Blockchain Based E-Voting System - using Multichain Framework

ANSHI AGARWAL, AYUSHI BANSAL, and KRITHIGA MURUGAVEL, University of California, Davis

Abstract - **Building an electronic voting system that satisfies the legal requirements of legislators has been a challenge for a long time. Distributed ledger technologies is an exciting technological advancement in the information technology world. Blockchain technologies offer an infinite range of applications benefiting from sharing economies. Our project uses blockchain technology as a service to implement distributed electronic voting systems and base architecture is loosely based on [Hjálmarsson et al. 2018] and [del [n.d.]]. The report elicitates the requirements of building electronic voting systems and identifies the legal and technological limitations of using blockchain as a service for realizing such systems. We used Multichain framework to implement permissioned Blockchain for our use-case. Our system, curbs the need to have a central Election Commission to conduct election (replacing it with Trusted third party), as EC can be rigged in many cases, thus making the whole process more secure. The system has high throughput and low latency. As a part of future work, this work can be extended to include more techniques to make the system more robust to external attacks.**

Additional Key Words and Phrases: Blockchain, ZKP, Multichain, Proof of Authority, Private Public Key Encryption

ACM Reference Format:
Anshi Agarwal, Ayushi Bansal, and Krithiga Murugavel. 2018. Blockchain Based E-Voting System - using Multichain Framework. *ACM Trans. Graph.* 37, 4, Article 111 (August 2018), 10 pages. 10.1145/1122445.1122456

1 INTRODUCTION

Voting, whether traditional ballet based or electronic voting (e-voting), is what modern democracies are built upon. In all the democracies, security of an election is an important factor that need to be considered, while building any type of voting system. The computer security field has studied the possibilities of electronic voting systems, with the goal of minimizing the cost of having a national election, while fulfilling and increasing the security conditions of an election. For a robust e-voting scheme, a number of functional and security requirements needs to be fulfilled including: transparency, accuracy, auditability, system and data integrity, secrecy/privacy, availability, and distribution of authority.

Replacing the traditional pen and paper scheme with a new election system is critical to limit fraud and having the voting process traceable and verifiable [Schaupp and Carter 2005]. Electronic voting machines have been viewed as flawed, by the security community, primarily based on physical security concerns. Anyone with physical access to such machine can sabotage the machine, thereby affecting all votes cast on the aforementioned machine.

But with the latest Blockchain technology these shortcoming of the Electronic voting machine. A blockchain is a distributed, immutable, incontrovertible, public ledger. This new technology works through four main features:

- The ledger exists in many different locations: No single point of failure in the maintenance of the distributed ledger.
- There is distributed control over who can append new transactions to the ledger.
- Any proposed "new block" to the ledger must reference the previous version of the ledger, creating an immutable chain and thus preventing tampering with the integrity of previous entries.
- A majority of the network nodes must reach a consensus before a proposed new block of entries becomes a permanent part of the ledger.

These features of the Blockchain coupled with necessary cryptographic improvisations can help in achieving the level of security and integrity that was missing in electronic voting machine. Thus, our project exhausts all the properties of Blockchain technology to build a new modern democratic voting system that is robust and tamper-proof.
We used Multichain framework to implement e-voting application, using permissioned blockchain. Our architecture tries to optimize existing models for implementing e-voting using blockchain for the requirements and considerations discussed in subsequent section. Our key contributions are:

- Exploring various Blockchain platforms, their pros and cons to select the platform, that best suits our use-case.
- Identify the key security concerns of traditional voting and thus devised the model, which doesn't require Election Commission to conduct election.
- Implement the complete system using Google Cloud VM instances to create the various components of system and using Multichain as framework.
- Evaluate the performance of the system (robustness, latency and breadth of model).

The reminder of this report is organized as follows: In section 2, we talked about related work, in Section 3, we discussed about the experimental setup, including assumptions and pre-requisites of the system, architecture details and technicalities and justification of our design choices. In section 4, we discussed about the Implementation details including how the VM instances interacted and how the system worked. In section 5, we talked about results followed by Conclusion and Discussion in Section 6 and 7 respectively.

Authors' address: Anshi Agarwal; Ayushi Bansal; Krithiga Murugavel, University of California, Davis, Davis, California.

2 RELATED WORK

In order to incorporate all the requirements for e-voting system in our blockchain based system, we need to make many design choices related to features of blockchain to be employed: public, private and permissioned blockchain, smart contracts and consensus protocol to be used. Due to some issues with public blockchain, like scalability, it is more feasible to use permissioned blockchain. A lot of work has been done in designing voting system using blockchain.

Agora [ago [n.d.]] is an end-to-end verifiable blockchain based voting solution designed for governments and institutions. Agora uses their own token on the blockchain for elections, where governments and institutions purchase these tokens for each individual eligible voter.

A Smart Contract For Boardroom Voting [McCorry et al. 2017] with Maximum Voter Privacy, proposed the first implementation of a decentralized and self-tallying internet voting protocol with maximum voter privacy using the Blockchain, called The Open Vote Network (OVN). The OVN is written as a smart contract on the public Ethereum blockchain.

Digital Voting with the use of Blockchain Technology [Andrew Barnes and Perry [n.d.]] proposed an integration of the blockchain technology to the current voting system in the UK in which the voters can vote at a voting district or on a web browser at home.

Netvote [nev [n.d.]] is a decentralized blockchain-based voting network on the Ethereum blockchain. Netvote utilizes decentralized apps (dApps) for the user interface of the system. The Admin dApp allows election administrators to set election policies, create ballots, establish registration rules and open and close voting. The Voter dApp is used by individual voters for registration, voting and can be integrated with other devices (such as biometric readers) for voter identification. The Tally dApp is then used to tally and verify election results.

Dell EMC [del [n.d.]] paper uses Multichain to implement E-voting system. The paper lists out all the specific components required and the features of each of these. But the paper lacks to give specific implementation details. Paper by [Hjálmarsson et al. 2018], provides other cryptographic aspects needed to ensure safety of identity during election process. Our base architecture takes most of the key concepts from these two papers.

