BLOCKCHAIN BASED E VOTING SYSTEM

Senior Design Project

By

Name		Id
Mehedi Hasan	**-**	**1610882042**
Ishtiaque Shahriar	**-**	**1511814642**
Umma Habiba Sultana	**-**	**1620421042**

Instructor

Md. Shahriar Karim

Assistant Professor, North South University.

LETTER OF TRANSMITTAL

August, 2021

To

Dr. Mohammad Rezaul Bari
Associate Professor and Chairman,
Department of Electrical and Computer Engineering,
North South University, Bashundhara R/A, Dhaka

Subject: Submission of Capstone project Report on "My area – An E Voting Platform for online voting using Blockchain Technology".

Dear sir,
With due respect, we would like to submit our Capstone Project Report on "My Area- An E Voting Platform for online voting using Blockchain Technology" as a part of our BSc program. This report deals with a E Voting Platform where users can vote from anywhere after registration process. We used Blockchain Technology for strong security. We tried our level best to make the report meaningful and informative. The capestone project was very much valuable to us as it helped us to gain experience from practical field. It was a great learning experience for us. We tried to the maximum competence to meet all the dimensions required from this report. We will be highly obliged if you are kind enough to receive this report and provide your valuable judgment. It would be our immense pleasure if you find this report useful and informative to have an apparent perspective on the issue.

Sincerely yours,

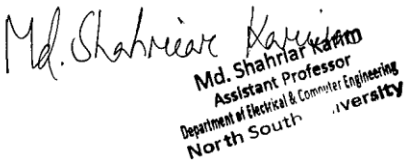
Mehedi Hasan, Ishtiaque Shahriar, and Umma Habiba Sultana

Department of ECE, North South University,
Bashundhara R/A

APPROVAL

The capstone project entitled **"My Area – An E – Voting Platform for online voting "**. " By Mehedi Hasan(ID# 1610882042), Ishtiaque Shahriar(ID# 1511814642) and Umma Habiba Sultana(ID# 1620421042), is approved in partial fulfillment of the requirement of Degree of Bechelor of Science in Computer Science and Engineering on September 2020, has been accepted as satisfactory.

Supervisor:



……

Md. Shahriar Karim
Assistant Professor,
Dpartment of Electrical and Computer Engineering
North South University
Basundhara R/A

Department Chair:

……

Dr. Mohammad Rezaul Bari
Associate Professor and Chairman
Department of Electrical and Computer Engineering
North South University
Bashundhara R/A, Dhaka.

AUTHOR' DECLARATION

This is our truthful declaration that the "Capstone Project Report" we have prepared is not a copy of any "Capstone Project Report" previously made by any other team. We also express our honest confirmation in support of the fact that the said "Capstone Project Report" has neither been used before to fulfill any other course related purpose nor it will be submitted to any other team or authority in future.

……..

Mehedi Hasan

Department of Electrical and Computer Engineering

North South University

……..

Md. Ishtiaque Shahriar

Department of Electrical and Computer Engineering

North South University

……..

Umma Habiba Sultana

Department of Electrical and Computer Engineering

North South University

ACKNOWLEDGEMENT

First of all, we wish to express our gratitude to the Almighty for giving us the strength to perform our responsibilities and complete the report.

The capstone project program is very helpful to bridge the gap between the theoretical knowledge and real life experience as part of Bachelor of Science (BSc) program. This report has been designed to have a practical experience through the theoretical understanding. Furthermore, **North South University** gave us plenty of privileges regarding technical facilities. So, we also like to admire the cooperation of our authority.

We also acknowledge our profound sense of gratitude to all the teachers who have been instrumental for providing us the technical knowledge and moral support to complete the project with full understanding. We would like to convey our gratitude to our faculty **Md. Shahriar Karim** for his stimulating inspiration, kind guidance, valuable suggestions, sagacious advice and kind cooperation throughout the period of word undertaken, which has been instrumented in the success of our project. At this level of understanding it is often difficult to understand the wide spectrum of knowledge without proper guidance and advice. His suggestions and guidance have made the report a good manner.

We like to express our heartiest gratitude to our family and friends for their moral support to carve out this project.

Abstract

Voting is a very important issue which can be beneficial in terms of choosing the right leader in an election. So people want a trust worthy voting system. The goal of our project is to replace the traditional pen and paper election with a new election system which has the ability to limit fraud and improve voting system by increasing security and decreasing the cost of hosting a nationwide election. We discussed the characteristics of our proposed **Blockchain-Based Electronic Voting** protocol in this paper. We implement this **Blockchain** system to turn election protocol into an automated control system without relying any single point of authority. This paper gives a comprehensive overview of our proposed system. We used **Hyperledger Fabric framework** in this project. As this framework is very flexible in term of **cryptography** , it was very suitable for our area. For the main functionality implementation we used **Java.** This system will ease the voting process and make people willing to participate in this new technology.

Table of Content

Chapter 1

INTRODUCTION

Now a days, election has become a sensitive issue for any democratic country. Sometimes people don't believe the voting system for the vulnerability of the process. In this era information security is a big issue. E-Voting Systems have been the challenge of recent researches, with the goal to limit the cost of an election and also ensuring the security, privacy and other requirements. To fulfill this goal many countries have already taken the initiative to improve their voting system by using blockchain. So we wanted to build such type of system.

The goals of our project is to gain the trust of the people by giving them a secure trustworthy voting system. Our voting system will maximize user participation by allowing them to vote from anywhere and from any device that has internet connection. For the people who have no internet there will be voting booth but lesser booth than traditional voting system. So, there will be less physical infrastructure. And also no need of paper ballot and lesser booth will reduce the cost of organizing a vote. Our system eliminates the errors of manual count and gives every voter the guaranty of their votes being correctly issued and accounted.

