Permissioned Blockchain based E-Voting System using Hyperledger Fabric

Manav Ranawat
Department of Information Technology
Sardar Patel Institute of Technology
Mumbai, India
manavranawat0@gmail.com

Luv Gupta
Department of Information Technology
Sardar Patel Institute of Technology
Mumbai, India
luvkoolg15@gmail.com

Chirag Jain
Department of Information Technology
Sardar Patel Institute of Technology
Mumbai, India
chiragajain291@gmail.com

Varsha Hole
Department of Information Technology
Sardar Patel Institute of Technology
Mumbai, India
varsha_hole@spit.ac.in

***Abstract*— With the rise of democracy, elections have been accused quite much of the lack of transparency and security. Although societies worldwide are rapidly adopting technology, creating a secure E-voting system that offers fairness and anonymity to current voters while ensuring transparency & flexibility has always been a challenge for a long time. The current EVM system reduces the time for casting a vote and announcing the results compared to the traditional paper ballot system; however, it still has many issues that can put the election authorities at risk. People need to know that their vote is not being tampered with and is successfully accounted for by the system. General e-voting system uses a centralized system which gives one organization complete control over the system. Hence the E-voting system can be implemented using Blockchain technology, which would provide a secure and decentralized e-voting platform, i.e., peer-to-peer transaction, and maintains a ledger where every vote cast will be considered allowing the users to see the results in real-time without having the permission to edit the vote after election gets over. It involves using Hyperledger fabric and Chaincode to create a highly maintainable, large-scale, and cost-effective E-voting solution within a personalized private blockchain.**

Keywords*— *E-voting, Blockchain, Hyperledger, Chaincode, Distributed Ledger, Private Network, Security, Privacy

I. INTRODUCTION

Elections play an essential role in a democratic country such as India, where the country's residents elect the leader. Elections keep a nation functioning reasonably as they allow individuals the option to choose their government. But with great power comes the responsibility of making sure that the Elections are conducted in all fairness. Looking at the current situation of conducting Elections, it won't be wrong to say that there is a vast mistrust between the people and the authorities' running elections. This is because of previous experiences where a lot of fraudulent activities have been performed during the Election process. Also, elections are a long, tiring, and cumbersome process which causes people to avoid it altogether. Thus, we need to design a voting process that needs to be secure, reliable, and easily accessible to the people.

Old conventional voting systems like Ballot based voting have been part of the voting process for quite some time. A voter marks their vote on a paper, puts it in a sealed ballot box. Once the election process is over, the ballot boxes are opened before the election authority, and the counting process begins manually. The election authority compiles the results and announces the winner. This manual procedure of counting votes takes a lot of time and is not error-free. Incidents like booth capturing, ballot stealing, and fake/bogus paper votes are very common in such a process which gradually led to its downfall.

In the late '90s, the second voting method, the Electronic Voting Machine (EVM), was introduced which served as an alternative to paper-based voting systems. It tried to overcome the existing problems of the ballot system; however, it came with its problems. The EVM could be re-programmed or replaced with a corrupt memory chip by a malicious person who had access to the machines, which can alter the election results. The most critical issue with the device is a centralized database which means if malware is inserted, it will tamper with the entire database. Therefore, the EVM's are subject to security and distrust worldwide.

To address the difficulties which come with traditional voting systems as mentioned above, Digital Voting or E-voting methods can be implemented to enhance reliability, secrecy, integrity and reduce investment in workforce and technological equipment. But just like any other technology E-voting too has its disadvantages which need to be kept in mind to make a near-optimum Voting system. A whole new field of crime has emerged - identity theft, which makes Fake voting one of the cruel practices throughout the voting process. Furthermore, intruders can significantly impact the poll or calculation of votes by damaging the intelligent systems. Thus, specific protection policies need to be put in place which guarantees a consistent voting and counting process.

To dissipate the existing problems of both conventional and E-voting systems, Blockchain can be added to improve upon the existing E-Voting systems. Blockchain technology is a decentralized ledger maintained within a distributed network of peer nodes that maintains a coherent understanding of truth. Blockchain technology is used in many cryptocurrencies such as Bitcoin and Ethereum. Blockchain eliminates the need for a central server to manage the network, thus ensuring trust. The nodes in the network each maintain a copy of the ledger which includes transactions of the system that have been validated by a consensus protocol and then put into blocks on the network. These blocks and in turn the transactions are permanent and immutable.

In an E-voting System, these transactions are in the form of Votes. Blockchain technology ensures that each voter can cast his vote only once which cannot be modified at a later stage. Blockchain-enabled e-voting systems reduce voter fraud and increase voter access. Any eligible voter can vote anonymously from any corner of the world using a computer or a laptop.

Many of the existing Blockchain technologies like Bitcoin and Ethereum use public permissionless blockchain technology. Public blockchains are open to anyone where participants can join the network and interact anonymously. But an E-voting system demands some form of authentication to make sure that the votes are coming from an eligible voter. Also, some form of

authorization should exist in the network to make sure that no single user can operate on the network independently. This is where Hyperledger Fabric, which is an open-source enterprise-grade permissioned distributed ledger technology (DLT) comes into play.

