Grenze International Journal of Engineering and Technology, Jul Issue

Hierarchical Blockchain based E-Voting System

Hritvik Patel[1], Anish Poddar[2], Anish Sekhar[3] and Prof. Sunitha R[4]
[1] PES University, Department of Computer Science, Bengaluru, India
Email: hritvik.patel4@gmail.com
[2-3] PES University, Department of Computer Science, Bengaluru, India
Email: {anishpoddar2307, anishsekh1999}@gmail.com
[4] PES University, Department of Computer Science, Bengaluru, India
Email: sunithar@pes.edu

***Abstract*—This paper describes the working of an end-to-end hierarchical blockchain-based voting system, presented as a fully automated, secure, and highly scalable online alternative to traditional methods of voting such as electronic voting machines, which require labor-intensive manual tallying. The system is divided into a hierarchy of clusters with self-contained blockchains, which aids in scalability by reducing the intensive network traffic common to traditional blockchains. It also drastically reduces computation time for election results by performing intermediate computations on votes while passing them to higher levels in the hierarchy and utilizing parallel processing capabilities for vote tallying.**

***Index Terms*— Blockchain, e-voting, voting, big data, hierarchical blockchain, elections.**

I. INTRODUCTION

A. Project Premise

Elections in India currently make use of electronic voting machines (EVMs) to record votes. These are small physical devices that record votes using a dedicated button and run unalterable software etched into their chips, with no connection to the internet. EVMs come with several advantages, such as built-in guarantees of the idempotency of a vote, and an invulnerability to large-scale tampering due to their inherently discrete and disconnected nature. However, the process of tallying votes using EVMs is highly manual due to the sheer number of EVMs whose votes must be tallied (since each EVM can only record a maximum of 2000 votes, as mentioned in [1], and approximately 418 million votes were cast in the 2019 Indian general election, as shown in [2], EVMs number in the millions), which delays the calculation of results greatly. Furthermore, the secure transportation of EVMs to central locations for counting poses a sizeable logistical challenge. We propose an alternative to this in the form of an online blockchain-based voting system that ensures vote immutability and anonymity and provides the convenience of online voting and automated tallying while mitigating the risks commonly associated with online voting systems.

B. E-voting systems in use today

E-voting systems are not very widely used in the modern world despite several past and ongoing attempts to do so in various countries. However, many countries like Switzerland, the Netherlands, and Norway have used e-voting for their elections [3]. The first country to use internet voting in its elections was Estonia, which employed it during local elections in October 2005 and received participation from 9,317 online voters, as

Grenze ID: 01.GIJET.7.2.9

detailed in [4]. Votes were cast using electronic chip-enabled ID cards held by each voter, which verified the voter's identity using a digital signature. A voter could return to alter his or her vote until the end of the voting period. Subsequent research and analysis has revealed several security issues in Estonia's system, such as the possibility of votes being inadvertently altered without the voter's knowledge, though the likelihood of this occurring was extremely remote. Nonetheless, following the success of this pilot, internet-based elections in Estonia have continued to be held for municipal and parliamentary elections over the years, with an increasing percentage of votes being cast online over the years. Switzerland uses several e-voting systems in several of its cantons (as explained in [5]), such as Geneva, *Neuchâtel*, and Zurich. Voters in Geneva prove their eligibility using ID numbers on their ballot sheets (whose chances of being randomly guessed are one in five billion), cast their votes and confirm their identities using their date of birth and the PIN on their ballot sheet. The system used in Zurich provides these details in a registration letter rather than a ballot sheet. Both models store encrypted votes in a database. However, e-voting has not been observed to significantly raise voter turnout in Switzerland. It seems to appeal mostly to voters that occasionally abstain from voting and prefer the added convenience of online voting from any device connected to the internet [6].

C. Blockchain voting – an overview

Blockchain technology implements a decentralized, distributed and immutable ledger for storing transactions and records. A blockchain consists of a network of nodes where each node contains the copy of the transactions that took place in the application [7]. Any data in a blockchain is very difficult to change or remove once it has been written into it [8], and therefore most blockchains implement strict consensus mechanisms to ensure that the data being written is legitimate. Blockchain systems have numerous applications, but their most prominent use so far has been in the realm of cryptocurrency transactions. The use of blockchain technology in voting systems, while hitherto observed only in limited cases like Agora's post-election verification of Sierra Leone's elections on the blockchain in March 2018, or Moscow's attempts at designing a blockchain voting system in September 2019, remains a field in which there is enduring interest. Blockchains offer a variety of attractive features in this domain, such as the ability to store votes immutably, provide verification to the voter that their vote was accurately registered, and store multiple copies of the blockchain across all nodes in the blockchain network to avoid reliance on a single point of failure. We have attempted to incorporate some of these features into our proposed model.