Chapter 2
RELATED TERMS

2.1 BLOCKCHAIN

*B*lockchain is a decentralized peer to peer network accompanied by a public ledger.The blockchain is a data structure, where data is stored in a distributed ledger that cannot be tampered with or deleted. This makes the ledger immutable. It is an ordered data structure that contains blocks of transactions. Each block in the chain is linked to the previous block in the chain. The first block in the chain is referred to as the foundation of the stack. Each new block created gets layered on top of the previous block. Each block in the stack is identified by a hash placed on the header. This hash is generated using the Hashing Algorithm.

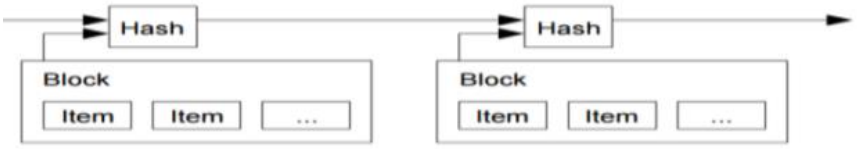


Fig 1.A Visual illustration of Blockchain

Source Bitcoin: A Peer-to-Peer Electronic Cash System, S. Nakamoto

2.2 SMART CONTRACT

A smart contract is an agreement between all the people in a blockchain network in the form of computer code [2]. They cannot be changed. The transactions that happen in a smart contract are processed by the blockchain which means they can be sent automatically without a third party. This means there is no one to rely on. The transactions only happen when the conditions in the smart contract are met. There is no third party, so there are no issues with trust.

2.2.1 How Smart Contract Work

Smart contracts work by following simple “if/when…then…” statements that are written into code on a blockchain. A network of computers executes the actions when predetermined conditions have been met and verified. These actions could include releasing funds to the appropriate parties, registering a vehicle, sending notifications, or issuing a ticket. The blockchain is then updated when the transaction is completed. That means the transaction cannot be changed, and only parties who have been granted permission can see the results [8].

Within a smart contract, there can be as many stipulations as needed to satisfy the participants that the task will be completed satisfactorily. To establish the terms, participants must determine how transactions and their data are represented on the blockchain, agree on the "if/when...then…" rules that govern those transactions, explore all possible exceptions, and define a framework for resolving disputes.
Then the smart contract can be programmed by a developer – although increasingly, organizations that use blockchain for business provide templates, web interfaces, and other online tools to simplify structuring smart contracts.

2.3 CONENSUS PROTOCOL

A consensus algorithm is a procedure through which all the peers of the blockchain network reach a common agreement about the present state of the distributed ledger.

2.4 REQUIRMENTS OF ELECTRONIC VOTING SYSTEM

- Only users with the right to vote should be able to cast a vote.
- Each voter should be able to vote only once.
- Votes should not be able to be modified or destroyed.
- Anyone should be able to independently verify that all votes have been correctly counted.
- Voting systems should be accessible by people with special needs and without requiring specific equipment or abilities.
- Voting systems should detect errors, faults and attacks and recover voting information to the point of failure.

2.5 BENEFITS OF USING BLOCKCHAIN

- They're immutable, which means a smart contract can never be changed and no one can tamper with or break a contract.
- They're distributed, which means that the outcome of the contract is validated by everyone in the network, just like any transaction on a blockchain. Distribution makes it impossible for an attacker to force control over the network
- Every vote cast is linked to the public key of the voter on the blockchain. By allowing each public key to cast only one vote.
- Every vote cast will be encrypted before being stored on the blockchain, making it impossible for anyone to know the content of that vote.
- If any malicious action is made on the blockchain, it will be detected by the system and considered invalid.

- As soon as some data is stored on the blockchain it can no longer be deleted.

Chapter 3

PROPOSED METHODOLOGY

The proposed system involves a client server architecture integrated with a block chain system. There are mainly five parts of the system.

Admin: Admin first takes important information about voters and enroll them into the system.

Fabric CA: The system creates a public key and password and sends them to voter via admin.

Voter: Voters then can access their account and cast votes and after the voting they can also check for confirmation if his/her vote is counted or not.

Encryption Server: Casted vote then goes to the encryption server after validating so that no one can temper with that content of the vote.

Blockchain Server: The encrypted file then sent to the blockchain server to store into the ledger

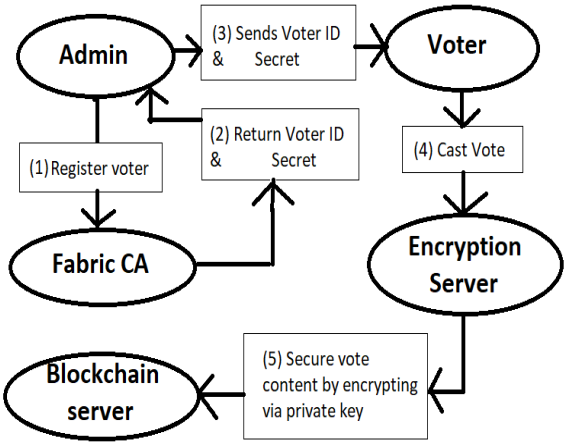


Figure 3.1: Client Server Architecture

We used Raspberry Pi and a fingerprint scanner to handle the data validation of a voter in the network. It scans thumbprint and Raspberry Pi processes the extracted data and sends the data to database in order to verify the voter.

