E-VOTING - Application of Blockchain and Threshold Cryptography

1. Introduction

Advancements in technology have always made us wonder if voting could also be managed electronically without being physically present to cast your vote standing in long queues at polling booths. Widespread suspicion on the accuracy and correctness of the results and conduct of elections among people have them thinking of developing a system that can offer transparency, auditability, immutability and integrity.

Many countries have tried their hand at e-voting or i-voting. Though each of them has encountered some or the other issues, yet it is worth pursuing due to increased polling percentage. Various private companies have come up with different solutions to conduct voting.

Estonia is the only country to offer internet voting for all their elections even today[1]. Every Estonian citizen has a national identity card that enables them to authenticate themselves. Several security issues were raised against the i-voting system[2], even asking the government to shut down this system. But the ease of election management and cooperation from each citizen have kept it going.

Ben Adida had developed an open-source web-based system known as Helios[3]. Though this system enabled auditability, as the voters were able to view their encrypted votes and then cast their votes finally, the centralised server's security was not ensured. The emergence of blockchain ensured transparency and immutability, which resulted in using it as the backbone of various e-voting application by private entities like voteWatcher[4] etc. Researchers have been keen on developing various solutions[5] [6] to e-voting that satisfies each of the requirements like integrity, privacy, auditability, verifiability, transparency etc.

The major disadvantage with most of the systems mentioned is that the role of a trusted party is huge. Here we are trying to design our e-voting system that can satisfy these requirements as well as be able to run without a trusted party that can control the results of the election.

---

[1] "i-Voting — e-Estonia." https://e-estonia.com/solutions/e-governance/i-voting/. Accessed 6 Jun. 2020.
[2] "Security Analysis of the Estonian Internet Voting System ...." https://dl.acm.org/doi/10.1145/2660267.2660315. Accessed 6 Jun. 2020.
[3] "Helios: Web-based Open-Audit Voting - Usenix." 30 Jan. 2008, https://www.usenix.org/legacy/event/sec08/tech/full_papers/adida/adida.pdf. Accessed 6 Jun. 2020.
[4] "VoteWatcher." http://votewatcher.com/. Accessed 6 Jun. 2020.
[5] "Secure Voting System Using Ethereum's Blockchain." 1 Jan. 2018, https://scholarworks.boisestate.edu/cgi/viewcontent.cgi?article=1174&context=cs_facpubs. Accessed 6 Jun. 2020.
[6] "(PDF) ethVote: Towards secure voting with distributed ledgers." 29 Apr. 2020, https://www.researchgate.net/publication/341000573_ethVote_Towards_secure_voting_with_distributed_ledgers. Accessed 6 Jun. 2020.

2. Theory Behind the System

The proposed E-Voting system uses the application of two concepts: Blockchain Technology and Threshold Cryptography (Secret sharing).

2.1 Blockchain

A Blockchain can be described as a data structure that holds a time-stamped series of immutable transactional records that ensures security, transparency, and decentralization[7]. It can be viewed as a distributed public ledger i.e a chain or records stored in the forms of blocks which are controlled by no single authority[8].

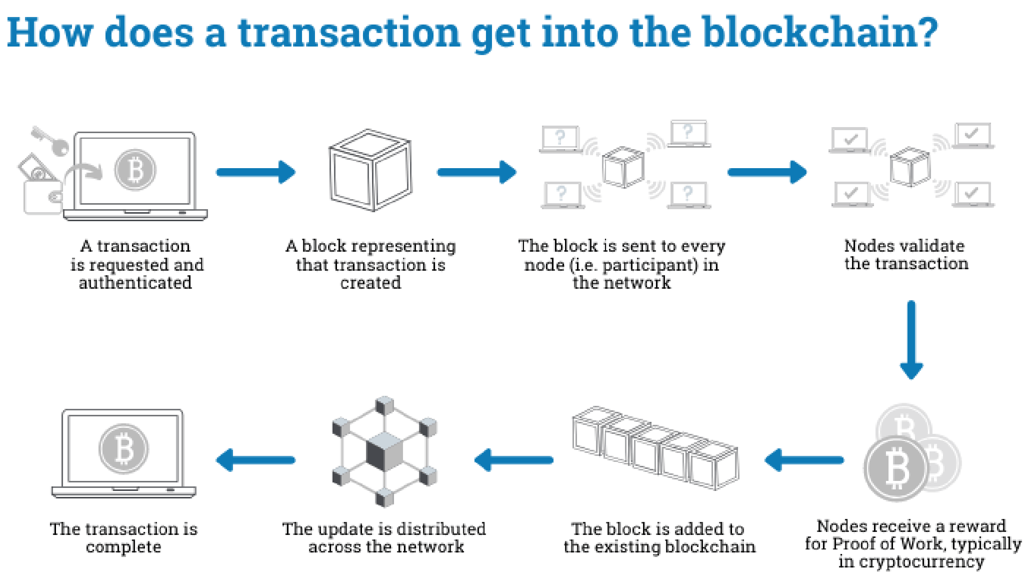
- Everyone connected to the blockchain network has a copy of the ledger, making the information open to them, thereby maintaining **transparency**.
- Each transaction on a blockchain is secured with a **digital signature** that proves its **authenticity, integrity and non-repudiation**.
- Each block in a blockchain network stores some transactional data along with the hash of its previous block. So, if the data is modified,the hash of the block will also be modified and such attempts can be easily tracked. This makes the blockchain resistant to data-tampering, assuring **immutability**.
- Blockchain is an append-only data structure and anyone in the network can propose to add a new block. But the majority of the network nodes must reach an agreement, commonly referred to as consensus before a proposed new block of entries becomes a permanent part of the ledger. Some of the commonly used **consensus algorithms**[9] are:
    a. **Proof of Work** (PoW) : Here, the validators, commonly called as miners try to prove that they have engaged in a significant amount of computational work to solve a mathematical puzzle. The objective is to select a miner for the next block generation. The PoW protocol may set a condition for the validity of the block, say for example, the first 30 bits of the hash should be zero. All the miners try giving inputs as a parameter, in a brute-force manner along with the actual data in the block, so that the resultant hash satisfies the condition that considers a block valid. Once a miner finds such a hash, the verification can be done very easily. The first miner who produces a valid hash will be accepted and rewarded in the form of cryptocurrency of the system.

