



# MOTILAL NEHRU NATIONAL INSTITUTE OF TECHNOLOGY ALLAHABAD

PROJECT TOPIC : **SECURE E-VOTING USING BLOCKCHAIN**

MCA-4 Department of Computer Science & Engineering

Project Instructor : Dr. J Sathish Kumar

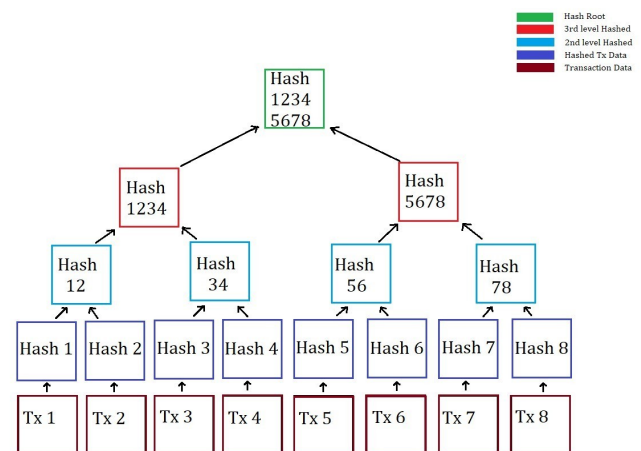
Team Members : Harman Singh \* Rakhi \* Shiv Lal Jat

## 1.INTRODUCTION

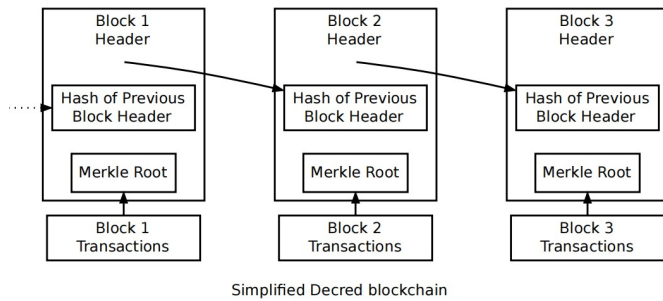
**1.1 Overview of Project:** Democratic voting is a crucial and serious event in any country. The most common way in which a country votes is through a paper based system, but is it not time to bring voting into the 21st century of modern technology? Digital voting is the use of electronic devices, such as voting machines or an internet browser, to cast votes. These are sometimes referred to as e-voting when voting using a machine in a polling station, and i-voting when using a web browser. Security of digital voting is always the biggest concern when considering to implement a digital voting system. With Such monumental decisions at stake, there can be no doubt about the system's ability to secure data and defend against potential attacks. One way the security issues can be potentially solved is through the technology of blockchains.

**1.2 Overview of Technology:** Blockchain technology was first used within Bitcoin and is a public ledger of all transactions. A blockchain stores these transactions in a block, the block eventually becomes completed as more transactions are carried out. Once complete it is then added in a linear, chronological order to the blockchain. The initial block in a blockchain is known as the 'Genesis block' or 'Block 0'. The genesis block is usually hard coded into the software; it is special in that it doesn't contain a reference to a previous block. Once the genesis block has been initialised 'Block 1' is created and when complete is attached to

the genesis block. Each block has a transaction data part, copies of each transaction are hashed, and then the hashes are paired and hashed again, this continues until a single hash remains; also known as a merkle root.



The block header is where the merkle root is stored. To ensure that a transaction cannot be modified each block also keeps a record of the previous blocks header, this means to change data you would have to Figure 1: Hash table 5 modify the block that records the transaction as well as all following blocks, as seen in Figure 2. A blockchain is designed to be accessed across a peer-to-peer network, each node/peer then communicates with other nodes for block and transaction exchange. Once connected to the network, peers start sending messages about other peers on the network,



Simplified Decred blockchain

this creates a decentralised method of peer discovery.