Chapter 4
ARCHITECTURE

We will use Hyperledger Fabric for building our blockchain network .Hyperledger Fabric is a framework for the development of blockchain system. The reasons behind using hyperledger fabric are :
It has permissioned architecture which means as a developer you can choose who will be able to access your blockchain and on what level they get access. This is not possible in Ethereum or Bitcoin(other platforms for developing blockchain based system). We have different nodes with different responsibilities so we will need permissioned architecture. Also it supports pluggable consensus mechanism means we can impalement different consensus algorithm according to our system's need which doesn't allow Bitcoin or Ethereum. Then performance wise hyperledger is better and fast. Bitcoin has less that 7 transactions per second. But in hyperledger fabric, it has 1000 to 20000 transactions per second depending on the consensus algorithm we choose. And also it features, multilanguage smart contract support for languages like Go, Java, Javascript, typescript [1][3].

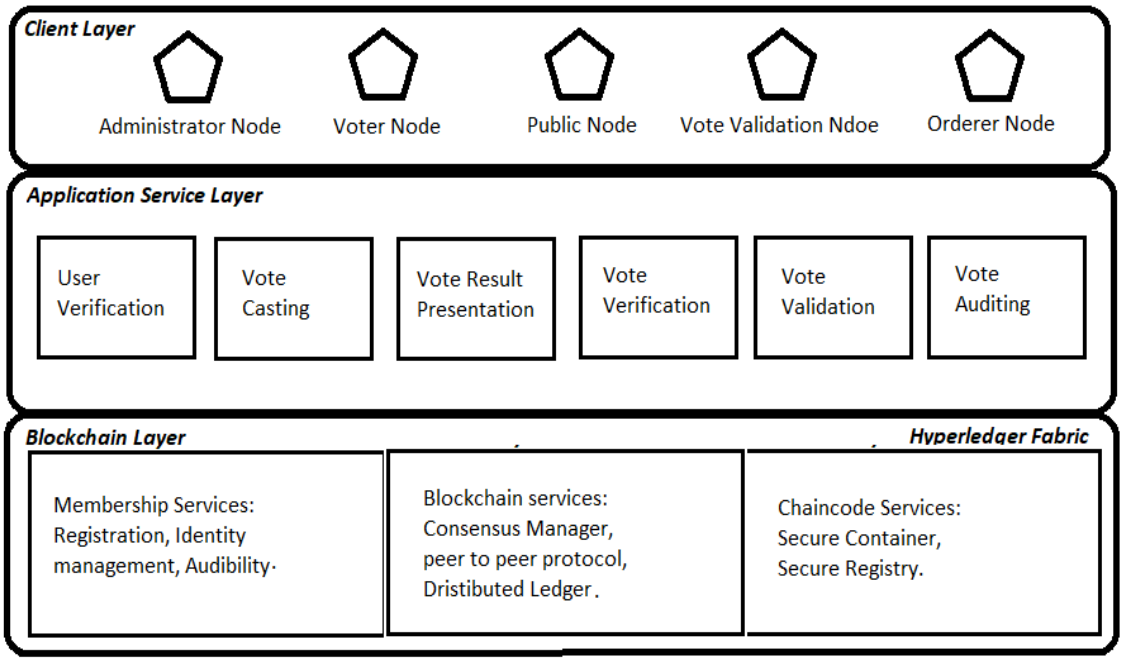


Figure 4.1: Different Layers of the System

Client layer: contains the various electronic devices and systems with which users interact with the blockchain e-voting system. These devices are the peer nodes of the e-voting blockchain that interact via smart contracts in the Hyperledger Fabric. The different types of peer nodes are assigned with different responsibilities.

Application Service Layer: consists of a set of services that are available in the e-voting system. The level of access control and the defined permissions level determines the type of services that a node can access in the blockchain.

Blockchain Layer: is composed of the Hyperledger Fabric. This layer contains the fabric membership service which indentifies the peers on the network, then the blockchain services such as consensus protocol and the blockchain database or the ledger and the chaincode services.

ROLES OF THE NODES

Administrator Node: These nodes are used to configure blockchain network channels, assign roles or responsibilities to the nodes of the blockchain, grant permissions, set the level of access control for specific nodes.

Voter Node: The primary purpose of these nodes are to cast votes.

Public Node: These are the nodes that enable view-only public access to transaction of the blockchain.

Vote Validation Node: These are responsible for vote validation.

Ordering Node: They validate the transactions and commit new blocks to the leadger.

LOGIC OF OUR SMART CONTRACT

Transaction input – from client App:

```
submitVote(electionName,voterID,candidateName);
```

these inputs go to the blocks.

```
candidateName.count++;
```

these counts helps tallying votes.

```
voterID.castBallot =True;
```

This lets voter vote for only one time.

Chapter 5

HARDWARE PART

In this part the voter will be registered and validated with finger print while participating in voting process. This will be helpful for identify the wrong voters and will be more secured [12].

