USING BLOCKCHAIN FOR VOTING

By

ANAY DHARMENDRA LOYA (161080062)

(An autonomous Institute)

Department of Computer Engineering

and Information Technology

Veermata Jijabai Technological Institute – 400 019

March 2018

Abstract

Blockchain is offering new opportunities to develop new types of digital services. While research on the topic is still emerging, it has mostly focused on the technical and legal issues instead of taking advantage of this novel concept and creating advanced digital services. In this project report, we are going to leverage the open source Blockchain technology to propose a design for a new electronic voting system that could be used in local or national elections. The Blockchain-based system will be secure, reliable, and anonymous, and will help increase the number of voters as well as the trust of people.

Table of Contents

1. INTRODCTION

1.1 Democratic Voting

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. An e-Voting system has to have heightened security in order make sure it is available to voters but protected against outside influences changing votes from being cast, or keep a voter's ballot from being tampered with.

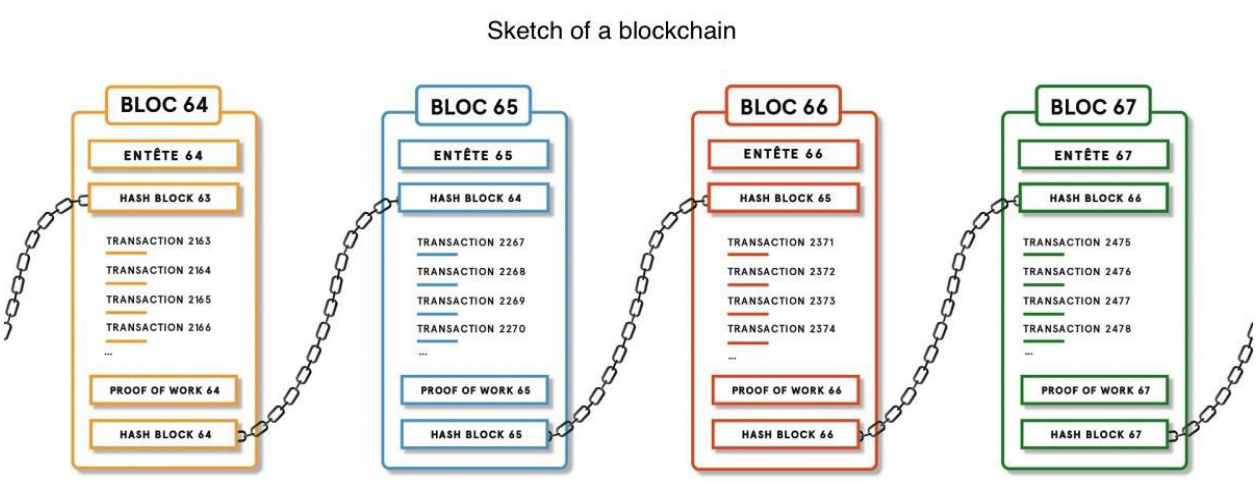
One way the security issues can be potentially solved is through the technology of blockchains.

1.2 What is a Blockchain ?

Blockchain technology originates from the underlying architectural design of the cryptocurrency bitcoin. Blockchain was first introduced by Satoshi Nakamoto (a pseudonym), who proposed a **peer-to-peer payment system** that allows cash transactions through the Internet **without relying on trust or the need for a financial institution** . Blockchain is secure by design, and an example of a system with a high byzantine failure tolerance .

Bitcoin is considered the first application of the Blockchain concept to create a currency that could be exchanged over the Internet relying only on cryptography to secure the transactions. Blockchain 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 to form a stack called a Blockchain.

But a block does not only store a transaction . We can make it more useful using a smart contract .



1.3 What is a Smart Contract ?

A smart contract is a computer code running on top of a blockchain containing a set of rules under which the parties to that smart contract agree to interact with each other. If and when the pre-defined rules are met, the agreement is automatically enforced. The smart contract code facilitates, verifies, and enforces the negotiation or performance of an agreement or transaction. It is the simplest form of decentralized automation.