The purpose of the nodes within the network is to validate unconfirmed transactions and recently mined blocks, before a new node can start to do this it first has to carry out an initial block download. The initial block download makes the new node download and validate all blocks from block 1 to the most current blockchain, once this is done the node is considered synchronised.

## 2.SYSTEM REQUIREMENT AND SPECIFICATION





### Hardware Requirements (Minimum)

System:	64-bit versions of Microsoft Windows 10, 8, 7
HardDisk:	1.5 GB hard disk space + at least 1 GB for caches
Monitor:	1024x768 minimum screen resolution
RAM:	4 GB RAM minimum, 8 GB RAM recommended

### Software Requirements(Minimum)

Operating System	Linux, windows 7, 8 or higher
------------------	-------------------------------

### 2.1 Software Tools Required :

			
GANACH E - CLI	WEB3.JS	NODE.JS	SOLIDITY
Part of truffle suite for Ethereum. * Simulating full client behaviour. * Makes developing Ethereum applications fast and easy.	Collection of libraries. * Allows interaction with local or remote Ethereum code. * Uses HTTP, IPC or Websocket for interaction.	NodeJS is an open-source, cross-platform javascript runtime environment. * Server side scripting. * Optimizes throughput and scalability of web app.	Object oriented language to write smart contracts. * Implements smart contracts.

## 3.IMPLEMENTATION

### MODULE DESCRIPTION

#### 3.1 BLOCKCHAIN AS A SERVICE FOR E-VOTING

Here, we consider existing electronic voting systems, blockchain based and non-blockchain-based, and evaluate their respective feasibility for implementing a national e-voting system. Based on this, we devised a blockchain-based electronic voting system, optimizing for the requirements and considerations identified.

In the following subsection, we start by identifying the roles and components for implementing an e-voting smart contract then, we evaluate different blockchain frameworks that can be used to realize and deploy the election smart contracts. In the last subsection, we will discuss the design and architecture of the proposed system.

#### 3.2 Election as a smart contract :

Defining a smart contract includes identifying the roles that are involved in the agreement (the election agreement in our case) and the different components and transactions in the agreement process.

We start by explaining the election roles followed by the election process.

##### 3.2.1 Election Roles :

Elections in our proposal enable participation of individuals or institutions in the following roles. Where multiple institutions and individuals can be enrolled to the same role.

##### i. Election administrators:

Manage the lifecycle of an election. Multiple trusted institutions and companies are enrolled with this role.

The election administrators specify the election type and create the aforementioned election, configure ballots, register voters, decide the lifetime of the election and assign permissioned nodes.

##### ii. Voters:

For elections to which they are eligible for, voters can authenticate themselves, load election ballots, cast their vote and verify their vote after an election is over.

Voters can be rewarded for voting with tokens when they cast their vote in an election in the near future, which could be integrated with a smart city project.

**iii. Nodes:** When the election administrators create an election, each ballot smart contract, representing each voting district, are deployed onto the blockchain. When the ballot smart contracts are created, each of the corresponding district nodes are given permission to interact with their corresponding ballot smart contract.

When an individual voter casts his vote from his corresponding smart contract, the vote data is verified by all of the corresponding district nodes and every vote they agree on are appended onto the blockchain when block time has been reached.

**3.2.2. Election Process :** In our work, each election process is

represented by a set of smart contracts, which are instantiated on the blockchain by the election administrators. A smart contract is defined for each of the voting districts of the election so multiple smart contracts are involved in an election. For each voter with its corresponding voting district location, defined in the voters registration phase, the smart contract with the corresponding

location will be prompted to the voter after the user authenticates himself when voting.

The following are the main activities in the election process:

**(i) Election creation :**

Election administrators create election ballots using a decentralized app(dApp). This decentralized app interacts with an election creation smart contract, in which the administrator defines a list of candidates and voting districts.

This smart contract creates a set of ballot smart contracts and deploys them onto the blockchain, with a list of the candidates, for each voting district, where each voting district is a parameter in each ballot smart contract.