II. Literature Survey

A. Electronic voting machines – a review

This paper [9] reviews EVMs, the machines currently used to record votes in Indian elections. It first explains the old system of voting using paper ballots, then examines electronic voting and the current system used in India. It goes on to discuss some core properties every e-voting system should have. It then compares various voting systems, from the EVMs used in India to biometric EVMs. The paper concludes by talking about issues with EVMs today, and the need for further development. This was a long, time-consuming process (the delays add up) that was prone to errors. This also involved around 8,000 tons of paper and 400,000 phials of indelible ink and required some 2.5 million strongboxes to store them under heavy security until the votes were counted. It also took 3-4 days (all day) to even count all the votes. Additionally, senior citizens and disabled people may have trouble getting to voting centres and booths.

B. Success factors of Geneva's e-voting system

This paper [10] examines the success of Geneva's e-voting system. The citizen is given an option to choose their method of voting from the following: physical ballots, mail voting, or e-Voting.

a. The voter received a voting card which had a unique 16-digit number allocated for this unique vote, a 4-digit control key and a 6-digit secret code. If the code is revealed, voting using the other two modes is no longer possible, as a human operator then visually controls the card.
b. The servers are protected by several layers of firewalls and cannot be accessed directly from the Internet. Database servers are redundant and automatically replicated.
c. This paper was mainly looked at to check the viability of an online system in something as important as national elections. One of the biggest challenges for online voting systems has been security, because an online system is exposed to attack and anyone can try and hack it. Considering that such attacks could be sponsored by other nation states, the resources of these hackers are potentially huge.

d. Despite such great threats, Geneva successfully used an online application for their voting, and it worked without any major problems. This shows that as long as enough care is devoted to developing the security of a system and extensively testing it, an online voting system is a viable means of conducting national elections.

B. Blockchain-based E-voting system

This paper [11] discusses the feasibility of using blockchains in e-voting systems, and proposes an implementation of a novel blockchain based e-voting system.

a. Nodes in the blockchain network are divided into two types: district nodes, that execute smart contract logic and append transactions to the blockchain, and bootnodes, that help the district nodes discover each other.
b. Voters are registered by election administrators, that provide each voter with a unique non-reusable digital wallet containing one vote's worth of 'currency', as well as information about which district the voter can cast votes in. Voters are verified by an electronic ID and PIN number.
c. Upon casting a vote, the voter interacts with a ballot smart contract with the same voting district as is defined for any individual voter. This smart contract interacts with the blockchain via the corresponding district node, which appends the vote to the blockchain if the majority of the corresponding district nodes reach mutual consensus. Votes are tallied on the fly.
d. Once a vote is cast, a corresponding transaction ID is generated and given to the voter so they can verify if their vote was correctly cast and recorded in the blockchain.

C. Blockchain-Based Electronic Voting System for Elections in Turkey

This paper [12] proposes a hierarchical blockchain voting system based on a clustered architecture to improve the scalability and efficiency of the system when applied to national elections.

a. The paper details the various advantages of a blockchain-based voting system, such as decentralization and immutability. It then outlines the need to develop a generalized version of such a system that is more scalable and avoids the problem of synchronization delays caused by larger blockchain networks with a single shared blockchain across the entire network.
b. It then goes on to offer an alternative to a single fully connected blockchain network in the form of a hierarchical blockchain network divided into smaller clusters, which are far faster and easier to synchronize internally. Each cluster contains its own fully self-contained blockchain, and does not communicate with other clusters at the same level.
c. At the lowest level of the hierarchy, each cluster stores only one vote per block in its internal blockchain; subsequent levels store batches of votes per block, passed to them from the lower levels. Our model distinguishes itself from this approach by passing batches of votes as a consolidated sum of votes per candidate rather than a group of connected votes, reducing network overhead by drastically lowering payload size.

