SEVA: A Smart Electronic Voting Application Using Blockchain Technology

Jacob Abegunde and Joseph Spring, *University of Hertfordshire UK*
Hannan Xiao, *King's College London UK*

***Abstract*—The development of electronic voting applications remains an active area of research and this has led to the proposal and implementation of many models based on blockchains. However, most of the proposed models are partially decentralized solutions, in which the blockchain is used as a storage media for votes while the application is written in programming tools such as HTML, CSS, and JavaScript. This makes them vulnerable to attacks such as Denial of Service (DoS) attacks, Single Point of Failure (SPF), and fraudulent record modification.**

In this paper, we propose a fully decentralized electronic voting application, SEVA, in which we placed the whole application (code and data) in Ethereum to protect the application from vulnerabilities. Additionally, we propose a new consensus algorithm, Proof of Smart Vote (PoSV) for SEVA, as a viable energy-saving alternative to the energy-intensive Proof of Work (PoW). We implemented and evaluated SEVA with PoSV and compared it with a partially decentralized model of the application.

***Index Terms*—Fully Decentralized Application, Distributed Security, Trustless System, Blockchain, Smart Contract, E-voting.**

I. INTRODUCTION

Recent developments in technology together with the proliferation of smart devices has led to electronic voting (e-voting) becoming an attractive alternative to paper ballot systems [1], [2]. However, the credibility of most current e-voting applications are yet to be established since they are unable to satisfy all of the requirements for e-voting such as resilience, non-repudiation, universal verifiability, receipt-freeness, coercion free, robustness, transparency, and privacy. This has resulted in a lack of trust and acceptability for such applications.

In [3] and [4], it was conjectured that the privacy and security of an electronic vote could be preserved in two ways, firstly by encrypting the vote and secondly or by sending the vote through an anonymous communication channel. Although both methods are known to be good theoretically, they do not offer a guarantee of protection for voting information, neither do they meet the tamper-proof requirement for an e-voting system. A well-funded adversary could have the capability to decrypt communication as well as sniff anonymous channels in order to eavesdrop or replay messages.

Hao et al, proposed a two-round anonymous general purpose voting protocol for e-voting in 2010 [4]. It required no trusted third parties or private channels, and participants could execute the protocol by sending two rounds of public messages. Subsequent work [5], was released in 2012 to address the robustness and fairness problem identified in [4].

The work in [3] was released in 2014 as an improved version of [5] for large-scale elections. It was said to be significantly more efficient in terms of the number of rounds, computational cost, and bandwidth usage. However, the new version still has drawbacks since it is based on the principle of a centralized application. In order to address the deficiencies of [3], subsequent work [6], in which votes were stored in the blockchain was released in 2017 by the same group. However, as a partially decentralized application, it is still vulnerable to DoS and SPF.

Overall, most of the proposed blockchain e-voting applications described in [6]–[10] are partially decentralized solutions in which the application code is developed with programming tools such as HTML, CSS, and JavaScript, whilst utilizing the blockchain as a storage media for votes. Although these solutions protect the votes against fraudulent modification since they are written in the blockchain, the application code remains vulnerable to various attacks as discussed above.

A. Motivations

It follows from the above discussion that, an appropriate e-voting solution must involve more than just vote encryption and sending votes through an anonymous channel. For this reason, we propose a Smart Electronic Voting Application (SEVA), a fully decentralized application hosted on an Ethereum Virtual Machine (EVM) network that protects the electronic voting application against vulnerabilities such as those discussed above.

The trend towards smartness continues to rise through the ongoing proliferation of smart devices and the deployment of smart protocols in IoT (as discussed in [11]). This ongoing trend towards smartness is the principal driving force for this work, in addition to the desire to protect electoral processes from adversaries.

B. Contribution

This work has three contributions. Firstly, the authors of e-voting applications such as [9] and [10] have acknowledged that their work is not fully decentralized. As far as we know, existing similar applications fall into the same category of partially decentralized solutions since they only use the blockchain as a backend database for the storage of votes. This work differs from such previous proposals in the sense that it combines the code and the data on EVM by coding all of the election functionalities into a smart contract for security and redundancy. Thus it is referred to as fully a decentralized solution.

Secondly, the Proof of Work (PoW) consensus mechanism is considered to be inappropriate for an e-voting protocol [12], due to its energy requirements and a block time of approximately 10 minutes, as required by Bitcoin [13]. Similarly, the Proof of Stake (PoS) consensus requires optimization

for the allocation of appropriate stakes for each vote. We therefore propose a new consensus mechanism, Proof of Smart Vote (PoSV) in this paper. To the best of our knowledge, this consensus mechanism has not been proposed or used elsewhere.

Thirdly, SEVA is a self-tallying e-voting application that also includes a two-factor authentication (2FA) mechanism in the form of a one-time password (OTP) as a security enhancement. Although the use of 2FA with OTP is a basic security standard in online authentication, very little is said about its potential usage in e-voting as far as we know.