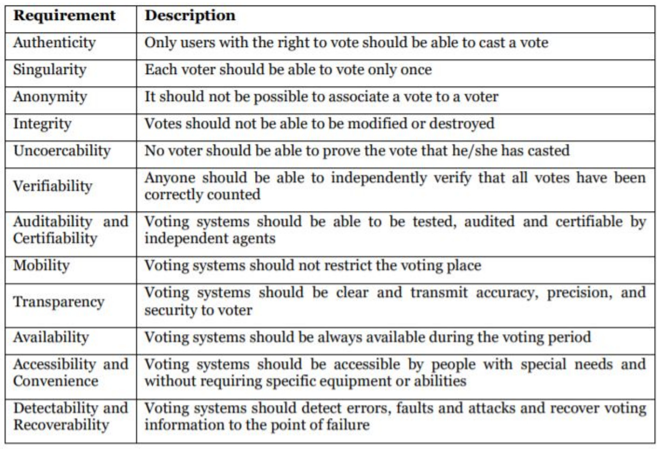
Our approach is based permissioned blockchain implementation. Public-based e-voting systems are inefficient concerning financial cost. This inefficiency occurs because of the high gas cost and the gas limits which are set for smart contracts in the network. Another risk using the public blockchain is that high traffic in the network, which could affect the throughput of votes in the system, making it less time efficient. Moreover, a 51 % attack poses a threat to a public blockchain, where any individual can sign transactions.

3 ARCHITECTURE

This section gives the high-level Architectural overview of our e-voting system, including the requirements that needs to be fulfilled fore-voting, components of our system and the design choices that we made along with the reason to back it.

3.1 Requirements

Following are the general requirements that needs to be fulfilled for implementing a practical e-voting system presented by order of relevance, from the most important to the least important [Yi 2019]:



Requirement	Description
Authenticity	Only users with the right to vote should be able to cast a vote
Singularity	Each voter should be able to vote only once
Anonymity	It should not be possible to associate a vote to a voter
Integrity	Votes should not be able to be modified or destroyed
Uncoercability	No voter should be able to prove the vote that he/she has casted
Verifiability	Anyone should be able to independently verify that all votes have been correctly counted
Auditability and Certifiability	Voting systems should be able to be tested, audited and certifiable by independent agents
Mobility	Voting systems should not restrict the voting place
Transparency	Voting systems should be clear and transmit accuracy, precision, and security to voter
Availability	Voting systems should be always available during the voting period
Accessibility and Convenience	Voting systems should be accessible by people with special needs and without requiring specific equipment or abilities
Detectability and Recoverability	Voting systems should detect errors, faults and attacks and recover voting information to the point of failure

Fig. 1. E-voting Systems Requirements

3.2 Components

The key components in our model are as follows:

- **Election Commission:** It is a government authority which contains information about the citizens of the country. The election commission specifies the election type and create aforementioned election, configure ballots and register voters. After the registration period ends, it initiates the start of election and the authority to manage the election is then transferred trusted third party.
  When a citizen wants to register as a voter, he interacts with Election Commission to verify its identity and on successful verification of identity, it is assigned a vote which it can use during voting process to vote for the desired candidate.
  When a citizen wants to contest election and thus file his nomination, it interacts with Election Commission and a public address (generated using Multichain) is generated by the Election Commission. This nominee is then added to the candidate list and other voters can vote for him.
  The detailed process of how a participant interacts with Election Commission is discussed in subsequent Section
- **Trusted Third Party:** This is a trusted entry, which is assumed to be non-malicious and its key purpose is to eliminate the need to have an Election Commission during voting process as the EC can be biased and thus influence the election process. It decides the duration of voting process. Each of the voter interacts with trusted third party during voting to cast his/her vote.
  This is basically synonymous to endorsers in Proof of Authority Consensus algorithm. But instead of having multiple endorsers, our system assumes only one central trusted third party. This idea of further extension is discussed in subsequent Section.

- **Participant:** A Participant is anyone with a valid ID who can either vote or can contest elections. The participants can be of two types: Voter and Candidate.Detailed steps that need to be followed by the voter and nominee is discussed further in the report.
  - **Voter**: Voter need to prove to EC that it is a valid citizen of the country without revealing its identity. After successful verification only, it is allowed to vote.
  - **Candidate**: Candidate also need to prove to EC that it is valid citizen of country and then have to provide a public address (generated out of Multichain) to EC which will be used by voter to transfer vote to it in voting process.

3.3 Design Choices

In order to satisfy various requirements needed in our e-voting system, we explored various design implementations (i) for making the voting system anonymous, (ii) for making the system immutable, (iii) for verifying the voters in the system and (iv) for restricting the people who can participate in the voting. We will look into each of these below.[Langhe and Langhe 2018]

Permissioned Blockchain
In order to make sure that only valid voters can be a part of the voting procedure and the valid voters can only access the blockchain, we propose to use a permissioned blockhain instead of a public blockchain. Here the voters who are verified by the election commission (government authorized entity) are only permissible to vote and gain access to add transactions and view the blockhain.

Anonymity
In order to give anonymity for a voter who is taking part in the election, the voter needs to verify to the Election commission that he is a valid ID holder without giving away his identity. This can be done using cryptographic techniques related to homomorphic encryption.
The major drawback of homomorphic encryption is that it is computation heavy and slow hence, can make the process slow. Using this is a tradeoff between latency of system and privacy/security.
The crypto technique that we are implementing for this application is Zero Knowledge Proof (ZKP).[Hjálmarsson et al. 2018]

Zero Knowledge Proof:
A ZKP [Morais et al. 2019] is a cryptographic proof detailing knowledge of an input x to an output f(x) without revealing x. In simpler words, ZKP allows you proving that you know some secret (Identity in this case) to somebody at the other "end" of communication without actually revealing it. It has three main properties:

- Completeness: If the statement is really true and both users follow the rules properly, then the verifier would be convinced without any artificial help.
- Soundness: In the case of the statement being false, the verifier would not be convinced in any scenario. (The method is probabilistically checked to ensure that the probability of falsehood is equal to zero)
- Zero-knowledge: The verifier in every case would not know any more information

An advanced version of ZKP is called **Non-Interactive zero knowledge Proof** (NIZKP). It is built by working with oneself and simulating a verifier, in such a way that the actual verifier is convinced that you couldn't have made up the NIZK proof without knowing the proven statement. This way the interaction between the verifier(EC) and the prover(voter) happens only once.