When the election is created, each corresponding district node is given permission to interact with his corresponding ballot smart contract.

**(ii) Voter registration :**

The registration of voter phase is conducted by the election administrators.

When an election is created the election administrators must define a deterministic list of eligible voters. This requires a component for a government identity verification service to securely authenticate and authorize eligible individuals. Using such verification services, each of the eligible voter should have an electronic ID and PIN number and information on what voting district the voter is located in. For each eligible voter, a corresponding wallet would be generated for the voter.

The wallet generated for each individual voter should be unique for each election the voter is eligible for and a NIZKP could be integrated to generate such wallet so that the system itself does not know which wallet matches an individual voter.

**(iii) Vote transaction :**

When an individual votes at a voting district, the voter interacts with a ballot smart contract with the same voting district as is defined for any individual voter.

This smart contract interacts with the blockchain via the corresponding district node, which appends the vote to the blockchain if consensus is reached between the majority of the corresponding district nodes.

**(iv) Tallying results :**

The tallying of the election is done on the fly in the smart contracts.

Each ballot smart contract does their own tally for their corresponding location in its own storage. When an election is over, the final result for each smart contract is published.

**(v) Verifying vote :** As was mentioned earlier, each individual voter receives the transaction ID of his vote. Each individual voter can go to his government official and present their transaction ID after authenticating himself using his electronic ID and its corresponding PIN.

The government official, utilizing district node access to the blockchain, uses the blockchain explorer to locate the transaction with the corresponding transaction ID on the

blockchain. The voter can therefore see his vote on the blockchain, verifying that it was counted and counted correctly.

### 3.3 Evaluating Blockchain as a service for E-Voting

The three blockchain frameworks that we consider for implementing and deploying our election smart contracts.

Those

are Exonum, Quorum and Geth.

#### 3.3.1 Ethereum :

Ethereum is an open-source blockchain-based platform used to create and share business, financial services, and entertainment applications.

- Ethereum users pay fees to use dapps. The fees are called "gas" because they vary depending on the amount of computational power required.

- Ethereum has its own associated cryptocurrency, Ether or ETH.

- Its cryptocurrency is now second only to Bitcoin in market value.

#### 3.3.2 Geth :

Go-Ethereum or Geth is one of three original implementations of the Ethereum protocol and it runs smart contract applications exactly as programmed without possibility of downtime, censorship, fraud or third party interference.

This framework supports development beyond the Geth protocol, and is the most developer-friendly framework of the frameworks we evaluated.

The transaction per second(transaction rate) is dependent on whether the blockchain is implemented as a public or private network. Because of these capabilities, Geth was the framework we chose to base our work on, any similar blockchain

framework with the same capabilities as Geth should be considered for such systems.

## 4.DESIGN AND IMPLEMENTATION

### 4.1 Aadhaar Card Validation

The voter tries to login themselves to cast a vote. In this phase voter first logs in using password. After successful login, to cast their vote voter has to authenticate themselves. For real-time authentication OTP verification is used for enhanced security.

#### Validating Aadhar using Verhoeff Algorithm

The Verhoeff algorithm is a checksum formula for error detection developed by the Dutch mathematician Jacobus Verhoeff and was first published in 1969.

It was the first decimal check digit algorithm which detects all single-digit errors, and all transposition errors involving two adjacent digits, which was at the time thought impossible with such a code. This algorithm has been used in application to verify aadhaar number.

```

68 function validate(aadharNumber) {
69   let c = 0;
70   let invertedArray = aadharNumber.split('').map(Number).reverse();
71
72   invertedArray.forEach((val, i) => {
73     c = d[c][p[(i % 8)][val]];
74   })
75   if(c === 0) {
76     if(localStorage.getItem(aadharNumber) === "1") {
77       console.log("Already Voted!");
78       alert("Already Voted");
79       return false;
80     }
81     else {
82       localStorage.setItem(aadharNumber, "1");
83       return true;
84     }
85   };
86   if(!(c === 0)) {
87     alert("Invalid Aadhaar Number");
88     return false;
89   }
90   return true;
91 }
92

```

## 4.2 Generation and OTP Verification

**OTP generation using firebase** We have used Firebase Authentication to sign in a user by sending an SMS message to the user's phone. The user signs in using a one-time code contained in the SMS message.