The rest of this paper is structured as follows. The relevant background is discussed in section 2, followed by our proposal and model in section 3. The system implementation is presented in section 4, with the system evaluation in section 5 while the conclusion and future work are presented in section 6.

II. SYSTEM MODEL

A. A Fully Decentralized Application

The remote e-voting solution can be broadly categorized into two approaches: the centralized e-voting application and the decentralized e-voting application. The centralized application model constitute single or multiple system(s) working as one from single or multiple location(s) with a single service point and single point of control. Its positive side includes consistency, efficiency and affordability, however, such applications are vulnerable to attacks such as DoS attacks, SPF and fraudulent record modification [1].

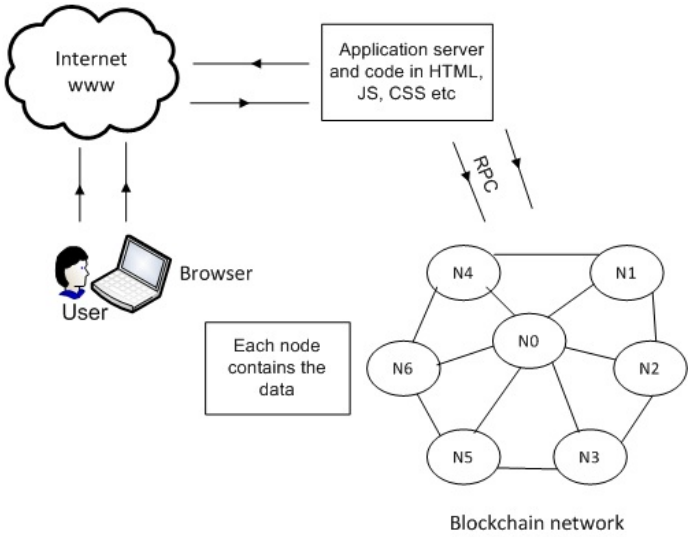


Fig. 1: Partially decentralized application workflow

An alternative model to the centralized model is the partially decentralized model as shown in Fig. 1, where the data of the application is hosted on peer-to-peer nodes which form a blockchain network, whilst the code is developed using for example HTML, CSS, and JavaScript. Although the blockchain network provides protection for the data against DoS, SPF and fraudulent modification of the records, the application code remains vulnerable to such attacks.

As shown in Fig. 2, SEVA utilizes a fully decentralized application model as a viable alternative to the partially decentralized application model shown in Fig 1. In SEVA, both the code and data of the application are hosted on an EVM network which is a blockchain-based software platform, to provide resilience, redundancy, and tamper-proof protection.

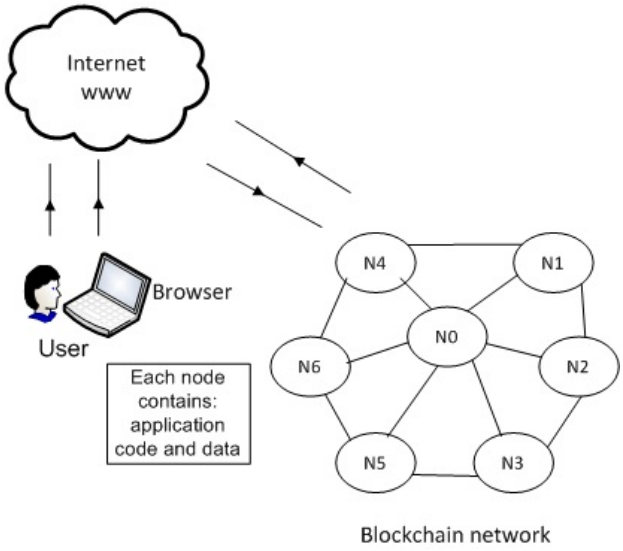


Fig. 2: Fully decentralized application workflow

One of the features of the fully decentralized architecture is that the data (votes) and the application code (smart contract) are immutable, meaning that none of the blockchain nodes can unilaterally make any change to the data or the code [8]. The ledger records (votes) are arranged into blocks and chained together using a hash function to form a public record [7] while the application functions are coded into the smart contract. This arrangement ensures that the data and the code are held securely and are tamper-proof. We therefore believe that this model is more suitable for the deployment of e-voting applications than the centralized or partially decentralized architecture.

B. System Entities

The proposed model consists of the following entities, each with a distinct role.

Voters: the set of people who are eligible and authorized to vote.

Candidates: the subset of voters who are standing for electoral post as indicated on the ballot paper by the electoral commission.

Electoral Commission (EC): an independent authority empowered by law to conduct elections and announce the results.

Political Parties: the different political parties that are represented in the election.

Local Authority: the local constituency in which election is to be held.

Blockchain Network: a non-trusted peer-to-peer network that maintains records that are accessible to all of the stakeholders.

Blockchain Admins: a team of system administrators, members of the EC.

Mining Nodes: a set of nodes that are responsible for adding records of votes to the public ledger.