Hyperledger Fabric provides a plethora of features that are required in a secure E-Voting System. Hyperledger can use pluggable Consensus Protocols that do not require a native cryptocurrency which eliminates the costly mining process thus they can work at the same operational cost as any other distributed system. The fabric has a highly modular and configurable architecture. It also comes with Membership arrangements for ensuring who can get an entry into the system, thus only eligible voters can use the network. Since hyper-ledger fabric offers parallel transactions and limits the number of users in the system, it increases the throughput and number of transactions per second. The combination of all these design features makes Fabric one of the better performing platforms available today both in terms of transaction processing and transaction confirmation latency, and it enables privacy and confidentiality of transactions and the smart contracts that implement them.

The remaining part of the proposed paper has been organized into six sections. Section II gives a critical assessment of previous work published. Section III describes the proposed system; in Section IV we discuss the Methodology. Section V details the experimental analysis obtained. Section VI illustrates the conclusions, and finally, Section VII talks about the future work.

II. Related Work

The problems in electronic voting are quite complicated due to many requirements. Therefore, there are many works devoted to this research that proposes different approaches ranging from the use of public blockchain to the use of private blockchain with different cryptographic techniques.

The methodology proposed in paper [1] makes use of Framework as a Service (FaaS) which has the specialty of both Platform as a Service (PaaS) and Software as a Service (SaaS). It uses Hyper-ledger fabric based FaaS which encapsulates several levels of application services within the infrastructure. FaaS makes the entire system platform independent which means it can be compatible with any existing government election system.

The paper [2] proposes the use of different technologies to ensure a safe and smooth election process. Blind signature is implemented which is a digital signature mechanism with an additional property that doesn't allow the one signing to know what exactly is to be signed. Idemix or Identity Mixed developed by IBM is used to ensure the anonymity and privacy of the voters. The paper also creates a possibility of cancellation of the registration. Observers were added to detect any presence of fraud attempts.

Paper [3] tries to overcome the disadvantage of the existing system by incorporating hyper-ledger private blockchain. A wallet is created for every candidate in a political race to challenge in. This decision is made by political decision overseers. For a voter, a private key is generated from which he/she can decode the polling form and vote for a candidate. After the voting phase is over, each voter gets a hash that will be used if the voter wants to check the results after the elections. Symmetric key cryptography is used to encrypt the data to provide anonymity and security.

To prevent fake votes, paper [4], [5], [13] requires voters to verify with a unique voter ID issued during the time of registration along with fingerprint, OTP, and facial recognition as part of two-factor authentication. SHA one way hashing is used to generate a fixed-length hash containing voter id and the vote.

Paper [6], [7], [9], [17] designed a system where a voter after registration gets a crypto wallet with a certain amount of currency. When a vote is made then a transaction fee is transferred from the voter's wallet to the candidate ensuring that the voter can vote only once because he won't have the amount to make another vote. The paper uses Ganache, which uses the Truffle ecosystem to give a local and virtual blockchain for testing. This allows development of private blockchain for Ethereum. Meta-mask is used to generate a crypto wallet. It allows controlling the money and stores the private keys generated by a 12-word seed phrase.

Paper [8] proposes a decentralized public blockchain system that solves the problems raised by EVM. It uses the Merkle Tree encryption algorithm to ensure the integrity of the data stored in the node. Since one cannot track the voter's vote, it maintains voters' anonymity. Their proposed system allows offline and online voting. Here offline users are authenticated using fingerprint biometric whereas online users have a login portal.

Smart Contracts are implemented to reduce the challenges faced by the adoption of blockchain technology. Paper [10], [11] uses smart contracts running in the chain to ensure security and privacy while the hash generated from voter information provides scalability and anonymity. A miner is nominated by a smart contract to improve the speed of transactions. The miner generates the desired hash with Proof of Work as the consensus algorithm.

Many countries have adopted E-voting systems and Estonia was the pioneer country conducting online elections between the years 2005 and 2007. The paper [12] focuses on the electronic voting system for elections in Turkey while the paper [16] focuses on Thai Elections. They propose a multi-level system based on the necessity of the country. The government is expected to provide the necessary authorization for the citizens to vote. The consensus algorithm used is Delegated Proof of Stake (DPoS). To decrease the latency, the chains in the system are divided into levels. The synchronization between data in different levels is done periodically to attain consistency.

Various cryptographic algorithms and digital signatures are used to enhance the overall security of the system. Encrypting data through a homomorphic encryption algorithm is suggested in papers [14], [15]. Shamir Secret Share has been used as the second layer of security. Voting certificates are issued to keep the voters anonymous. The system works even if a node fails or is inoperable.

III. Proposed Work

Our proposed solution is an architecture involving Blockchain Technology that takes care of the entire Election Process. The system is designed to carry out the Election process digitally so that anyone can vote securely from the comfort of their homes. The main aim of our proposed solution is to design

a system that conducts the election process in a decentralized fashion, thus keeping the entire process fair, open, and independently verifiable. Hyperledger Fabric is employed to design the network, where people can register themselves in the system to vote. The detailed System Block Diagram is shown in Fig 1.

The network consists of multiple organizations which form the backbone of the system, each of these organizations has several Peers who maintain the network's ledger of transactions. The network contains a single channel where all the network participants will be interacting to carry out the Election process. The entire election process is divided into three phases - 1) Registration Phase, 2) Voting Phase & 3) Tallying Phase. One assumption in our proposed solution is that all the users who will be registering themselves with the Blockchain Network are legally eligible to vote and that their Identity is already verified against a State/Government Database using some National Identity Number or any other authentication technique like Biometrics, etc. We have assumed these details because they are trivial and do not add any value to our objectives.