It is a mechanism involving digital assets and two or more parties, where some or all of the parties deposit assets into the smart contract and the assets automatically get redistributed among those parties according to a formula based on certain data, which is not known at the time of contract initiation.

2. REQUIRMENT

2.1 PRESENT DIGITAL VOTING

A number of digital voting systems are currently in use in countries around the world. Let's take example of India's EVM. Contrary to claims by Indian election authorities, these paperless EVMs suffer from significant vulnerabilities as follows.

- **EVM Software Isn't Safe**

The electronic voting machines are safe and secure only if the source code used in the EVMs is genuine. Shockingly, the EVM manufacturers, the BEL and ECIL, have shared the 'top secret' EVM software program with two foreign companies, Microchip (USA) and Renesas (Japan) to copy it onto microcontrollers used in EVMs. This process could have been done securely in-house by the Indian manufacturers.

- **Possibility of Mendling**

The most deadly way to hack Indian EVMs is by inserting a chip with Trojan inside the display section of the Control unit. This requires access to the EVM for just two minutes and these replacement units can be made for a few hundred rupees. Bypassing completely all inbuilt securities, this chip would manipulate the results and give out "fixed" results on the EVM screen.

- **Storage and Counting are Concerns**

The EVMs are stored at the district headquarters or in a decentralized manner in different locations. Election Commission's concern for EVM safety becomes apparent only during elections, where as security experts say that voting machines must remain in a secure environment throughout their life cycle.

We can address the above security concerns using using open source code which d rely on Blockchain technology to secure votes, and decentralize the system

3. DESIGN

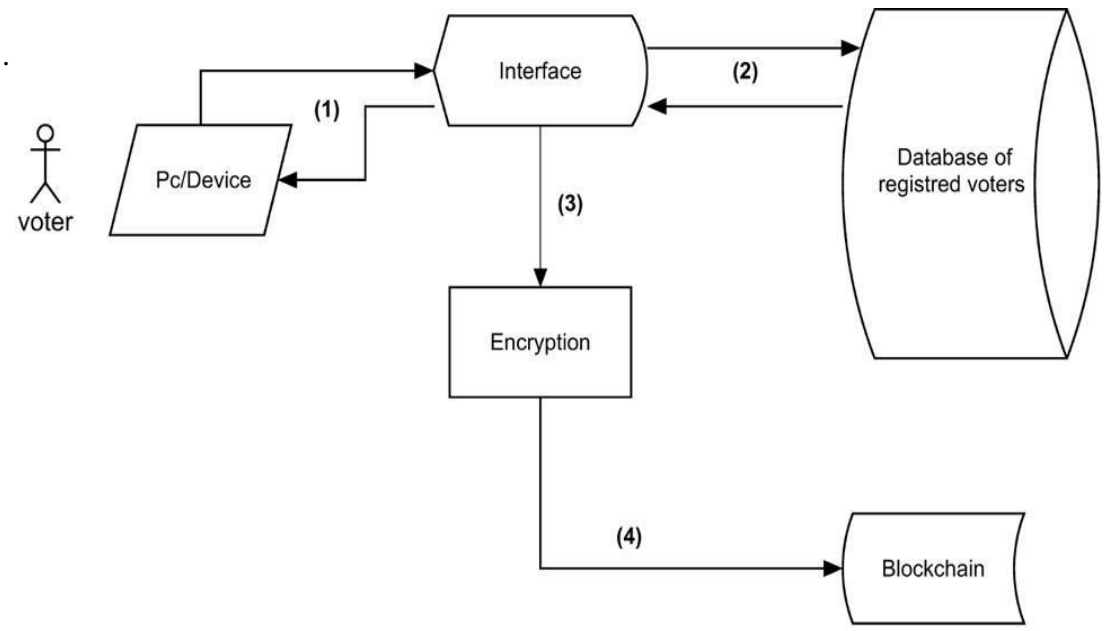


Figure: Basic Design of Proposed System

3.1 ANALYSIS

(1)Requesting to vote :

It will be checked if he is valid voter and is not voting twice through the database.