Non-mining Nodes: a set of nodes whose role is to observe and verify all election records and transactions.

Smart Contracts: software code that manages the election process and performs vote tabulation and counting.

C. System Requirements

The voter registration (and voting) are divided into constituencies. Voters can only register and vote in their constituencies since the e-voting application will be deployed on a constituency basis. In each constituency, the blockchain network will be a permissioned network with an equal number of mining nodes owned by each political party, the EC, and the local authority. This is to increase transparency and reduce the risk of a 51% attack through collusion [13].

Each voter will be automatically provided with an Ethereum account (for voting) during the application run, once they are authenticated successfully. Voters will have their Ethereum accounts credited with ETH (Ethereum currency), to pay for the gas needed to execute the smart contract. Voters will connect to the application via their browser, hence there will be no need for client software. The voting user interface (website) will be protected using asymmetric cryptography (SSL certificates), however, voters will be responsible for the security of their devices when using the online application.

It is assumed that the EC has an accurate database of registered voters which will be written to the blockchain and will be updated regularly. It is also assumed that adversaries are capable of monitoring public communication channels, decrypting encrypted traffic, and performing general network-level attacks such as eavesdropping, replay attacks, masquerade attacks, DoS attacks, and record modification.

III. SYSTEM DESIGN

A. Smart E-Voting Model

Fig. 3 shows the architecture for the smart e-voting application which consists of the user interface, the smart contract, and the ledger record, combined on an EMV which is a blockchain infrastructure on a peer-to-peer network of nodes [8]. Intercommunication between nodes is via encrypted multicast. The votes are stored in bundles of records called blocks, which are chained together using a hash function to form a ledger [8].

The user interface of the application is divided into an administrative and voter interface. The administrative user interface consists of *nodejs* which provides an administrative interface to the EVM for coding and deployment of the smart contract. The voters' user interface consists of *web3js*, which provides an interface with the smart contract to present the voting menu to the voters.

All peer-to-peer nodes run the same application code as smart contract and contain both the administrative and the voters' interface which communicates with the backend via

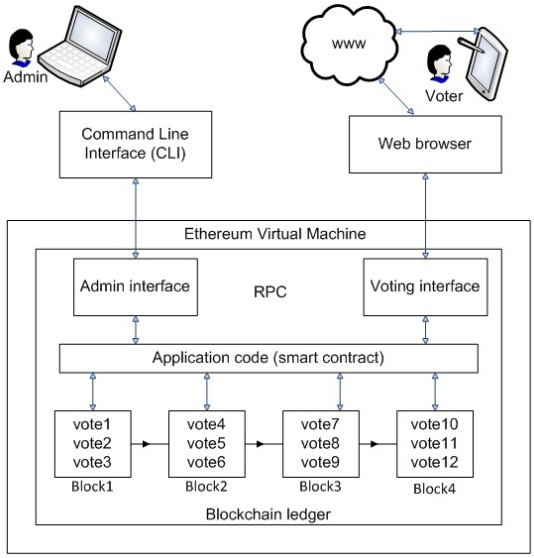


Fig. 3: Smart e-voting architecture

Remote Procedure Call (RPC). The application server (on EVM) is viewed as a network of nodes that form the blockchain network, in which each node contains the ledger (data), the smart contract (application code), and the user interface combined.

B. Smart E-Voting Phases

The smart e-voting process consists of three major phases: the registration phase, the voting phase, and the tallying and counting phase.

Registration Phase: The first step in any voting process is the registration of voters. It involves the process of checking and confirming the identity and eligibility of voters. It is done periodically before the voting day. The end product of this phase is the voter list. Voter registration usually closes before the voting phase.

Voting Phase: The voting phase is subdivided into three stages as follows: initialization, authentication, and voting.

The initialization process is carried out by the Electoral Commission and involves preparing all the components of the application for the election process. It involves blockchain network preparation, deployment of the smart contract, and publication of the election website address.

All eligible voters need to authenticate against the voting application in order to cast their vote. The authentication process requires a voter to provide the information used for registration during the registration process: a pre-set voter identification and password are required for the first stage of the authentication process.

If the first stage of the authentication process is successful, then a code (OTP) will be sent to the voter's known mobile number that was provided during the registration process. Once the code is received on the voter's mobile phone and entered into the application, the 2-stage authentication process is completed. A voting Ethereum account is then generated and allocated to the voter.

The voting process involves the selection of the preferred candidate from a drop-down list of candidates, (as shown in Fig. 7a). After the selection of the preferred candidate, the voter clicks on 'vote' to cast their vote. The vote is tallied, counted, and written to the blockchain by the smart contract. The transaction is identified by a Transaction Identification Number (TIN) which is the Ganache account number. The TIN is written into a file to be sent to the voter's registered mobile phone number or email address, after the election. This TIN could be used to trace and verify the vote after the election.