The easiest way to add phone number sign-in to your app is to use FirebaseUI, which includes a drop-in sign-in widget that implements sign-in flows for phone number sign-in, as well as password-based and federated sign-in.

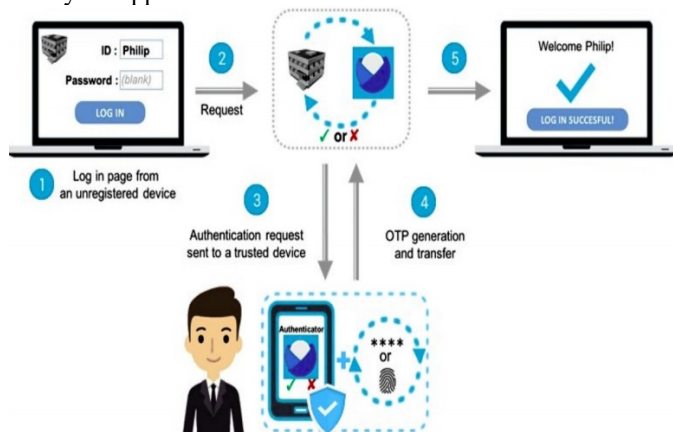
This document describes how to implement a phone number sign-in flow using the Firebase SDK.

## 4.3 APP VERIFICATION

To use phone number authentication, Firebase must be able to verify that phone number sign-in requests are coming from your app.

There are two ways Firebase Authentication accomplishes this: reCAPTCHA verification: In the event that SafetyNet cannot be used, such as when the user does not have Google Play Services support, or when testing your app on an emulator, Firebase Authentication uses a reCAPTCHA verification to complete the phone sign-in flow. The reCAPTCHA challenge can often be completed without the user having to solve anything.

Please note that this flow requires that a SHA-1 is associated with your application.



## 4.4 Displaying Candidate list

Sets the ballot smart contract to the address of the wallet which created the election, the voting district of the smart contract to the district which the ElectionCreation contract provided and then proceeds to fill the Candidates struct with the list of candidates provided and the number of votes for each candidate to 0.

The constructor also stores the time of the creation of the contract along with the time when the contract is to expire. Candidate is also one of the roles as defined by our smart contract.

A candidate is a person who stands a chance to get elected after successful completion of the elections. In our smart contract we have initialized a map which has the names of all the candidates that are taking part in the election.

```

bytes32[] public candidateList;

/* This is the constructor which will be called once when you
   deploy the contract to the blockchain. When we deploy the contract,
   we will pass an array of candidates who will be contesting in the election*/
function Voting(bytes32[] candidateNames) {
  candidateList = candidateNames;
}

```

## 4.5 Allowing verified voter to vote for candidate of his choice

### Check for a valid candidate

This function only allows valid candidate to vote. It checks whether the candidate who is trying to vote is present in the list of valid candidates or not. function validCandidate(bytes32 candidate) returns (bool)

```

function validCandidate(bytes32 candidate) returns (bool) {
  for(uint i = 0; i < candidateList.length; i++) {
    if (candidateList[i] == candidate) {
      return true;
    }
  }
  return false;
}