Consensus
In our model, since we are implementing a permissioned blockchain, we use proof of authority (POA) consensus algorithm [Wang et al. 2018]. In Proof of authority based networks, transactions and blocks are validated by a set of approved "safe computing base" entities called the validators (in our case the trusted third party) who is assumed to be impartial in their decisions. The validator's reputation and identity are put at stake during consensus. As an incentive, the validators get paid for the service they provide. In our model, we only have one validator, but this can be extended to include multiple validators as a future work. Moreover, using a private network limits the possibility for an eavesdropper to monitor traffic or read the incoming data.[del [n.d.]]

Multichain (Platform for Permissioned BlockChain)
Multichain is an open-source blockchain platform, it is a permission blockchain.[mul [n.d.]]Since voting is a closed event and only participants with permission can take part in the election, we are using permissioned blockchain where only authorised personal are allowed to join the network where as a public blockchain platform like Ethereum allows any number of participants to join the network.

Ethereum [Eth [n.d.]] is an open Blockchain platform that lets anyone build and use decentralized applications that leverage Ethereum Platform. Ethereum community support is very active. An advantage of using Ethereum for Voting application [Zhang et al. 2019] is that we can use the Smart contract to validate and store voting count states in the Blockchain. However, it comes at the cost of transacting in public network. Also, mining computation is heavy in the case of Ethereum Public or Private Platform to give consensus to blocks. Thus, making us choose Multichain for the our application.

The key features of Multichain are:

- Atomic two-way exchanges of assets between participants.
- Permission management.
- Provides a simple API and command-line interface

3.4 Architectural Overview

Our overall architecture is divided into two sub-sections. First is Registration process and the second is voting process.

3.4.1 ***Registration.*** During this process, Voter and Candidates register with the Election Commission. During this phase, they need to verify that they are valid citizens of the country. A lot of cryptographic concepts like ZKP, digital signature are used. In the response, voters and candidates receive necessary asset or entity that they would need during voting process. Detailed information of how the interactions occur between Election Commission and Participant is explained in subsequent section.

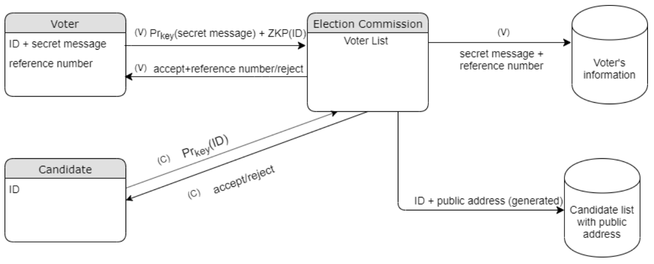

Voter
ID + secret message
reference number
(V) Pr_{key}(secret message) + ZKP(ID)
(V) accept+reference number/reject
Election Commission
Voter List
(V)
secret message + reference number
Voter's information
Candidate
ID
(C) Pr_{key}(ID)
(C) accept/reject
ID + public address (generated)
Candidate list with public address

Fig. 2. Registration Process Overview

3.4.2 ***Voting.*** During this phase, Voter interact with Trusted Third Party to first verify its identity. This acts a second check for only allowing valid voter to vote. Then the voter transfers it vote (asset) to concerned candidate.

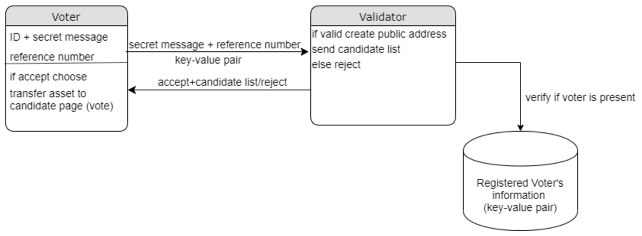

Voter
ID + secret message
reference number
if accept choose
transfer asset to candidate page (vote)
secret message + reference number
key-value pair
accept+candidate list/reject
Validator
if valid create public address
send candidate list
else reject
verify if voter is present
Registered Voter's information (key-value pair)

Fig. 3. Voting Process Overview

3.5 E-voting scenarios

This section specifies in detail the various e-voting scenarios that can arise before, during and after elections, in further sections we will discus how to solve all these issues.

- **Multiple voting:** This is one of the most important scenarios which needs to be handled carefully. When a voter tries to cast his vote multiple times, he/she should not be allowed to cast his vote more than once at any scenario.
- **Identity of the voter/candidate verified:**This scenario deals with the voters who are not allowed to vote in the election, for eg, if they do not belong to the district where elections are taking place.
- **Identity impersonation:** This scenario makes sure that any voter cannot impersonate as someone else during the voting procedure. During registration phase, we need to enforce that the voter is genuine and there was no case of identity theft.
- **Election Commission Rigging:** In this scenario we deal with the use case when any political party tries to manipulate the elections by influencing the commission. In this case the election commission can no longer be trusted and can release the identity of voter to the political party.
- **Voter can verify his transaction:** This allows the voter to verify his transfer of asset/vote to the candidate to make the voting process tamper free and transparent.
- **Tampering while tallying votes:** Generally in a voting, there is a possibility of tampering. While tallying the votes an individual can make a counting mistake or count more for one candidate which can affect the decision.

4 EXPERIMENTAL SETUP

We utilized Google Cloud Platform for creating virtual machines for all the entities of our architecture i.e. voter, candidate, election commission and trusted third party. All the virtual machines reside in the same private network in our implementation, but can interact even if placed in different sub-networks. We have utilized Python to create an application on top of the multichain API framework for e-voting.
We utilized the multichain API for the underlying blockchain requirements which we explain in the the below sections.

4.1 Installing multichain

In this we explain the procedure we utilized for downloading and installing multichain on the respective virtual machines.