Tallying and Counting Phase: The tallying process refers to the tabulation of votes according to the candidates voted for. The counting process is the summation of votes for each candidate as the votes are tabulated. In the application simulation, the votes are tabulated and counted automatically on the fly by the smart contract which implies that SEVA is a self tallying e-voting application. However, the result is not shown or published until the end of the voting window.

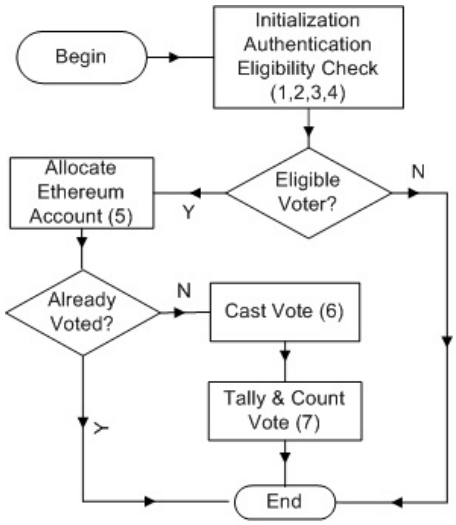


Fig. 4: Smart e-vote flowchart

C. Smart E-Voting Algorithm

Each steps of the smart e-voting algorithm is as shown in Fig. 4. The algorithm is discussed below:

Step 1: Closing of voters' registration record and initialization of the application by the system administrators of the electoral commission.

Step 2: Publishing of the smart contract and its web address to declare the start of the election.

Step 3: The voting process begins when voters log in to the electoral commission system to authenticate themselves.

Step 4: After a successful 2FA, the voter is granted permission to interact with the smart contract.

Step 5: A voting Ethereum (Ganache) account is allocated to authenticated voters to cast their votes

Step 6: Voters select their preferred candidate and cast their votes.

Step 7: Votes are automatically tallied and counted via the smart contract on the fly and the results are displayed at the end of the election.

D. Proof of Smart Vote (PoSV)

In order to prevent identity forging as in Sybil attacks Ethereum uses proof of work (PoW) as a consensus mechanism which has been widely criticized in the literature for its inefficient use of energy [1]. This is because for every block to be mined in a PoW consensus, all of the mining nodes are required to engage in intensive mathematical work and this consumes a tremendous amount of energy. Although the security of a decentralized application is partly credited to the PoW, the same PoW is also responsible for the high transaction fees, high latency, slow convergence, and scalability problems associated with most decentralized applications [14]–[16].

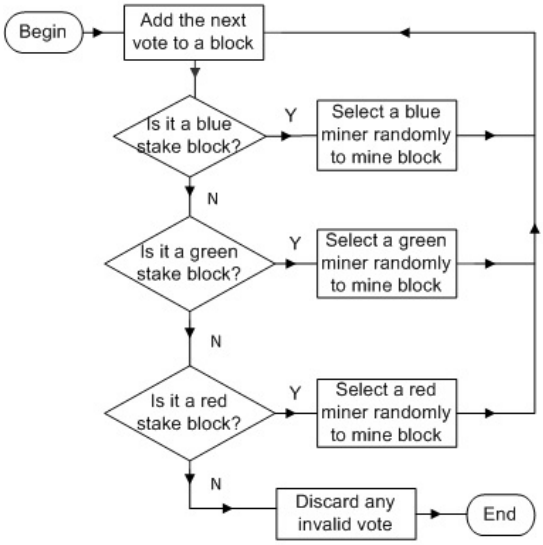


Fig. 5: Proof of smart vote (PoSV) flowchart

An efficient consensus mechanism has therefore been an ongoing area of research in blockchain technology which has led to the proposal of various consensus mechanisms. The most prominent of these is the Proof of Stake (PoS) which is considered a viable alternative to PoW. In a PoS consensus, the miner of a block is chosen based on the stake owned by the participant. This implies that the miner with the largest stake is chosen since it is rational for it to protect its stake by mining the data according to the rule, while all other nodes do the verification.

For the purpose of SEVA, PoW is considered to be heavy-weight and too expensive whilst PoS needs to be optimized to allocate stake to votes and blocks based on the party of choice in the votes. Therefore, in order to improve the efficiency, scalability, and security of the mining process, a new concession mechanism, Proof of Smart Vote (PoSV) is proposed, a modified version of PoS. The flowchart for the PoSV is shown in Fig. 5.

In the PoSV mechanism, the mining privilege is allocated to the mining nodes of the political party for which a voter voted. For each block of votes to be mined, the mining node is selected randomly from the nodes of the party that has the greatest stake on the block to be mined. In other words, for a block that contains predominantly 'red vote' to be mined, the

miner is randomly selected from the list of 'red miners', since they have the largest stake in that block. This is consistent with the protocol since mining of votes takes place after the votes have been tallied and counted hence the stake of each vote (and block) is known to the application before mining.

The PoSV is a modified version of PoS which is fine-tuned for smart e-voting not only to reduce the energy wastage in PoW, but to also reduce the convergence time and consequently reduce the latency and improve the performance of the application. To the best of our knowledge, PoSV is unique to SEVA and has not been used elsewhere.