(2)Casting a vote:

Voters will have to choose to either vote for one of the candidates. Casting the vote will be done through a friendly user interface.

(3)Encrypting votes:

The information related to each vote will be encrypted using SHA one-way hash function that has no known reverse to it

(4)Adding the vote to the Blockchain:

After a block is created, and depending on the candidate selected, the information is reco rded in the corresponding Blockchain. Each block gets linked to the previously cast v

4. PROPOSED SYSTEM

4.1 The Blockchain

The first transaction added to the block will be a special transaction that represents the candidate. When this transaction is created it will include the candidate's name and will serve as the foundation block, with every vote for that specific candidate placed on top of it

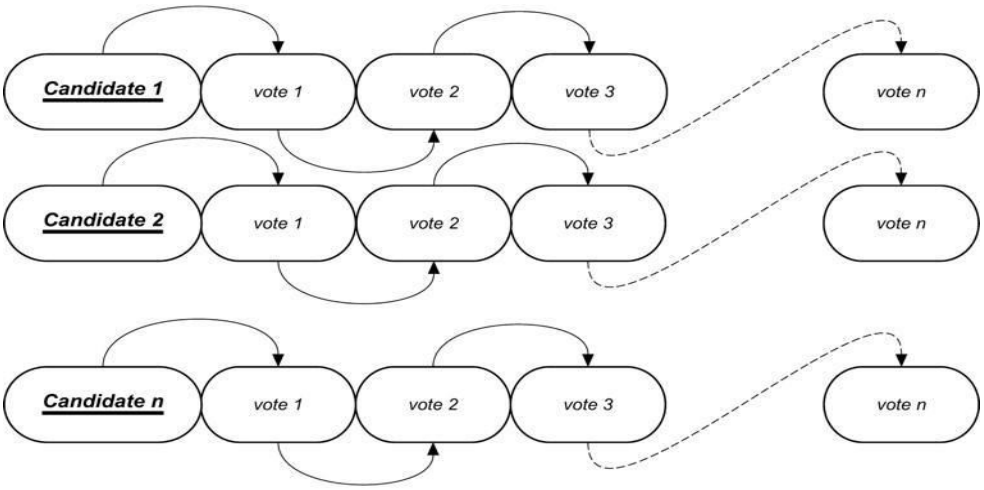


Figure : A Simple Representation of the Blockchain Structure of each Candidate

To ensure that the system is secure, the block will contain the previous voter's information. If any of the blocks were compromised, then it would be easy to find out since all blocks are connected to each other. The Blockchain is decentralized and cannot be corrupted; no single point of failure exists. The Blockchain is where the actual voting takes place. The user's vote gets sent to one of the nodes on the system, and the node then adds the vote to the Blockchain. The voting system will have a node in each district to ensure the system is decentralized.

Each block in the stack is identified by a hash placed on the header. This hash is generated using the Secure Hash Algorithm (SH A-256) to generate an almost idiosyncratic fixed-size 256-bit hash. The SHA-25 6 will take any size plaintext as an input, and encrypt it to a 256-byte binary value. The SHA-25 6 is always a 256-bit binary value, and it is a strictly one-way function. The figure below shows the basic logic of the SHA-256 encryption .