III. System Overview

A. Introduction

Our model implements a hierarchical blockchain-based voting system divided into self-contained clusters, each with its own separate blockchain. The overall architecture was influenced by the approach followed in [12] but differs in several key aspects. Votes cast from a basic frontend interface are processed and converted to a suitable format before they are sent to random clusters at the lowest level, following which they are subjected to an ordering process within the cluster and written to several dedicated storage nodes within it. At periodic intervals, lower levels in the hierarchy consolidate and pass votes up to higher levels (in doing which the individual identity of each vote is lost, anonymizing its caster). Finally, when election results are calculated, all clusters at the highest level of the hierarchy send their blockchains to a central processing location for tallying, which is performed in a highly parallelized manner for efficient computation of results.

B. Justification for using a hierarchical model

Traditional blockchains share a single common blockchain data structure across all nodes in the blockchain network. While this increases security greatly by increasing the number of nodes that must reach consensus and providing greater data replication, such an architecture is harder to scale efficiently due to the high network traffic incurred by its design. This problem becomes especially prominent in the arena of elections, which are

often conducted at large scale. Our model attempts to achieve a trade-off between security and efficiency by dividing our architecture into smaller blockchain clusters organized in a hierarchical manner. Clusters at a single level all contain their own internal blockchains that are not shared or synchronized with other clusters in the same level, thus greatly reducing network overhead and increasing the rate of recording votes. Periodically, all votes received by a cluster are aggregated into batches and passed to higher levels, where they will be written as single blocks instead of distinct votes. This implements some degree of precomputation to aid the final calculation of results.

IV. System Design

This section describes the individual components of our architecture in detail and traces the flow of a single vote through these components from the moment of casting the vote to the time it is written persistently onto the blockchain in a cluster. It will also detail the process of consolidation of votes and how aggregated votes are passed up to higher levels in the hierarchy. A high-level system overview diagram is shown in Fig. 1.

A. System modules

The proposed model may be logically divided into the following distinct modules:

a. A web server, that renders the voting website, permits voters to cast votes, authenticates their identity using a shared database, and passes all verified votes to random clusters in the blockchain hierarchy for further processing.
b. A shared database server, that authenticates a voter based on their credentials and ensures that they have not already voted in the current election. It also contains references to important election metadata and assets such as the names of parties and their representatives, and image assets such as candidate images and party logos (which are stored externally on the cloud).
c. A blockchain cluster, which is a set of machines that collectively record a single blockchain data structure among themselves and achieve consensus when writing to it. The cluster itself consists of the following distinct types of machines:
    1. A “forwarder”, that receives all incoming votes from lower levels of the hierarchy and passes them to other machines in the cluster to be processed, synchronized, and finally written persistently to the blockchain. It is also responsible for requesting the storage nodes to initiate the process of calculating election results.
    2. A blockchain storage node, that contains a locally stored copy of the blockchain for the cluster it belongs to, and periodically passes up aggregates of votes contained within it to clusters at a level just higher than its current level. The storage nodes can also send their local blockchains to external cloud storage for computation of election results.
    3. An orderer, which is responsible for deciding on the order in which the votes it receives are to be written to the storage nodes. This is decided between all orderers in the cluster via a quorum.
    4. A time server, that periodically instructs the orderer nodes to initiate the ordering process and send the ordered vote sequences to be written to the storage nodes.
    5. An SQLite database server responsible for generating unique batch IDs when passing up aggregated votes to higher levels in the hierarchy.
d. An Apache Spark cluster that runs parallelized computations to calculate election results based on consolidated votes stored in the clusters at the highest level of the blockchain hierarchy.

B. System flow

Election creation

Initially, administrators log into the voting website and access a form for creating an election. Here, they can enter information pertaining to the election such as party and representative names, as well as party logos and candidate images. This information is uploaded to cloud storage and retrieved when rendering the UI that will be visible to the voter.

Voter registration

Voters can then register on the voting website by entering information such as their voter ID and date of birth. If the registration is successful, the voter will be provided with an 8-character secret key that they must later enter as a credential when logging in to vote.