[7] "Blockchain-Based E-Voting System - IEEE Conference ...." https://ieeexplore.ieee.org/document/8457919. Accessed 6 Jun. 2020.
[8] "Blockchain Technology Explained: Introduction, Meaning, and ...." https://hackernoon.com/blockchain-technology-explained-introduction-meaning-and-applications-edbd6759a2b2. Accessed 6 Jun. 2020.
[9] "What Is a Blockchain Consensus Algorithm? | Binance Academy." https://academy.binance.com/blockchain/what-is-a-blockchain-consensus-algorithm. Accessed 6 Jun. 2020.

b. **Proof of Stake** (PoS) : Here, Instead of investing in expensive hardware to solve computationally expensive problems, validators invest in the coins (cryptocurrency of the system) by locking up some of their coins as stake in a wallet[10]. Validators validate blocks by betting on the next block that will be selected to be added to the chain. If your block is selected, you'll receive a proportion of transaction fees, depending on your stake. If there is an attempt to cheat by selecting an invalid transaction, a proportionate portion of your stake is lost. PoS encourages validators through an incentive mechanism to reach an agreement.

- Blockchain is **programmable**. One way of doing it is through **Smart contracts**. They enable the coding of simple contracts that are **self-executable**, i.e when certain conditions in the code of these contracts are met, they are automatically deployed. The Ethereum, an open source blockchain platform has introduced smart contracts in the Blockchain ecosystem

Each transaction refers to the transfer of some data or cryptocurrency. The following diagram summarises the flow of control in a transaction[11].


How does a transaction get into the blockchain?
A transaction is requested and authenticated
A block representing that transaction is created
The block is sent to every node (i.e. participant) in the network
Nodes validate the transaction
Nodes receive a reward for Proof of Work, typically in cryptocurrency
The block is added to the existing blockchain
The update is distributed across the network
The transaction is complete

Though Blockchain has evolved to many levels since inception, there are two broad categories in which blockchains can be classified majorly i.e. Public and Private blockchains.

➔ **Public blockchain** is a permissionless ledger and grants access to read and ability to create a transaction to everyone. Anyone with internet is eligible to access it.They

[10] "Consensus Algorithms in Blockchain - GeeksforGeeks." https://www.geeksforgeeks.org/consensus-algorithms-in-blockchain/. Accessed 6 Jun. 2020.
[11] "Blockchain Explained: How does a transaction get into the ...." https://www.euromoney.com/learning/blockchain-explained/how-transactions-get-into-the-blockchain. Accessed 6 Jun. 2020.

usually reward their network participants for performing the mining process and maintaining the immutability of the ledger. e.g Bitcoin Blockchain

➔ **Private Blockchain** on the contrary is permissioned blockchain that allows organisations to employ distributed ledger technology without making the data public. The rules of a private blockchain can be changed according to different levels of permissions, exposure, number of members, authorization etc.

2.2 Secret Sharing

[12]**Secret sharing** is the method of dividing a secret amongst the participants in such a way that the reconstruction of the original secret is possible only if at least a specified number of participants come together. The minimum number of shares required to reconstruct the secret is referred to as the threshold of the system. For a system with n participants and the threshold as k, the secret can be reconstructed only with a group of size at least k come together. Such a secret sharing system is called a **(k,n) threshold scheme**.

A **secure secret sharing system** is one in which a person/group with less than k shares infers no more information than a person/group with no shares. Mathematically such a system can be defined as:
Given a secret S, an (n,k) secure secret sharing system divides it into n shares $S_1, S_2, ..., S_n$ so that:

1. S can be computed from knowing k or more shares out of the n shares generated.
2. Knowing any number of shares less than k does not reveal any information about S.

If k = n, then all the shares are required to reconstruct S.

Shamir's Secret Sharing

[13]**Shamir's secret sharing** algorithm defines one such secure secret sharing system. It is based on the idea that k points are needed to describe a polynomial of degree k-1.
All operation are done in a finite field $\boldsymbol{F}$, of size $\boldsymbol{P}$ (a prime number) subjected to the constraints: $0 < k \leq n$ and $S < \boldsymbol{P}$. A smooth curve appears disorganized in a finite field, ensuring that no insights are inferred despite knowing the degree of the polynomial and knowing less than k number of shares.

