BLOCKCHAIN BASED BALLOT SYSTEM

Project report submitted

in partial fulfilment of the requirement for the degree of

Bachelor of Technology

By

Durgesh Barwal (1515013)

Hariom Kumar Sinha (1515014)

Mohit Kumar Shrivastava (1515021)

Saumya Raj (1515039)

Ujjawal Sinha (1515057)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

Deemed to be University, BHUBANESWAR

(April 2018)

CERTIFICATE

This is certifying that the work contained in the project report entitled "BLOCKCHAIN BASED BALLOT SYSTEM", submitted by "Saumya Raj (1515039), Durgesh Barwal (1515013), Hariom Kumar Sinha (1515014), Mohit Shrivastava (1515021) and Ujjawal Sinha (1515057)" is a record of bonafide work carried out by them, in the partial fulfilment of the requirement. This work has been carried out under my supervision during year 2017-2018, under my guidance.

Date:

Mr. Rajat Kumar Behera

(Project Guide)

Associate Professor

School of Computer Engineering

KIIT University

DECLARATION

We declare that this written submission represents our ideas in our own words and where other's ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date:

Saumya Raj

Durgesh Barwal

Hariom Kumar Sinha

Ujjawal Sinha

Mohit Kumar Shivastava

ACKNOWLEDGEMENT

Apart from our efforts, the success of this project depends largely on the encouragement and guidance of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take immense pleasure in thanking and warmly acknowledging the continuous encouragement, invaluable supervision, timely suggestions and inspired guidance offered by our project mentor, Mr. Rajat Kumar Behera, Associate Professor, School of Computer Engineering, KIIT University, in bringing this report to a successful completion. We also express our sincere thanks to all our friends who have patiently extended all sorts of help for accomplishing this undertaking. Also, we would like to express our heartfelt thanks to each of our beloved parents for their blessings, for their help and wishes for the successful completion of this project. Finally, we extend our gratefulness to one and all that are directly or indirectly involved in the successful completion of this project work.

Durgesh Barwal (1515013)

Hariom Kumar Sinha (1515014)

Mohit Kumar Shrivastava (1515021)

Saumya Raj (1515039)

Ujjawal Sinha (1515057)

CONTENT

LIST OF FIGURES AND TABLE

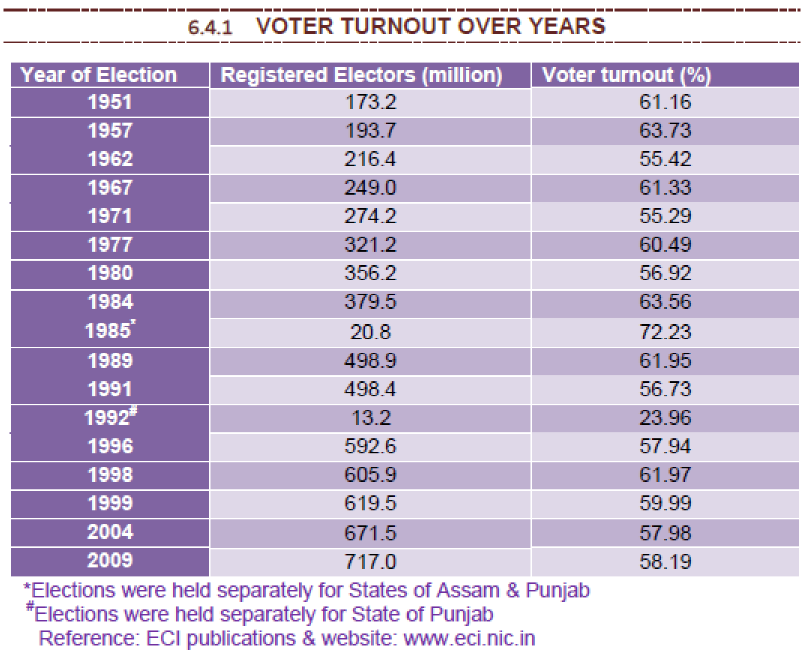
ABSTRACT

In this project, we are going to leverage the open source Blockchain concept to propose a design for new Blockchain based Ballet System that could be use in local and national elections. The goal of this project is to keep the integrity of the vote casted by the voter in the elections. The proposed Blockchain-based Ballet System will be secure, reliable and anonymous and will help to increase the number of voters and their trust. In past few years we have seen questions being raised on tampering of votes in Indian voting system. As we already seen Blockchain security in Bitcoin we learn the same and introduce it into our current voting system which is criticize. Using this project, we try to overcome the security issues in present voting system. This project is mathematically protected and is maintained by a network of peers. Digital signatures authorise individual voter's transaction basically votes which are stored in the Blockchain.

CHAPTER 1

INTRODUCTION

India us the largest democracy in the world. Democratic voting is crucial and serious event in any country. The most common ways to do the votes are: Traditional paper-based voting system which is not secure and the other one is Electronic voting system more secure than the traditional voting system. The most common ways to do the votes are: Traditional Voting System: In traditional voting system paper is used for voting process. There is no scope for automation in paper ballot system. Post-election, it takes a huge amount of time to count the votes before declaring the results. Casting votes using paper ballot is a time-consuming task. The cost of expenditure on the paper ballot is way higher than on EVMs. In few places where the governance is corrupt, they can easily insert several bogus paper votes in the ballot and then it becomes impossible to track the honest votes. Electronic Voting System: Electronic voting is a term used to describe the act of voting using electronic systems to cast and count votes. It eliminates the possibility of invalid and doubtful votes which, in many cases, are the root causes of controversies and election petitions. It makes the process of counting of votes much faster than the conventional system. It reduces to a great extent the quantity of paper used thus saving a large number of trees making the process eco-friendly. It reduces cost of printing (almost nil) as only one sheet of ballot paper is required for each Polling Station. It reduces transportation cost related to transporting conventional ballot papers and the ballot boxes in which ballot papers are. In India even after using the electronic voting system people are losing their trust in government because of the insecurity of electronic voting machine and hence the turnout is decreasing.



6.4.1 VOTER TURNOUT OVER YEARS

Year of Election	Registered Electors (million)	Voter turnout (%)
1951	173.2	61.16
1957	193.7	63.73
1962	216.4	55.42
1967	249.0	61.33
1971	274.2	55.29
1977	321.2	60.49
1980	356.2	56.92
1984	379.5	63.56
1985*	20.8	72.23
1989	498.9	61.95
1991	498.4	56.73
1992#	13.2	23.96
1996	592.6	57.94
1998	605.9	61.97
1999	619.5	59.99
2004	671.5	57.98
2009	717.0	58.19

*Elections were held separately for States of Assam & Punjab
#Elections were held separately for State of Punjab
Reference: ECI publications & website: www.eci.nic.in

Table 1.1: Statistical Report on Voting Turnout in India

The main goal of this project is to increase the security in voting system and gain people trust in their government also attract the more numbers of voters. This project helps to reduce the paperwork and workload in current voting system. Here votes are immutable, no bogus votes are allowed and reduced time delay in posting the results of the voting process. We Can keep the historic reference of votes.

CHAPTER 2

LITERATURE REVIEW

2.1 BLOCKCHAIN

Blockchain is a decentralised distributed ledger of records or blocks, where each block contains a set of transaction, hash of the previous block and a timestamp, verifiable by everyone in the network. It is typically managed in a p2p network paradigm. It is an immutable distributed ledger. Once a data is recorded it cannot be changed retroactively as it will require to change all the subsequent blocks. Therefore, recently mined blocks are less reliable and those which are mined before are more reliable.

2.2 TYPES OF BLOCKCHAIN

1. Public Blockchains

   State of the art public Blockchain protocols based on Proof of Work (PoW) consensus algorithms are open source and not permissioned. Anyone can participate, without permission. (1) Anyone can download the code and start running a public node on their local device, validating transactions in the network, thus participating in the consensus process – the process for determining what blocks get added to the chain and what the current state is. (2) Anyone in the world can send transactions through the network and expect to see them included in the blockchain if they are valid. (3) Anyone can read transaction on the public block explorer. Transactions are transparent, but anonymous/pseudonumous. Examples: Bitcoin, Ethereum, Monero, Dash, Litecoin, Dodgecoin, etc. Effects: (1) Potential to disrupt current business models through disintermediation. (2) No infrastructure costs: No need to maintain servers or system admins radically reduces the costs of creating and running decentralized applications (dApps).

2. Federated Blockchains or Consortium Blockchains

   Federated Blockchains operate under the leadership of a group. As opposed to public Blockchains, they don’t allow any person with access to the Internet to participate in the process of verifying transactions. Federated Blockchains are faster (higher scalability) and provide more transaction privacy. Consortium blockchains are mostly used in the banking sector. The consensus process is controlled by a pre-selected set of nodes; for example, one might imagine a consortium of 15 financial institutions, each of which operates a node and of which 10 must sign every block in order for the block to be valid. The right to read the blockchain may be public or restricted to the participants. Example: R3 (Banks), EWF (Energy), B3i (Insurance), Corda. Effects: (1) reduces transaction costs and data redundancies and replaces legacy systems, simplifying document handling and getting rid of semi manual compliance mechanisms. (2) in that sense it can be seen as equivalent to SAP in the 1990's: reduces costs, but not disruptive!

3. Private Blockchains

   Write permissions are kept centralized to one organization. Read permissions may be public or restricted to an arbitrary extent. Example applications include database management, auditing, etc. which are internal to a single company, and so public readability may in many cases not be necessary at all. In other cases, public audit ability is desired. Private blockchains are a way of taking advantage of blockchain technology by setting up groups and participants who can verify transactions internally. This puts you at the risk of security breaches just like in a centralized system, as opposed to public blockchain secured by game theoretic incentive mechanisms. However, private blockchains have their use case, especially when it comes to scalability and state compliance of data privacy rules and other regulatory issues. They have certain security advantages, and other security disadvantages (as stated before). Examples: MONAX, Multichain. Effects: (1) reduces transaction costs and data redundancies and replaces legacy systems, simplifying document handling and getting rid of semi manual compliance mechanisms. (2) in that sense it can be seen as equivalent to SAP in the 1990's: reduces costs, but not disruptive!

2.3 BLOCKCHAIN IN BITCOIN

The blockchain is a public ledger that records bitcoin transactions. It is implemented as a chain of blocks, each block containing a hash of the previous block up to the genesis block of the chain. A novel solution accomplishes this without any trusted central authority: the maintenance of the blockchain is performed by a network of communicating nodes running bitcoin software. Transactions of the form payer X sends Y bitcoins to payee Z are broadcast to this network using readily available software applications. Network nodes can validate transactions, add them to their copy of the ledger, and then broadcast these ledger additions to other nodes. The blockchain is a distributed database – to achieve independent verification of the chain of ownership of any and every bitcoin amount, each network node stores its own copy of the blockchain. Approximately once every 10 minutes, a new group of accepted transactions, a block, is created, added to the blockchain, and quickly published to all nodes. This allows bitcoin software to determine when a particular bitcoin amount has been spent, which is necessary in order to prevent double-spending in an environment without central oversight. Whereas a conventional ledger records the transfers of actual bills or promissory notes that exist apart from it, the blockchain is the only place that bitcoins can be said to exist in the form of unspent outputs of transactions.

2.4 PROOF OF WORK (POW)

A proof-of-work (PoW) system (or protocol, or function) is an economic measure to deter denial of service attacks and other service abuses such as spam on a network by requiring some work from the service requester, usually meaning processing time by a computer. The concept was invented by Cynthia Dwork and Moni Naor as presented in a 1993 journal article. The term "Proof of Work" or POW was first coined and formalized in a 1999 paper by Markus Jakobsson and Ari Juels. An early example of the proof-of-work system used to give value to a currency is the shell money of the Solomon Islands.

A key feature of these schemes is their asymmetry: the work must be moderately hard (but feasible) on the requester side but easy to check for the service provider. This idea is also known as a CPU cost function, client puzzle, computational puzzle or CPU pricing function. It is distinct from a CAPTCHA, which is intended for a human to solve quickly, rather than a computer. Proof of space (PoS) proposals apply the same principle by proving a dedicated amount of memory or disk space instead of CPU time. Proof of bandwidth approaches have been discussed in the context of cryptocurrency. Proof of ownership aims at proving that specific data are held by the prover.

There are two classes of proof-of-work protocols.

- Challenge-response protocols assume a direct interactive link between the requester (client) and the provider (server). The provider chooses a challenge, say an item in a set with a property, the requester finds the relevant response in the set, which is sent back and checked by the provider. As the challenge is chosen on the spot by the provider, its difficulty can be adapted to its current load. The work on the requester side may be bounded if the challenge-response protocol has a known solution (chosen by the provider) or is known to exist within a bounded search space.