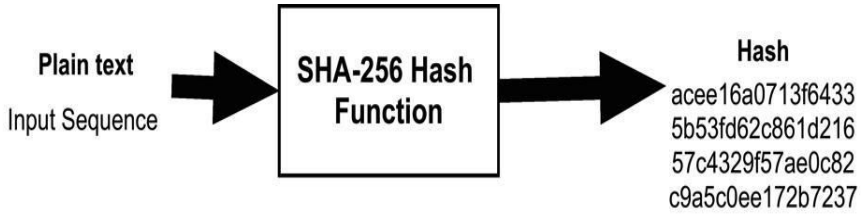


Figure : Basic Function of the SHA-256 Hash

Each header contains informatio n that links a block to its previous block in the chain, which creates a chain linked to the very first block ever created, which is referred to as the foundation. The primary identifier of each block is the encrypted hash in its header. A digital

fin gerprint that was made combining two types of information: the information concerning the new block created, as well as the previous b lock in the chain.

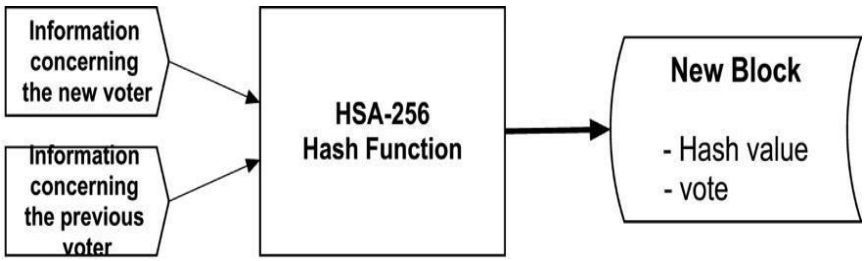


Figure : Creation of new Block containing a Hash Value and a Vote

5. IMPLEMENTATION

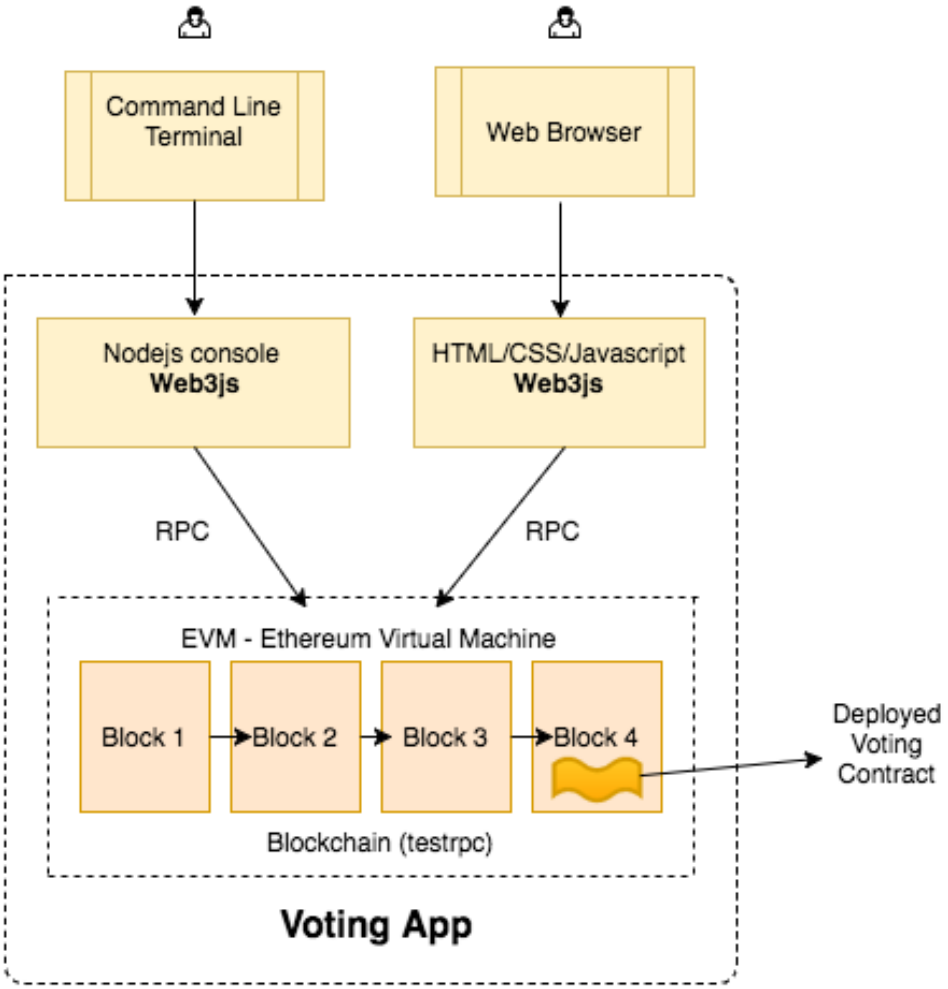


Figure : Basic Structure of Voting App

5.1 Language Used: Solidity

Solidity is a contract-oriented programming language for writing smart contracts.[1] It is used for

implementing smart contracts[2] on various blockchain platforms. We have chosen Ethereum as our Blockchain platform.