5.1 Block Diagram

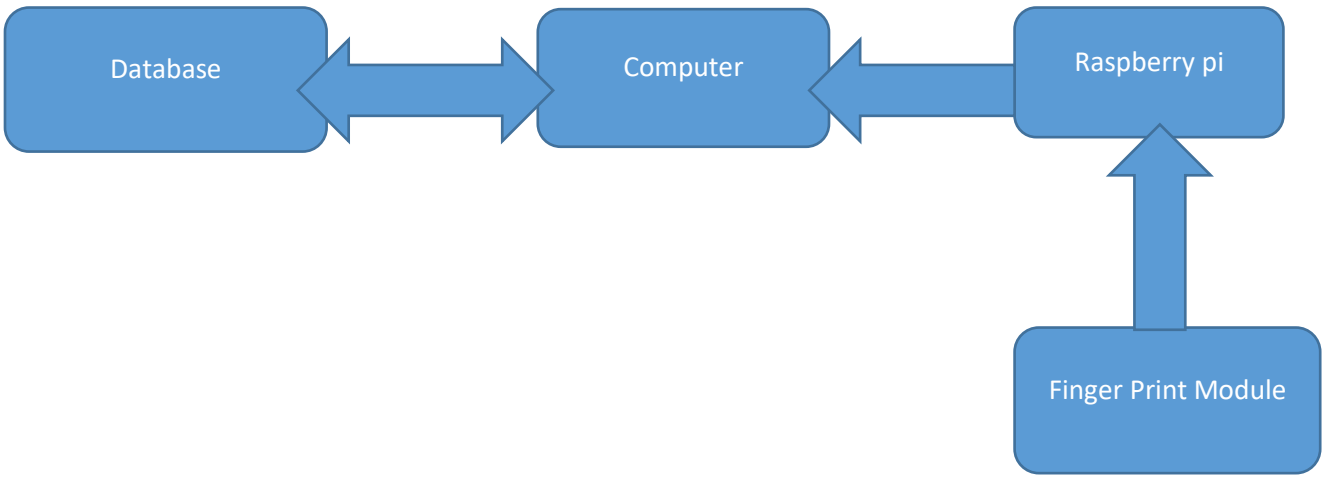


Figure 5.1: Block Diagram of Biometric Validation

In this diagram the database is connected to the computer. The computer can access the information through database. There is also a finger print module connected to the micro controller. Here we use Raspberry pi. This micro controller is connected to the computer.

5.2 Working Process

When the voter will put finger on the finger print module the computer will start process to match that print with the database. If the print matches with the valid entity, the system will allow the voter to see the voting page. After giving vote the voter will be a logged out of the system and will not be able to vote further as the database will save that voter id as voted. So, no voter will be able to vote more than once. No one will also able to give vote of other voter. Before that process the registration process must be completed well. The registration will also be in biometric registration with voter id. So, this is how this hardware part work with the database to validate the actual vote.

Chapter 6

IMPLEMENTATION SECTION

*S*ome implements of important functionality are given below:

Ballot:

```
'use
strict';


class Ballot {


  /**
   *
   * Ballot
   *
   * Constructor for a Ballot object. This is what the point of the application
is - create
   * ballots, have a voter fill them out, and then tally the ballots.
   *
   * @param items - an array of choices
   * @param election - what election you are making ballots for
   * @param voterId - the unique Id which corresponds to a registered voter
   * @returns - registrar object
   */
  constructor(ctx, items, election, voterId) {


    if (this.validateBallot(ctx, voterId)) {


      this.votableItems = items;
      this.election = election;
```

```
      this.voterId = voterId;
      this.ballotCast = false;
      this.ballotId = Math.random().toString(36).substring(2, 15) +
Math.random().toString(36).substring(2, 15);
      this.type = 'ballot';
      if (this.__isContract) {
        delete this.__isContract;
      }
      if (this.name) {
        delete this.name;
      }
      return this;


    } else {
      console.log('a ballot has already been created for this voter');
      throw new Error ('a ballot has already been created for this voter');
    }
  }


  /**
   *
   * validateBallot
   *
   * check to make sure a ballot has not been created for this
   * voter.
   *
   * @param voterId - the unique Id for a registered voter
   * @returns - yes if valid Voter, no if invalid
   */
  async validateBallot(ctx, voterId) {


    const buffer = await ctx.stub.getState(voterId);

    if (!!buffer && buffer.length > 0) {
      let voter = JSON.parse(buffer.toString());
      if (voter.ballotCreated) {
        console.log('ballot has already been created for this voter');
        return false;
      } else {
        return true;
```

```
      }
    } else {
      console.log('This ID is not registered to vote.');
      return false;
    }
  }
}
module.exports = Ballot;
```

Election

```
class
Election
{


  /**
   *
   * validateElection
   *
   * check for valid election, cross check with government. Don't want duplicate
   * elections.
   *
   * @param electionId - an array of choices
   * @returns - yes if valid Voter, no if invalid
   */
  async validateElection(electionId) {


    //registrarId error checking here, i.e. check if valid drivers License, or
state ID
    if (electionId) {
      return true;
    } else {
      return false;
    }
  }
  /**
   *
   * Election
   *
```

```
   * Constructor for an Election object. Specifies start and end date.
   *
   * @param items - an array of choices
   * @param election - what election you are making ballots for
   * @param voterId - the unique Id which corresponds to a registered voter
   * @returns - registrar object
   */
  constructor(name, country, year, startDate, endDate) {

    this.electionId = Math.random().toString(36).substring(2, 15) +
Math.random().toString(36).substring(2, 15);


    if (this.validateElection(this.electionId)) {


      //create the election object
      this.name = name;
      this.country = country;
      this.year = year;
      this.startDate = startDate;
      this.endDate = endDate;
      this.type = 'election';
      if (this.__isContract) {
        delete this.__isContract;
      }
      return this;


    } else {
      throw new Error('not a valid election!');
    }


  }


}
module.exports = Election;
```