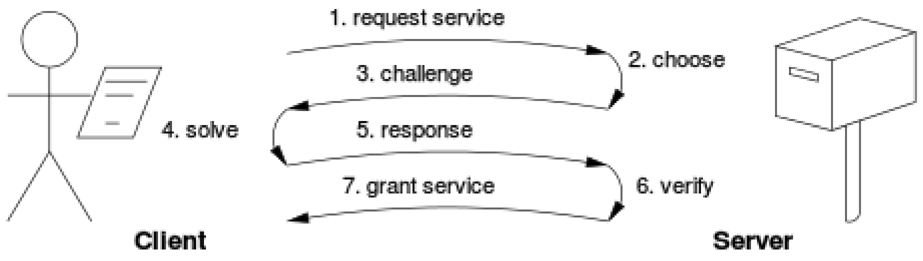


Figure 2.4.1: Challenge response protocol

- Solution-verification protocols do not assume such a link: as a result, the problem must be self-imposed before a solution is sought by the requester, and the provider must check both the problem choice and the found solution. Most such schemes are unbounded probabilistic iterative procedures such as Hashcash.

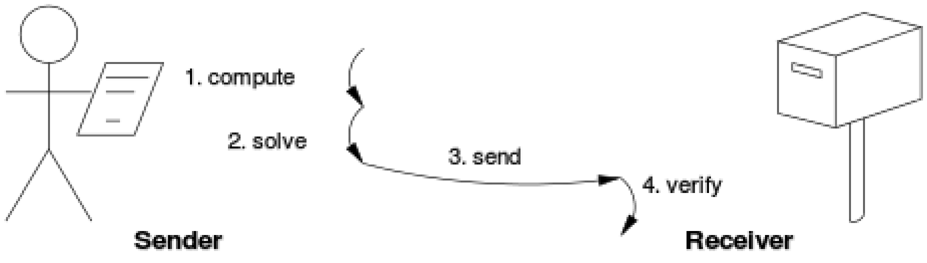


Figure 2.4.2: Solution verification protocol

List of POW:

- Integer square root modulo a large prime.
- Weaken Fiat–Shamir signatures.
- Ong–Schnorr–Shamir signature broken by Pollard.
- Partial hash inversion: This paper formalizes the idea of a proof of work (POW) and introduces "the dependent idea of a bread pudding protocol", a "re-usable proof of work" (RPOW) system as Hashcash.
- Hash sequences.
- Puzzles.
- Diffie–Hellman-based puzzle.
- Moderate.
- Mbound.
- Hokkaido.
- Cuckoo Cycle.
- Merkle tree based.
- Guided tour puzzle protocol.

CHAPTER 3

PROBLEM DEFINITION

3.1 Problem Statement

Finding a way to overcome the hitches in the traditional voting system such as speculation of integrity of votes casted.

3.2 Proposal

We have paved a way to make the voting system more robust and secured by removing the paper ballots and reducing the effort of working individuals, thus enhancing the overall smoothness of voting system and eliminating the chances of any false casted vote. We also have worked on providing several verification processes to run the process with integrity. We also provided voters a way to check their casted vote in the results by implementing new ideas.