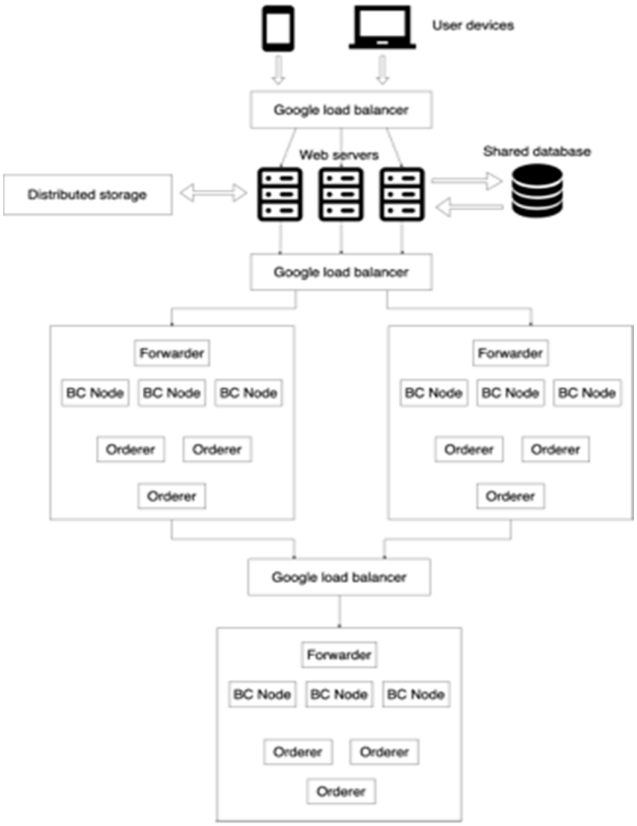

Figure 1. High-level system overview diagram

Voter login

Once registered, the voter is redirected to a login page where they enter some identifying information along with the secret key received on registration. The web server then sends this information to one of the shared database servers to verify the voter's credentials, and ensure they have not yet cast a vote in the current election. If the voter passes both these checks, they are redirected to the voting page.

Casting a vote

The voting page contains a list of candidates and party names as well as their logos and images., which are retrieved from the cloud storage bucket they were previously put into by the administrator. It also contains a button the voter can use to cast their vote. Once this button is pressed, the vote data is sent to the shared database, which then generates a unique ID for that vote and sends it back to the web server. Once the ID has been received, all vote data excluding the voter's voter ID (for anonymity reasons) is passed to the backend to be written to the blockchain clusters.

Lowest-level cluster processing

Once the vote data leaves the web server to be written to the backend, it is sent to a load balancer that then forwards it to one of the clusters at the lowest level of the hierarchy. Here, the vote goes through the following sequence of steps:

1) The vote is first received by the forwarder node of the cluster (see Section III-Ac1). The forwarder simply chooses a random storage node (see Section III-Ac2) within its cluster and forwards the received vote data to it (see Fig. 2).
2) This storage node then chooses a random orderer node (Section III-Ac3) within the cluster and forwards the vote data to it (see Fig. 3).

3) The orderer node first writes the received vote to an internal queue (called the receiver queue), and then broadcasts it to all other orderers in the same cluster, where it will then be written to their respective receiver queues (see Fig. 4).
4) Every few minutes (the exact interval can be adjusted), the time server for the cluster (see Section III-Ac4) calls an API endpoint on all the orderer nodes in the cluster to trigger the ordering process and initiate a timeout (see Fig. 5). As long as a timeout is active, any external votes received by the forwarder are retained in a temporary queue in the forwarder until the ordering process is complete, and steps 1 through 3 of this process do not occur until the timeout is completed. Similarly, any votes that may have passed through the forwarder just before the timeout was activated but have not yet reached an orderer in the cluster will also be retained in a timeout queue on whichever orderer happens to receive it.
5) Once the orderer nodes enter their timeout state, each orderer sends all the votes it has received since the last timeout as a batch to all other orderers in its cluster (see Fig. 6). Assuming there are 'n' orderers in the cluster, at the end of this process, every single orderer in the cluster will possess exactly 'n' batches of votes, one of which is its own, and the other (n-1) of which have been received from the other orderers. Ideally, these batches are completely identical (assuming step 3 executed successfully for each orderer), but since the timing server cannot trigger a timeout on each orderer with perfect simultaneity, it is possible that the first orderer to trigger its timeout may not have received some votes present in the other orderers as they could not send those votes to it in time.
6) Once each orderer has exactly 'n' batches of votes, it counts the frequencies of each distinct vote in all these batches (votes at this level are distinguished by the unique ID given to them by the shared database, as explained in Section III-Bd). All distinct votes that occur more than (n/2) times in these batches are added to a final consensus batch. It should be noted that since all the orderers have exactly the same batches, they compute the same consensus batch, so it is impossible for the orderers to "disagree" on the final set. All votes in the consensus batch are then sorted by their unique ID to obtain an unambiguous sequence of votes to be written to the blockchain (as demonstrated in Fig. 7).
7) After the consensus batch has been calculated and sorted, all votes contained within it are processed to deterministically generate a number between 1 and 'n' (where n is the number of orderers in the cluster). Each storage node arrives at the same result since they all receive the same consensus batch. Each orderer already contains a number in this range as an environment variable specified during its creation. The orderer for which this number is the same as the number generated from the consensus batch is chosen to broadcast the consensus batch to all the storage nodes in the cluster to be written to their blockchains (see Fig. 8).
8) Once the storage nodes in the cluster receive the generated consensus batch from the deterministically chosen orderer, they write the received batch to a locally stored blockchain structure (currently stored as a CSV file secured with AES128 encryption). Each storage node then generates a number between 1 and 'm' by processing the consensus batch data (similar to the number generated in the previous step), where 'm' is the number of storage nodes in the cluster. The storage node whose environment variable is the same as the generated number aggregates all the votes in the received batch into a single set of vote sums per candidate.
9) The chosen storage node then requests a new unique 'batch ID' for this aggregate from the SQLite server node within the cluster (see Section III-Ac5). Furthermore, all nodes in each cluster contain two additional environment variables containing the level number and cluster ID of the current cluster to uniquely identify each cluster and avoid communication between non-consecutive levels of the hierarchy. The batch ID, along with the level number and cluster ID, is added to the aggregate data and passed up to another Google load balancer, which redirects it to a cluster in the next lowest level in the hierarchy (see Fig. 9).