IV. SYSTEM IMPLEMENTATION

A. System Preparation

An Ethereum Integrated Development Environment (IDE) is prepared for the development of SEVA. The IDE replaces the need to implement the whole blockchain infrastructure [2]. The necessary open-source software was downloaded, installed, configured, and linked together to form the required IDE. The IDE was then used in the writing, testing, and deployment of the smart contract for SEVA, with the implementation running on Ubuntu Linux. The core components of the IDE are discussed below.

Nodejs and NPM: Node.js is a JavaScript platform for general-purpose programming that allows users to build network applications. Node.js is used for the application distribution across network nodes.

Solidity Compiler: Solidity is the most popular programming language for writing smart contracts on Ethereum. It turns human-readable Solidity code into Ethereum bytecode, which Ethereum network nodes can understand and execute. It is a strongly typed object-oriented, high-level language for implementing smart contracts [2].

Ganache: is a software utility used for simulating personal blockchain accounts, developing and deploying smart contracts. It is used to simulate client behaviour and to provide RPC functions and features for communication between the user interface, application code and the data.

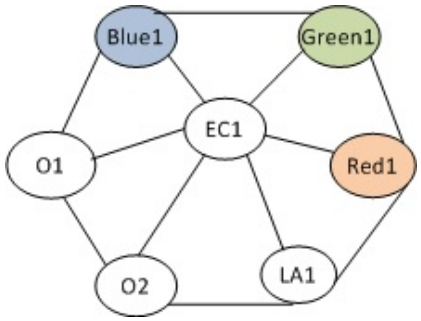
Truffle Framework: The Truffle Framework is a testing framework for the Ethereum IDE. It is a project management tool used for testing, compiling, migration, and deployment of smart contracts on an Ethereum network.

After the installation, configuration, and testing of the IDE, the smart contract for the smart e-voting application is written and compiled using solidity.

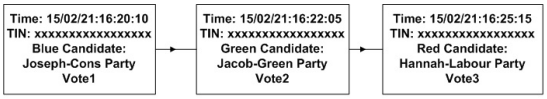
B. Smart E-Voting Application Simulation

The initial experimental setup is as shown in Fig. 6a. A permissioned blockchain is considered in which there are seven participating nodes representing different entities: the three political parties (Blue, Green, and Red), the electoral commission, the local authority, and two independent observers. The three political parties are considered as the biggest stakeholders in the election. Therefore, a mining node is allocated to each of them as follows: Blue1 node for Blue party, Green1 node for Green party and Red1 node for Red party. For equal representation, each political party is assigned the same number of mining nodes in all our simulation runs.

The remaining four non-mining nodes are allocated as follows: EC1 node is for the electoral commission, LA1 node is for the local authority, whilst O1 and O2 nodes are for the two independent observers. These are non-mining nodes whose roles are monitoring and validation of transaction. For transparency purposes, individual members of the public or corporate bodies such as the press, and public observers can host their non-mining monitoring nodes, to play the role of observers and verifiers, as may be allowed by the stakeholders. In order to satisfy the anonymity requirement, each vote on the ledger is an encrypted record consisting of timestamp, TIN, and the candidate or party of choice as shown in Fig. 6b.



(a) Node simulation



(b) E-vote information

Fig. 6: Application simulation

The nodes' addresses and authentication requirements were set up in the configuration files. The number of voters considered in the first simulation run was ten based on ten Ganache accounts. After setting up these parameters in the configuration files, the written smart contract code was deployed and the application web address and service port were published using the truffle framework [12], [17]. Voters were then able to access the user interface (voting menu) as shown in Fig. 7a, and to cast their votes using their browsers. Fig. 7b shows the administrative interface for verification of the configuration and the Ganache accounts.

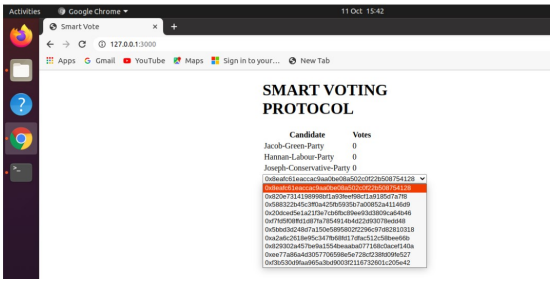
At the end of the voting window, the allocated Ganache account was sent to the voter's mobile phone or email address as the Transaction Identification Number (TIN). The TIN can be used to verify and confirm the vote but not the voter, since the confirmation will only show the party, the candidate of choice, and the timestamp of the vote. Thus, the voter's identity is anonymized and disassociated from the actual vote to discourage coercion and vote-buying.