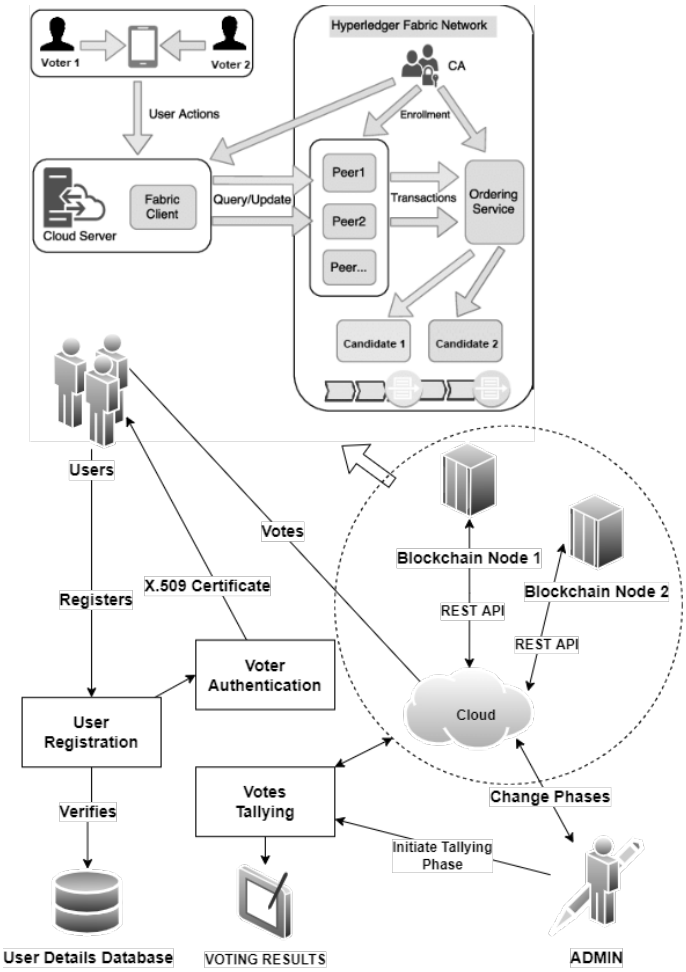

Fig. 1. Proposed System Block Diagram

During the Registration Phase, eligible voters can register themselves onto the system. To do this each organization in the network has an Admin who is responsible for enrolling and registering users onto the network. This way the Blockchain is permissioned as only the admin can permit users to transact on the network.

During the Voting Phase, all the registered voters who have a valid X.509 certificate can vote. This X.509 certificate is used to verify whether the person voting has the necessary permissions to vote on the network's channel or not. Each person is allowed to vote only once, and they cannot change their vote once it is cast. Also, the identity of the voter is hashed and stored to provide anonymity and the voting data is encrypted using Asymmetric Key Encryption to ensure the security of data. Every voting transaction goes through a series of steps to ensure the highest mode of security, these steps are explained in detail in Section III, and they form the most important part of the proposed solution.

The users can register themselves in any of the three phases, but they can only vote during the voting phase. Once a vote is cast each voter can individually and secretly verify that their vote is cast to the party they intended to vote for. Thus, the voters have a sense of trust that their vote was not tampered with and was recorded as intended.

During the Tally Phase, the results of the Elections are determined. To generate a Tally of the Votes special Users are registered on the network who have Admin Role and only they can query all the votes on the network. The Tally of the votes is then stored on the Blockchain from where users can access it to know who has won the Election. Only these special Users with Admin Roles can change the ongoing Phase of the network and hence they are responsible for carrying out each Phase of the Election Process properly.

IV. Methodology

A. Key Concepts

Some of the key components of Hyperledger Fabric using which we can design, and build are listed down below. Every component described below plays an important role in the working of the system.

- *Asset* - These are things tangible or intangible which can be transferred from one entity to another. Assets can be modified using chaincode by performing transactions. They are represented as a collection of key-value pairs.
- *Ledger* - Ledger consists of two things i.e., Blockchain and the Global State Database. It is immutable; hence it cannot be modified once set. It maintains and stores information about the current state (Global State) and about the transactions that led to the current state (Blockchain). It is expressed as key-value pairs.
- *Global State* - It is the current state of the database. It consists of the latest values for all the keys present in the blockchain. It is updated every time when a key is changed. The system can directly get the latest values from the global state making the process faster.
- *CouchDB* - It is a state database that enables you to model data on the ledger as JSON and run complex queries on data values instead of keys. Using CouchDB, we can deploy indexes with your chaincode, making queries more efficient and allowing you to query enormous datasets.
- *Orderer* - It is a node known as the orderer node which performs the ordering of the transactions. Along with

other orderer nodes, it forms the ordering service. It keeps track of the organizations that are permitted to construct channels. It also enforces basic channel access control.