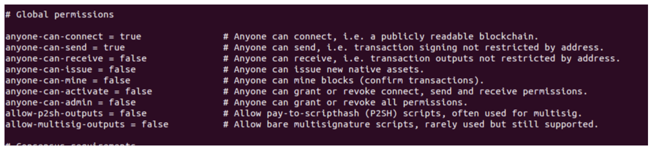
(1) Download multichain in all the instances from the multichain website.
wget http://www.multichain.com/download/multichain-1.0-alpha-21.tar.gz
(2) Unzip the tar file.
tar -xvzf multichain-1.0-alpha-21.tar.gz
(3) Store it in the execution enviornment in the VM or real system. cd multichain-1.0-alpha-21
sudo mv multichaind multichain-cli multichain-util /usr/local/bin

4.2 Creating a blockchain

In the multichain framework, blockchain is synonymous to multichain, so we will use both interchangeably to refer to the same.
Create Blockchain in each VM or real machine using the below command:
multichain-util create <blockchainname>

4.3 Adjusting Blockchain Settings

This section describes the adjustment in the configuration of the multichain that we perform before it is instantiated. We edit a params.dat file which has all the configurations of the multichain described according to the requirements of our system architecture.
Since the multichain is created on the election commisssion as we will describe in the implementation details later, all these changes in the params.dat file are performed on the Election Commission VM.
On the EC server run,
nano /.multichain/survey/params.dat



```
# Global permissions

anyone-can-connect = true               # Anyone can connect, i.e. a publicly readable blockchain.
anyone-can-send = true                  # Anyone can send, i.e. transaction signing not restricted by address.
anyone-can-receive = false              # Anyone can receive, i.e. transaction outputs not restricted by address.
anyone-can-issue = false                # Anyone can issue new native assets.
anyone-can-mine = false                 # Anyone can mine blocks (confirm transactions).
anyone-can-activate = false             # Anyone can grant or revoke connect, send and receive permissions.
anyone-can-admin = false                # Anyone can grant or revoke all permissions.
allow-p2sh-outputs = false              # Allow pay-to-scripthash (P2SH) scripts, often used for multisig.
allow-multisig-outputs = false          # Allow bare multisignature scripts, rarely used but still supported.
```

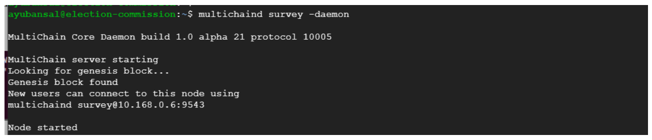
Fig. 4. params.dat File of Mulichain

Executing this command opens the params.dat file where we can change some global configurations according to our need, in our case since we have a verification step before instantiating into multichain we are going to set anyone can connect to true as visible in the Figure 4
Since we give only 1 asset to each voter and none to candidates. We set the send permission to anyone.
Since we have only one signal at a time we set multiple signal option to False. The rest of the options are left to default. These options can be changed according to the architecture requirements to make the application more flexible.

4.4 Initialize the Blockchain

In this section, we describe the prerequisite steps to instantiate the multichain before we begin adding blocks to it and perform any other operation.

(1) When we want to instantiate the multichain platform and we start running daemon process. This step is carried out by the Election Commission in the beginning before starting registration.
multichaind <blockchain-name> -daemon



```
ayubansal@election-commission:~$ multichaind survey -daemon

MultiChain Core Daemon build 1.0 alpha 21 protocol 10005

MultiChain server starting
Looking for genesis block...
Genesis block found
New users can connect to this node using
multichaind survey@10.168.0.6:9543

Node started
```

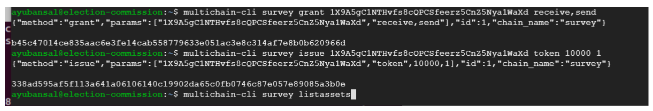
Fig. 5. Initialize Blockchain

As we can see in the Figure 5, an address is assigned which can be used by the other users in the network to connect to the multichain and the node is started.

(2) This address is sent to the voters after their validity is established by the Election Commission so that the voter can use it to connect to the multichain network.
(3) After the voter connects to the multichain network it generates its own address using the following command:
multichain-cli <blockchain name> getnewaddress
(4) This specific address of the voter is sent back to the EC for letting it give permissions to the voter.
(5) The EC sets the permission for the voter using the address it just received.
multichain-cli <blockchainname> grant <address> receive,send

4.5 Issue the "token" Asset

In this section, we describe the main unit which is exchanged in our system for vote transfer from a voter to the candidate which marks as a transaction in the block of the blockchain.
We create an asset named "vote" which cannot be subdivided, and place 10,000 of them on the election commission. The number 10,000 sets a limit to 10,000 voters in this system.
multichain-cli survey issue <EC-address> token 10000 1



```
ayubansal@election-commission:~$ multichain-cli survey grant 1X9A5gC1NTHvfs8cQPCSfeerz5CnZ5Nya1WaXd receive,send
{"method":"grant","params":["1X9A5gC1NTHvfs8cQPCSfeerz5CnZ5Nya1WaXd","receive,send"],"id":1,"chain_name":"survey"}

b45c47014ce835aac6e3fe14cab558779633e051ac3e8c314af7e8b0b620966d
ayubansal@election-commission:~$ multichain-cli survey issue 1X9A5gC1NTHvfs8cQPCSfeerz5CnZ5Nya1WaXd token 10000 1
{"method":"issue","params":["1X9A5gC1NTHvfs8cQPCSfeerz5CnZ5Nya1WaXd","token",10000,1],"id":1,"chain_name":"survey"}

338ad595af5f113a641a06106140c19902da65c0fb0746c87e057e89085a3b0e
ayubansal@election-commission:~$ multichain-cli survey listassets
```

Fig. 6. 1000 tokens issued to EC

We will discuss in the implementation details how a voter receives his vote asset after his verification by the election commission and how we make sure that each voter casts only one vote and receives only one asset during the whole voting process.
At any point of time, any entity (i.e. election commission, voter or candidate) can check the asset count in the multichain by utilizing the following API command:
multichain-cli <blockchain name> listassets
These are the total vote assets present in the system and do not represent asset of any individual entity.

4.6 Sending asset from one entity to other

In this section we describe how one entity can transfer asset to another entity.
Since in the above step we instantiated our multichain with 10,000 vote assets using election commission server, the EC has the authorization to distribute these assets to the voter after verification.
Also, in the voting process the voter will transfer his vote asset to the candidate he wishes to vote for. The multichain API provides the below mentioned command for this transaction.
multichain-cli survey sendassettoaddress <receiver-address> token <qty>
Here in case of election commission sending asset to the voter, we can replace the receiver address by the voter's address and qty by 1. And in case of voting the voter can replace receiver address by the candidate's address he wants to cast vote for.