Voter

```
'use
strict';


            class Voter {
              /**
               *
               * Voter
               *
               * Constructor for a Voter object. Voter has a voterId and registrar that the
               * voter is .
               *
               * @param items - an array of choices
               * @param election - what election you are making ballots for
               * @param voterId - the unique Id which corresponds to a registered voter
               * @returns - registrar object
               */
              constructor(voterId, registrarId, firstName, lastName) {


                if (this.validateVoter(voterId) && this.validateRegistrar(registrarId)) {


                  this.voterId = voterId;
                  this.registrarId = registrarId;
                  this.firstName = firstName;
                  this.lastName = lastName;
                  this.ballotCreated = false;
                  this.type = 'voter';
                  if (this.__isContract) {
                    delete this.__isContract;
                  }
                  if (this.name) {
                    delete this.name;
                  }
                  return this;


                } else if (!this.validateVoter(voterId)){
                  throw new Error('the voterId is not valid.');
                } else {
                  throw new Error('the registrarId is not valid.');
                }
```

```
  }


  /**
   *
   * validateVoter
   *
   * check for valid ID card - stateID or drivers License.
   *
   * @param voterId - an array of choices
   * @returns - yes if valid Voter, no if invalid
   */
  async validateVoter(voterId) {
    //VoterId error checking here, i.e. check if valid drivers License, or state
ID
    if (voterId) {
      return true;
    } else {
      return false;
    }
  }


  /**
   *
   * validateRegistrar
   *
   * check for valid registrarId, should be cross checked with government
   *
   * @param voterId - an array of choices
   * @returns - yes if valid Voter, no if invalid
   */
  async validateRegistrar(registrarId) {


    //registrarId error checking here, i.e. check if valid drivers License, or
state ID
    if (registrarId) {
      return true;
    } else {
      return false;
```

```
    }
  }

}
module.exports = Voter;
```

Voteable Item

```
'use
strict';

    class VotableItem {

      /**
     *
     * VotableItem
     *
     * Constructor for a VotableItem object. These will eventually be placed on
    the
     * ballot.
     *
     * @param votableId - the Id of the votableItem
     * @param description - the description of the votableItem
     * @param voterId - the unique Id which corresponds to a registered voter
     * @returns - registrar object
     */
    constructor(ctx, votableId, description) {

      this.votableId = votableId;
      this.description = description;
      this.count = 0;
      this.type = 'votableItem';
      if (this.__isContract) {
        delete this.__isContract;
      }
      return this;
```

```
  }
}
module.exports = VotableItem;
```

Chapter 7

CONSENSUS ALGORITHM

We are using Raft ordering service as a consensus algorithm [6]. Every time a transaction happends our orderer nodes have to have consensus about the transaction whether to write the transaction into the blockchain or not. 50% of the nodes has to agree to make a valid transaction happen. Our system has 5 ordering nodes , if two of them crashes still there will be no problem while validating transactions. And to attack a raft node the attacker must has to have the private key of the raft node which is impossible for the attacker to know unless the person who has access to the ralf node is corrupted. There are also some other ordering services such as “kafka” ,”solo” etc but we are using “Raft ” because it makes our system faster and also securethan others.

To understand Raft, we shall first look at the problem which the Raft protocol tries to solve that is achieving consensus. Consensus means multiple servers agreeing on the same information. Lets describe it with visuals.

So, lets first define the process used when a client interacts with a server to clarify the process.

Process: The client sends a message to the server and the server responds back with a reply.

A consensus protocol tolerating system must have the following features:

1. **Validity:** If a process decides (read/write) a value, then it must have been proposed by some other correct process.
2. **Agreement:** Every correct process must agree on the same value.
3. **Termination:** Every correct process must terminate after a finite number of steps.
4. **Integrity:** If all correct processes decide on the same value, then any process has the said value.

Now, there can be two types of systems assuming only one client (for the sake of understand) :

1. **Single System Server:** The client interact with a system having only one server with no backup. There is no problem for achieving consensus in such a system.

Figure 7.1: Single System Server

2. **Multiple Server system :** The client interacts a system having a multiple servers such system can be of two types.

   Symmetric: Any of the multiple servers can respond to the client and all other servers are supposed to sync up with the server that responds to the client's request.
   Asymmetric: Only the elected leader server can respond to the client. All other server then sync up the leader server.

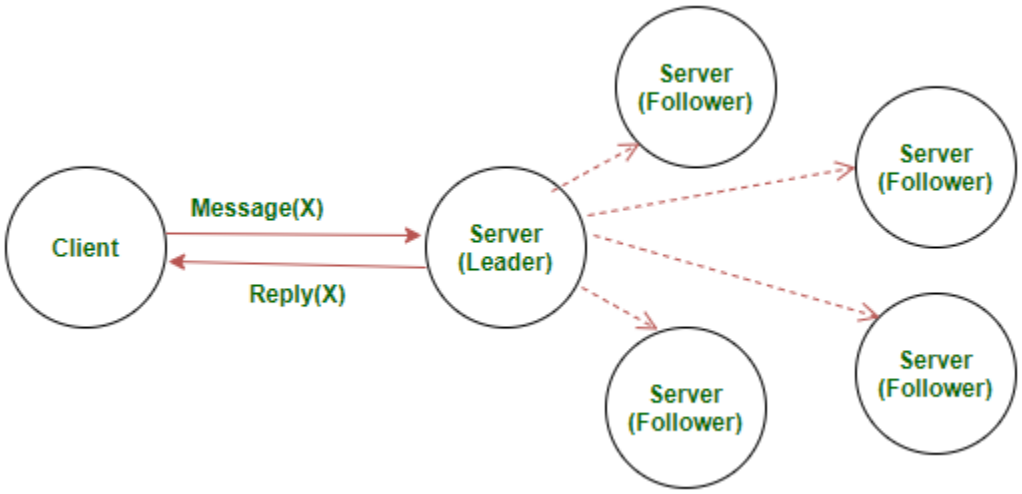


Figure 7.2: Multiple server labelled raft visual

Chapter 8

SECURITY ISSUES

***A*uthentication Vulnerability:** Each individual is identified and authenticated by our “Fabric-Certificate-Authority” with public and private key. Every voter need to sign after giving their vote with their private key to validate the vote.

Prone to Sybill Attack: It is known against centralized system, where an individual creates a large number of nodes to disrupt the network. But, since it is a private network and no one has the ability to do so, our system is prone to sybill attack.

Immutability: A record with date and time stamps of all transactions in multiple copies on the ledger are maintained to avoid any doubt. And these blocks can’t be rewritten, edited or deleted**.**

Privacy of the people: Since every user is identified by a public key and the stored vote is encrypted, so they can vote anonymously.