- *Peers* - It is one of the most important core elements, consisting mostly of a set of peer nodes. It keeps track of the ledger and chain code. Peers are linked by channels and can be organized according to the demands for managing ledgers and contracts. Endorsement and Committing to the Ledger are two significant functions played by peers in the network. The peer can hold many ledgers, each of which is controlled by one or more chaincodes.
- *Anchor Peer* - Gossip is used to ensure that peers in different organizations are aware of each other. The channel configuration must provide at least one Anchor Peer. They help in speeding up the transaction process and provide higher availability and redundancy.
- *Membership Services Provider (MSP)* - On a permissioned blockchain network, Membership Services authenticates, authorizes, and manages identities. The MSP abstraction is implemented via Public Key Infrastructure (PKI).
- *Channel* - The purpose of the channel is to prepare a private network and perform confidential transactions. This network has several peers. Each transaction of the network is executed on Channel. Each channel has its separate ledger which is stored in each peer on the channel.

B. Network Architecture

The Network consists of two organizations namely, Org1 & Org2 each of which contains 2 Peers. Each of these Organizations consists of 1 Admin by default. There are 3 Orderer nodes configured in the network. Both the Organizations are allowed to create Channels by the Ordering Service hence both Organizations are part of the Consortium. In our case, the Network is designed for a single Channel where both the Organizations are invited to validate transactions, which will be used by all entities to conduct the Elections digitally.

This network configuration is described in the 'crypto-config.yaml' file and is consumed by the 'cryptogen' tool to create identities and cryptographic material for bootstrapping the network. This crypto-config file contains the network topology and allows us to generate a set of certificates and keys for both the Organizations and the components that belong to those Organizations. Each Organization is provisioned with a unique root certificate (ca-cert) that binds specific components (peers and orderers) to that Org. By assigning each Organization a unique CA certificate, we are mimicking a typical network where a participating Member would use its own Certificate Authority. Transactions and communications within Hyperledger Fabric are signed by an entity's private key (Keystore), and then verified by means of a public key.

Next, the configtxgen tool is used to create Orderer genesis block, channel configuration block, and anchor peers for each Organization. Configtxgen consumes a file - configtx.yaml, that contains the definitions for the network and is used to build the channel configuration. The channel configuration is stored on the ledger, and it governs every block that is added to the network. This configuration defines which organizations are members of the channel, their anchor peers, and the location of their MSP directory in turn allowing us to store the root certificates for each Org in the Genesis Block; now any network entity communicating with the ordering service can have its digital signature verified. The config file also contains which ordering nodes can add new blocks to the network, the Consensus Ordering Algorithm, and the policies that define how updates are made on the channel. In our case, all the organizations are part of the channel and all the Orderer nodes can add blocks to the network. The consensus algorithm used is RAFT which is a Crash Fault-Tolerant Algorithm. The Policy type which we use is ImplicitMeta, also all peer organizations will be able to read and write data to the ledger. Consequently, the majority of channel members sign channel configuration updates, and that a Majority of channel members need to approve a chaincode definition before a chaincode can be deployed to a channel.

We also need to fine-tune some parameters according to the working of our system which in turn affects the Latency and Throughput of the system. These parameters are the BatchTimeout which is set to 1 sec and BatchSize which is 100 transactions in one block having a maximum size of 99MB.

Hyperledger follows a Chaincode Lifecycle which includes several steps to deploy the Chaincode successfully on the channel which will then be installed on peer nodes from where it can be queried to invoke transactions and query data. The chaincode is first packaged by a single Peer to make sure that each peer is using the same version of the Chaincode. It is then installed on the provided peer nodes on the network. Each organization needs to approve the chaincode before it can be committed to the network. Hence in the next step, each organization approves the chaincode and it is only after the Majority of them approve it that the chaincode can be used in the channel. Once the approval process is done one of the Organizations commits the chaincode to the channel by collecting endorsements from enough peers on the network. Once a chaincode is committed to the network it can be queried by clients to communicate with the blockchain.

The entire network is containerized i.e., every node on the network runs as a Docker Container. This includes the 4 peers, 3 orderers, 2 certificate authorities, 4 CouchDB, and the deployed chaincode containers.

C. Registration Phase

During this phase, eligible voters can register themselves with the Blockchain Network. They can register themselves with any one of the organizations with an identification name that has to be unique within an organization. The registration process is carried out by special actors in the network which is known as Admin (each organization has one Admin). This Admin user is responsible for enrolling and registering new users on the network. Now there are certain roles that a user can have which are - client, peer, admin, and orderer. When we are registering a new user, we need to assign a role to that user who will define the things that a user can do in the network. For our implementation, we have 2 types of roles - client and admin. The client users are the eligible voters who can perform operations like voting in the network, querying the votes, and querying the results. The admin user can perform certain high security and

critical tasks like changing the phase of the Election process and generating a Tally of the results. Once a user is enrolled and registered with an organization then they are granted an X.509 certificate which is required to ascertain his identity on the network. This X.509 certificate encapsulates the user's identity and is an important aspect of the security model because they determine the exact permissions over resources and access to information that users have in a blockchain network. The X.509 certificates adopt the traditional Public Key Infrastructure (PKI) hierarchical model using Digital Certificates to maintain a sense of trust in the network.

D. *Voting Phase*

During this phase, all the users of the network with a valid X.509 certificate can vote for their preferred candidate/party. We need to keep 2 criteria in mind while registering a user vote on the blockchain which are that the Vote should not be traced back to the voter thus maintaining the anonymity of the voter and the vote itself should not be openly accessible to anyone who is administering the network. The votes made by the voters are registered as key-value pairs on the blockchain and can be visible on the CouchDB which stores the global state (current data snapshot) of the network. Hence, we need to design a database model in such a way that the above two criteria are met.