4.7 Verifying asset count

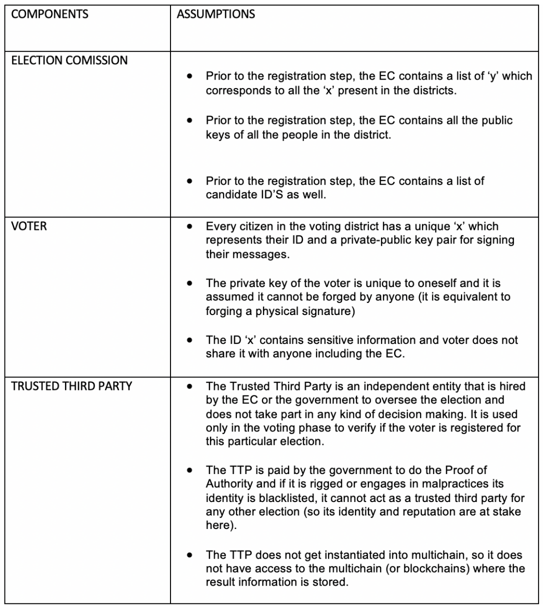
Any entity can verify his/her asset count at anytime during the process using the following command:
multichain-cli survey gettotalbalances
This command takes in account all the transactions in the blocks of the multichain and calculates the total balance that a specific entity owns.
To verify the correctness of the system, at any point in time all the voters must have either 0 or 1 vote asset to them, if not then there is some malicious transaction that occurred.

5 IMPLEMENTATION DETAILS

We start with virtual machines for all the entities of the system architecture as detailed in the experimental details section.
Before the e-voting is started, multichain is installed on each machine with the help of a shell script. The Election Commission creates the multichain and edits the params.dat file according to the architecture requirements. EC initializes the blockchain and gets a new address for itself. The EC creates an asset named vote in a specified quantity. For the purpose of our project we issued 10,000 vote assets, which means our system can support maximum 10,000 voters for one election.
After all the initiation process, the e-voting starts with the voter registration and then finally ends up in voting. We will explore them in detail in the following subsections.

5.1 Assumptions

Even before the registration for the election starts, there are a few things and settings that needs to be present in our entities, are system makes some prior assumptions that these are already present with these entities.



COMPONENTS	ASSUMPTIONS
ELECTION COMISSION	• Prior to the registration step, the EC contains a list of 'y' which corresponds to all the 'x' present in the districts. • Prior to the registration step, the EC contains all the public keys of all the people in the district. • Prior to the registration step, the EC contains a list of candidate ID'S as well.
VOTER	• Every citizen in the voting district has a unique 'x' which represents their ID and a private-public key pair for signing their messages. • The private key of the voter is unique to oneself and it is assumed it cannot be forged by anyone (it is equivalent to forging a physical signature) • The ID 'x' contains sensitive information and voter does not share it with anyone including the EC.
TRUSTED THIRD PARTY	• The Trusted Third Party is an independent entity that is hired by the EC or the government to oversee the election and does not take part in any kind of decision making. It is used only in the voting phase to verify if the voter is registered for this particular election. • The TTP is paid by the government to do the Proof of Authority and if it is rigged or engages in malpractices its identity is blacklisted, it cannot act as a trusted third party for any other election (so its identity and reputation are at stake here). • The TTP does not get instantiated into multichain, so it does not have access to the multichain (or blockchains) where the result information is stored.

Fig. 7. Assumptions

5.2 Voter Registration

The voter registration is the first step in the electronic voting. This step is used to verify the identity of the voter without compromising his identity to the election commission in case it is rigged. The registration of voter is comprised of two parts, first verification by the election commission and then receiving multichain access on successful verification.
To avoid identity disclosure, we utilize Zero Knowledge Proof technique where in the identity of the voter is sent in a form where it cannot be extracted and used against the voter.

Zero Knowledge Proof Verification

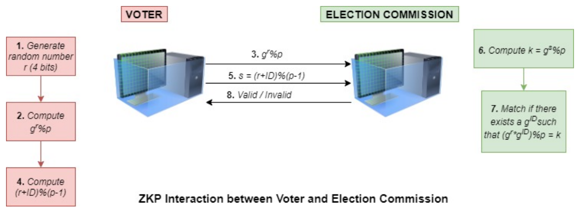

VOTER
ELECTION COMMISSION
1. Generate random number r (4 bits)
2. Compute $g^r\%p$
4. Compute (r+ID)%(p-1)
3. $g^r\%p$
5. s = (r+ID)%(p-1)
8. Valid / Invalid
6. Compute $k = g^s\%p$
7. Match if there exists a g^{ID} such that $(g^r * g^{ID})\%p = k$
ZKP Interaction between Voter and Election Commission

Fig. 8. ZKP

In the Figure 8 we have a voter and a election commission exchanging messages to prove the identity of the voter. We have a generator g and prime number p which is decided for the ZKP algorithm which is known to both voter and election commission. The following steps take place in sequence:

(1) The voter generates a random number of 4 bits.
(2) The voter computes $g^r\%p$.
(3) The voter sends the computed value $g^r\%p$ to the election commission.
(4) After this the voter computes a value $s = (r + id)\%(p - 1)$, here r is the random number generated in the step (1).
(5) This value is sent to the election commission.
(6) The election commission computes $k = g^s\%p$
(7) The election commission has a list of valid g^{id} which it matches by checking if there exists a g^{id} in this list such that $(g^r * g^{id})\%p == k$

This makes sure that the identity of the voter is verified without disclosing his identity.

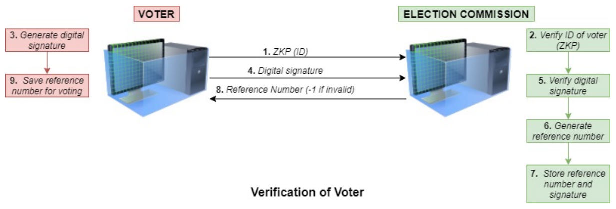

VOTER
ELECTION COMMISSION
3. Generate digital signature
9. Save reference number for voting
1. ZKP (ID)
4. Digital signature
8. Reference Number (-1 if invalid)
2. Verify ID of voter (ZKP)
5. Verify digital signature
6. Generate reference number
7. Store reference number and signature
Verification of Voter

Fig. 9. Verification

The verification of voter can be compiled in the below mentioned steps in reference to Figure 9:

(1) The voter exchanges ZKP messages with the election commission as described in detail above.
(2) The election commission verifies the identity.
(3) The voter generates a digital signature to avoid impersonation.
(4) The voter sends his digital signature to the EC for verification.
(5) The election commission verifies the digital signature of the voter.