```

## 4.6 VoteForCandidate:

This function allows voters to vote. The requirement for a voter to vote, is that the mapping of the address of the voter is set to its default, false. If that is the case, the function guarantees that the election time limit has not been reached. If both requirements are satisfied, the contract retrieves the index of which candidate was voted for and increases his vote count by 1 and sets the mapping to true, so that the voter can never vote again in this particular election.

```

// This function increments the vote count for the specified candidate. This
// is equivalent to casting a vote
function voteForCandidate(bytes32 candidate) {
  if (validCandidate(candidate) == false)
    throw;
  votesReceived[candidate] += 1;
}

```

## 4.7 Secure logout from the application and transaction hash generation



The candidate list has the names of all the participating candidates with a button alongside their names. The button has the functionality of incrementing the vote count of the corresponding candidate by 1. The `voteForCandidate()` function as defined in the earlier section is called. The following function is used for secure logout:

```
function disable() {
    $('#vote1').addClass( "disabled" );
    $('#vote2').addClass( "disabled" );
    $('#vote3').addClass( "disabled" );
    $('#vote4').addClass( "disabled" );
    document.cookie = "show=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC";
    document.cookie = "aadhaar=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC";
    window.location = '/app';
}
```

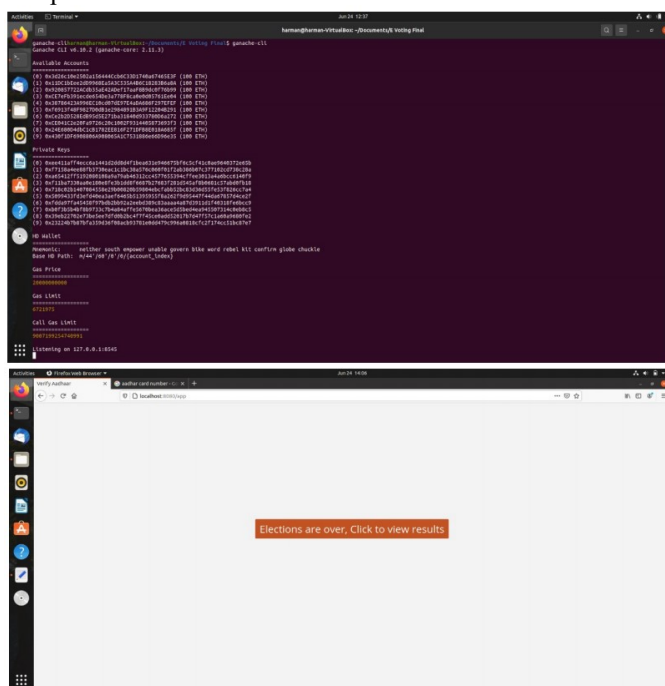
## 4.8 Result generation after polling has reached its completion time

### Result phase:

Result phase begins only once our time bound election has been completed. The processing and tallying of votes is done in results phase.

Results are generated and displayed on the website in tabular form. Users can verify their votes using their own public key.

This provides transparency to the voting system. There is a result button which will take you to the result page. The results can be displayed only once the entire election has been completed.



## 5.CONCLUSION

We have introduced a unique, blockchain-based electronic voting system that utilizes smart contracts to enable secure and cost-efficient election while guaranteeing voters privacy. By comparison to previous work, we have shown that the blockchain technology offers a new possibility for democratic countries to advance from the pen and paper election scheme, to a more cost and time efficient election scheme, while increasing the security measures of the today's scheme and offer new possibilities of transparency.

E-voting is still a controversial topic within both political and scientific circles. Despite the existence of a few very good examples, most of which are still in use; many more attempts have either failed to provide the security and privacy features of a traditional election or have serious usability and scalability issues.

On the contrary, blockchain-based e-voting solutions, including the one we have implemented using the smart contracts and the Ethereum network, address (or may address with relevant modifications) almost all of the security concerns, like privacy of voters, integrity, verification and non-repudiation of votes, and transparency of counting. Yet, there are also some properties that cannot be addressed solely using the blockchain,

for example authentication of voters (on the personal level, not on the account level) requires additional mechanisms to be integrated, such as use of biometric factors.