V. SYSTEM EVALUATION

A. Transparency Accuracy and Integrity

In order to evaluate the performance of SEVA with respect to transparency, accuracy, and integrity, we need to show that:

(a) User interface showing the voting menu



(b) Admin interface showing ten voting accounts

Fig. 7: Voting interface and admin interface

- votes are tallied as cast
- votes are recorded as cast
- votes are counted as cast
- the result is recorded and published as counted

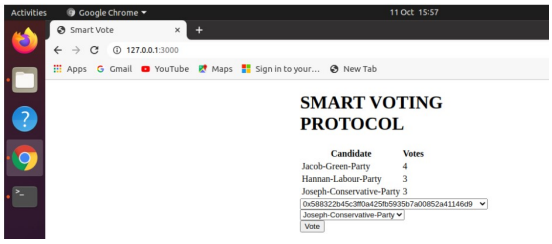
At the end of the initial run of the simulation with three mining nodes and ten voters, the voting result interface was examined as shown in Fig. 8a, to ensure that it corresponds with the votes cast at the user interface. On the voting interface, four votes were cast for the candidate Jacob - Green party, while three votes each were cast for candidates Hannan - Red party and Joseph - Blue party. The displayed votes-count on the report interface in Fig. 8a matches the votes cast in the voting interface in Fig. 7a.

In addition to the above verification, the transaction logs were examined to verify the backend process and the ledger record. As shown in Fig. 8b, the backend log also reflected the votes cast on the voting interface and displayed on the report interface. This constitutes a double verification showing that votes are tallied, counted, and recorded as cast.

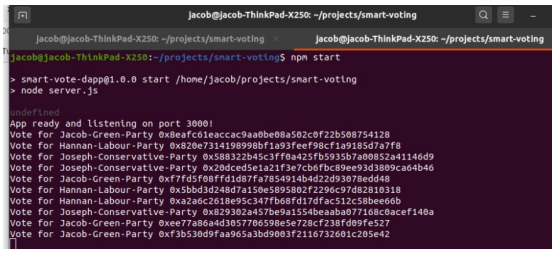
B. Anonymity and Coercion-resistance

In evaluating anonymity and coercion-resistance features, Fig. 7a and Fig. 7b show respectively, the user interface for voters and the voting accounts generated for users. These contained no voter identification information. Similarly, the vote information which is written into the ledger by the smart contract as shown in the transaction log in Fig. 8b contained no voter identification information.

This combination shows that voter's identification information is not associated with the vote, and is replaced with the Ganache account as designed. The votes could be verified by the voter after the election using the TIN. However, the absence of voter identification in the vote information will make it difficult for voters to prove how they cast their vote to a coercer for reward purpose. The verification confirms to voters that their votes have not been altered, it does not confirm the identity of the voters to a coercer, hence there is no motivation to coerce.

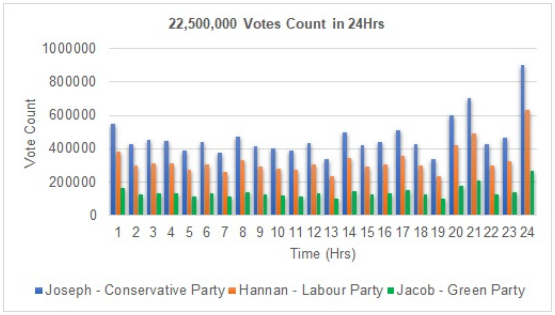


(a) The result of first ten votes - counted on the fly

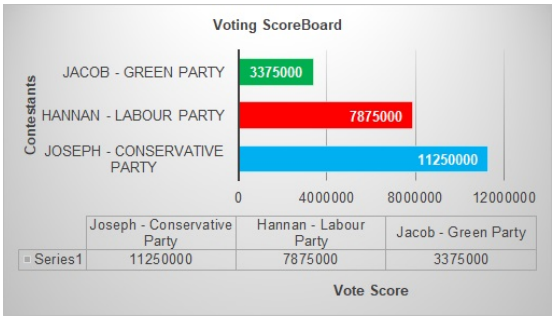


(b) The log of first ten votes - counted on the fly

Fig. 8: Voting result and voting logs



(a) Graph of over 22 million votes count in 24 Hours



(b) Voting scoreboard at the close of the election

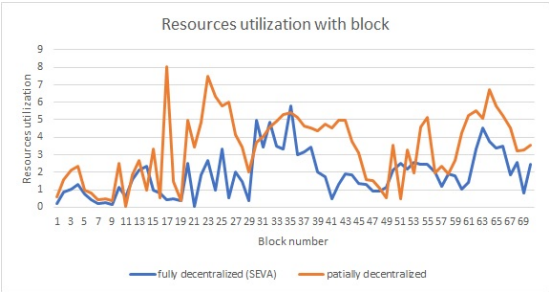
Fig. 9: Voting graph and voting scoreboard