(6) If the voter has a valid identity proof and a valid digital signature, the EC generates a unique reference number for this election for the specific voter.
(7) The EC stores the generated reference number and the signature for future verification when voting starts.
(8) The reference number is sent back to the voter for future reference. -1 is sent in case any verification resulted in false.
(9) The voter saves the reference number for voting.

After the voter is verified, the voter needs to get access to the multichain.

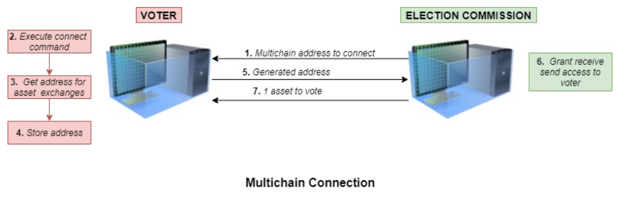

VOTER
ELECTION COMMISSION
2. Execute connect command
3. Get address for asset exchanges
4. Store address
1. Multichain address to connect
5. Generated address
7. 1 asset to vote
6. Grant receive send access to voter
Multichain Connection

Fig. 10. Multichain access

As we see in the Figure 10 the following steps take place in getting a multichain for the voter.

(1) The election commission after verifying the voter sends the address of the multichain to connect to.
(2) The voter executes the multichain API command to connect to the received multichain address.
(3) The voter uses multichain API to get a new address for the future operations.
(4) The voter stores its address for future reference during voting.
(5) The voter sends the generated address to the EC.
(6) The EC grants receive and send access to the voter using its address. This allows the voter to perform transactions and add them to the block in the multichain network.
(7) The voter sends 1 vote asset to the voter to perform voting. Since we give only 1 asset vote in this step, it ensures that each voter can only vote once by sending his/her asset to his/her chosen candidate.

This completes the voter registration procedure successfully with important benefits such as identity protection, prevention of impersonation and one person one vote.

5.3 Candidate Registration

Along with the voter registration, the EC is also performing candidate registration at the same time in which it verifies the identity of the candidate and allows him multichain access.

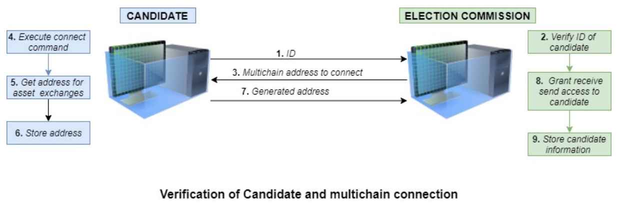

CANDIDATE
ELECTION COMMISSION
4. Execute connect command
5. Get address for asset exchanges
6. Store address
1. ID
3. Multichain address to connect
7. Generated address
2. Verify ID of candidate
8. Grant receive send access to candidate
9. Store candidate information
Verification of Candidate and multichain connection

Fig. 11. Candidate Registration

In Figure 11, we perform the following steps for the candidate registration and verification,

(1) The candidate sends his ID to the election commission. We do not need to perform ZKP here, since the identity of candidates standing up for an election is known to all and does not need to be protected or hidden.
(2) The election commission verifies the ID of the candidate to check if he is a valid candidate for the specific election. We assume that the election commission has a list of valid candidates for this election from the government already.
(3) If verified, the EC sends the multichain address to the candidate for him/her to be able to connect to it.
(4) The candidate uses the multichain API command to connect to the multichain using its address received.
(5) It gets an address for the future asset exchanges using multichain API.
(6) The generated address is stored safely for future use.
(7) The address is sent to the EC.
(8) The EC grants receive send access to the candidate
(9) Finally the EC stores the candidate information along with his generated address for voting process.

This completes all the prerequisite steps to the voting process and now we can introduce the trusted third party and allow the voters to cast their votes.

5.4 Trusted Third Party

Now the voter can proceed onto the voting procedure. To maintain the sequential nature between registration for voting and actually voting, we close the registration process after some time depending on the scale of the specific election. The process of registration and voting were not parallel since the trusted third party is introduced in voting and before we start voting, the trusted third party should have access to a list of all valid voters.
As discussed in our experimental setup details, the trusted third party acts as a endorser and performs Proof of Authority consensus before allowing the voter to add its block to the multichain.

Trusted Third Party communicates with EC
In this step the TTP communicates with the EC after the registration procedure (both voter and candidate registration) is over to receive the voter and candidate information stored by the EC.

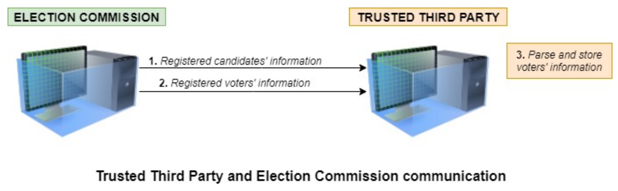

ELECTION COMMISSION
TRUSTED THIRD PARTY
1. Registered candidates' information
2. Registered voters' information
3. Parse and store voters' information
Trusted Third Party and Election Commission communication

Fig. 12. TTP-EC communication

From the Figure 12, it is easy to see that the election commission sends the candidate and voter information to the trusted third party. Here the TTP parses and stores the voter information in a tuple format for easy verification of voter before it is allowed to proceed to voting. The TTP does not need to parse the candidate information and can send it directly to a valid voter from which the voter can choose one to vote for.
After receiving all the candidate and voter information the TTP can start receiving voting requests from the voters.

Trusted Third Party verifies voters
In this the TTP performs the consensus algorithm i.e. the Proof of Authority by authorizing any incoming voters to vote (add a block to the multichain) based on their reference number and signature stored previously in the registration phase.