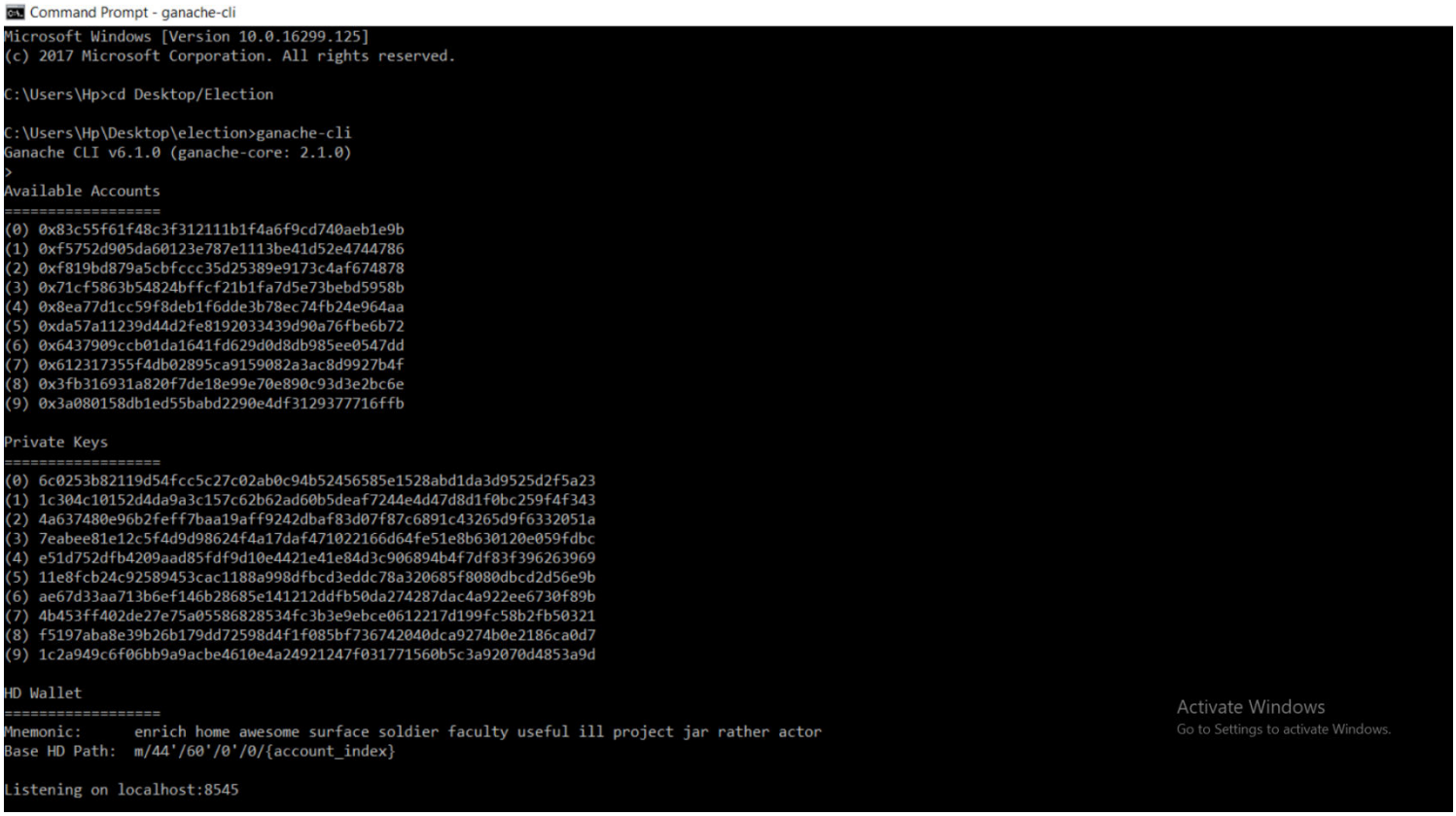
5.2 Dependency Used: Web3.js

The **web3**.js library is a collection of modules which contain specific functionality for the ethereum ecosystem. • The **web3**-eth is for the ethereum blockchain and smart contracts.

5.3 STEPS

5.3.1 Initialize Blockchain

Using Ganche(test-rpc) dependency initialize our blockchain with at least 10 accounts along with their private keys as shown below.



5.3.2 Writing the Smart Contract

Below is the voting contract code with inline comment explanation:

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
pragma solidity ^0.4.18;
// We have to specify what version of compiler this code will
compile with


contract Voting {
  /* mapping field below is equivalent to an associative array or
hash.
  The key of the mapping is candidate name stored as type bytes32
and value is
  an unsigned integer to store the vote count
  */


  mapping (bytes32 => uint8) public votesReceived;


  /* Solidity doesn't let you pass in an array of strings in the
constructor (yet).
  We will use an array of bytes32 instead to store the list of
candidates
  */


  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) public {
    candidateList = candidateNames;
  }
```

```
  // This function returns the total votes a candidate has received
so far


function totalVotesFor(bytes32 candidate) view public returns
(uint8) {
    require(validCandidate(candidate));
    return votesReceived[candidate];
  }


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



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


}
```

5.3.3 Deploy our Smart-Contract

First create a contract object (VotingContract below) which is used to deploy and initiate contracts in the blockchain.