Generation:

Choose k-1 random positive integers, $a_1, a_2, ..., a_{k-1}$ such that $a_i < \boldsymbol{P}$.
Using these, build the polynomial $f(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{k-1}x^{k-1}$ where $a_0 = S$.
Pick random x_i values for i = 1, ..., n and $x_i \neq 0$. Distribute the points (xi, f(xi)) as the individual secret shares of each of the n participants.

[12] "Secret sharing - Wikipedia." https://en.wikipedia.org/wiki/Secret_sharing. Accessed 6 Jun. 2020.
[13] "Shamir's Secret Sharing - Wikipedia." https://en.wikipedia.org/wiki/Shamir%27s_Secret_Sharing. Accessed 6 Jun. 2020.

Reconstruction:

Given a subset of k shares of the form (x_j,y_j), the coefficients of the polynomial can be found using Lagrange Interpolation.

$$f(x) \; = \sum_{j=1,\ldots,\,n} y_j \,.\; l_j(x)$$

where,

$$l_j = \frac{(x-x_1)(x-x_2)....(x-x_{j-1})(x-x_{j+1})....(x-x_n)}{(x_j-x_1)(x_j-x_2)....(x_j-x_{j-1})(x_j-x_{j+1})....(x_j-x_n)}$$

S is obtained from f(0).

Instead of calculating all coefficients of f(x), S= f(0) can be also be directly calculated by

$$L(0) \; = \sum_{j=0}^{k-1} f(x_j) \prod_{m=0,\, m\neq j}^{k-1} \frac{x_m}{x_m - x_j}$$

3. The Technology behind the solution

The voting system will be built on top of the **Ethereum Blockchain**. Ethereum offers a modern secure open-source decentralised public blockchain solution that supports use cases beyond the scope of simply currency transfers unlike Bitcoin. The Ethereum blockchain offers the following features over other public chains:

Smart Contracts: These are special types of accounts on the Ethereum blockchain that have some immutable code associated with it. This code and its logic can be used to facilitate operations beyond simple currency transfers on the ethereum chain. Smart contracts are used in our solution to carry out the actual voting process. The encrypted vote can be passed to our vote casting smart contract as a transaction. The smart contract then processes the necessary and casts the user's (external account) vote. Smart contracts are written in a special language called Solidity, a Turing complete language.

Ethereum Virtual Machine (EVM) : The code in smart contracts are executed on a virtual machine called Ethereum Virtual Machine before committing the result to the blockchain. Since the data on the blockchain is permanent, the EVM helps in executing transactions on it and filtering out faulty transactions.

For the purpose of the project, a private ethereum blockchain network will be used exclusively for the voting system. Below mentioned are the different technologies used/involved in this system.

Geth: Geth is an ethereum client written in the Go language. This is used to spawn out ethereum nodes and run the blockchain on the node. It also provides an interface to interact with the blockchain. Also provides CLI wallets for the blockchain.

Web3.js: A JavaScript library for interacting with an Ethereum chain over RPC. Used for calling smart contract methods and sending transactions to the blockchain.

Truffle: Truffle is a development suite for the Ethereum blockchain. It has several tools that simplify the process of developing dApps (decentralised apps) for Ethereum much faster. It also comes with pre-built support for the Chai assertion library and Mocha testing library.

MetaMask: Provides a GUI frontend for your Ethereum wallet as a browser extension. This helps the users of the system to cast votes easily.

Ganache: Ganache is a tool used to spawn a personal development blockchain. This chain has no miners and cannot be used for production. However, it provides a quick simulated environment for developing dApps and running tests before deploying to your production chain.

Twilio: Twilio provides a cloud-based API for sending SMS to voter mobile phones. This is used to send OTP messages to their phones during registration and login phases.

React: A frontend library in JavaScript developed by Facebook. Used for building the website frontend.

MongoDB: A Document Oriented Model based database management system. Used for maintaining list of details of eligible voters for elections.

For the secret key sharing, NodeJS libraries for SMTP and secret sharing are used for generating, splitting and sharing the keys with the concerned parties.

4. Entities/Roles

This section describes the various roles and their significance in the process. There are three entities involved:

Registration Authority

An admin entity responsible for creating elections, registering voters for each election and adding candidates to an election. In a real election, this role is assumed by the Election Commision or a similar body. Candidate representatives, if required, are also allowed to constitute this body.