Higher-level cluster processing

All higher -level clusters receive vote aggregates rather than single votes at their forwarder nodes. However, the code for each component of a blockchain cluster has been written in a generic fashion, meaning that each cluster behaves identically regardless of its level in the hierarchy. Naturally, this means that the vote data received at the lowest-level clusters must conform to the same format as an 'aggregate' received at any higher level. To achieve this conformity, the web server represents each single vote it receives as an 'aggregate' of 1 vote before it sends the vote to a lowest-level cluster (this aggregate contains 1 vote for the candidate voted for, and 0 votes for all

other candidates competing in the election). Therefore, a lowest-level cluster is unaware that it is receiving single votes, as it "thinks" it is receiving aggregates that contain one vote each from the web server.

Result calculation

Once the stipulated period for an election has been completed, the administrator may log into the voting website and press a button to trigger the calculation of results. This initiates the following sequence of steps:

1. A request is sent from the administrator's device to the web server to begin calculating election results. The web server then uses the Google Cloud SDK to initiate a Spark job on the Apache spark cluster described in Section III-Ae. The spark cluster then calls an API endpoint on the forwarders of all clusters at the highest level of the hierarchy to request them to upload their respective blockchains to a cloud storage bucket for calculating the final tally of votes.
2. The forwarders at each highest-level cluster send this request to a randomly chosen storage node within their respective clusters. These storage nodes upload their blockchains to the storage bucket mentioned in step 1. They then send a success response to the forwarder, which relays this response to the Spark cluster.
3. Once all highest-level clusters have uploaded their blockchains to the storage bucket, the Spark cluster begins to add up votes received for all candidates to obtain the total number of votes per candidate. This process is highly parallelized, and processes billions of votes in the order of minutes.
4. Once the final tally of votes has been generated, the vote totals for all candidates are put in sorted order and sent back to a random web server from the Spark cluster. This random web server composes an email containing the ranked list of candidates (and their vote totals) as an attachment, and sends it to all registered administrators for the election, concluding the election.