Every X.509 certificate contains 2 important things which are the Issuer DN & a Serial Number, now a combination of the Issuer DN and Serial number is unique globally. So, a combination of these two parameters is chosen as the key. But the Issuer DN consists of the identity of the person to whom the certificate was assigned. Hence the unique string thus generated is hashed using SHA256 Hashing Algorithm to generate a fixed-length hash which is guaranteed to be unique since the input string is unique, this way we make sure there are no key collisions. Now to generate the value of this voting transaction we take the candidate/party which the voter votes for and append it to the end of the key which we generated before to make a new string which again is guaranteed to be unique since a voter can vote only once and the hashed value is unique, so their combination is unique too. This new string is then encrypted using RSA Encryption using a Public Key. We encrypt the voting data instead of hashing because we will need the actual voting data at a later stage. Also, we are encrypting the votes to make sure that the second criteria are met as mentioned above. Thus, the key which is then hashed a unique string and the value which is the encrypted string consisting of the voting data is stored on the blockchain after a successful invoke transaction. A summary of the above steps is presented in the summary of the above steps is presented in Fig 2.

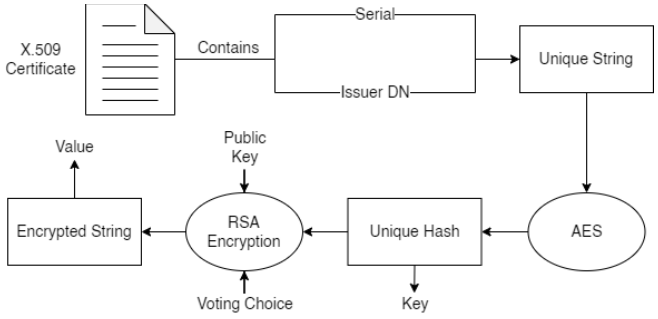

Fig. 2. Voting Phase Key-Value Pair Generation

E. *Tally Phase*

During this phase the tallying that is the total of the vote for each candidate is performed. This can only be done by the admin (government authorities). Here we need to ensure that even the admin should not know which candidate a voter has voted for.

When the admin starts this phase, it will fetch all the votes which are key-value pairs from the CouchDB database. The key is a unique hashed value that represents a unique voter. Since it is hashed, even the admin cannot know which voter does the key represents. The value was encrypted using the RSA encryption algorithm. We will now decrypt the value using the RSA decryption algorithm. The decryption can only be performed using the private key of the admin. Additionally, to perform decryption an extra paraphrase is required which was used during the encryption process. This paraphrase is only known to the admin. If the paraphrase does not match, then decryption won't be performed. This is done to provide additional security in the process. Even if someone gets access to the admin account, they cannot decrypt the vote as they do not know the paraphrase. After decryption, it will give a string which is the concatenation of the hashed key and the name of the candidate that the voter has voted for. Comparing the hashed key that we got from the value with the actual key, we can verify that the vote is genuine, and no malicious activity has occurred. Finally, we increment the count by 1 for the candidate that the voter has voted for. Performing this process for each vote, we will get the result which will tell us how many votes each candidate has received in total and thereby we get the candidate that wins the election with maximum votes.

F. *Chaincode*

The chaincode which executes the business logic of our system is described below:

- *initLedger* - As the name suggests, it is the initialization of the ledger. Here we perform the initial steps that are set up for the system. The two main things done here are starting the voting process by making the phase as registration phase and the vote count for each candidate is initialized to 0.
- *queryVoteById* - This method can only be accessed by clients(voters). If the client has performed his vote, then this will return the candidate for which he has voted.
- *queryAllVotes* - This can only be performed by the admin (government authority) and can only be called during the Tally Phase. This will return the list of key-value pairs of all the votes that have been performed. Since the key is hashed, the admin cannot know which voter has voted for which candidate.
- *queryPhase* - It will return the phase which is currently going on i.e., either of Registration, Voting, or Tally Phase
- *getAllVotesClient* - It returns key-value pairs where the key is the candidate's name and value are the number of votes it received.
- *countVotes* - It can only be performed by the admin and only during the Voting Phase. This function counts the number of votes each candidate has received.
- *changePhase* - Only admins are allowed to perform this. Using this they can change from one phase to another.

- *voteCan* - Using this method voters are allowed to cast a vote for a candidate. This can only be formed by clients(voters) since admins are not allowed to vote. The current phase must be the Voting Phase for the client to vote. Here similarly as explained in the voting phase, a key is generated for the voter which is a unique hash value, and it is compared with the key present in the global state. This is done to avoid any malicious attack on the system or to avoid any man-in-the-middle attack. If both values match, then the vote is registered successfully. If the voter had already voted for a candidate earlier then he/she is not allowed to vote.

V. Results & Observations

To evaluate the performance of our system the network was configured on a Linux Virtual Machine running Ubuntu 18.04 with 8GB of RAM and 4 logical processors. The backend was designed using NodeJS which used the Node SDK provided by Hyperledger Fabric to interact with the Blockchain. The NodeJS server was running on a single port, and it was responsible for handling the entire load of the network. A Frontend Web Application was created using ReactJS for end-users to interact with the network. Hyperledger Explorer was integrated with the network to monitor the network statistics, chaincode, individual blocks, and associated data.