CHAPTER 4

ARCHITECTURE DESIGN

4.1 CLASS DIAGRAMS

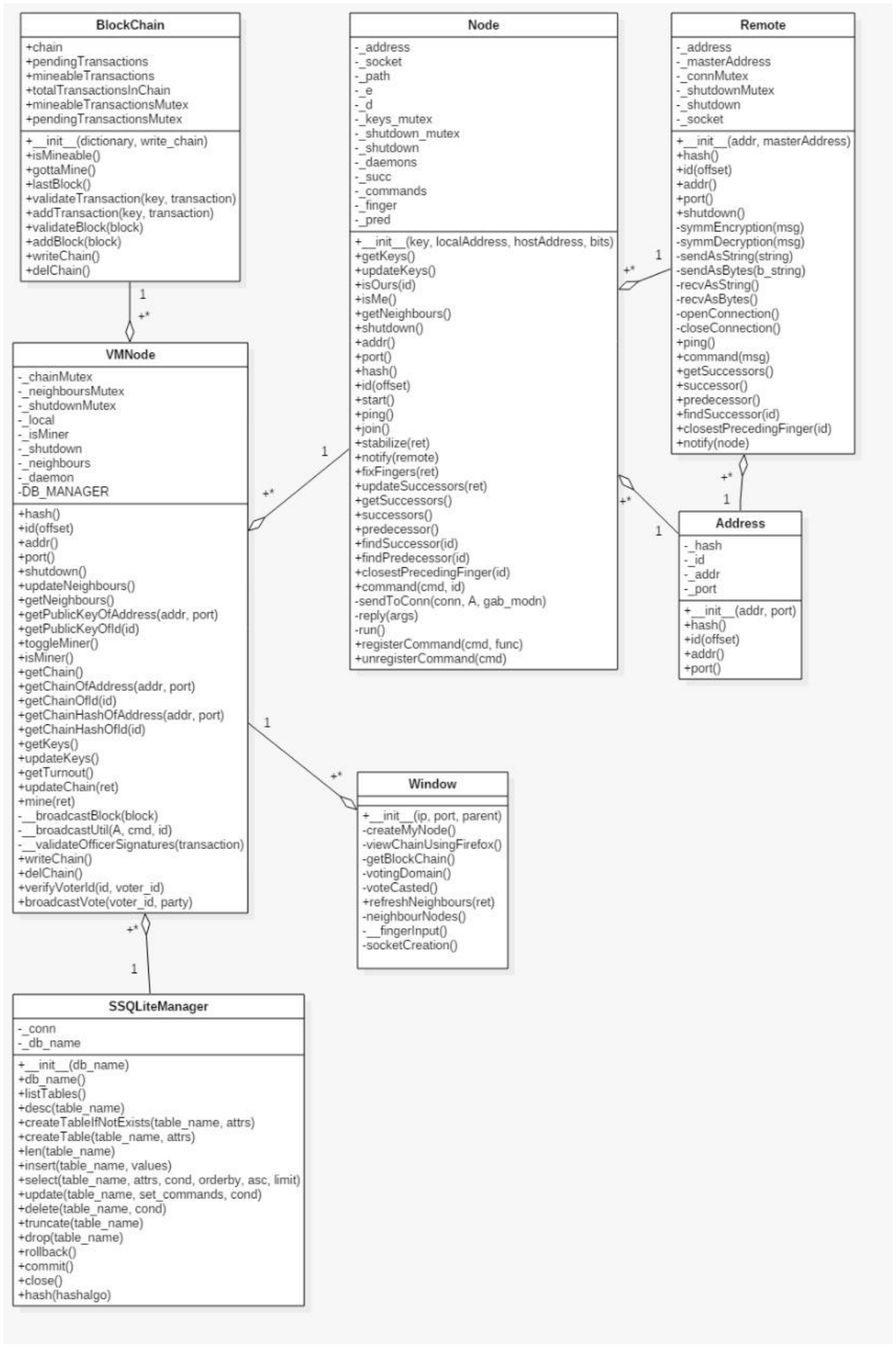


Figure 4.1.1 Voting Machine

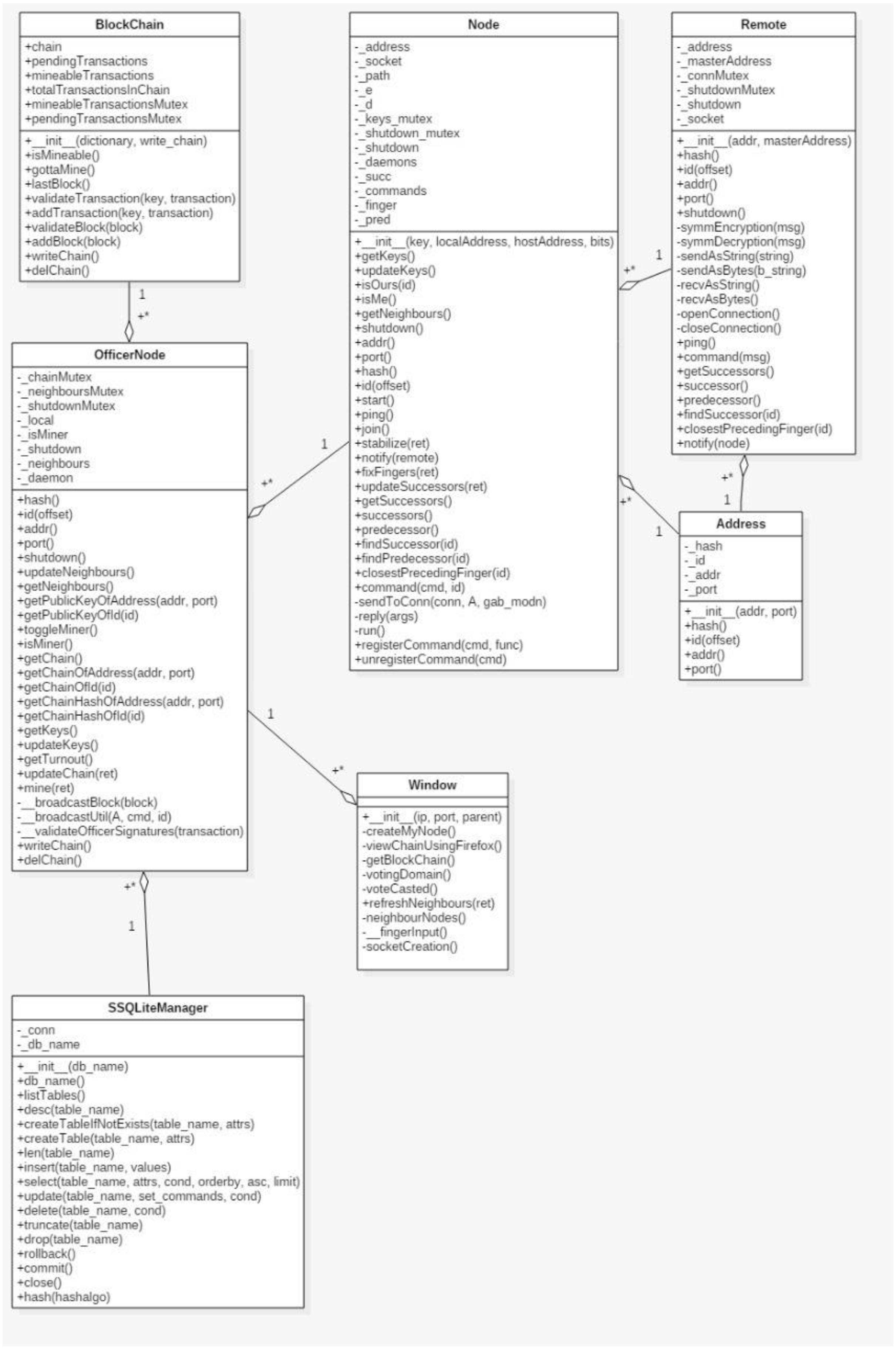


Figure 4.1.2 Officers

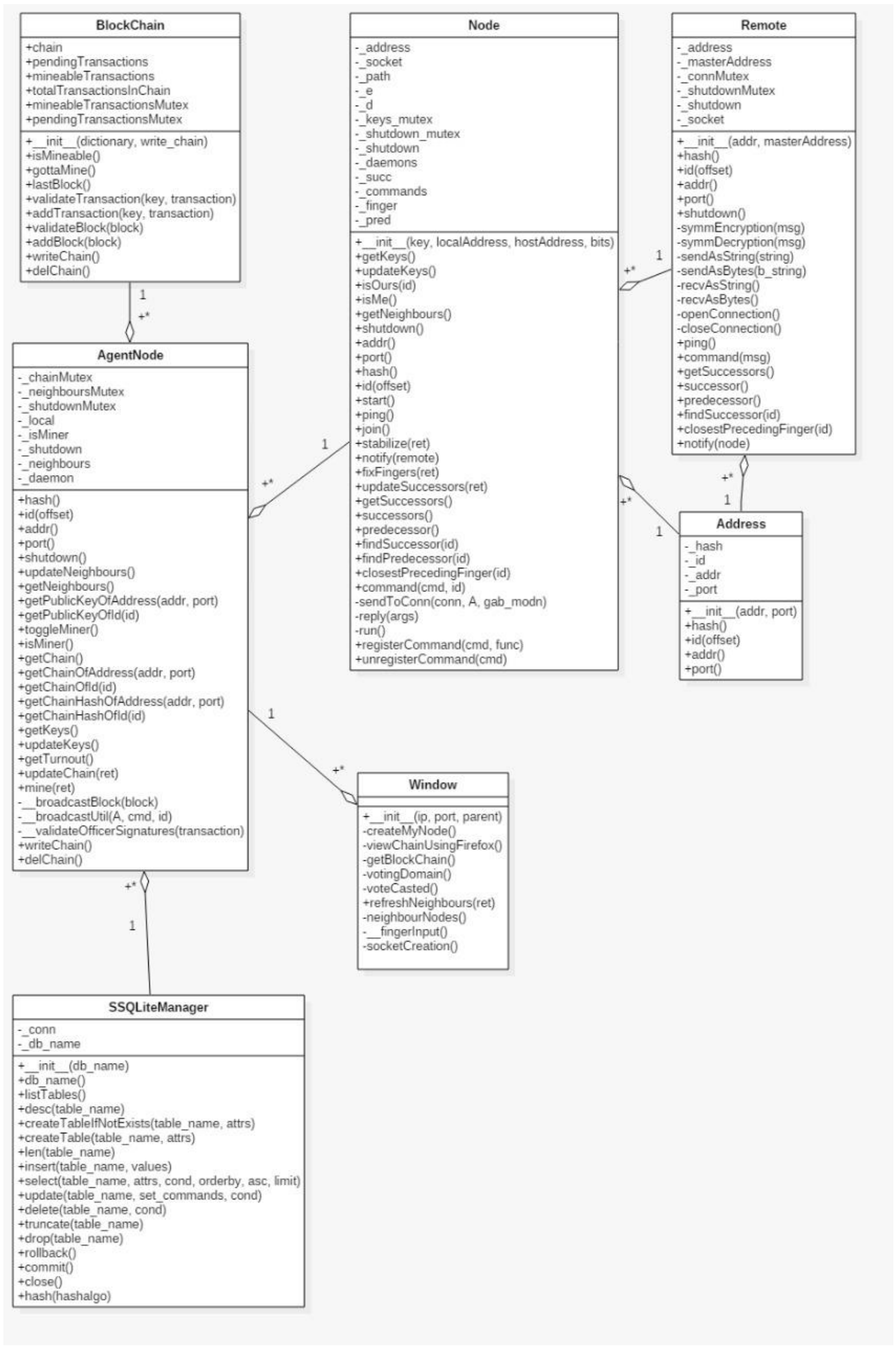


Figure 4.1.3 Party Agent

4.2 USE CASE DIAGRAMS

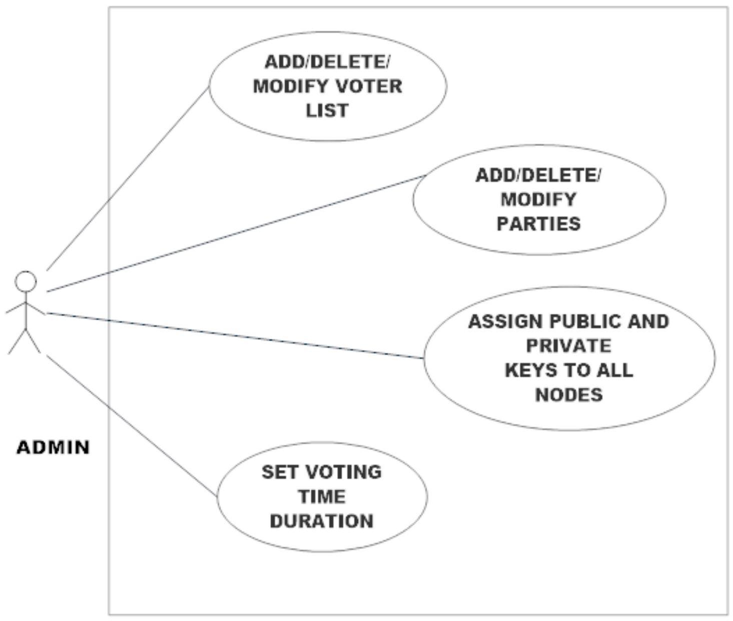

ADD/DELETE/ MODIFY VOTER LIST
ADD/DELETE/ MODIFY PARTIES
ASSIGN PUBLIC AND PRIVATE KEYS TO ALL NODES
SET VOTING TIME DURATION
ADMIN