C. Voters Scalability

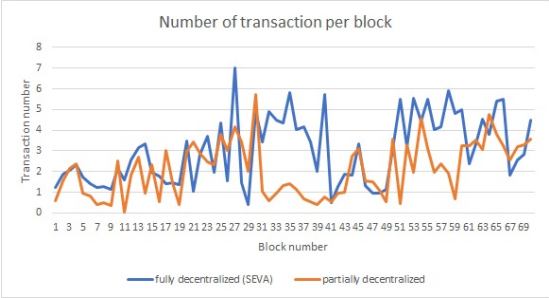
In order to test the capability of SEVA in handling a large number of voters (voters' scalability), the number of nodes is kept constant while the number of voters is increased gradually from 10 to fourteen million, the size of registered voters in the state of Florida in the year 2020. The number was then increased to 22 million to match the number of registered voters in the state of California, which had one of the highest numbers of registered voters in the US in the year 2020,

according to the US voters' registration record that is available in the public domain. The output graphs and scoreboard for the simulation election runs are as shown in Fig. 9a and Fig. 9b.

The casting of the votes for the simulations occurred at random time interval to represent different voting time. The result shows that the number of voters voting during different time windows has no effect on the performance of the application since it takes just a few seconds to cast a vote after successful authentication. The votes cast were counted on the fly and kept in memory pools while awaiting mining.



(a) Resources utilization graph for 3 mining nodes



(b) Block transaction graph for 3 mining nodes

Fig. 10: Utilization and transactions

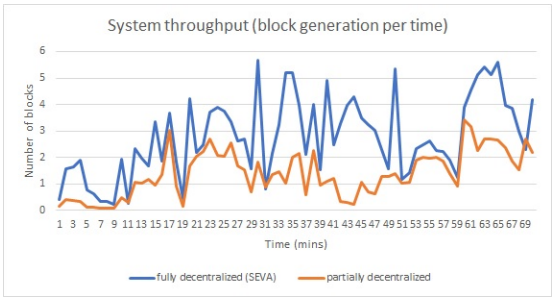
D. SEVA vs Partially Decentralized Application

In order to compare the robustness of SEVA with a partially decentralized application, we simulated a partially decentralized model that does not host the application code on Ethereum, but stores votes in the blockchain network using the same number of nodes. The resource utilization graph in Fig. 10a shows lower resource utilization for SEVA than the partially decentralized model as a result of the efficiency of smart contract functionalities.

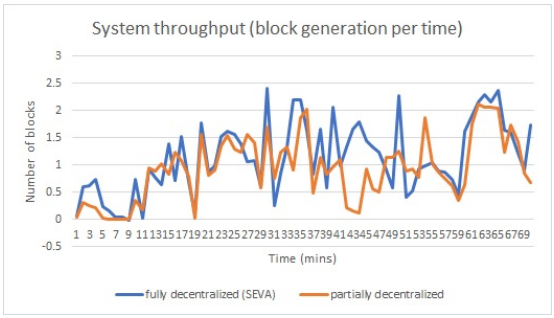
Similarly, in order to compare the efficiency of SEVA with a partially decentralized model, the above simulation was repeated with the same number of nodes. The number of transactions per block are as shown in Fig. 10b and it indicate a higher performance for SEVA in terms of the number of transactions per block. In addition, the graph in Fig. 11a shows a better performance for SEVA than a partial decentralized model in terms of the number of blocks generated and mined with time (throughput).

E. Mining Nodes Scalability

In order to increase the resilience of the application and mitigate the risk of DoS and SPF, there is a need to increase the



(a) Throughput graph for 3 mining nodes



(b) Throughput graph for 30 mining nodes

Fig. 11: Throughput comparison

number of mining nodes. The increase in the number of mining nodes is a test of nodes' scalability. In order to investigate the scalability of mining nodes in SEVA, the number of voters is kept at twenty two million while the number of mining node is increased to ten for each of the three political parties.

In this simulation run, it was observed that increasing the number of mining nodes has no effect on the application response time, however, the number of blocks generated per time (throughput) is slightly lower for both SEVA and the partially decentralized model as shown in Fig. 11a and Fig. 11b. However, the result still shows a better throughput for SEVA than the partially decentralized model as shown in Fig. 11b.

We attributed this lower throughput to communication overhead which increases with the number of nodes, but more test results will be necessary to confirm this. A laptop of low-end hardware configuration of 2.0GHz i5 Intel processor, 16G RAM and 512G hard disk drive was used in the simulation run, so we would expect a better performance on a high-end machine.

The overhead of SEVA includes the time cost of development and implementation, which is higher than the partially decentralized model because of the need for Ethereum IDE and coding of functionalities into smart contract. In addition to this, the overhead of the processing power and communication could make the application a bit expensive to implement and run as the number of nodes increases. However, this could be considered as the cost of security and hence constitute a trade-off. The potential scalability issue is addressed by dividing the election into smaller chunks known as constituencies, each of which has independent blockchain networks and run a different instance of the application.

F. Threat Analysis and Mitigation

SEVA consists of several distributed nodes working together as one, which provides resilience against security attacks by eliminating the possibility of SPF inherent in the centralized and partially decentralized application models. For an attack to have a significant effect on the application, a large percentage of the nodes will have to be attacked or compromised simultaneously.