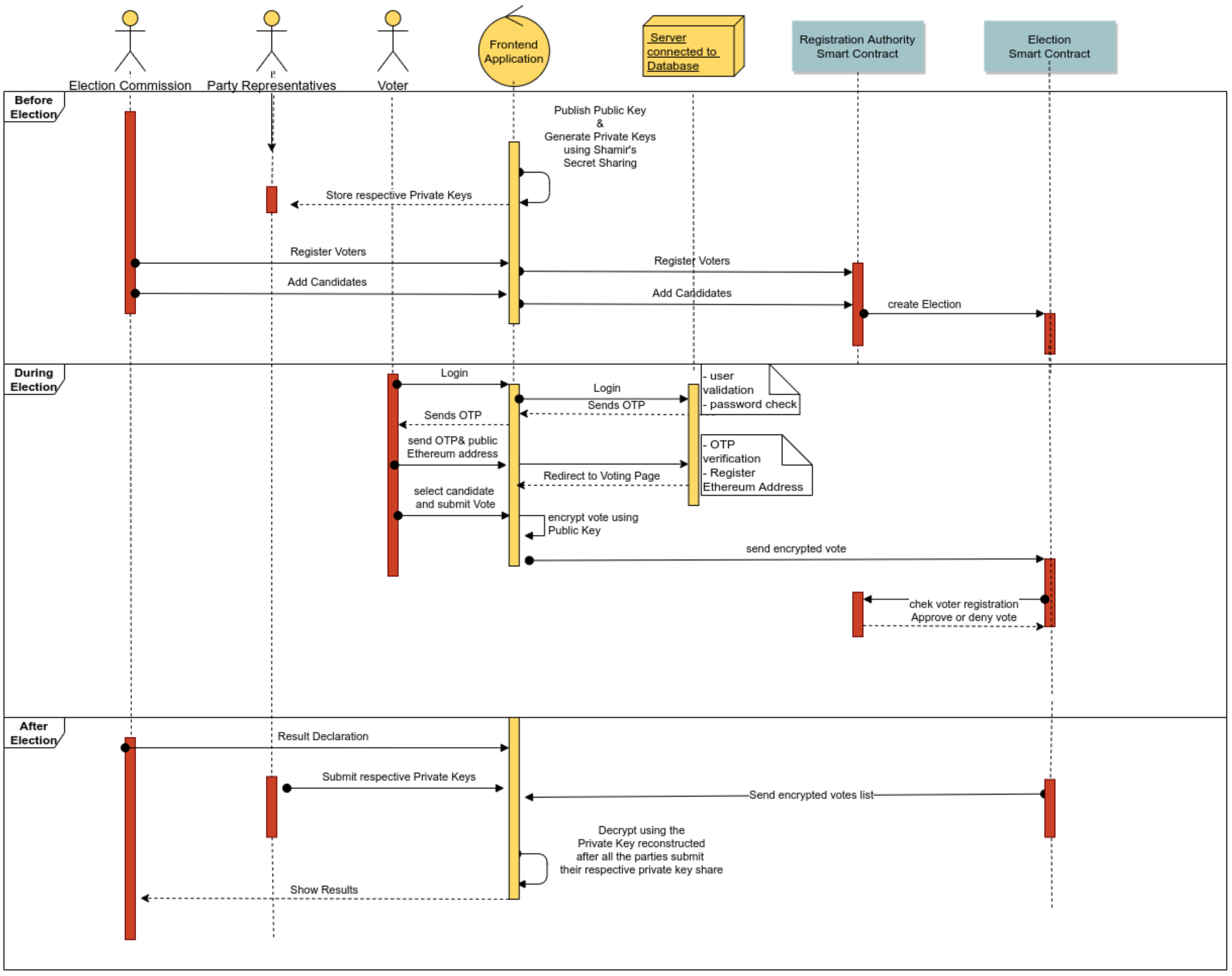
Candidates

The superset of all individuals/bodies contesting different elections. The votes cast by voters correspond to one of the candidates in an election.

Voters

The superset of all individuals registered for different elections. These individuals cast their vote during the election and determine the winning candidate for an election.

5. Workflow



Election Commission
Party Representatives
Voter
Frontend Application
Server connected to Database
Registration Authority Smart Contract
Election Smart Contract
Before Election
Publish Public Key & Generate Private Keys using Shamir's Secret Sharing
Store respective Private Keys
Register Voters
Add Candidates
Register Voters
Add Candidates
create Election
During Election
Login
Login
Sends OTP
Sends OTP
- user validation
- password check
send OTP& public Ethereum address
- OTP verification
- Register Ethereum Address
Redirect to Voting Page
select candidate and submit Vote
encrypt vote using Public Key
send encrypted vote
chek voter registration
Approve or deny vote
After Election
Result Declaration
Submit respective Private Keys
Send encrypted votes list
Decrypt using the Private Key reconstructed after all the parties submit their respective private key share
Show Results

Pre-election stage

Generation:
A public - private key pair (key_{pub}, key_{pri}) is generated by the “KeyGeneration” program hosted on the server. This requires all the candidates/representatives and Registration Authority to supply their email addresses for the program. The program then publishes the public key and distributes the private key shares securely over SMTP with TLS amongst the candidates and Registration Authority.

Starting Election:
The Registration Authority calls a smart contract method for creating an Election specifying the candidate list and stating the start and end time of the election. The key_{pub} for encrypting votes generated above is also supplied to the Election create method. The Registration Authority also maintains a database of eligible candidates, their Election IDs, and mobile numbers in a backend server B.

Registering Candidates:
The Registration Authority registers the candidate names and their descriptions using the smart contract method for adding ballot options.

Voter Ethereum Address Registration:
Voters authenticate themselves on the backend server B and, if eligible for an election, can register their Ethereum address to be used for their vote for that election. This is done after the user successfully clears an OTP challenge sent to their mobile number or email address for a two-step verification of their identity. Once the server receives the Ethereum address, it registers on the blockchain using a smart contract to register voting addresses. The mapping between user details and their chosen Ethereum address is not stored to maintain privacy. However, for the same reason, a user is not allowed to modify their address later.