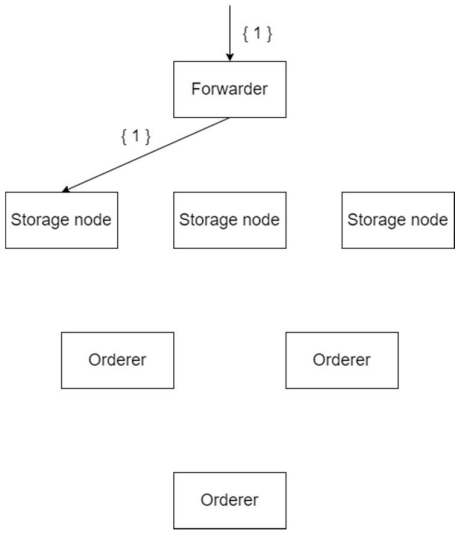

Figure 2. Forwarder sending vote batch to random storage node

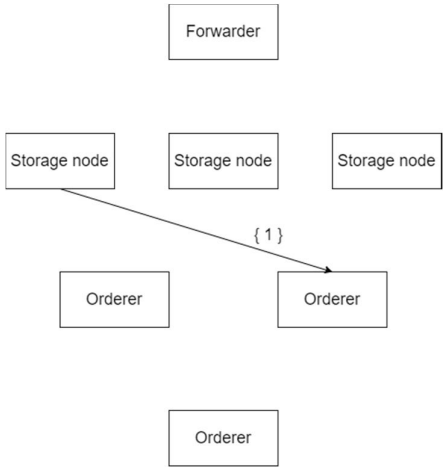

Figure 3. Storage node sending vote batch to random orderer

V. Experimental Results

The current system utilizes Spark to calculate election results by processing data from all the highest-level clusters using parallel computing. When tested on a load of 1 billion votes, it produced results in approximately 7.5 minutes, with 3 minutes taken for all computations, without taking into consideration delays caused by downloading the blockchain(s) from the cloud storage bucket.

On sending 20,000 requests asynchronously using a Python script with randomized votes, a rate of approximately 5 requests per second was reached. The code utilized a sliding-window method that required exactly 20 active requests to be open at any time, sending more requests as other requests received responses. The backend was unable to handle over 20 concurrent requests at a time due to computing constraints.