Certain parameters were chosen according to the Hyperledger Blockchain Performance Metrics White Paper to do a performance analysis of the network. These parameters are:

1) ***Latency (measured in seconds) -***

   (Time when response received) – (Submit time)

2) ***Throughput (measured in transactions per minute***

   (Total number of operations performed successfully) / (Time taken)

Both parameters were measured for both Invoke (write) and Query (read) Transactions. Apart from these Metrics, we performed Load Testing on the Network to check how many Concurrent Requests the System can handle without failing to commit any Transactions. We have also obtained results for the number of Transactions the System can commit per minute and the number of Blocks the System generates per minute to Commit these transactions.

To measure the performance of the Read queries we subjected the network to different numbers of requests ranging from 100-1000 requests and found the evaluation parameters in each case which is summarized in Table I. The query chaincode which was used in this evaluation was queryVoteById which would get the candidate which a particular voter had voted for. A lower value of Latency and a higher value of Throughput are indicative of an optimum and high-performance system. We obtained Read Latency in the range 0.124 secs to 0.165 secs and Read Throughput in the range 357.37 tpm to 482.05 tpm (where tpm is transactions per minute and in case of reading queries these transactions are read/query operations).

To measure the performance of the Write queries or Invoke transactions we added another parameter along with the number of requests which was the Batch Size. So instead of sending all the requests at once, they were batched together in varying sizes and then sent concurrently to the backend for evaluation. This was done for multiple reasons, firstly because Hyperledger internally commits multiple transactions in the same block which is dependent on the BatchSize which we defined during the Network configuration. Thus, providing the network with batches of transactions from the very beginning would be beneficial as we are not unnecessarily queuing transactions that we know won't be committed until the previous set of transactions is committed. Secondly, our current implementation is restricted to a single NodeJS server, and hence as the number of transactions increases, the Load on the server increases. Each transaction opens a connection to a Gateway that uses gRPC internally which happens synchronously which causes a gRPC timeout and reduces performance. For our evaluation, we ran multiple simulations where users were randomly registered with any one of the two organizations, and they would vote randomly from a set of predefined candidates. The network was subjected to 200 and 400 requests in batches of sizes 25, 50, 100, and 200. The results of this simulation have been summarized in Table II & III. The Transaction Latency obtained was in the range of 0.213 secs to 0.215 secs for 200 requests and varying batch sizes, and in the range of 0.213 secs to 0.216 secs for 400 requests. The Transaction Throughput lied in the range of 255.22 tpm to 274.32 tpm in case of 200 requests of varying batch sizes, and in the range of 254.16 tpm to 280.93 tpm in case of 400 requests.

We also performed Load Testing on the network to measure how many concurrent requests the system could handle. We found out that the system could concurrently commit 400 transactions successfully. This means that when 400 Invoke Transactions or Voting transactions were sent at the same time all 400 of these transactions we committed successfully.

TABLE I. Latency & Throughput for Query Operations

No. of Requests	Latency	Throughput
100	0.16515 s	363.30609 tpm
200	0.160445 s	373.95992 tpm
300	0.16467 s	364.36509 tpm
400	0.16526 s	363.06975 tpm
500	0.16789 s	357.36834 tpm
600	0.16033 s	374.23593 tpm
700	0.15203 s	394.64782 tpm
800	0.13278 s	451.86677 tpm
900	0.12631 s	475.00924 tpm
1000	0.12447 s	482.05161 tpm

TABLE II. Latency for Invoke Transactions of varying Batch Sizes

Batch Size	200 Requests	400 Requests
25	0.235089 s	0.236065 s
50	0.219065 s	0.219265 s
100	0.213215 s	0.21357 s
200	0.21872 s	0.21529 s

TABLE III. THROUGHPUT FOR INVOKE TRANSACTIONS OF VARYING BATCH SIZES

Batch Size	200 Requests	400 Requests
25	255.2214 tpm	254.1673 tpm
50	273.8913 tpm	273.6415 tpm
100	281.4061 tpm	280.9383 tpm
200	274.3233 tpm	278.6841 tpm

The observations from the above results are as follows:

- The average Read Latency decreases as the number of requests are increasing. This can be attributed to the fact that multiple concurrent requests are batched, and response is generated in those batches. As the batch size increases more requests are served per unit time. The variation of the read latency with the number of requests is plotted in Fig 3.
- The average Read Throughput increases as the number of requests are increasing. Just like Latency even here multiple concurrent requests are batched and response is generated in those batches. As the batch size increases more requests are served per unit time. The variation of the Read Throughput with the number of requests is plotted in Fig 4.
- The average Invoke Latency decreases as the Batch Size Increases which is attributed to the fact that more requests are served per unit time. The average Latency is the same for 200 and 400 requests because, in the end, it is the batch size that decides the metrics of the system. The variation of Invoke Latency with the number of requests for different batch sizes is plotted in Fig 5.
- The average Invoke Throughput increases as the Batch Size Increases which is attributed to the fact that more requests are served per unit time. The average Throughput is the same for 200 and 400 requests for the same reasons as mentioned previously. The variation of Invoke Throughput with the number of requests for different batch sizes is plotted in Fig 6.
- Fig 7. shows a graph of the number of transactions which were committed every minute. We can see that 400 transactions that were committed successfully and around 8 blocks were created to include these transactions in the blockchain.