Election Stage

During the specified duration of the election, registered Ethereum accounts from the pre-election stage can access the block chain and register their vote by calling the “Vote” smart contract. Each vote will appear as a transaction in the blockchain with the “From” field giving the voter’s public Ethereum wallet address and the data field will contain the vote encrypted with key_{pub} generated during the pre-election stage. The encrypted vote consists of a one-hot encoded vector representing the ballot options and a random long string. This ensures an attacker cannot bruteforce all possible vectors and compare to find the selected option.Upon mining the transaction on to a block, the voter is returned the transactionHash for his vote which he/she can use to convince him/herself of the vote inclusion. The “Vote” smart contract also maintains a list of the encrypted votes in its storage memory. Only one vote is allowed per voter and the block chain provides a ledger of who have voted in the election.

Post-election stage:

After the election ends automatically after the stipulated time, the Registration Authority along with the candidates/representatives can call the “ResultDeclaration" program hosted on the server. The list of encrypted votes publicly maintained by the “Vote” smart contract along with individual secret shares of key_{pri} are needed to call the function. This program will then reconstruct the key_{pri}, count the “valid” votes and declare the result. The reconstructed key_{pri} is not disclosed at any point of time and still remains a secret.

6. Implementation

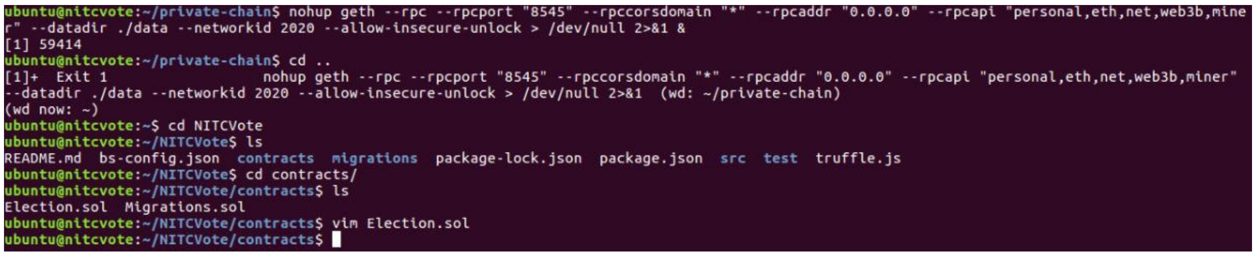
This section details the implementation of the e-voting system. The various components of the system are listed below.

Backend Server: A server written in ExpressJS is responsible for maintaining a MongoDB database of eligible voters, authenticating voters, sending OTP using Twilio and registering Ethereum account addresses submitted by eligible voters. The server also hosts the KeyGeneration programs to generate public keys and distribute private key shares to candidates of an election and the Tally program used to decrypt and tally the votes from the blockchain. The code is present at https://github.com/appu313/NITCVote-backend.

Private Blockchain: A private blockchain is deployed using Geth (Go-Ethereum). Any node part of the blockchain’s underlying network can connect and run a full node to be part of the blockchain network. One of the nodes is present at http://104.211.163.93:30303. Queries to the blockchain can be made via RPC to any of the nodes and one of them is http://104.211.163.93:8545. Smart contracts written according to above mentioned specifications were deployed on this blockchain. The smart contracts were based on an existing code from GitHub of Johannes Mols’ ethVote project [14] and was modified to fit use cases and logic of this e-voting system. The smart contracts used are present at https://github.com/farisshajahan/NITCVote.