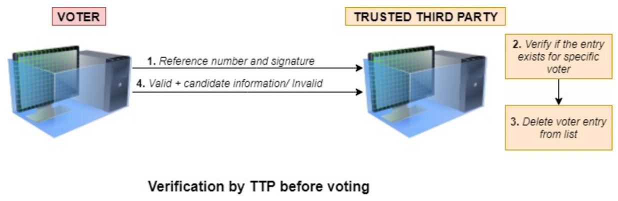

VOTER
TRUSTED THIRD PARTY
1. Reference number and signature
4. Valid + candidate information/ Invalid
2. Verify if the entry exists for specific voter
3. Delete voter entry from list
Verification by TTP before voting

Fig. 13. TTP-Voter communication

It can be easily inferred from the Figure 13 that we perform voter verification using the list received from the EC.
The EC sends the file containing mapping of reference numbers to the signature of specific voters. The following steps are performed,

(1) The voter who wishes to vote sends his reference number received in the registration phase and the signature generated in the registration phase.
(2) The trusted third party scans the list of the voter entries and matches is there is an entry corresponding the received pair.
(3) If an entry is found, the TTP deletes it from the list, so that the voter cannot come again for verification or voting process.
(4) If found valid, the voter is sent back a valid confirmation and the valid candidates addresses and names to choose from, otherwise an invalid is sent.

The assumption here is that if the voter is found invalid, then he won't receive the candidates' addresses and thus can't a vote at all, therefore making sure that no invalid votes are casted and counted. Now a valid voter has a vote asset to transfer and list of all candidates' addresses to choose from and hence can go ahead an perform voting i.e. transfer his vote to chosen candidate and add his valid transaction (block) into the multichain.

5.5 Voting

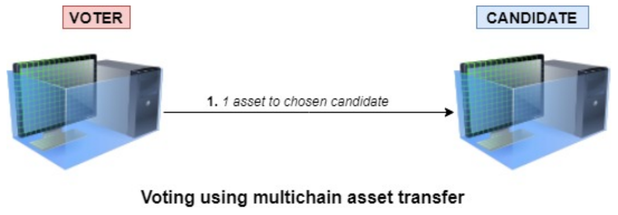

VOTER
CANDIDATE
1. 1 asset to chosen candidate
Voting using multichain asset transfer

Fig. 14. Voting

In the Figure 14 the voter is transferring a vote asset to the candidate of choice, this adds a block to the multichain signifying the transaction between the specific voter and specific candidate of one vote transfer.
This finally marks the completion of the voting process.

5.6 Blockchain

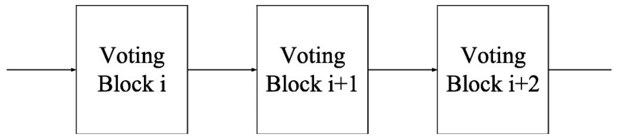
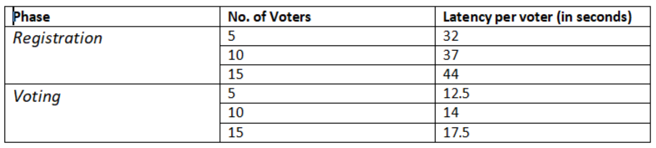


Fig. 15. Blockchain

When the voter transfers his asset (vote) to the candidate using the address created by multi-chain networks, the transaction is recorded as a block in the blockchain initiated in the multichain network.The final results of the voting process can be declared after checking the vote asset count of all the candidates and declaring the winner as the one having the highest vote assets in his/her balances
GitHub Repo Link Blockchain-Based-e-voting

6 EVALUATION

6.1 Quantitative evaluation

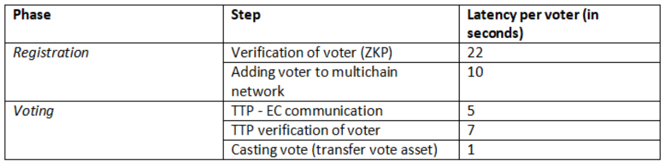


Phase	No. of Voters	Latency per voter (in seconds)
Registration	5	32
	10	37
	15	44
Voting	5	12.5
	10	14
	15	17.5

Fig. 16. Registration and Voting Latencies

From the Figure 16, we see that the overall latency increases with the increase in scale of the system. As we increase the number of voters participating in the network, we see a gradual increase in the latency.

The average latency over all the performed experiments came out to be 37.67 seconds for registration and 14.67 seconds for voting. The latency for registration is much more in comparison to the voting process because of the verification of the voter by the election commission. The Zero Knowledge Proof step in the registration process takes considerable amount of time and hence increases the overall latency. This is a trade off between security and throughput since by including ZKP algorithm we ensure protection for the voter from any political party but it results in increase in latency.
Despite of this the throughput is exponentially better than the manual voting procedure which usually takes a whole day.
Let us explore the detailed amount of time taken for each step within registration and voting.



Phase	Step	Latency per voter (in seconds)
Registration	Verification of voter (ZKP)	22
	Adding voter to multichain network	10
Voting	TTP - EC communication	5
	TTP verification of voter	7
	Casting vote (transfer vote asset)	1

Fig. 17. Latency of steps in registration and voting for 10 voters

From Figure 17, the hypothesis that the ZKP i.e. verification of voter takes the most time in the registration phase is proven, since we can easily see that it takes more than twice the amount of time taken in other steps in the registration phase.
For voting we see that the amount of time required in the TTP-EC communication and Proof of Authority consensus (verification of voter by TTP before voting) is almost equal. The actual amount of time taken for casting the vote (transfer of vote asset from voter to candidate) is small since it is just a transaction from voter's wallet to the candidate's wallet and adds a block for this transaction in the multichain.