Figure 4.2.1 Admin-System Interaction

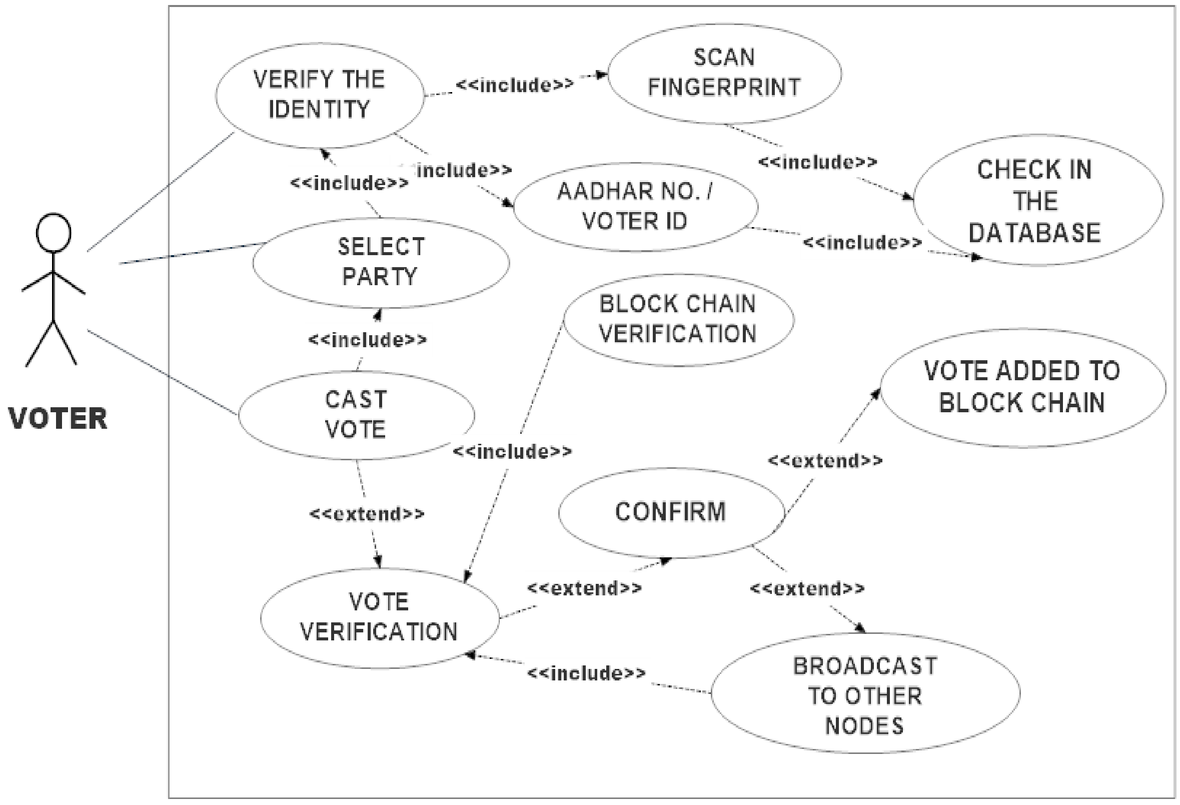


Figure 4.2.2 Voter-Voting Machine System

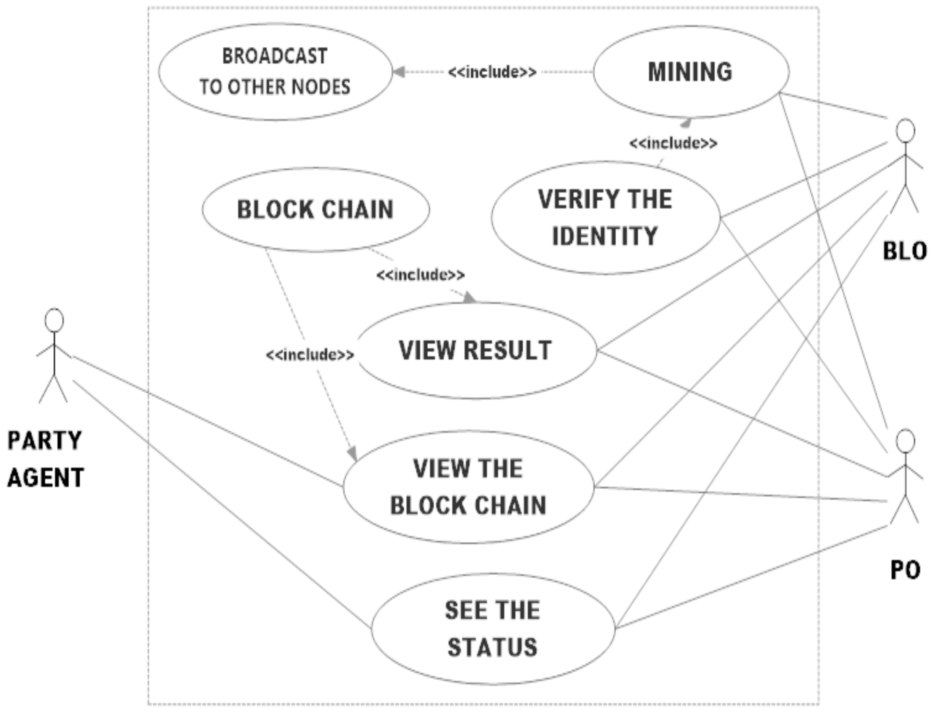


Figure 4.2.3 Vote Verification

4.3 COMPONENT DIAGRAM

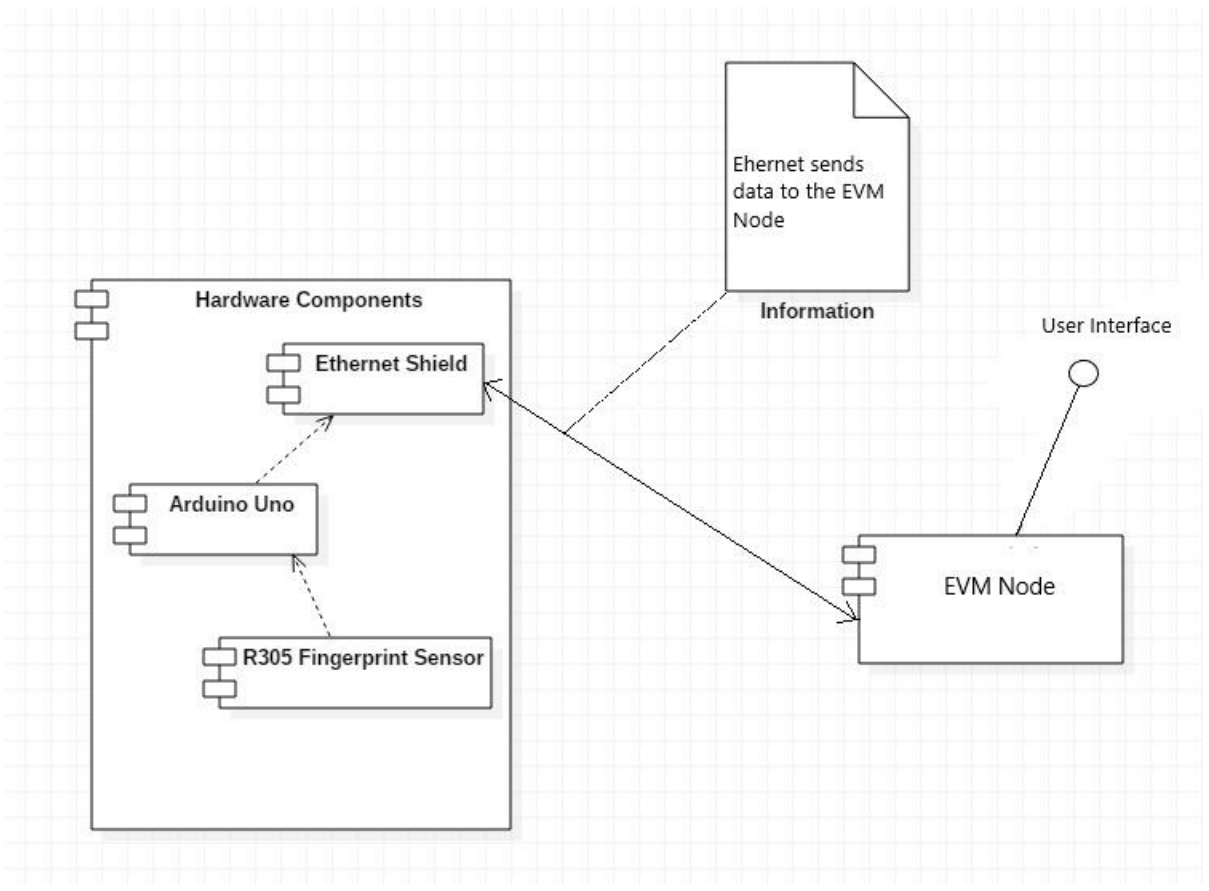


Figure 4.3.1 Hardware Architecture

4.4 ACTIVITY DIAGRAM

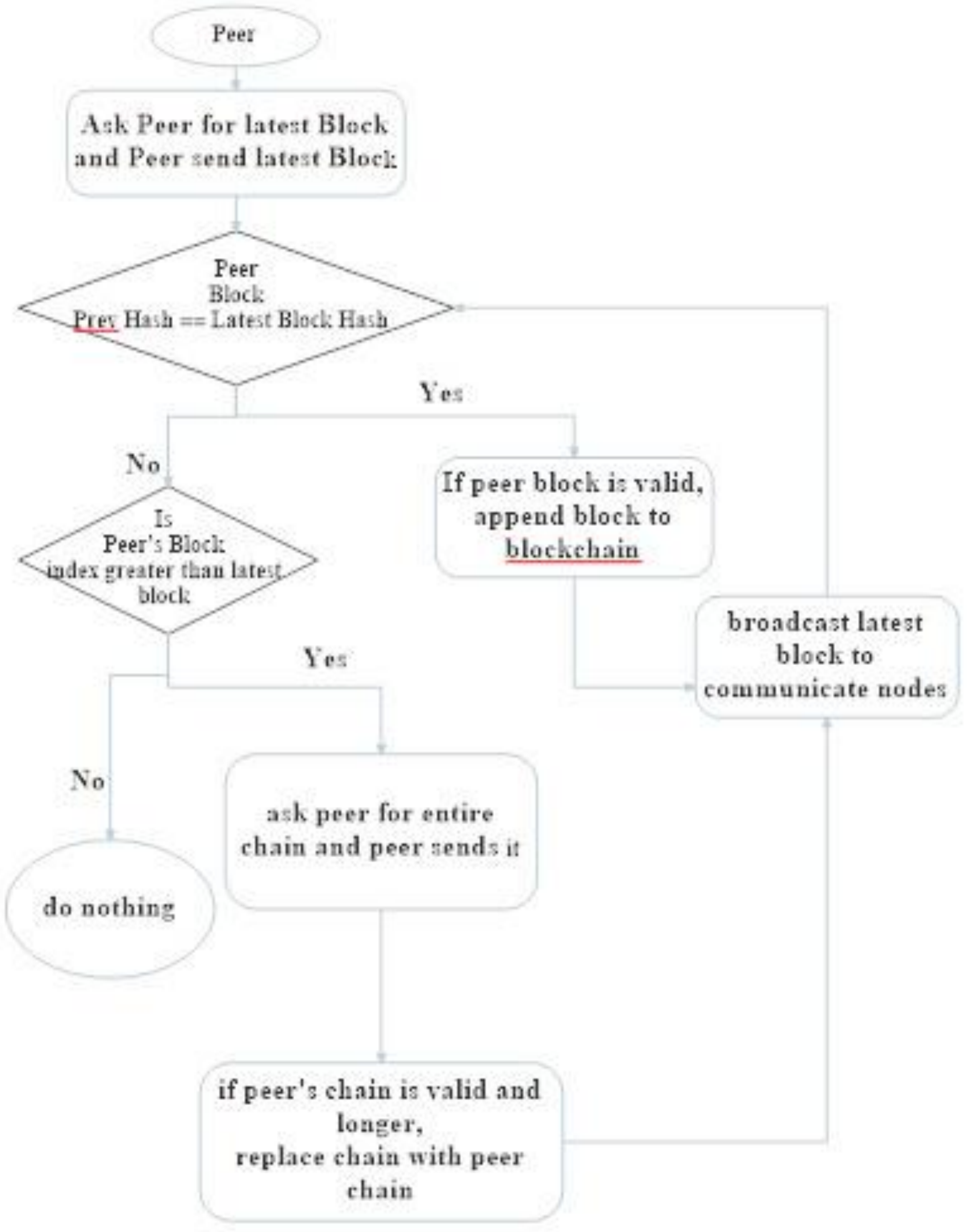

Peer
Ask Peer for latest Block and Peer send latest Block
Peer Block Prev Hash == Latest Block Hash
Yes
No
If peer block is valid, append block to blockchain
Is Peer's Block index greater than latest block
broadcast latest block to communicate nodes
Yes
No
ask peer for entire chain and peer sends it
do nothing
if peer's chain is valid and longer, replace chain with peer chain

Figure 4.4.1 P2P communication among nodes

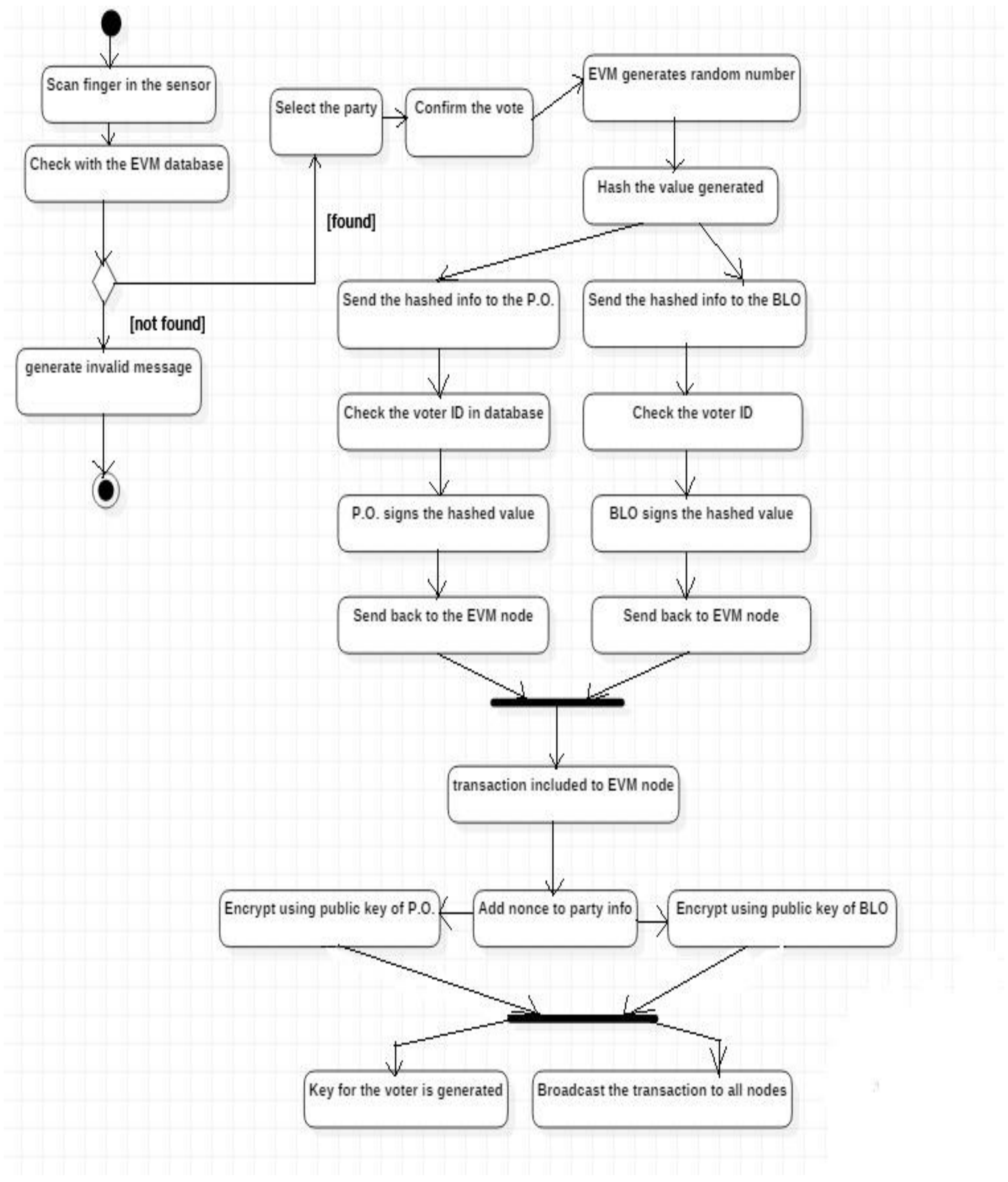

Scan finger in the sensor
Check with the EVM database
[found]
[not found]
generate invalid message
Select the party
Confirm the vote
EVM generates random number
Hash the value generated
Send the hashed info to the P.O.
Send the hashed info to the BLO
Check the voter ID in database
Check the voter ID
P.O. signs the hashed value
BLO signs the hashed value
Send back to the EVM node
Send back to EVM node
transaction included to EVM node
Encrypt using public key of P.O.
Add nonce to party info
Encrypt using public key of BLO
Key for the voter is generated
Broadcast the transaction to all nodes

Figure 4.4.2 Voting Process

4.5 DEPLOYMENT DIAGRAM

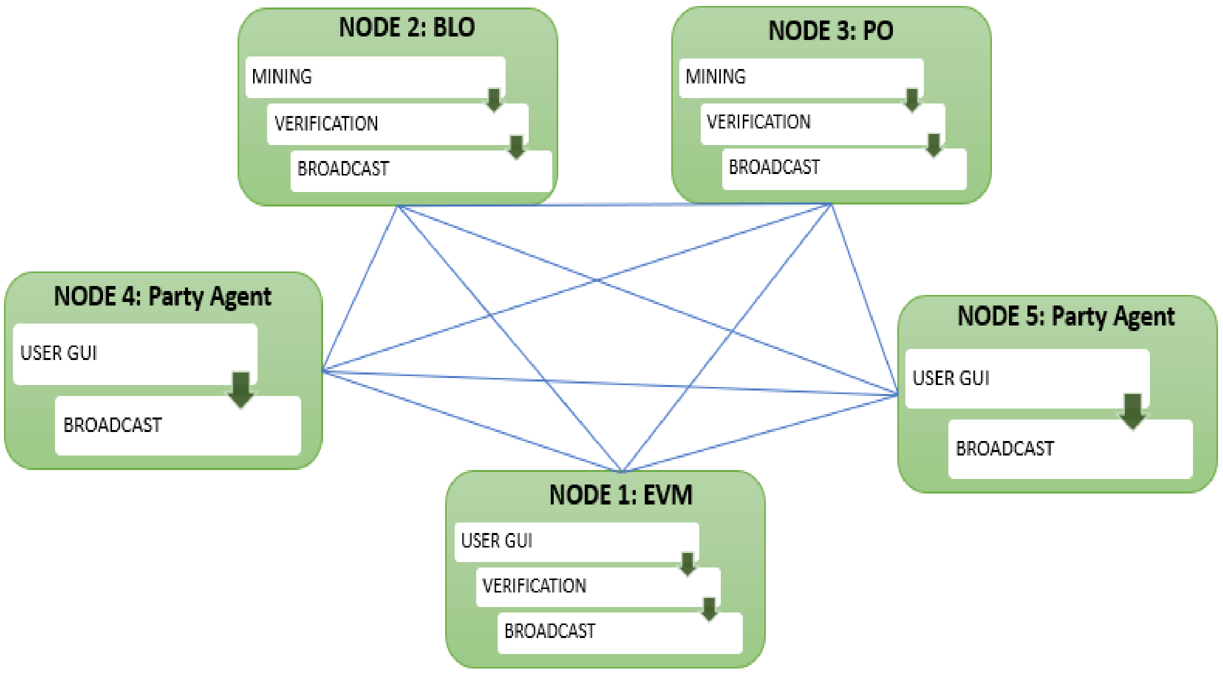


Figure 4.5.1 Overview of Nodes

CHAPTER 5

REQUIREMENT SPECIFICATIONS

5.1 Functional Requirements

5.1.1 Fingerprint match

Input: Fingerprint.

Output: Id associated with the fingerprint.

5.1.2 Cast vote

Input: Voter id and party name.

Output: True with symmetric key if vote was successful, else False.

5.2 Non-functional Requirements

5.2.1 Usability:

The system interface is easy to learn for both voters and connected nodes members. This complete user documentation can able to explain how to achieve common task in the connecting nodes. Error messages is provided in the system whenever required.

5.2.2 Reliability

The database should be consistent and correct across the nodes and any of the officers should not adhere to rules and regulations.

5.2.3 Performance:

System should work near real-time which means there should be an acceptable time delay such as max 4-5 seconds between request and response.

5.2.4 Supportability:

Our implementation works on Linux platform.

5.2.5 Scalability:

Yes.

5.2.6 Security:

It is secure until the system works adhered to the proposed project.

5.3 Constraints

1. R305 fingerprint sensor has false acceptance rate (FAR) < 0.001% and false rejection rate (FRR) <0.1%
2. R305 fingerprint sensor can capture image up to resolution of 500 dpi only.
3. Fingerprint module can only store 127 fingerprint templates in the provided Flash memory.
4. Arduino is another critical component. Since it will send data to the EVM node for processing in case of its collapse both image processing part and web-stream module will fail again.
5. Cost of the hardware should be in a reasonable margin to be affordable.
6. To run the whole project, battery life is another important issue. Recharging should be continuing for long term voting on the same connected nodes.
7. All nodes should me in a common network.
8. All nodes should have consistent database and correct.
9. Both the officers (BLO and ext.-officer) should be live for the voting procedure to run.