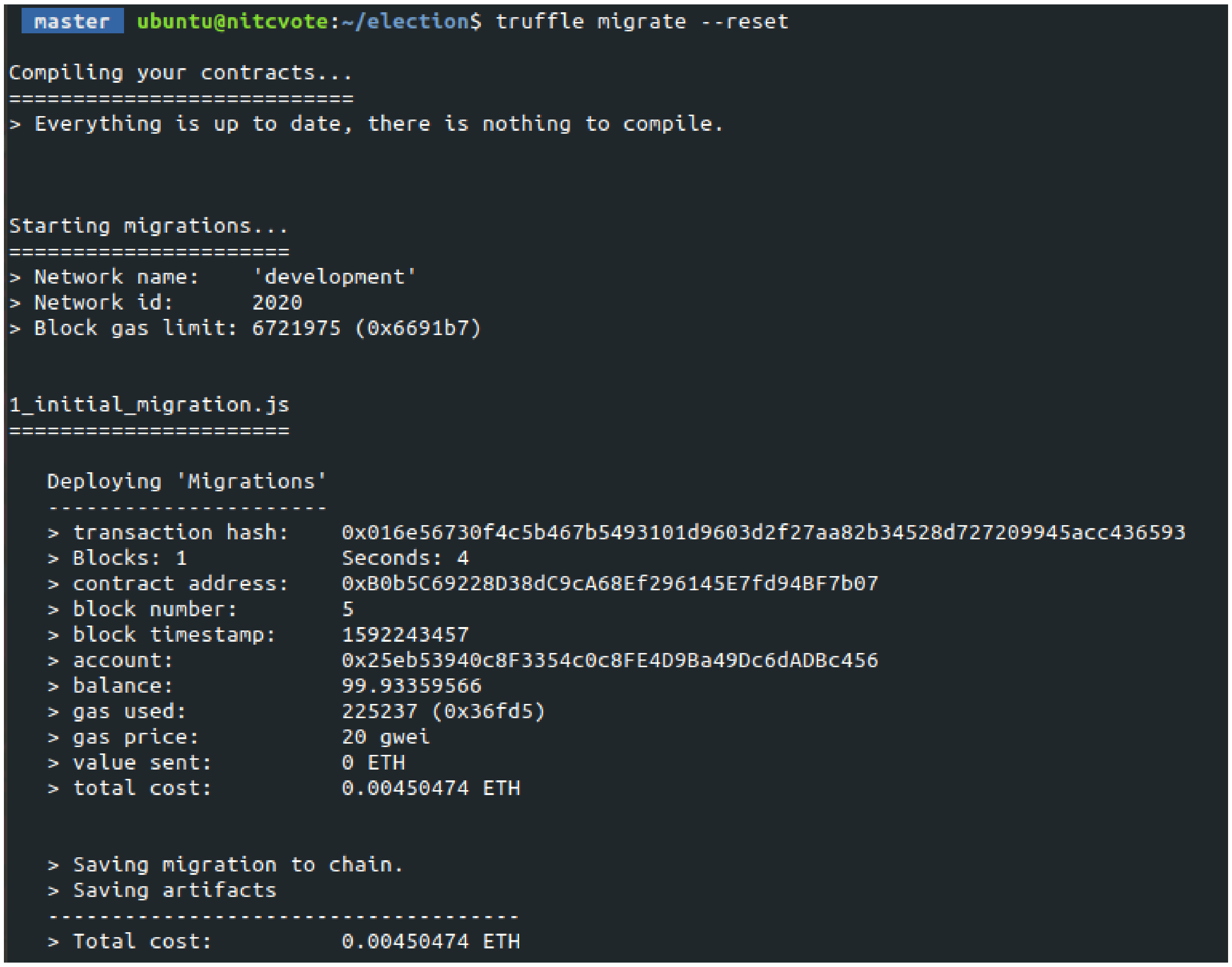
Frontend Web Application: The frontend web app deployed on a server connects all the different components of the e-voting system. The frontend is written in React and uses the Axios library to connect to the backend server for authentication and registration requests and uses the Web3 library for interacting with the private Ethereum blockchain. All logical separations are made in the frontend code using React components for each of them, with each having their own states. The frontend is based on an existing code from GitHub of Johannes Mols’ ethVote project [15] and was modified to fit use cases of this voting system. The application is currently hosted at https://nitcvote.fossmeet.com and code can be found at https://github.com/farisshajahan/NITCVote-react.

Screenshots

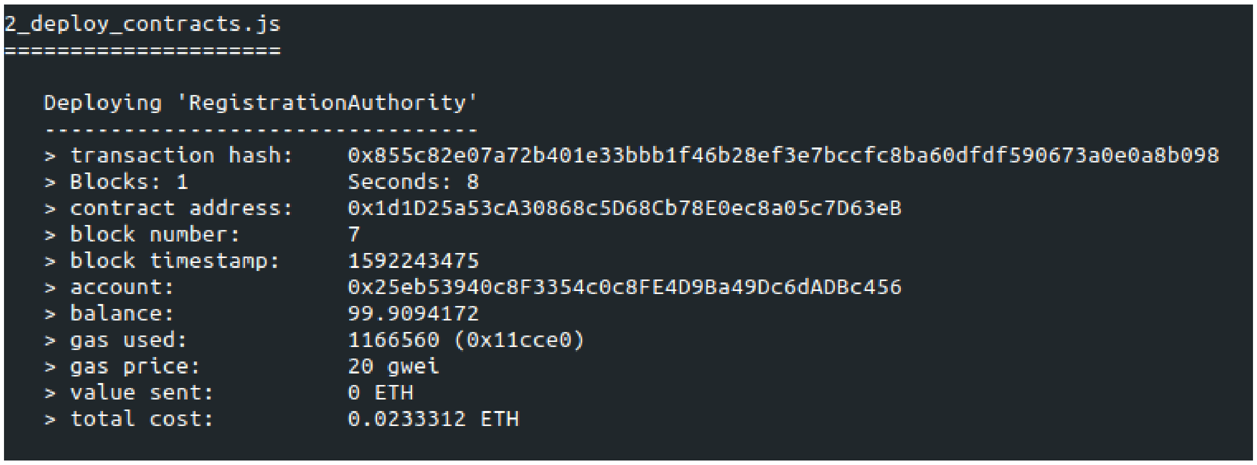


```
ubuntu@nitcvote:~/private-chain$ nohup geth --rpc --rpcport "8545" --rpccorsdomain "*" --rpcaddr "0.0.0.0" --rpcapi "personal,eth,net,web3b,mine
r" --datadir ./data --networkid 2020 --allow-insecure-unlock > /dev/null 2>&1 &
[1] 59414
ubuntu@nitcvote:~/private-chain$ cd ..
[1]+  Exit 1                  nohup geth --rpc --rpcport "8545" --rpccorsdomain "*" --rpcaddr "0.0.0.0" --rpcapi "personal,eth,net,web3b,miner"
--datadir ./data --networkid 2020 --allow-insecure-unlock > /dev/null 2>&1  (wd: ~/private-chain)
(wd now: ~)
ubuntu@nitcvote:~$ cd NITCVote
ubuntu@nitcvote:~/NITCVote$ ls
README.md  bs-config.json  contracts  migrations  package-lock.json  package.json  src  test  truffle.js
ubuntu@nitcvote:~/NITCVote$ cd contracts/
ubuntu@nitcvote:~/NITCVote/contracts$ ls
Election.sol  Migrations.sol
ubuntu@nitcvote:~/NITCVote/contracts$ vim Election.sol
ubuntu@nitcvote:~/NITCVote/contracts$
```