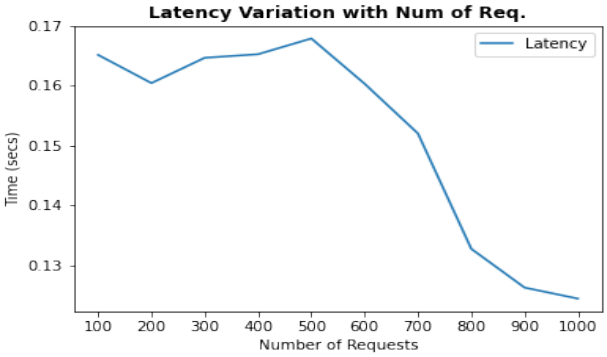

Fig. 3. Read (Query) Latency Variation

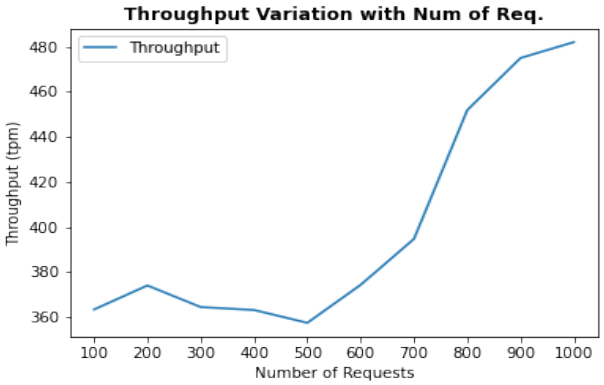

Fig. 4. Read (Query) Throughput Variation

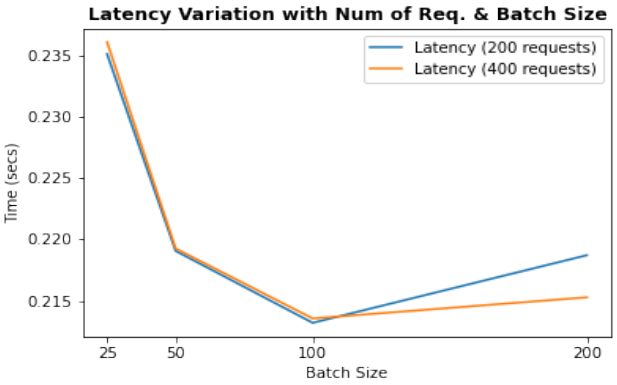

Fig. 5. Transaction (Invoke) Latency Variation

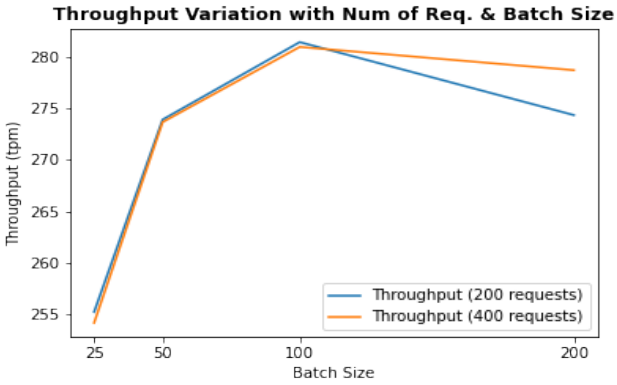

Fig. 6. Transaction (Invoke) Throughput Variation

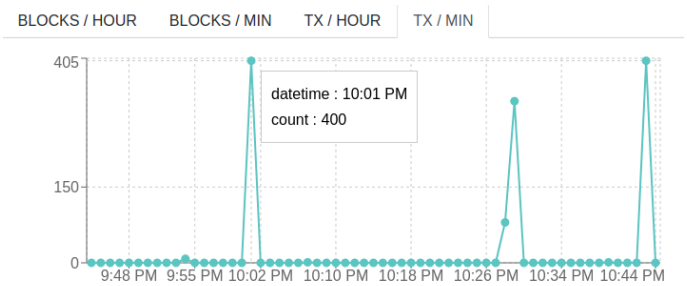

Fig. 7. Transactions generated per minute (Explorer)

VI. Conclusion

In this project we have designed a Permissioned Blockchain System which is decentralized for conducting Elections digitally in a fair, open, and independently verifiable manner. An eligible user can register themselves with the Blockchain system to get a X.509 Identity using which they can vote once in the Elections. The Blockchain was designed using Hyper-ledger Fabric with a Single Channel. The E-voting System was designed using 2 Organizations - Org1 & Org2 each of them having 2 Peers; Also, there were 3 Orderers in the System which used RAFT Ordering Algorithm. We implemented the Smart Contracts (Chaincode) which make sure that the User can only vote once completely Anonymously and only when the admin permits users to vote by controlling the Phase. We developed several APIs using NodeJS and Node SDK to allow clients to interact with the blockchain. We developed a Web Application using ReactJS using which a User can vote from any place and at any time within the Voting Phase to make his/her vote. We were able to achieve 400 concurrent Transactions using the Node SDK. We performed Load Testing & measured parameters like Latency & Throughput. The average Transaction Latency we obtained was 0.22128 s and the average Transaction Throughput was 271.534162 transactions per minute. The average Read Latency we obtained was 0.15193 s and the average Read Throughput was 399.98806 transactions per minute.