CHAPTER 6

IMPLEMENTATION

6.1 P2P

6.1.1 Introduction

In Computer Science, CHORD is a protocol and an algorithm for p2p distributed hash table (DHT). A DHT stores key-value pairs where it assigns keys to different computers (nodes) and values for which it is responsible for. CHORD specifies how keys will be assigned to nodes and how a node can discover value for a given key.It is one of the four original DHT protocols, along with CAN (Content Addressable Network), Tapestry, and pastry. It was introduced in 2001 by Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balkrishnan, and was developed at MIT (Massachusetts Institute of Technology).

6.1.2 Working

A m-bit identifier (key) is given to each node using consistent hashing. Cryptographic hashing, like md5, sha-1, sha-224, sha-3, blake-2s, etc; is generally used as hash functions in consistent hashing to have less probability of collisions. In our project we have used md-5 as hash function (It could be changed in config.py under “Chord p2p schema”). As the keys and nodes are uniformly distributed in the same identifier space, nodes can easily join or leave the network without causing disruption.

In chord each nodes and keys are arranged in a conceptual circle with max. 2^m nodes, ranging from 0 to 2^m - 1.

Each node has a successor and a predecessor. Successor of a node is the next node in the circle clockwise, and predecessor is the next node in the circle anti-clockwise. If every 2^m fields in the circle are occupied then successor and predecessor of node 1 is node 2 and 2^m - 1 respectively. But usually, there are holes in the circle, i.e.; successor of node 100 could be node 201, i.e.; no node has key in range 101 to 200.As nodes can join or leave the network, every node keeps track of a segment of the circle, so that they can find their correct successors and predecessor. Every node will be responsible for keys ranging from one more than its predecessor’s key and itself.

Basic queries are “which node is responsible for key k?” could be found in O(N) time (where, N is the no of nodes in the circle). Just find the successor of key k and locate it by following successor or predecessor pointers. In fig-1, if node 1 is queried for key 7, then node 1 will send this query to its successor node 2, node 2 will send this query to node 3, 3 will send to 4, 4 will send to 5, 5 will send to 6, 6 will send to 8. Node 8 will say that it is responsible for key 7 and contact node 1 directly. It required 6 hops.

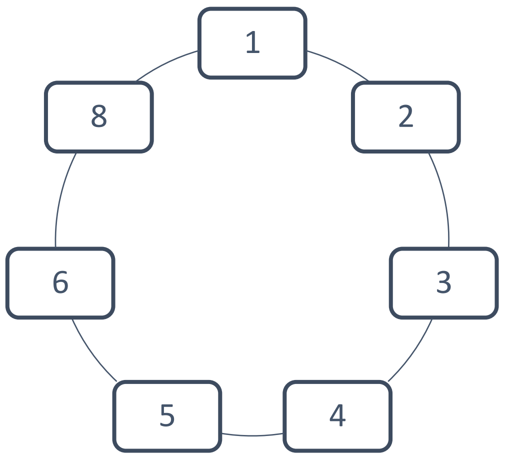


Figure 6.1.1: Conceptual circle depicting p2p network (with successor and predecessor only) (m = 3)

To avoid this linear search, along with successor and predecessor address every node keeps a finger table with at most m entries. The ith entry in the finger table of node n is successor of (n + 2^{i-1}) mod 2^m. The first entry in the finger table is the node's immediate successor, second entry is the successor of the node's successor, third entry is the successor of successor of successor of node's successor, and so on. Every time a node wants to lookup a key k, it will pass the query to the closest predecessor or successor, until a node finds out its immediate successor is responsible for the key k. This is O(log N) time. In fig-2, if node 1 is queried for key 7, it will first find the closest predecessor of key 7 in its finger table, i.e.; 5. Instead passing the query to node 2, node 1 will pass this query to node 5. Node 5 will also find the closest predecessor of key 7 in its finger table, i.e.; 6. Node 5 will pass this query to node 6. Node 6 will do the same and pass this query to node 8. Node 8 will say that it is responsible for this key and directly contact node 1. It only required 3 hops.

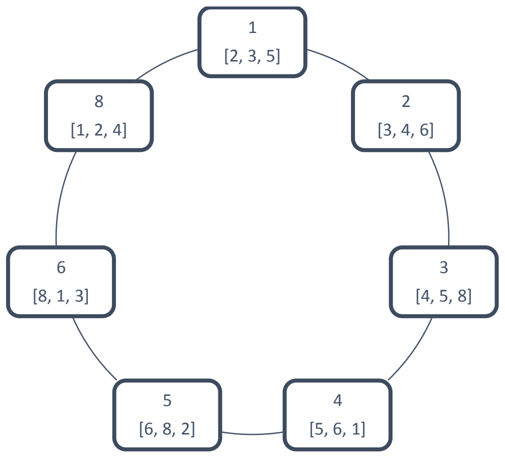


Figure 6.1.2: Conceptual circle depicting p2p network (with successor, predecessor and finger table) (m = 3)

When a node joins the network, it asks the host node to find its successor and fill its first entry of the finger table (O(log N) time). Then a daemon process (fixFinger) is executed for updating the finger table entries which runs again and again after a short interval. Updating algorithm is simply pick a random finger table entry and update its successor. Every node sends a heartbeat to its successor about informing that it is alive and it is the valid predecessor. For this daemon processes (stabilize and notify) are executed, which runs again and again after a short interval. For finding the successor of a key, first the node checks if its predecessor is responsible for that key, if yes returns itself as the successor, otherwise finds the predecessor of that key. For finding the predecessor of a key, the node first finds the closest preceding node of the key, query that node for finding the closest preceding node of the same key, until a responsible node for the key is found.

6.2 Arduino: Fingerprint Sensor

Hardware device which is a wearable component consists of three main units: Arduino Uno, Arduino Ethernet Shield, R305 fingerprint sensor module. Arduino Uno is a microcontroller board for write and upload computer code to the physical board. Ethernet shield allows an Arduino board to connect to the internet using the open source Ethernet library and through Ethernet cable. R305 fingerprint sensor module with TTL UART interface for direct communications to microcontroller UART.

Hardware models are listed below:

- ✓ Arduino Uno R3 ATmega (Comfortable with USB Cable)
- ✓ Ethernet Shield W5100 for Arduino 328 Uno
- ✓ R305 Fingerprint module
- ✓ Ethernet Patch Cable

In this project Arduino Uno is used as an intermediary device to communicate/send the fingerprint ID to the server and also act as a hardware interface to be connected by various hardware components such as R305 fingerprint module (store the fingerprint data in the module and can configure it in 1:1 or 1: N mode for identifying the person.) and Ethernet shield for sending the data (here User ID) to the PC for the verification with the local database.

Devices	**Features**
Arduino Uno R3 ATmega	Microcontroller: ATmega328P Operating voltage: 5V Digital I/O Pins: 14 Analog Input Pins: 6 USB power supply A power jack A reset button
Ethernet Shield W5100 for Arduino 328 Uno	Connection speed: 10/100Mb Connection with Arduino on SPI port Operating voltage: 5V (supplied from the Arduino board) Ethernet Controller: W5500 with internal 32K buffer
R305 Fingerprint module	Power DC: 3.6V-6.0V Interface: UART TTL Image acquiring time: <0.5s Matching Mode: 1:1 and 1: N Baud rate: (9600*N) bps, N=1-12 Average searching time: <0.8s Window dimension: 18mm*22mm Template size: 512 bytes Storage capacity: 256

Table 6.2.1: Arduino Components Features

Apart from these hardwires we use Arduino Software(IDE) which makes easy to write code and upload it to the board. It can run on Windows, Mac OS X and Linux. The sketches/programs are written in the text editor and saved with the file extension.ino.

At first, we enrolled all the fingerprint templates (for eligible voters for a particular booth) using “Adafruit Fingerprint” an open source sensor library and stored in the onboard FLASH memory. There are basically two requirements for the project.

1. Enrolling:
   First is we enrolled fingerprints of all the voters to an assigning ID #’s to search them latter. We enrol the voters fingerprint using Windows software with serial monitor to enter assigning ID. All the enrolled fingerprint templates stored in the Flash memory of Fingerprint module.
2. Searching:
   Once the EVM node is ready and a new voter place their finger in sensor, the corresponding ID is output in the serial monitor. We use this ID to match the corresponding Aadhar Number/Voter Number in the EVM node database. In the searching program we collect this assigned ID and send it to the EVM node through Ethernet cable, which is powered up by Ethernet shield and its open source library “Ethernet Library” to connect to another PC/ Internet.

Connection Sketches:

1) Wiring for R305 module with Arduino Uno

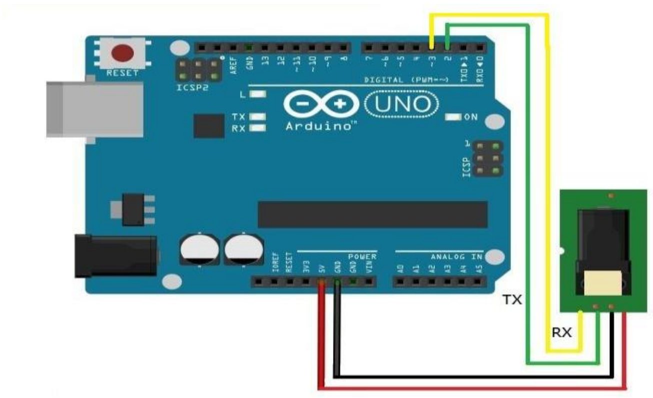


Figure 6.2.1: Wiring for R305 module with Arduino Uno

2) Ethernet shield with Arduino Uno

Figure 6.2.2: Ethernet shield with Arduino Uno

Screenshot of the connected hardware:

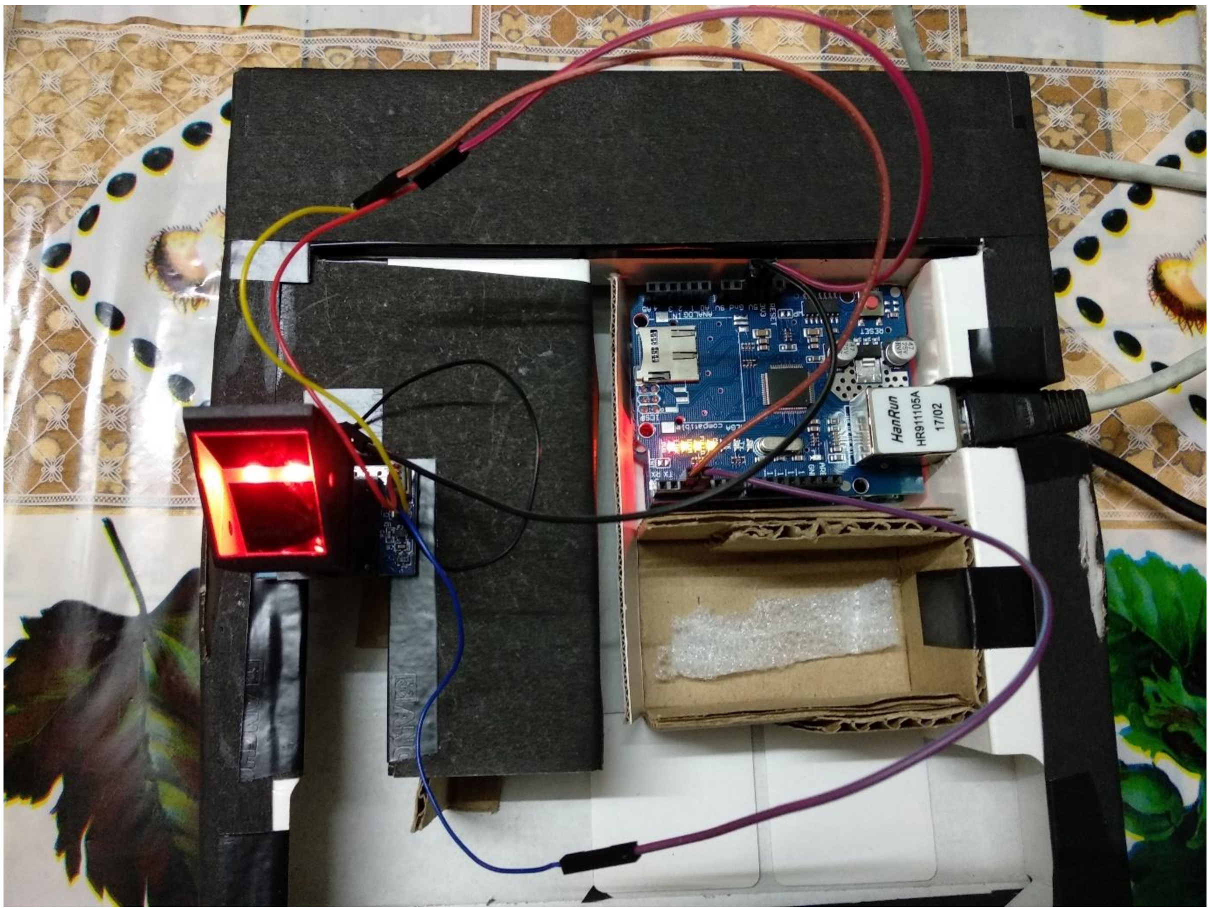


Figure 6.2.3: Working Fingerprint Sensor

6.3 PyQt5 FRAMEWORK

6.3.1 What is PyQt?

Qt is set of cross-platform C++ libraries that implement high-level APIs for accessing many aspects of modern desktop and mobile systems. These include location and positioning services, multimedia, NFC and Bluetooth connectivity, a Chromium based web browser, as well as traditional UI development.
PyQt5 is a comprehensive set of Python bindings for Qt v5. It is implemented as more than 35 extension modules and enables Python to be used as an alternative application development language to C++ on all supported platforms including iOS and Android. PyQt5 may also be embedded in C++ based applications to allow users of those applications to configure or enhance the functionality of those applications.