Geth deployment on 104.211.163.93

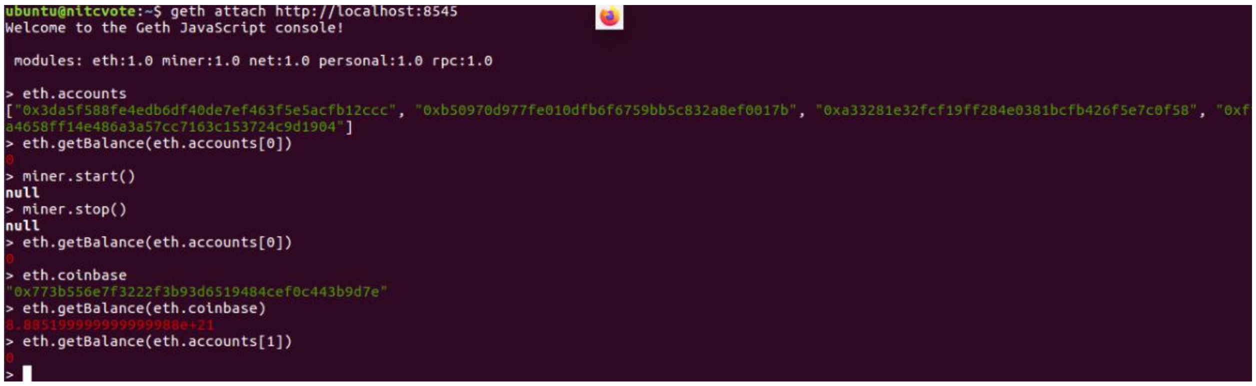


```
 master  ubuntu@nitcvote:~/election$ truffle migrate --reset

Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.



Starting migrations...
======================
> Network name:    'development'
> Network id:      2020
> Block gas limit: 6721975 (0x6691b7)


1_initial_migration.js
======================

   Deploying 'Migrations'
   ----------------------
   > transaction hash:    0x016e56730f4c5b467b5493101d9603d2f27aa82b34528d727209945acc436593
   > Blocks: 1            Seconds: 4
   > contract address:    0xB0b5C69228D38dC9cA68Ef296145E7fd94BF7b07
   > block number:        5
   > block timestamp:     1592243457
   > account:             0x25eb53940c8F3354c0c8FE4D9Ba49Dc6dADBc456
   > balance:             99.93359566
   > gas used:            225237 (0x36fd5)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00450474 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00450474 ETH
```



```
2_deploy_contracts.js
=====================

   Deploying 'RegistrationAuthority'
   ---------------------------------
   > transaction hash:    0x855c82e07a72b401e33bbb1f46b28ef3e7bccfc8ba60dfdf590673a0e0a8b098
   > Blocks: 1            Seconds: 8
   > contract address:    0x1d1D25a53cA30868c5D68Cb78E0ec8a05c7D63eB
   > block number:        7
   > block timestamp:     1592243475
   > account:             0x25eb53940c8F3354c0c8FE4D9Ba49Dc6dADBc456
   > balance:             99.9094172
   > gas used:            1166560 (0x11cce0)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.0233312 ETH
```

Smart contracts deployment to the geth blockchain



```
ubuntu@nitcvote:~$ geth attach http://localhost:8545
Welcome to the Geth JavaScript console!

 modules: eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0

> eth.accounts
["0x3da5f588fe4edb6df40de7ef463f5e5acfb12ccc", "0xb50970d977fe010dfb6f6759bb5c832a8ef0017b", "0xa33281e32fcf19ff284e0381bcfb426f5e7c0f58", "0xf
a4658ff14e486a3a57cc7163c153724c9d1904"]
> eth.getBalance(eth.accounts[0])
0
> miner.start()
null
> miner.stop()
null
> eth.getBalance(eth.accounts[0])
0
> eth.coinbase
"0x773b556e7f3222f3b93d6519484cef0c443b9d7e"
> eth.getBalance(eth.coinbase)
8.885199999999999988e+21
> eth.getBalance(eth.accounts[1])
0
> 
```

Geth RPC console: displaying accounts on the chain, starting miners and displaying account balance

More Screenshots:
https://docs.google.com/document/d/11kxxI78ztRbj73CDmlshr3waTWLoZNEPFyt3QCDnf4w/edit?usp=sharing

7. Analysis

Features of the e-voting system

1. **Secrecy**: All votes are encrypted with the public key generated during the pre-election phase. The corresponding private key is shared amongst the participating candidates and the Registration Authority and is not disclosed even after the election results are announced. This ensures that even though the votes are permanently recorded in the blockchain, no information about the unencrypted vote can be deduced.
2. **Non-coercibility**: Because the votes are encrypted, and the entire private key is not known to anyone, the voter will not be able to prove who he/she voted for.
3. **Convenience**: MetaMask, a plug-in for Chrome, acts as an Ethereum browser, allowing users to manage their Ethereum wallet and interact with decentralized applications and smart contracts without running a full node. The private key is stored safely, and each

transaction has to be authenticated by the user. This helps increase the accessibility of the Ethereum blockchain to the average user.