Transparency: Since the blockchain is transparent to every node of the network, everyone could confirm that the number of votes casted and counted are the same.

Chapter 9

THREAT MODEL

9.1 What is Threat Model?

Threat modeling in cybersecurity is a way of identifying, listing, prioritizing, and mitigating potential threats in order to protect systems and data [7]. The idea of anticipating threats is as old as the world itself. But systematic threat modeling is a relatively new approach. In the realm of software security, threat models describe specific threats to the systems and data, ranging from technology-specific threats such as web vulnerability exploits to broader risks such as unauthorized system access or insecure physical data storage. You can perform threat analysis and modeling at any level.

Threat modeling can be considered the next practical step after a broader risk assessment. Once you have identified security risks that can affect your operations and business processes, security experts (in this case the threat modelers) define specific threat models for each risk factor. A threat model is simply a set of parameters that define a threat, such as the underlying risk factor, identified threat actors, potential attack vectors, business impact and remedies

9.2 Defining Threat Models for Web Application Development

Web application rely on and interact with many other systems, resources and data stores. An effective web application threat model should take the big picture into account, because as always in security – an attacker only needs one gap to get through. This is why you can't build a threat model that only focuses only on one aspect.

A real-life attacker has the whole attack surface to play with: the web server, the network infrastructure, data storage, external cloud services, and much more. In fact, it doesn't even have to be an external attacker – weak passwords and the actions of rogue admins can also pose critical threats, even if they have nothing to do with application development. This is why it's important to involve a wide variety of stakeholders in the threat modeling process.

Threat modeling can be done by anyone and at any stage of development. The method you use depends on your needs and capabilities, but here are 5 basic steps as an example [7]:

1. **Define security requirements** so you know what to protect from threats and **what these threats could target.**
2. **Map out the application structure** to get an idea of all the data flows and actors involved. A data flow diagram is the usual way of doing this.
3. **Identify threats and threat categories** that could pose a risk to your application based on defined security criteria. This is the main threat identification and analysis step.
4. **Mitigate threats** to ensure that identified risks can't translate into real-life attacks. After all, the purpose of the exercise is it improve security, not just paint a picture.
5. **Check that threats have been mitigated** to make doubly sure that everything specified in your threat model is now a purely theoretical risk.

To be effective, a threat model should be an integral part of the software design and development process. Ideally, the threat model for a web application should be defined and maintained from the very start of development, but it can also be added to an existing application. Just like test suites, the threat model should reside close to the code and be rigorously updated every time the security conditions change. For example, if an application switches from using a local database to cloud storage, this may introduce new threats and require changes to the threat model.

9.3 Why You Should Use Threat Modeling

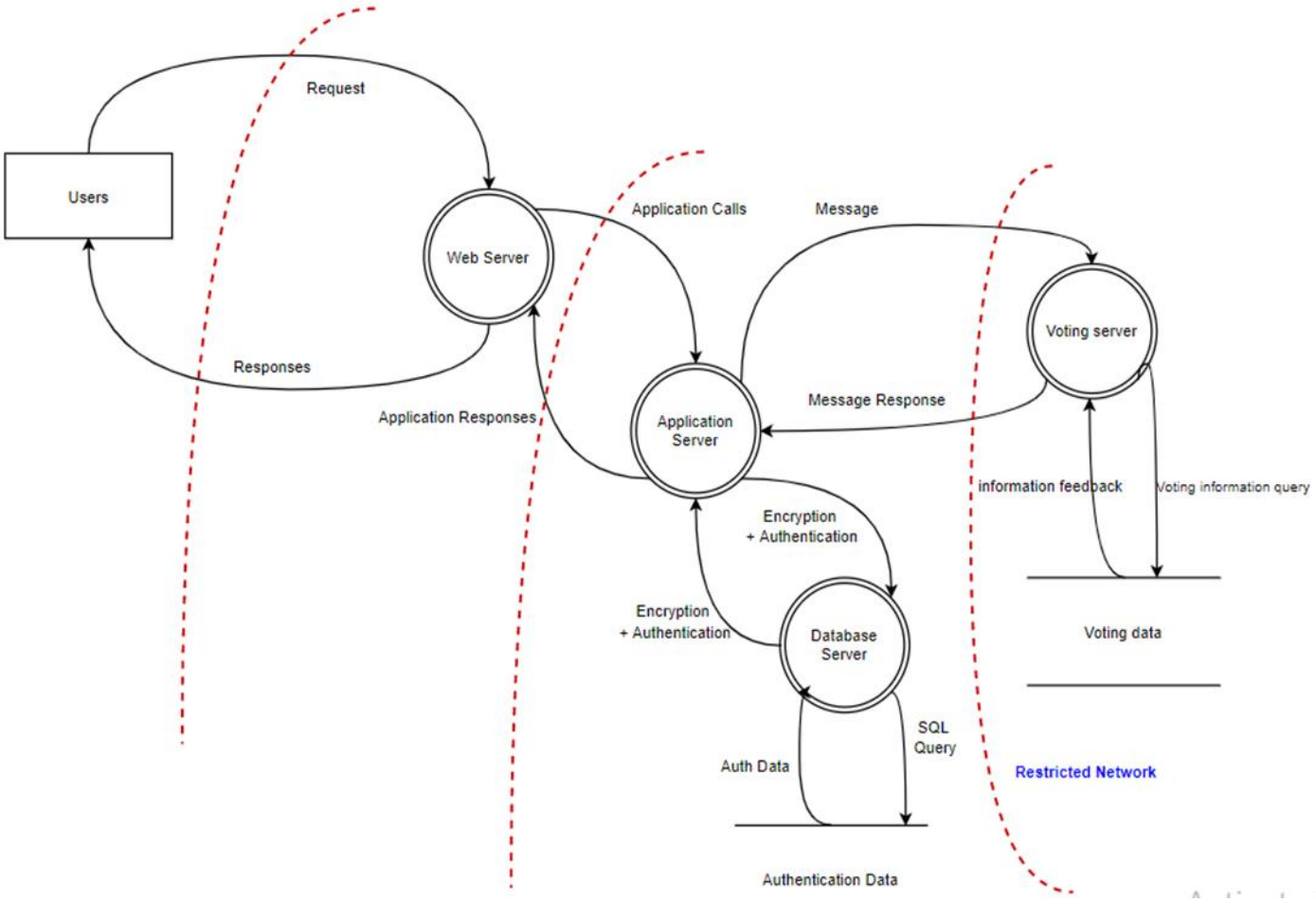
When dealing with IT security, each specialist will tend to focus on their area of expertise, so for a web developer, security will mean web vulnerabilities, a network engineer will think mainly about network security, and for a workplace admin, security will be all about malware, phishing, and unauthorized private devices. The big advantage of threat modeling is that it provides a high-level view of potential security issues by focusing on threats, not weaknesses. For example, if your security requirement is to protect confidential data in company systems, your threat model will need to include everything that could lead to a compromise of this information [7].