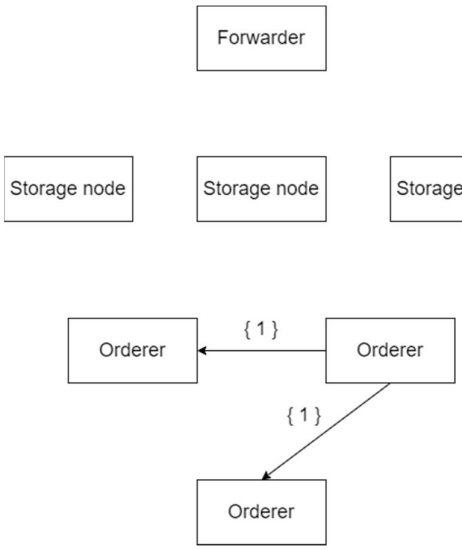

Figure 4. Orderer forwarding batch to all peer orderers

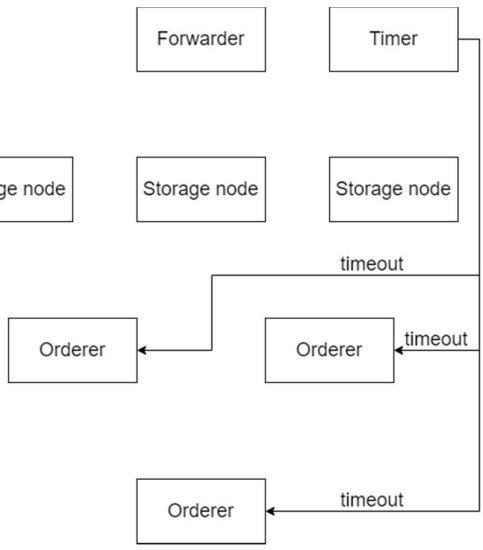

Figure 5. Timer initiating timeout on all orderers in cluster

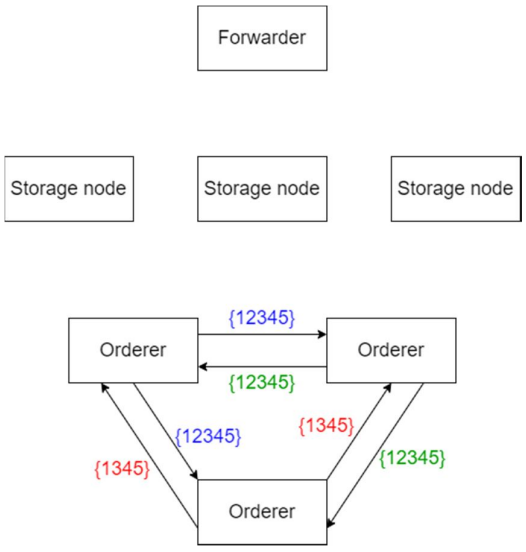

Figure 6. All orderers broadcasting their batches to each other

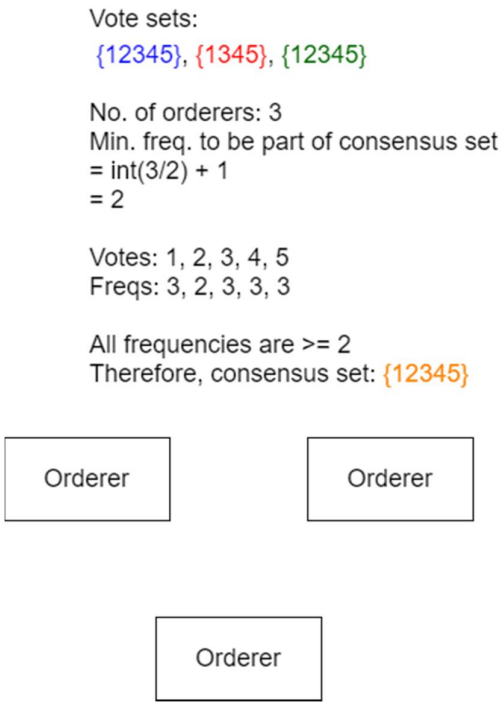

Figure 7. Calculation of consensus set of vote batches

We sent 10,000 requests to our backend using a pool of 12 workers on a 4-core machine in a Go script to load test our system. All the requests were serviced within 352 seconds, resulting in a rate of approximately 22.7 successful requests per second and a total rate of 28 requests per second regardless of success/failure. A 20% vote loss was observed at this load. The average response time for a successful request was observed to be about 10 ms.

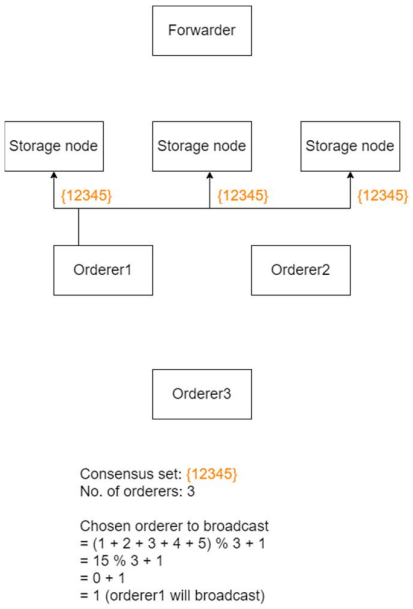

Figure 8. Chosen orderer broadcasting consensus set to all storage nodes

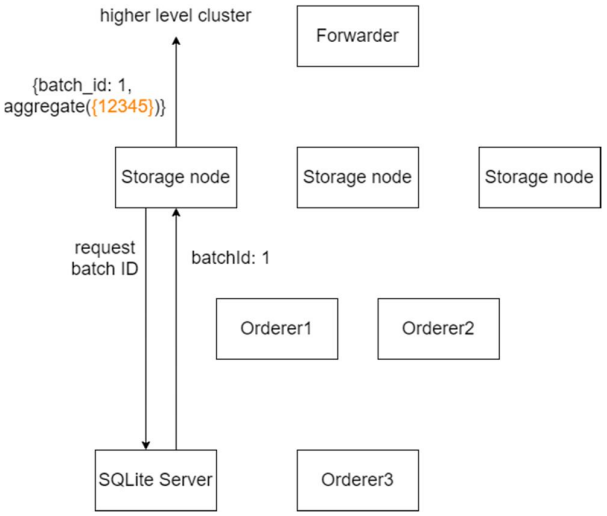

Figure 9. Chosen storage node passing up consensus batch to higher-level cluster

VI. Conclusions

The proposed system is decentralized, secure, highly scalable, and immutable. It remains highly available even during the synchronization process when votes are being ordered and is designed to be generic so code can be identical across clusters regardless of their level in the hierarchy. Furthermore, all data stored in the blockchain is encrypted to ensure the privacy and anonymity of votes.

The primary advantage of our model is its scalability and its ability to be adapted to elections in large or small countries and states. The number of machines in a single blockchain cluster, the number of levels in the hierarchy, and the number of clusters per level are all parameters that can be tweaked and adjusted according to the requirements of a particular election. Additionally, it is possible to limit access to the voting website only to machines secured within voting centres, even though the current model allows voting from all mobile and personal electronic devices owned by voters.

The proposed model successfully implements several key blockchain concepts, such as decentralization and immutability. However, it must be admitted that the model is not truly a blockchain architecture in the strictest sense and has some key differences that distinguish it from a traditional blockchain. The consensus mechanism described in Section III-Be6 is more akin to a quorum than commonly seen forms of blockchain consensus such as proof-of-work, since it implicitly trusts all votes that reach the forwarder. The initial intention was to encrypt all cross-cluster and interlevel communication using RSA 4096. While it is possible for this to be integrated into the existing system in the future, it should be noted that it does not exist in the current implementation.