6.3.2 Why PyQt?

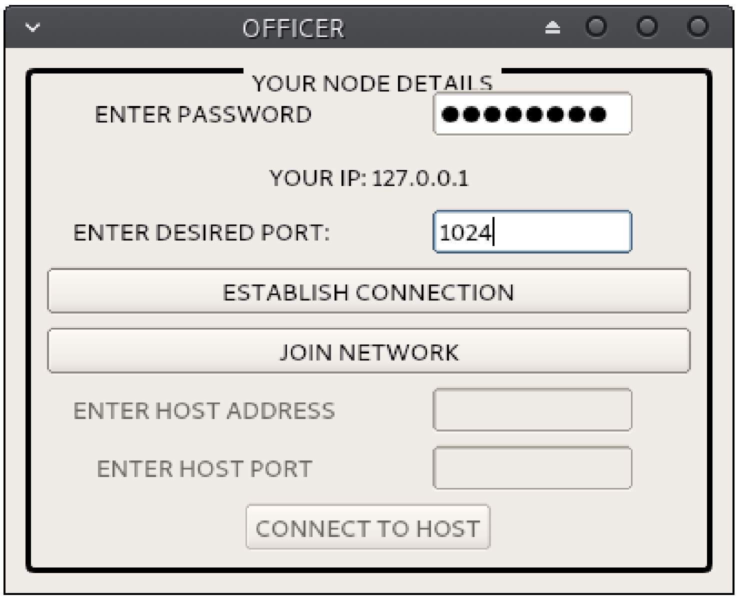
PyQt brings together the Qt C++ cross-platform application framework and the cross-platform interpreted language Python. Qt is more than a GUI toolkit. It includes abstractions of network sockets, threads, Unicode, regular expressions, SQL databases, SVG, OpenGL, XML, a fully functional web browser, a help system, a multimedia framework, as well as a rich collection of GUI widgets. Qt classes employ a signal/slot mechanism for communicating between objects that is type safe but loosely coupled making it easy to create re-usable software components. Qt also includes Qt Designer, a graphical user interface designer. PyQt is able to generate Python code from Qt Designer. It is also possible to add new GUI controls written in Python to Qt Designer.
Python is a simple but powerful object-orientated language. Its simplicity makes it easy to learn, but its power means that large and complex applications can be created. Its interpreted nature means that Python programmers are very productive because there is no edit/compile/link/run development cycle.
Much of Python's power comes from its comprehensive set of extension modules providing a wide variety of functions including HTTP servers, XML parsers, database access, data compression tools and, of course, graphical user interfaces. Extension modules are usually implemented in either Python, C or C++. Using tools such as SIP it is relatively straight forward to create an extension module that encapsulates an existing C or C++ library. Used in this way, Python can then become the glue to create new applications from established libraries.
PyQt combines all the advantages of Qt and Python. A programmer has all the power of Qt but is able to exploit it with the simplicity of Python.

6.3.3 Pyqt5 modules used:

- QtCore -- The QtCore module contains the core non-GUI functionality. This module is used for working with time, files and directories, various data types, streams, URLs, mime types, threads or processes.
- QtGui -- The QtGui contains classes for windowing system integration, event handling, 2D graphics, basic imaging, fonts and text.

- QtWidgets -- The QtWidgets module contains classes that provide a set of UI elements to create classic desktop-style user interfaces.
- QtMultimedia -- The QtMultimedia contains classes to handle multimedia content and APIs to access camera and radio functionality.

GUI Screenshots:



ENTER PASSWORD ●●●●●●●●●

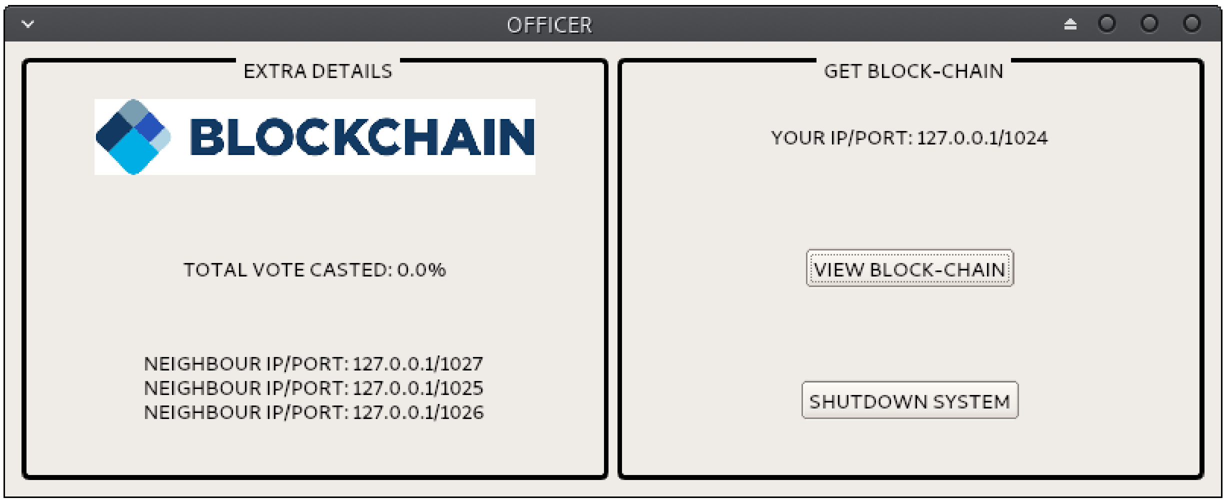
YOUR IP: 127.0.0.1

ENTER DESIRED PORT: 1024

ENTER HOST ADDRESS

ENTER HOST PORT

Figure 6.3.1: BLO Login Window

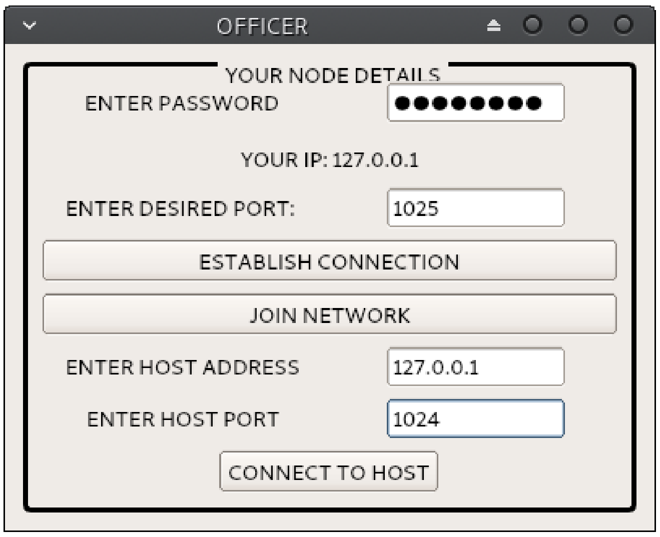


TOTAL VOTE CASTED: 0.0%

NEIGHBOUR IP/PORT: 127.0.0.1/1027
NEIGHBOUR IP/PORT: 127.0.0.1/1025
NEIGHBOUR IP/PORT: 127.0.0.1/1026

YOUR IP/PORT: 127.0.0.1/1024

Figure 6.3.2: BLO After Login Window


OFFICER

YOUR NODE DETAILS

ENTER PASSWORD ●●●●●●●●

YOUR IP: 127.0.0.1

ENTER DESIRED PORT: 1025

ESTABLISH CONNECTION

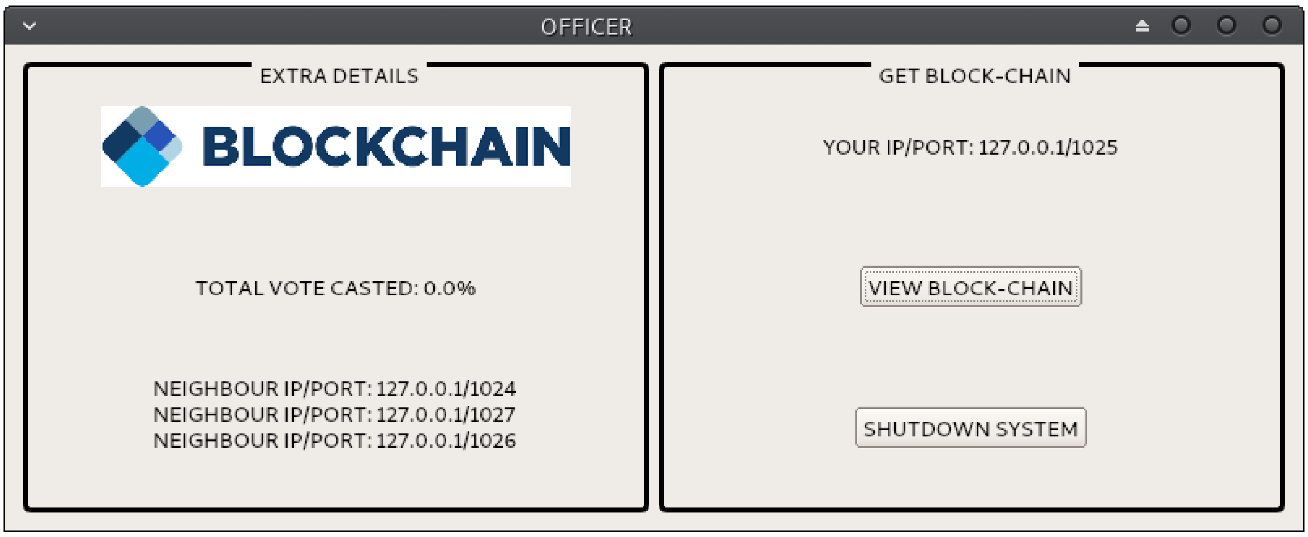
JOIN NETWORK

ENTER HOST ADDRESS 127.0.0.1

ENTER HOST PORT 1024

CONNECT TO HOST

Figure 6.3.3: External Officer Window


OFFICER

EXTRA DETAILS

BLOCKCHAIN

TOTAL VOTE CASTED: 0.0%

NEIGHBOUR IP/PORT: 127.0.0.1/1024
NEIGHBOUR IP/PORT: 127.0.0.1/1027
NEIGHBOUR IP/PORT: 127.0.0.1/1026

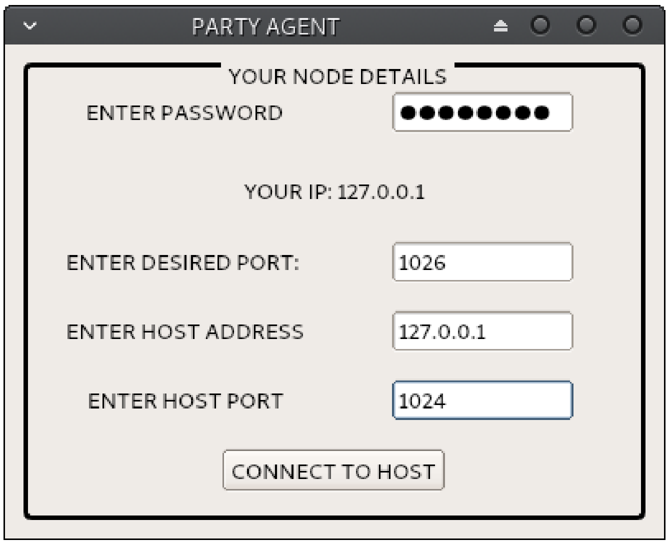
GET BLOCK-CHAIN

YOUR IP/PORT: 127.0.0.1/1025

VIEW BLOCK-CHAIN

SHUTDOWN SYSTEM

Figure 6.3.4: External Officer After Login Window


PARTY AGENT

YOUR NODE DETAILS

ENTER PASSWORD ●●●●●●●●

YOUR IP: 127.0.0.1

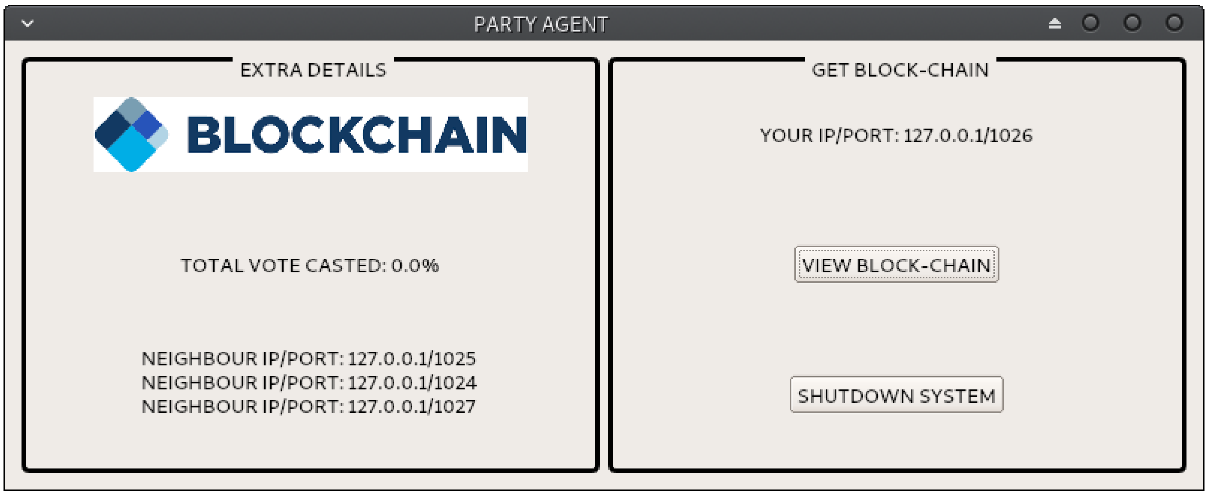
ENTER DESIRED PORT: 1026

ENTER HOST ADDRESS 127.0.0.1

ENTER HOST PORT 1024

CONNECT TO HOST

Figure 6.3.5: Party Agent Login Window


PARTY AGENT

EXTRA DETAILS

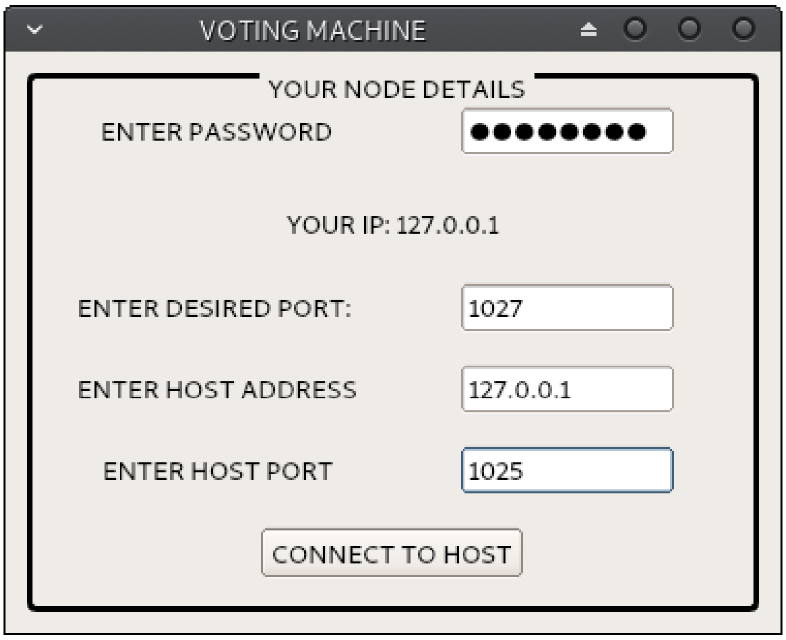
TOTAL VOTE CASTED: 0.0%

NEIGHBOUR IP/PORT: 127.0.0.1/1025
NEIGHBOUR IP/PORT: 127.0.0.1/1024
NEIGHBOUR IP/PORT: 127.0.0.1/1027

GET BLOCK-CHAIN

YOUR IP/PORT: 127.0.0.1/1026

Figure 6.3.6: Party Agent After Login Window


VOTING MACHINE

YOUR NODE DETAILS

ENTER PASSWORD ●●●●●●●●●

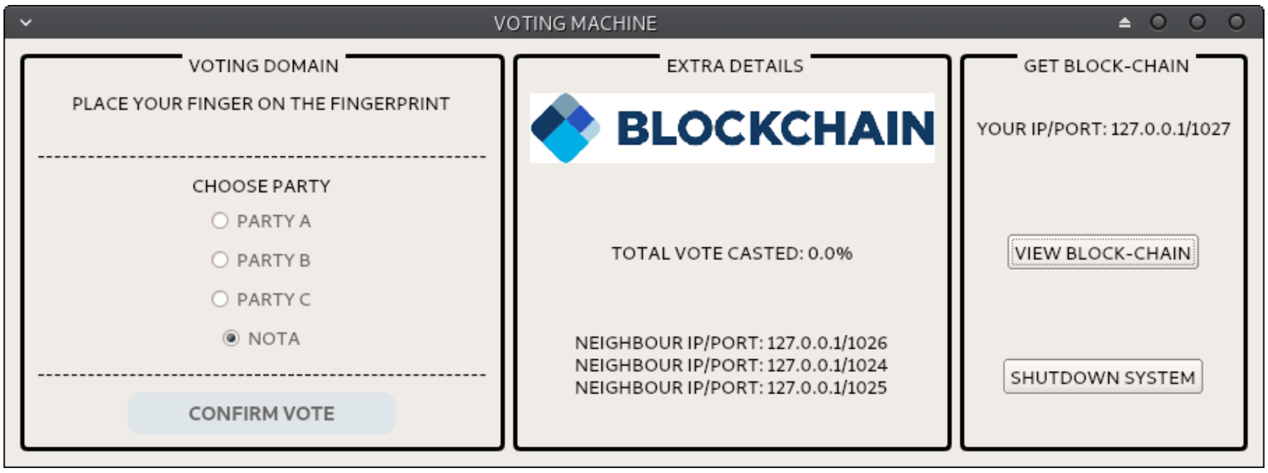
YOUR IP: 127.0.0.1

ENTER DESIRED PORT: 1027

ENTER HOST ADDRESS 127.0.0.1

ENTER HOST PORT 1025

Figure 6.3.7: Voting Machine Login Window


VOTING MACHINE

VOTING DOMAIN

PLACE YOUR FINGER ON THE FINGERPRINT

CHOOSE PARTY

- PARTY A
- PARTY B
- PARTY C
- [✓ NOTA]

EXTRA DETAILS

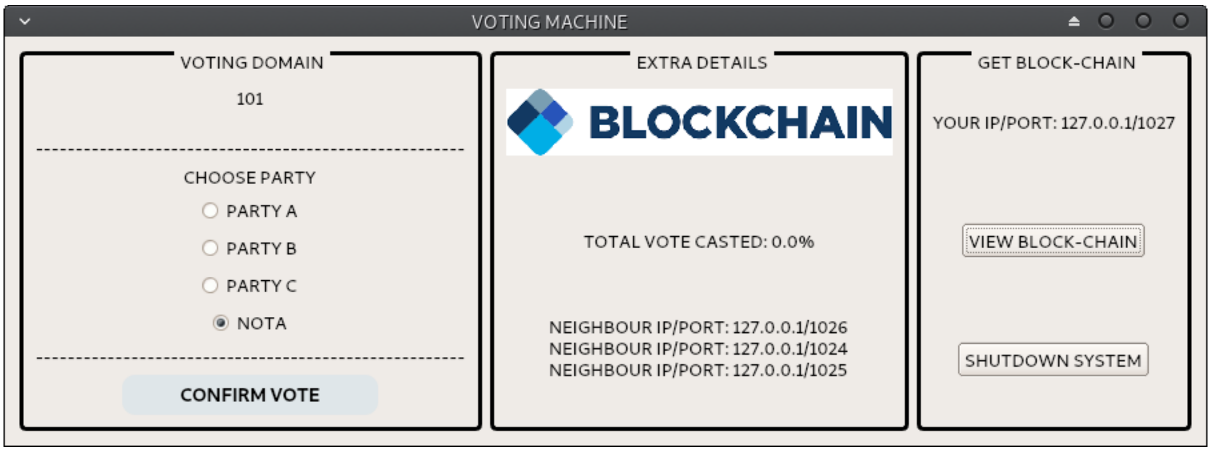
TOTAL VOTE CASTED: 0.0%

NEIGHBOUR IP/PORT: 127.0.0.1/1026
NEIGHBOUR IP/PORT: 127.0.0.1/1024
NEIGHBOUR IP/PORT: 127.0.0.1/1025

GET BLOCK-CHAIN

YOUR IP/PORT: 127.0.0.1/1027

Figure 6.3.8: Voting Machine After Login Window


VOTING MACHINE

VOTING DOMAIN

101

CHOOSE PARTY

PARTY A
PARTY B
PARTY C
[✓ NOTA]