VII. Future Work

Our current implementation is a Proof of Concept for conducting Election process Digitally where we have fulfilled all the criteria as mentioned in Article 324 of the Constitution. We have designed our system with certain assumptions which have to be taken care of in a production level network. The number of organizations in the network will depend on the number Wards in the country. Also, the number of Peers per Organization need to increase which further increases the performance of the system. Currently we are assuming that the voter's eligibility is verified against a Government Database which needs to be implemented. Also, a user can register himself with any organization and can vote for any candidate but in a real-world scenario the voter can only register using an organization in his ward and can only vote for candidates present in his ward. Thus, these things can be added to the proposed technique to further enhance the performance of the system which could be used potentially to conduct Elections globally.

References

[1] Milon Biswas, Prodipta Promit Mukherjee, and Arika Afrin Boshra, `A Hyper-ledger Fabric Framework as a Service for Improved Quality E-voting System', IEEE Region 10 Symposium (TENSYMP) 2020 At Dhaka, Bangladesh Volume: 10th

[2] Denis Kirillov, Vladimir Korkhov, Vadim Petrunin, Mikhail Makarov, Ildar M. Khamitov and Victor Dostov, `Implementation of an E-Voting Scheme Using Hyperledger Fabric Permissioned Blockchain', Computational Science and Its Applications, ICCSA 2019 (pp.509-521)

[3] Lakshmi Priya K, M.Naveen Kumar Reddy, and L. Maruthi Manohar Reddy, `An Integrated and Robust Evoting Application Using Private Blockchain', Fourth International Conference on Trends in Electronics and Informatics (ICOEI 2020) IEEE Xplore Part Number: CFP20J32-ART; ISBN: 978-1-7281-5518-0

[4] Mayur Chaudhari and Prof. Dinesh Patil (HOD), 'Election System using BlockChain Technology', International Journal of Scientific Research & Engineering Trends, Volume 7, Issue 3, May-June-2021, ISSN (Online): 2395-566X

[5] S K Geetha, S Sathya, and Sree T Sakthi, 'A Secure Digital E-Voting Using Blockchain Technology', Journal of Physics: Conference Series 2021 International Conference on Computing, Communication, Electrical, and Biomedical Systems (ICCCEBS), Volume 1916, 2021 25-26 March 2021, Coimbatore, India

[6] Saad Moin Khan, Aansa Arshad, Gazala Mushtaq, Aqeel Khalique, Tarek Husein, 'Implementation of Decentralized Blockchain E-voting,' EAI Endorsed Transactions on Smart Cities, vol. 4, no.10, June 2020.

[7] Prof Chaithra S, JK Hima, Rakshita Amaresh, 'Electronic Voting System Using Blockchain,' International Research Journal of Engineering and Technology (IRJET), vol. 07, no. 07, July 2020.

[8] R. Bosri, A. R. Uzzal, A. A. Omar, A. S. M. T. Hasan, and M. Z. A. Bhuiyan, "Towards a Privacy-Preserving Voting System Through Blockchain Technologies,"2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Confon Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology

[9] Sahla Sherin O, Anna Joshy, Neethu Subash, 'Ethereum Blockchain-based Secure E-voting System', International Journal of Innovative Technology and Exploring Engineering (IJITEE), Blue Eyes Intelligence Engineering & Sciences Publication. ISSN: 2278-3075, Volume-9, Issue-5, March 2020

[10] S. T. Alvi, M. N. Uddin and L. Islam," Digital Voting: A Blockchain-based E-Voting Systemusing Biohash and Smart Contract,"2020 Third International Conference on Smart Systems and inventive Technology (ICSSIT), 2020, pp. 228-233, DOI: 10.1109/ICSSIT48917.2020.9214250.

[11] F. Hj´almarsson, G. K. Hreiarsson, M. Hamdaqa, and G. Hj´almt´ysson," Blockchain-Based E-Voting System,"2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 2018, pp.983-986, DOI: 10.1109/CLOUD.2018.00151.

[12] R. Bulut, A. Kantarcı, S. Keskin and S¸ Bahtiyar, 'Blockchain-Based Electronic Voting System for Elections in Turkey',2019 4th International Conference on Computer Science and Engineering (UBMK), Sept. 2019, pp. 183-188.

[13] Lahane, Anita & Patel, Junaid & Pathan, Talif & Potdar, Prathmesh, 'Blockchain technology-based e-voting system,' ITM Web of Conferences, Jan 2020.

[14] Seiwoong Choi, Jihun Kang, and Kwang Sik Chung, 'Design of Blockchain based e-Voting System for Vote Requirements', Journal of Physics: Conference Series The 5th International Conference on Data Mining, Communications and Information Technology (DMCIT 2021), Volume 1944,16-18 April 2021, Hangzhou, China

[15] Ruhi Ta¸s, Ömer Özgür Tanrıöver, 'A Manipulation Prevention Model for Blockchain-Based E-Voting Systems', Security and Communication Networks, vol. 2021, Article ID 6673691, 16 pages,2021

[16] Krittaphas Wisessing, Phattaradon Ekthammabordee, Thattapon Surasak, Scott C.-H. Huang and Chakkrit Preuksakarn, "The Prototype of Thai Blockchain-based Voting System" International Journal of Advanced Computer Science and Applications(IJACSA), 11(5), 2020

[17] Poonam Patil, Seema Mane, "Decentralize Electronic Voting System Using Blockchain", International Journal of Computer Science Trends and Technology (IJCST) – Volume 9 Issue 1, Jan-Feb 2021