```
> abiDefinition =
JSON.parse(compiledCode.contracts[':Voting'].interface)
> VotingContract = web3.eth.contract(abiDefinition)
> byteCode = compiledCode.contracts[':Voting'].bytecode
> deployedContract =
VotingContract.new(['Name1','Name2','Name3'],{data: byteCode, from:
web3.eth.accounts[0], gas: 4700000})
> deployedContract.address
> contractInstance = VotingContract.at(deployedContract.address)
```

VotingContract.new above deploys the contract to the blockchain. The first argument is an array of candidates who are competing in the election which is pretty straightforward. Let's see what are all in the hash in the second argument:

1. **data**: This is the compiled bytecode which we deploy to the blockchain.
2. **from**: The blockchain has to keep track of who deployed the contract. In this case, we are just picking the first account we get back from calling web3.eth.accounts to be the owner of this contract (who will deploy it to the blockchain). Remember that web3.eth.accounts returns an array of 10 test accounts ganache created when we started the test blockchain. In the live blockchain, you can not just use any account. You have to own that account and unlock it before transacting. You are asked for a passphrase while creating an account and that is what you use to prove your ownership of that account. Ganache by default unlocks all the 10 accounts for convenience.
3. **gas**: It costs money to interact with the blockchain. This money goes to miners who do all the work to include your code in the blockchain. You have to specify how much money you are willing to pay to get your code included in the blockchain and you do that by setting the value of 'gas'. The ether balance in your 'from' account will be used to buy gas. The price of gas is set by the network.