CONFIRM VOTE

EXTRA DETAILS

TOTAL VOTE CASTED: 0.0%

NEIGHBOUR IP/PORT: 127.0.0.1/1026
NEIGHBOUR IP/PORT: 127.0.0.1/1024
NEIGHBOUR IP/PORT: 127.0.0.1/1025

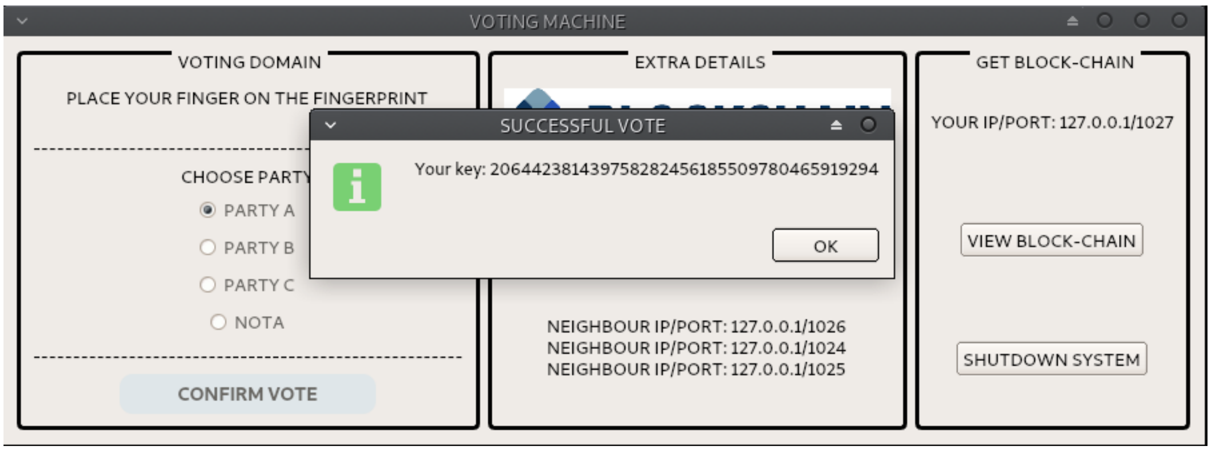
GET BLOCK-CHAIN

YOUR IP/PORT: 127.0.0.1/1027

VIEW BLOCK-CHAIN

SHUTDOWN SYSTEM

Figure 6.3.9: Voting Machine After Scanning FingerPrint


VOTING MACHINE

VOTING DOMAIN

PLACE YOUR FINGER ON THE FINGERPRINT

CHOOSE PARTY

[✓ PARTY A]
PARTY B
PARTY C
NOTA

CONFIRM VOTE

SUCCESSFUL VOTE

Your key: 206442381439758282456185509780465919294

OK

EXTRA DETAILS

NEIGHBOUR IP/PORT: 127.0.0.1/1026
NEIGHBOUR IP/PORT: 127.0.0.1/1024
NEIGHBOUR IP/PORT: 127.0.0.1/1025

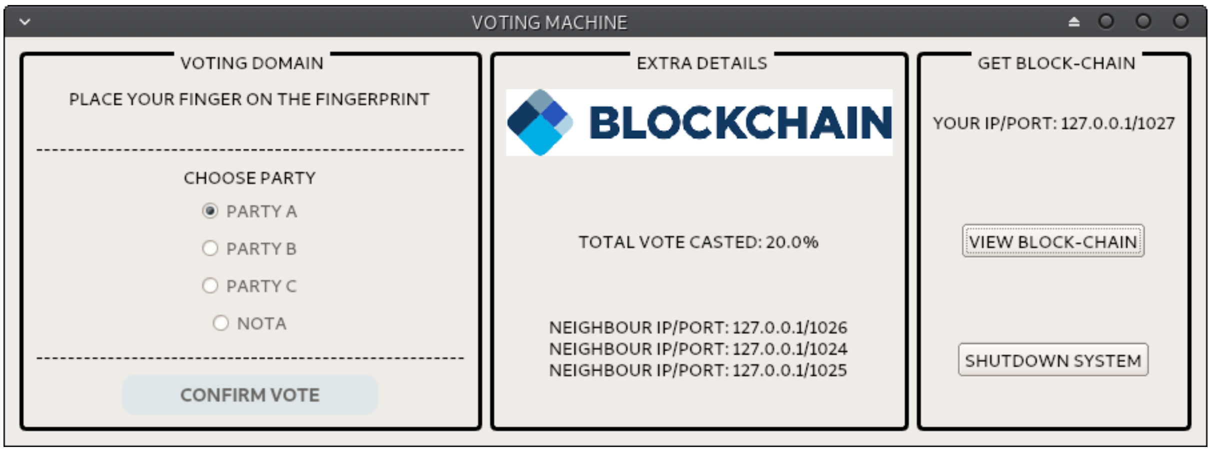
GET BLOCK-CHAIN

YOUR IP/PORT: 127.0.0.1/1027

VIEW BLOCK-CHAIN

SHUTDOWN SYSTEM

Figure 6.3.10: Voting Machine After Casting the vote


VOTING MACHINE

VOTING DOMAIN

PLACE YOUR FINGER ON THE FINGERPRINT

CHOOSE PARTY

[✓ PARTY A]
PARTY B
PARTY C
NOTA

CONFIRM VOTE

EXTRA DETAILS

TOTAL VOTE CASTED: 20.0%

NEIGHBOUR IP/PORT: 127.0.0.1/1026
NEIGHBOUR IP/PORT: 127.0.0.1/1024
NEIGHBOUR IP/PORT: 127.0.0.1/1025

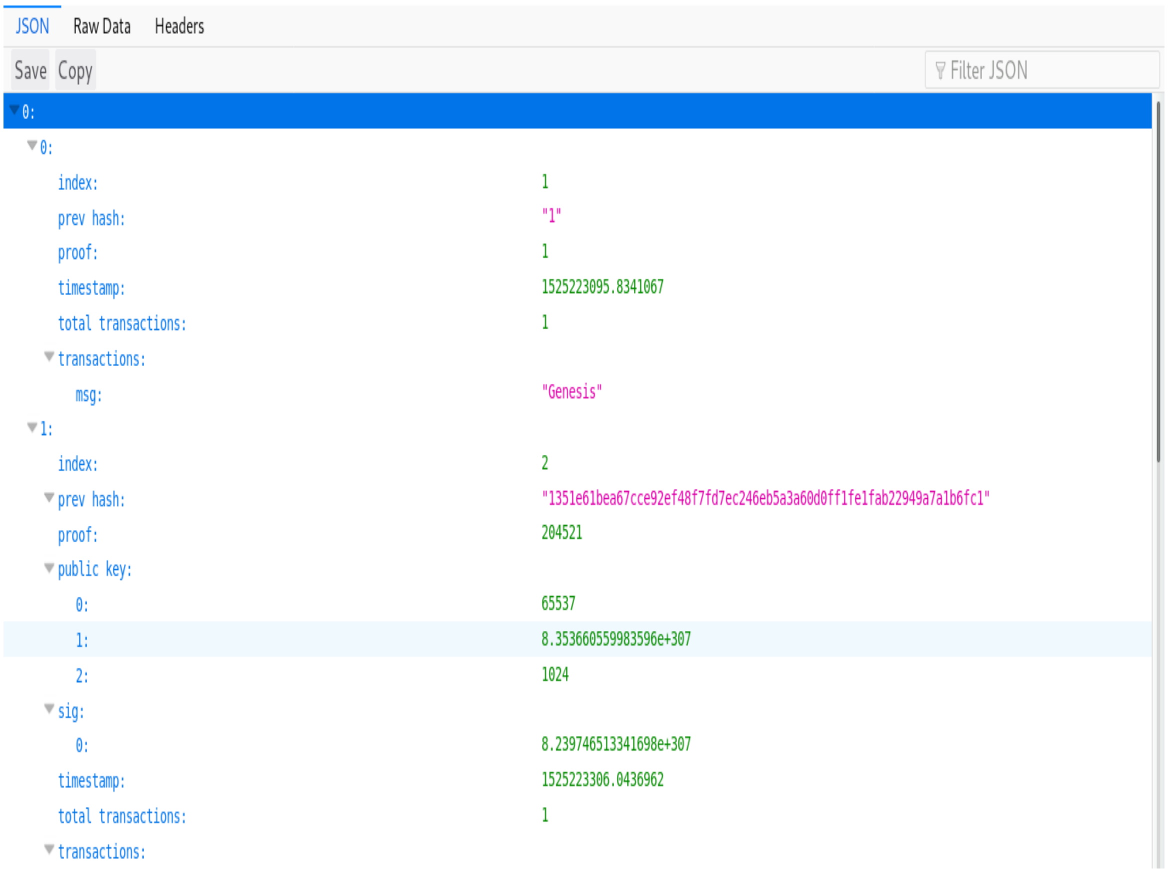
GET BLOCK-CHAIN

YOUR IP/PORT: 127.0.0.1/1027

VIEW BLOCK-CHAIN

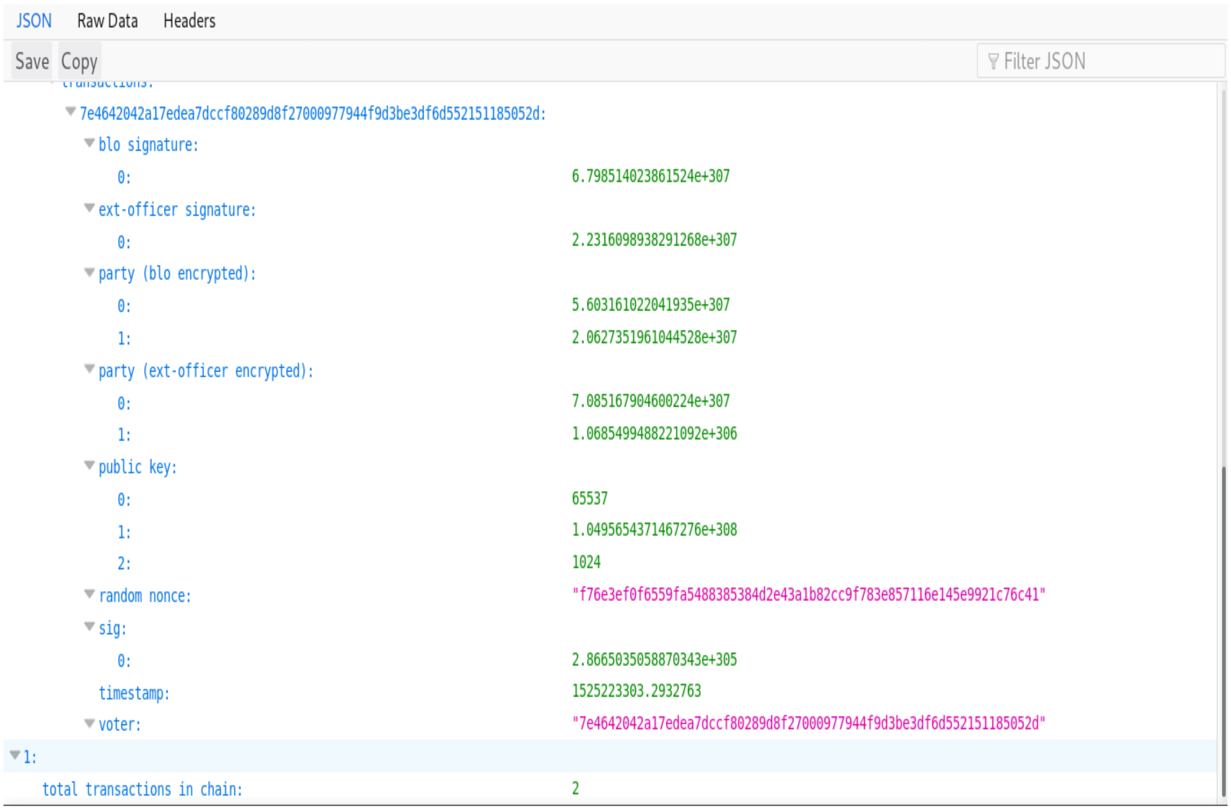
SHUTDOWN SYSTEM

Figure 6.3.11: Voting Machine After Accepting the Key



```
JSON   Raw Data   Headers
Save  Copy                                              Filter JSON
▼0:
  ▼0:
      index:                                  1
      prev hash:                              "1"
      proof:                                  1
      timestamp:                              1525223095.8341067
      total transactions:                     1
    ▼transactions:
        msg:                                  "Genesis"
  ▼1:
      index:                                  2
    ▼prev hash:                               "1351e61bea67cce92ef48f7fd7ec246eb5a3a60d0ff1fe1fab22949a7a1b6fc1"
      proof:                                  204521
    ▼public key:
        0:                                    65537
        1:                                    8.353660559983596e+307
        2:                                    1024
    ▼sig:
        0:                                    8.239746513341698e+307
      timestamp:                              1525223306.0436962
      total transactions:                     1
    ▼transactions:
```

Figure 6.3.12: Chain.json to represent blockchain (part-1)



```
JSON   Raw Data   Headers
Save  Copy                                              Filter JSON
    transactions:
    ▼7e4642042a17edea7dccf80289d8f27000977944f9d3be3df6d552151185052d:
      ▼blo signature:
          0:                                  6.798514023861524e+307
      ▼ext-officer signature:
          0:                                  2.2316098938291268e+307
      ▼party (blo encrypted):
          0:                                  5.603161022041935e+307
          1:                                  2.0627351961044528e+307
      ▼party (ext-officer encrypted):
          0:                                  7.085167904600224e+307
          1:                                  1.0685499488221092e+306
      ▼public key:
          0:                                  65537
          1:                                  1.0495654371467276e+308
          2:                                  1024
      ▼random nonce:                          "f76e3ef0f6559fa5488385384d2e43a1b82cc9f783e857116e145e9921c76c41"
      ▼sig:
          0:                                  2.8665035058870343e+305
        timestamp:                            1525223303.2932763
      ▼voter:                                 "7e4642042a17edea7dccf80289d8f27000977944f9d3be3df6d552151185052d"
▼1:
  total transactions in chain:                2
```

Figure 6.3.13: Chain.json to represent blockchain (part-2)

CONCLUSION

The entire project reflected a secured platform for voting system tempting the interest of voters throughout the region by providing robust features of no loss in integrity of the casted votes of the voters with the help of advancing footsteps of blockchain technology.

REFERENCES

[1] https://medium.com/@brian.smocovich/what-the-heck-is-a-blockchain-63b0e48d891e

[2] https://blockgeeks.com/guides/what-is-blockchain-technology

[3] https://www.comodo.com/resources/small-business/digital-certificates2.php

[4] https://pythonspot.com/pyqt5/

[5] https://www.thehackeruniversity.com/2014/01/23/pyqt5-beginner-tutorial/

[6] https://www.adafruit.com/product/751

[7] https://www.arduino.cc/en/Guide/ArduinoEthernetShield

[8] https://cdn-shop.adafruit.com/datasheets/ZFM+user+manualV15.pdf

[9] http://www.csi-india.org/Communications/CSIC/CSIC%20December%202017.pdf

[10] https://blockchainhub.net/blockchains-and- distributed-ledger- technologies-in- general/

[12] https://github.com/gaston770/python-chord

APPENDIX

Team Contributions

This is a Visionary team consists of 5 members. All of the members of our team are 3rd year students of School of Computer Engineering from Kalinga Institute of Industrial Technology deemed to be University, Bhubaneswar.

Each team member has different interest and specialization. We tried to assign the tasks fairly among the members and since every team member prefers to work on different subjects, this does not create a complication. We shared the development workload, since our system contains diverse components which are operate on distinct domains. However, this division is not strict. Every team member is updated and aware of the other member’s works and each member is ready to make contribution to other tasks of the project. Ujjawal Sinha and Saumya Raj are mainly worked on the Software module and Graphical User Interface component, Durgesh Barwal worked on the Hardware module, Communication development and Sensors module, Hariom Kumar Sinha and Mohit Kumar Shrivastava worked on the requirement specification module and architecture design module of the project. Apart from this all team members were helpful to others in case of any requirement.

We regularly met two or three times every week as a Visionary team in order to keep each other up-to- date about the progress. Also, we did some significant tasks such as overall system integration test, in collaboration. Apart from these, we had weekly meetings with our project guide Prof. Rajat Kumar Behera in order to take his advice about our project.