Making sense of security realities is always tricky, especially under time pressure, so having a solid threat model in place can be a great help for security decision-making. By looking at a valid and up-to-date model, you can get a good picture of the real threat environment and make your plans and decisions accordingly. This is especially important in DevOps and other agile development environments where you don't have the time or resources for a full security review every time you plan a sprint or release.

Simply put, threat modeling is a more formal way of thinking about what can go wrong in your systems and applications. By focusing on threats rather than weaknesses, you can zoom out of your technical niche and get a wider view of your web application security posture.

9.4 Threat Model of the System

We used this particular template for threat model [10].



Explanation of the Threat Model:

1. The model shows the vulnerability of the website.
2. In user level the website may attacked with illegal users.
3. Server can be buffered. So there may have problem in server level.
4. In database level the data should be secured and authentic.
5. The application should be up to the situation and user friendly.
6. Good observation from the system should be ensure to protect information.

These are the threats for our website.

Chapter 10

FRAMEWORK

10.1 What is Framework?

Frameworks are powerful tools that can make a web developer's job easier. They provide a standardized set of design and development conventions that can be applied and modified for your website [11].

10.2 Why Frameworks are Useful?

Essentially the allure of these is the amount of time that is saved and the resulting efficiency in getting a project rolled out faster because there's a lot less of the initial work to be done.

Frameworks take care of tasks like default browser settings, file structures, and layout templates for your website. This automation generates a uniform design for page elements like text, tables, forms, and buttons, amongst other things. Even complex navigation menus become standardized with frameworks as it can be consistently applied across your website with ease [11].

10.3 The Framework of Our System (Hyperledger Fabric)

We choose Hyperledger Fabric Framework for our system.

Hyperledger Fabric is an open source enterprise-grade permissioned distributed ledger technology (DLT) platform, designed for use in enterprise contexts, that delivers some key differentiating capabilities over other popular distributed ledger or blockchain platforms [9].

One key point of differentiation is that Hyperledger was established under the Linux Foundation, which itself has a long and very successful history of nurturing open source projects under open governance that grow strong sustaining communities and thriving ecosystems. Hyperledger is governed by a diverse technical steering committee, and the Hyperledger Fabric project by a diverse set of maintainers from multiple organizations. It has a development community that has grown to over 35 organizations and nearly 200 developers since its earliest commits.

Fabric has a highly modular and configurable architecture, enabling innovation, versatility and optimization for a broad range of industry use cases including banking, finance, insurance, healthcare, human resources, supply chain and even digital music delivery.

Fabric is the first distributed ledger platform to support smart contracts authored in general-purpose programming languages such as Java, Go and Node.js, rather than constrained domain-specific languages (DSL). This means that most enterprises already have the skill set needed to develop smart contracts, and no additional training to learn a new language or DSL is needed.

The Fabric platform is also permissioned, meaning that, unlike with a public permissionless network, the participants are known to each other, rather than anonymous and therefore fully untrusted. This means that while the participants may not fully trust one another (they may, for example, be competitors in the same industry), a network can be operated under a governance model that is built off of what trust does exist between participants, such as a legal agreement or framework for handling disputes.

One of the most important of the platform's differentiators is its support for pluggable consensus protocols that enable the platform to be more effectively customized to fit particular use cases and trust models. For instance, when deployed within a single enterprise, or operated by a trusted authority, fully byzantine fault tolerant consensus might be considered unnecessary and an excessive drag on performance and throughput. In situations such as that, a crash fault-tolerant (CFT) consensus protocol might be more than adequate whereas, in a multi-party, decentralized use case, a more traditional byzantine fault tolerant (BFT) consensus protocol might be required.

Fabric can leverage consensus protocols that do not require a native cryptocurrency to incent costly mining or to fuel smart contract execution. Avoidance of a cryptocurrency reduces some significant risk/attack vectors, and absence of cryptographic mining operations means that the platform can be deployed with roughly the same operational cost as any other distributed system.

The combination of these differentiating design features makes Fabric one of the better performing platforms available today both in terms of transaction processing and transaction confirmation latency, and it enables privacy and confidentiality of transactions and the smart contracts (what Fabric calls "chaincode") that implement them.