It is also possible to integrate code changes that generate a transaction ID for each individual vote for vote verifiability, which would enable a voter to know that their vote was successfully written to the blockchain. We have left out this part of the implementation presently since it runs the risk of causing public doubt in the reliability of the voting system, but it is possible to implement this as a way of maintaining an audit trail for each vote, which would be useful as an internal tool for election administrators.

Furthermore, most blockchains have ways of ensuring that all nodes in a blockchain network possess exactly the same sequence of blocks, such as assuming that the longest chain of blocks is the “correct” blockchain. The existing model does not currently contain any method for blockchain nodes with erroneous, missing or corrupted block data to obtain the accurate version of the blockchain from another storage node in its cluster. This is also an extension which can easily be added to the code to make the proposed model more similar to a real-life blockchain.

ACKNOWLEDGMENT

We would like to express our gratitude to our project guide, Prof. Sunitha R., Asst. Professor, Computer Science, PES University, for her continued support and unfailing guidance throughout the course of this project. We would also like to thank Prof. H. B. Prasad, Professor, Computer Science, PES University, for his incisive feedback and insights on the project architecture and design. Finally, we thank Prof. Shylaja S. S., Department Head, Computer Science, PES University for all the knowledge and support we have received from the department.

REFERENCES

[1] ECI, “Electronic Voting Machine – General Q/A”, 2018, Apr 3, Accessed on: Sep 3, 2020, [Online], Available: https://eci.gov.in/faqs/evm/general-qa/electronic-voting-machine-r2/

[2] EDM Division, “Performance of National Parties”, 2019, Oct 11, Accessed on: Sep 3, 2020, [Online], Available: https://eci.gov.in/files/file/10955-20-performance-of-national-parties

[3] R. Hanifatunnisa and B. Rahardjo, "Blockchain based e-voting recording system design," 2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA), 2017, pp. 1-6, doi: 10.1109/TSSA.2017.8272896.

[4] Dr. Alexander H. Trechsel, Ivar Tallo, E-Voting in the 2005 local elections in Estonia, Accessed on: Sep 1, 2020, [Online], Available: https://web.archive.org/web/20090402014151/http://www.coe.int/t/e/integrated_projects/democracy/02_activities/02_e-voting/00_evoting_news/finalreportevotingestoniacoe6_3_06.asp

[5] Republic and State of Geneva, The Geneva Internet Voting System, Accessed on: Sep 4, 2020, [Online], Available: https://www.coe.int/t/dgap/goodgovernance/activities/evoting/evoting_documentation/passport_evoting2010.pdf

[6] Ayed, Ahmed Ben. "A conceptual secure blockchain-based electronic voting system." International Journal of Network Security & Its Applications 9.3 (2017): 1-9.

[7] Sheer Hardwick, F., Gioulis, A., Naeem Akram, R., & Markantonakis, K.” E-Voting with Blockchain: An E-Voting Protocol with Decentralisation and Voter Privacy”. 2018, doi:10.1109/cybermatics_2018.2018.00262.

[8] K. Garg, P. Saraswat, S. Bisht, S. K. Aggarwal, S. K. Kothuri and S. Gupta, "A Comparative Analysis on E-Voting System Using Blockchain," 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), 2019, pp. 1-4, doi: 10.1109/IoT-SIU.2019.8777471.

[9] D. A. Kumar and T. U. S. Begum, "Electronic voting machine — A review," International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME-2012), Salem, India, 2012, pp. 41-48, doi: 10.1109/ICPRIME.2012.6208285.

[10] Michel Chevallier, Michel Warynski and Alain Sandoz State Chancery, Success Factors of Geneva’s e-Voting System, Apr. 2006, Available at https://www.researchgate.net/publication/259645027_Success_Factors_of_Geneva's_e-Voting_System

[11] Friðrik Þ. Hjálmarsson and Gunnlaugur K. Hreiðarsson, Blockchain Based E-Voting System, Jun. 13 2018, Available at http://hdl.handle.net/1946/31161

[12] R. Bulut, A. Kantarcı, S. Keskin and Ş. Bahtiyar, "Blockchain-Based Electronic Voting System for Elections in Turkey," 2019 4th International Conference on Computer Science and Engineering (UBMK), Samsun, Turkey, 2019, pp. 183-188, doi: 10.1109/UBMK.2019.8907102.