6.2 Qualitative evaluation

We evaluated our e-voting system based on the requirements listed out earlier that a practical e-voting system need to follow. Following are the various scenarios and cases that our system can handle:

- **One voter can cast only one vote:** This is one of the most important scenarios we handled. We are granting only 1 asset per voter by Election Commission during registration process. This avoids voter to cast multiple vote as he/she only has one vote issued by EC. After issuing 1 vote, EC deletes the public-key of the voter in its database, so that if the voter requests for another asset, it will be regarded as invalid citizen itself.
- **Identity of the voter/candidate verified:**This scenario deals with the voters who are not allowed to vote in the election, for eg. if they do not belong to the district where elections are taking place. We avoid voters from other constituency to vote by checking their identity during registration phase using zkp concepts.
- **Identity impersonation:** This scenario makes sure that any voter cannot impersonate as someone else during the voting procedure. During registration phase, we enforce that the voter send his secret message to the election commission by signing it with his own private key, and since the private key of a person is not known to anyone else , this ensures that it acts as a digital signature which no one can forge.
- **Election Commission Rigging:** In this scenario we deal with the use case when any political party tries to manipulate the elections by influencing the commission. In this case the election commission can no longer be trusted and can release the identity of voter to the political party. This invades the privacy of the voter in duress. For election commission to not be able to identify any voter, we are using ZKP(Zero Knowledge Proof) where during the registration phase the voter does not directly disclose his ID, instead he sends a value (x) which is related to his ID and is used by the EC to solve a logarithmic function to verify if the voter has the ID or not. This masks the identity of the voter from the election commission during registration phase. Also, we do not include the election commission during voting phase, which ensures that the election commission is unwary of who is voting.
- **Voter can verify his transaction:** This allows the voter to verify his transfer of asset/vote to the candidate to make the voting process tamper free and transparent. For the voter to verify his vote, we provide him/her with a transaction ID after his/her voting and he is given access to the blockchain so that he can check if his vote was counted correctly or not anytime during the process.
- **Tampering while tallying votes:** Generally in a voting, there is a possibility of tampering. While tallying the votes an individual can make a counting mistake or count more for one candidate which can affect the decision. To make the whole voting tallying procedure tamper free, we utilize blockchain where it is impossible to modify any previous transactions since it will change the hash of that block hence affecting the whole blockchain and it will fail during verification.

7 CONCLUSION

Blockchain is an emergent technology that is evolving on a daily basis. Similarly to other new technologies, blockchain adoption by organizations will evolve step by step, through small efforts, some failures, many successes and, hopefully, widespread adoption. The challenge is to assure the right balance between ensuring the governance, safety and resilience of the blockchain developed solutions while not infringing the innovation and development of this fast-evolving technology.

In this project we aim to contribute to the consolidation of the blockchain applications ecosystem by developing a blockchain-based electronic voting platform. It is widely recognized that electronic voting solutions have the potential to increase the level of participation in elections, due to the convenience of remote voting, while simultaneously reducing costs by eliminating the need for ballot printing or electronic voting machines purchase and maintenance. By using blockchain technology in the development of an electronic voting solution we expect to enhance the security of those solutions and provide a transparent and auditable electronic voting platform.

7.1 Shortcomings

- One of the major shortcomings of the architecture is single point of failure, i.e. if the trusted third party fails then it is impossible to perform the voting phase. To overcome this we can have multiple validators/endorsers instead of a single third party who can validate a voter's registration during the voting phase and reach a consensus among themselves using Proof of Authority as to the voter is valid or not.
- The steps are sequential, many voters cannot register at the same time, the EC can handle the verification and authentication of only one voter or candidate at a given time.
- Since everything is based online, it is impossible to check if anyone is voting under the influence of someone, it is the voter's responsibility to make the decision without somebody else influence.

7.2 Future Work

The method implemented by us for e-voting system using Blockchain, handles most of the voting scenarios and can be incorporated for practical applications. Scalability of our model depends on the capacity of the servers that can connect to the blockchain that is defined by the parameters of Multichain framework.

This working can be extended by including more Trusted Third Party servers (multiple instead of one) so that Proof of Authority can be better realised and thus avoiding any invalid transfer of votes. We can also make different TTP for each district, so that for nationwide voting cases, district-wise voting can be carried out separately. Also, we can have a shared database between Election Commission and Trusted Third Party with some security restriction, so that the overhead of transfer of voter's and candidate's information after registration process can be avoided.

REFERENCES

[n.d.]. https://pdfs.semanticscholar.org/84c7/c5b9df300d5d282038684654e2d47998b3dd.pdf

[n.d.]. https://static1.squarespace.com/static/5b0be2f4e2ccd12e7e8a9be9/t/5b6c38550e2e725e9cad3f18/1533818968655/Agora_Whitepaper.pdf

[n.d.]. https://citizendata.network/netvote/

[n.d.]. . https://www.multichain.com/.

[n.d.]. . https://gavwood.com/paper.pdf.

Christopher Brake Andrew Barnes and Thomas Perry. [n.d.]. Voting with the use of Blockchain Technology. ([n. d.]).

F. Þ. Hjálmarsson, G. K. Hreiðarsson, M. Hamdaqa, and G. Hjálmtýsson. 2018. Blockchain-Based E-Voting System. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. 983–986. https://doi.org/10.1109/CLOUD.2018.00151

Vaishnavi S. Langhe and Name-Vaishnavi S. Langhe. 2018. " E-voting using block chain Technology.

Patrick McCorry, Siamak F. Shahandashti, and Feng Hao. 2017. A Smart Contract for Boardroom Voting with Maximum Voter Privacy. *IACR Cryptology ePrint Archive* 2017 (2017), 110.

Eduardo Morais, Tommy Koens, Cees van Wijk, and Aleksei Koren. 2019. A Survey on Zero Knowledge Range Proofs and Applications. *CoRR* abs/1907.06381 (2019). arXiv:1907.06381 http://arxiv.org/abs/1907.06381

Ludwig Schaupp and Lemuria Carter. 2005. E-voting: From apathy to adoption. *J. Enterprise Inf. Management* 18 (10 2005), 586–601. https://doi.org/10.1108/17410390510624025

Wenbo Wang, Dinh Thai Hoang, Zehui Xiong, Dusit Niyato, Ping Wang, Peizhao Hu, and Yonggang Wen. 2018. A Survey on Consensus Mechanisms and Mining Management in Blockchain Networks. *CoRR* abs/1805.02707 (2018). arXiv:1805.02707 http://arxiv.org/abs/1805.02707

Haibo Yi. 2019. Securing e-voting based on blockchain in P2P network. *EURASIP Journal on Wireless Communications and Networking* 2019 (2019), 1–9.

Qixuan Zhang, Bowen Xu, Haotian Jing, and Zeyu Zheng. 2019. Ques-Chain: an Ethereum Based E-Voting System. *CoRR* abs/1905.05041 (2019). arXiv:1905.05041 http://arxiv.org/abs/1905.05041