10.4 Permissioned vs Permissionless Blockchain

In a permissionless blockchain, virtually anyone can participate, and every participant is anonymous. In such a context, there can be no trust other than that the state of the blockchain, prior to a certain depth, is immutable. In order to mitigate this absence of trust, permissionless blockchains typically employ a "mined" native cryptocurrency or transaction fees to provide economic incentive to offset the extraordinary costs of participating in a form of byzantine fault tolerant consensus based on "proof of work" (PoW) [9].

Permissioned blockchains, on the other hand, operate a blockchain amongst a set of known, identified and often vetted participants operating under a governance model that yields a certain degree of trust. A permissioned blockchain provides a way to secure the interactions among a group of entities that have a common goal but which may not fully trust each other. By relying on the identities of the participants, a permissioned blockchain can use more traditional crash fault tolerant (CFT) or byzantine fault tolerant (BFT) consensus protocols that do not require costly mining.

Additionally, in such a permissioned context, the risk of a participant intentionally introducing malicious code through a smart contract is diminished. First, the participants are known to one another and all actions, whether submitting application transactions, modifying the configuration of the network or deploying a smart contract are recorded on the blockchain following an endorsement policy that was established for the network and relevant transaction type. Rather than being completely anonymous, the guilty party can be easily identified and the incident handled in accordance with the terms of the governance model.

Any serious evaluation of blockchain platforms should include Hyperledger Fabric in its short list.

Combined, the differentiating capabilities of Fabric make it a highly scalable system for permissioned blockchains supporting flexible trust assumptions that enable the platform to support a wide range of industry use cases ranging from government, to finance, to supply-chain logistics, to healthcare and so much more.

Hyperledger Fabric is the most active of the Hyperledger projects. The community building around the platform is growing steadily, and the innovation delivered with each successive release far out-paces any of the other enterprise blockchain platforms.

Chapter 11

DISCUSSION

In this chapter we briefly explain the complications and limitations related to our findings. How we can improve the system is also explained in this section.

11.1 Complications and Limitations

In Blockchain based system, it is very important to select the right framework and algorithm. Because there is a relation with framework and cryptocurrency. There are some frameworks which work in term of cryptocurrency (like **Bitcoin**). When a transection take places there need to exchange cryptocurrency to complete the process. But sometimes we need transection without paying cryptocurrency. Like voting someone, selecting something etc. So, for these types of system we need a process that works without cryptocurrency. This flexibility also offers some frameworks (like Hyperledger Fabric). So, according to Hyperledger fabric we don't need pay while voting someone. So selecting framework is complicated issue. There is also an important issue. That is algorithm. If consensus algorithm selecting is wrong the whole process will be wrong. Related to our work to avoid crash fault tolerant and some other problem we faced some trouble to select the perfect algorithm.

There are also some limitations in our system. We didn't found a process to faster the voting system than the existing processes. We tried to secure the system. But we couldn't manage to make faster than those existing systems. There should be some logical operation in our chaincode to make faster the voting process. We hope that the limitations will be solved when we will work with it in future and also set some time limits.

11.2 Future Works

In future, we will work with this system where will try to make a time limitation of each vote. We will try to make a faster system. We will also try to use some APIs to make the system better looking and user friendly as the Hyperledger Fabric allows us to use many APIs.

Chapter 12

CONCLUSION

E **Voting System** created a new era in voting system. But it couldn't earn the faith of people for some **security issues**. Blockchain based e voting system will ensure the security issues. It is less time and energy consuming. It is also environment friendly and cost effective, because no ballot paper and ballot box is needed. It is a trust worthy system. For any democratic country this e-voting System will applicable. This system illustrates the impact of technology on democracy. Blockchain is a new technology. This can be used to make the website more secured. People should know about this new technology. As voting system is always related to the security, to built voting website Blockchain technology is always suitable. There is a relation between **Blckchain** and **Kryptography**. Hyperledger Fabric is a type of framework where using of kryptography is so flexible. For this quality **Blockchain Based Hyperledger Fabric** is very much usable technology. In any kinds of website become secured by using this technology. So, hopefully people will get benefit from this website and technology.

References

1.*A Design for Blockchain-Based Digital Voting System.* **Al-Rawy, Mahmoud and Elci, Atilla. 2018 .** 2018 , The 2018 International Conference on Digital Science , pp. 397-407 .

2.*Blockchain Based Secured E-voting by Using the Assistance of Smart Contract.* **Sadia, Kazi, et al. October 2019.** October 2019.

3.Blockchain-Based Electronic Voting Protocol. **Wei, Clement Chan Zheng and Wen, Chuah Chai. 2018.** 2018, Vol. 2, p. 6.

4.Blockchain-Based Electronic Voting System for Elections in Turkey. **Rumeysa Bulut, et al. September 2019.** September 2019.

5.Blockchain-Based E-Voting System. **Friðrik Þ. Hjálmarsson, et al. July 2018.** July 2018, Researchgate, p. 4.

6. Raft Consensus algorithm: h*ttps://www.geeksforgeeks.org/raft-consensus-algorithm/*

7**. Threat Model** : https://www.netsparker.com/blog/web-security/threat-modeling/

8**. Smart Contract**: https://www.ibm.com/topics/smart-contracts

9. **Framework:** https://hyperledger-fabric.readthedocs.io/en/release-2.2/whatis.html

10. **Threat Model Template** : https://online.visual-paradigm.com/diagrams/templates/threat-model-diagram/website-threat-modeling/

11. **What is framework**: https://careerfoundry.com/en/blog/web-development/responsiveness-with-a-front-end-framework/#:~:text=Frameworks%20are%20powerful%20tools%20that%20can%20make%20a,can%20be%20applied%20and%20modified%20for%20your%20website.

12. **Hardware Part**: https://www.academia.edu/36842236/ONLINE_VOTING_SYSTEM_USING_FINGERPRINT_SCANNER