHA-256	block wise
2	9	149	SHA-256	block wise
2	10	120	SHA-256	block wise
2	11	108	SHA-256	block wise
2	12	129	SHA-256	block wise
2	13	190	SHA-256	block wise

Table 6.13: System Performance Metrics Data with multiple thread, continued

difficulty level	no of thread	average time to mine (in ms)	type of hash	mode of mining
2	14	120	SHA-256	block wise
2	15	146	SHA-256	block wise
2	16	136	SHA-256	block wise
2	17	143	SHA-256	block wise
2	18	110	SHA-256	block wise
2	19	96	SHA-256	block wise
2	20	87	SHA-256	block wise
3	1	6662	SHA-256	block wise
3	2	2276	SHA-256	block wise
3	3	742	SHA-256	block wise
3	4	855	SHA-256	block wise
3	5	1318	SHA-256	block wise
3	6	2646	SHA-256	block wise
3	7	947	SHA-256	block wise
3	8	981	SHA-256	block wise
3	9	428	SHA-256	block wise
3	10	1263	SHA-256	block wise
3	11	953	SHA-256	block wise
3	12	715	SHA-256	block wise
3	13	781	SHA-256	block wise
3	14	892	SHA-256	block wise
3	15	440	SHA-256	block wise
3	16	610	SHA-256	block wise
3	17	767	SHA-256	block wise
3	18	759	SHA-256	block wise
3	19	926	SHA-256	block wise
3	20	706	SHA-256	block wise
4	1	26675	SHA-256	block wise
4	2	19473	SHA-256	block wise
4	3	22853	SHA-256	block wise
4	4	16723	SHA-256	block wise
4	5	6990	SHA-256	block wise
4	6	10516	SHA-256	block wise
4	7	6549	SHA-256	block wise
4	8	15630	SHA-256	block wise
4	9	7004	SHA-256	block wise
4	10	5551	SHA-256	block wise
4	11	14420	SHA-256	block wise
4	12	6439	SHA-256	block wise
4	13	26169	SHA-256	block wise
4	14	7463	SHA-256	block wise
4	15	8391	SHA-256	block wise
4	16	6769	SHA-256	block wise
4	17	7186	SHA-256	block wise
4	18	9070	SHA-256	block wise
4	19	15191	SHA-256	block wise
4	20	13114	SHA-256	block wise

APPENDIX F

A typical block in a block chain, with ballot size of 3, when represented as text looks as follows:

```
Block{id=1,          nonce=859055285,          timeStamp=1662907194197,
hash='00000c50da2bd5e3d344857b671b9fccf7e442b03a5516c042beecfab11bb31c',
previousHash='000002615941007553adbf068c1cb93f67ddfecf2787268784fd8f01a054
2115',
transactions=[Transaction{voterIds={1}, finalBallotPaper=
{Party1=          3214101728113432780610492910743573448438523697924095319
296594134 8276678268178738216732201075839057966186864884213291156861147
9537482792  781188961463600007413329802450746441124868302483719285718
71092243394619                                8871487795266331501626342930840023
4925311277174286547088293861077015158                                87819441262576850
552252490078888824038017539283898319100176405669047259
61330271139236488278601583121740463858311873501890584122256110235179089
64746706493437893512851759159013184529521523195423388981075992047082400
97026637803184962403442041024019937765132324516595175639832845326362929
8053042863132607248948034565728505480466317499731366696,
Party2=23971195694803979232665370406886412352220449898865346623044811751
38032182023611105895605755327077804117550939230941199616983985348262935
29398192139934217410299963636483880750359571875672972373012223073781920
59526870890500541727067964929665164319654936407551291739054916307961564
96209712589295847952031573720918571644128222564670451360466961701776188
35299006815542025280405674195667611818888904876898839054480483232876706
45846429163932853535713646242083593194371831221940016339924510974968519
92502532558888033149326939912947799080294621760622075794725872321815422
294197591651503411080969591326336153591422525127987725,
```


Party3=13918690118073971228306173197525044814501970334220720841960073158
68325863658767409125975656673911358485669133367502929949370145570212042
31472349436850733281110057620891062627400189567388368827189847297689889
14885011462142810711480097033330045605102666877946488487706195648155260
34711548154108053021846191868744075986908749725452084906208667877001693
14893541128310605983779592501546214032420406879674629011327563355671374
92226932363141959042258622084863547635541757367095643469980517063110287
67009050015783098050042483143455748973049533607495325314875682613836983
383750394088119072950721978538605632279817907460721141}}}