4. **Certifiability**: The e-voting is based on the cryptographically secure concepts of secret sharing. It also makes use of blockchain that guarantees immutability and authenticity.
5. **Transparency**: The code for pre and post election stages used to create the secret shares of the private key and for the result declaration are publicly available. All votes are recorded in the blockchain as transactions. Each transaction contains the ethereum account of the voter as well as the encrypted vote. Thus transparency is maintained at all stages in the election and is thus certifiable.
6. **Authentication**: Only eligible voters are allowed to register Ethereum addresses for their vote. Moreover, due to the use of a private blockchain network, a voter is required to be part of this network hence providing flexibility to implement the system within an organizational network.
7. **Accuracy**: The secure concepts of blockchain technology ensure that all votes are registered on the chain correctly. Any attempt to prevent votes from being added will be thwarted by miner nodes on the network.
8. **Uniqueness**: Smart contracts implemented in the blockchain ensures that multiple votes are not possible by the same voter.
9. **Integrity**: Immutable records of the encrypted votes available in the blockchain ensures that the votes cannot be modified.
10. **Verifiability**: A successful transaction indicates that the vote will be counted in the result. The final tally can also be compared to the total number of encrypted votes to ensure that all votes have been counted.
11. **Auditability**: The transactions can be used to audit that the vote of a voter has been cast properly. An audit option is given for the pre and post stage programs available in the server that allows anyone to check if these programs are behaving as per the code published. Anyone can also audit all transactions using the blockchain performed in a completely transparent manner by running a node.
12. **Reliability**: The blockchain network consists of multiple full nodes and miner nodes which maintain their own copies of all data. This ensures redundancy and in turn reliability with high fault tolerance.

8. Plan of Action

Installation of development tools and setup by All - June 7

Writing and testing smart contracts for vote casting by Nileena, Aparna - June 9

Deployment of blockchain network and client by Fariz, Reema - June 9

Decryption Server programs by Reema, Nileena - June 11

Authentication Server (using OTP) by Aparna, Fariz - June 11

References

[1] "i-Voting — e-Estonia." https://e-estonia.com/solutions/e-governance/i-voting/. Accessed 6 Jun. 2020.

[2] "Security Analysis of the Estonian Internet Voting System ...." https://dl.acm.org/doi/10.1145/2660267.2660315. Accessed 6 Jun. 2020.

[3] “Helios: Web-based Open-Audit Voting - Usenix." 30 Jan. 2008, https://www.usenix.org/legacy/event/sec08/tech/full_papers/adida/adida.pdf. Accessed 6 Jun. 2020.

[4] "VoteWatcher." http://votewatcher.com/. Accessed 6 Jun. 2020.

[5] "Secure Voting System Using Ethereum's Blockchain." 1 Jan. 2018, https://scholarworks.boisestate.edu/cgi/viewcontent.cgi?article=1174&context=cs_facpubs. Accessed 6 Jun. 2020.

[6] "(PDF) ethVote: Towards secure voting with distributed ledgers." 29 Apr. 2020, https://www.researchgate.net/publication/341000573_ethVote_Towards_secure_voting_with_distributed_ledgers. Accessed 6 Jun. 2020.

[7] F. Þ. Hjálmarsson, G. K. Hreiðarsson, M. Hamdaqa and G. Hjálmtýsson, "Blockchain-Based E-Voting System," *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, San Francisco, CA, 2018, pp. 983-986, doi: 10.1109/CLOUD.2018.00151.

[8] "Blockchain Technology Explained: Introduction, Meaning, and ...." https://hackernoon.com/blockchain-technology-explained-introduction-meaning-and-applications-edbd6759a2b2. Accessed 6 Jun. 2020.

[9] "What Is a Blockchain Consensus Algorithm? | Binance Academy." https://academy.binance.com/blockchain/what-is-a-blockchain-consensus-algorithm. Accessed 6 Jun. 2020.

[10] "Consensus Algorithms in Blockchain - GeeksforGeeks." https://www.geeksforgeeks.org/consensus-algorithms-in-blockchain/. Accessed 6 Jun. 2020.

[11] "Blockchain Explained: How does a transaction get into the ...." https://www.euromoney.com/learning/blockchain-explained/how-transactions-get-into-the-blockchain. Accessed 6 Jun. 2020.

[12] "Secret sharing - Wikipedia." https://en.wikipedia.org/wiki/Secret_sharing. Accessed 6 Jun. 2020.

[13] "Shamir's Secret Sharing - Wikipedia." https://en.wikipedia.org/wiki/Shamir%27s_Secret_Sharing. Accessed 6 Jun. 2020.

[14] https://github.com/johannesmols/ethVote

[15] https://github.com/johannesmols/ethVote-react