We have now deployed the contract and have an instance of the contract (variable contractInstance above) which we can use to interact with the contract.

5.3.4 Application Screen

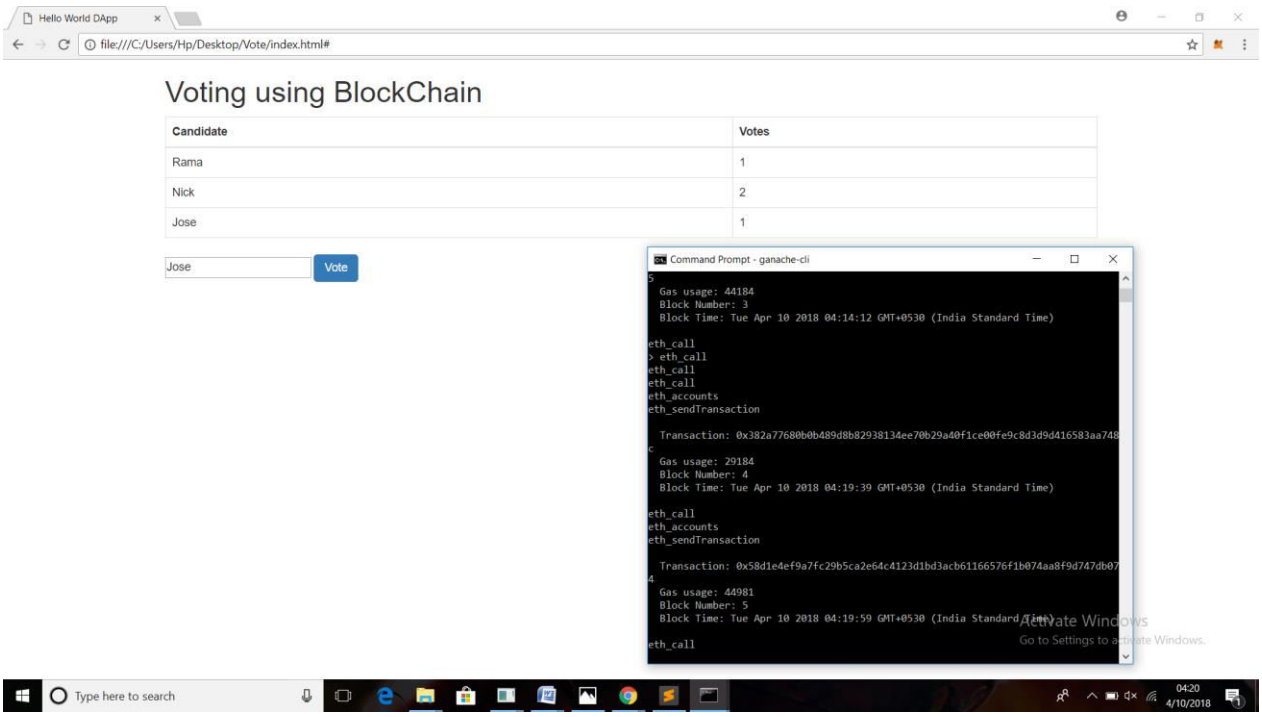


Figure: A new block 5 is created when vote is given

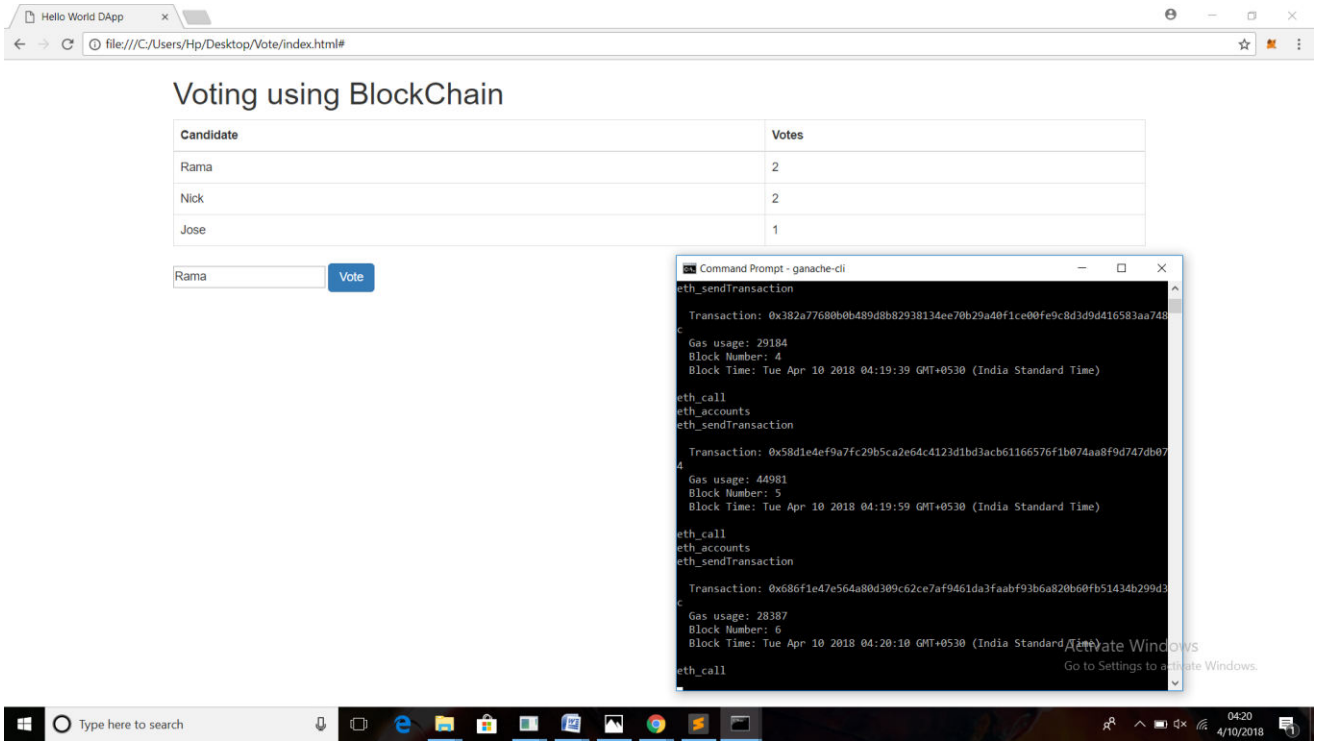


Figure: A new block 6 is created when vote is given

6. LIMITATION AND CONCLUSION

6.1 Limitation

- One of the drawbacks of our system is the inability to change a vote in case of a user mistake. The user will be able to cast its vote only once.
- Time required to process to single vote is much more than average time (in paper system) especially when number of voters is more.

6.2 Conclusion

- The system is decentralized and does not rely on trust. Any registered voter will have the ability to vote using any device connected to the Internet. The Blockchain will be publicly verifiable and distributed in a way that no one will be able to corrupt it.

- Also, due to the encryption mechanism we are using (as described in section 5.3) it would be close to Impossible for any person(s) to gain access to all the votes without first taking control of the entire service network.

7. REFERNCES

[1] https://yobicash.org/whitepaper.pdf

[2] http://www.indianevm.com/articles/ten-reasons-for-banning-indian-evms.pdf

[3] https://github.com/ethereum/go-ethereum/wiki/Building-Ethereum

[4] http://remix.ethereum.org/#optimize=false&version=soljson-v0.4.21+commit.dfe3193c.js

[5] https://bitcoin.org/en/developerguide#block-chain-overview