The two major attacks that could be staged against such an application are DoS and record modification. The DoS attack against mining nodes could be mitigated by increasing the number of nodes. The higher the number of mining nodes, the better the resilience of the blockchain network and hence the application that runs on it.

In theory, an attacker will need sufficient resources to attack 51% of the nodes simultaneously for attacks such as DoS, Replay, and record modification to have any significant effect [13], [17]. Therefore, the larger the number of nodes in the blockchain the better the security.

The record modification attack could manifest in the form of a brute-force attack on the private key to create fake transactions or modify existing ones. However, such transactions would not pass the validation test until at least 51% of the nodes collude to ratify the forged transaction [1], [12], [13]. Therefore, manipulated ballots will be rejected by the application due to failed validation processes and incorrect signatures.

This implies that the voting records in the backend and the application code written into smart contracts cannot easily be changed since they are protected by the immutability features of the blockchain. The randomly generated individual private keys could be a target of a brute force attack, however, they are 24 characters in length, hence brute-forcing keys within the voting window will be a non-trivial task for even a well-resourced adversary [18].

If on the contrary, it turns out to be a trivial task and the adversary creates fake votes or modifies existing ones, such action will render the fake votes or changed votes invalid and hence they will be rejected by the application. In addition, the votes have been counted on the fly before they are encrypted and stored on the disk for verification and recounting purposes.

VI. Conclusion and Future Work

The aim of developing SEVA is not to prevent attacks on e-voting applications, since this would not be a realistic goal. On the contrary, the aim is to make the job of the attackers more difficult and less likely to succeed. In order to achieve this, a fully decentralized voting application is proposed to replace the current partially decentralized models by hosting the application code and the data together on Ethereum for resilience and security.

Finally, a perfect e-voting solution is still a matter for the future, especially the one that will combine the receipt-free feature with the ability of voters to verify their votes after voting. The verifiability of the votes by voters is of a higher priority to us at the moment, so our implementation includes a voting receipt for voters to be able to verify that their votes were tallied and counted as cast. Our next step is to find a way to eliminate the receipt without removing the voters' ability to verify their votes.

References

[1] David Khoury, Elie F. Kfoury, Ali Kassem, and Hamza Harb. Decentralized voting platform based on ethereum blockchain. In *2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*, pages 1–6, 2018.

[2] Shalini Shukla, A.N. Thasmiya, D.O. Shashank, and H.R. Mamatha. Online voting application using ethereum blockchain. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 873–880, 2018.

[3] Feng Hao, Matthew N. Kreeger, Brian Randell, Dylan Clarke, Siamak F. Shahandashti, and Peter Hyun-Jeen Lee. Every vote counts: Ensuring integrity in large-scale electronic voting. *USENIX Journal of Election Technology and Systems (JETS)*, (3):1–25, August 2014.

[4] F. Hao, P. Y. A. Ryan, and P. Zieliński. Anonymous voting by two-round public discussion. *IET Information Security*, 4(2):62–67, 2010.

[5] Dalia Khader, Ben Smyth, Peter Y. A. Ryan, and Feng Hao. A fair and robust voting system by broadcast. In Manuel J. Kripp, Melanie Volkamer, and Rüdiger Grimm, editors, *5th International Conference on Electronic Voting 2012 (EVOTE2012)*, pages 285–299, Bonn, 2012. Gesellschaft für Informatik e.V.

[6] Patrick McCorry, Siamak F. Shahandashti, and Feng Hao. A smart contract for boardroom voting with maximum voter privacy. In Aggelos Kiayias, editor, *Financial Cryptography and Data Security*, pages 357–375, Cham, 2017. Springer International Publishing.

[7] Samuel Agbesi and George Asante. Electronic voting recording system based on blockchain technology. In *2019 12th CMI Conference on Cybersecurity and Privacy (CMI)*, pages 1–8, 2019.

[8] R. Hanifatunnisa and B. Rahardjo. Blockchain based e-voting recording system design. In *2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA)*, pages 1–6, 2017.

[9] Wei Cai, Zehua Wang, Jason B. Ernst, Zhen Hong, Chen Feng, and Victor C. M. Leung. Decentralized applications: The blockchain-empowered software system. *IEEE Access*, 6:53019–53033, 2018.

[10] Kevin Curran. E-voting on the blockchain. *The Journal of the British Blockchain Association*, 1(2):4451, 2018.

[11] Q. Wu, G. Ding, Y. Xu, S. Feng, Z. Du, J. Wang, and K. Long. Cognitive internet of things: A new paradigm beyond connection. *IEEE Internet of Things Journal*, 1(2):129–143, April 2014.

[12] Awsan A. H. Othman, Emarn A. A. Muhammed, Haneen K. M. Mujahid, Hamzah A. A. Muhammed, and Mogeeb A. A. Mosleh. Online voting system based on iot and ethereum blockchain. In *2021 International Conference of Technology, Science and Administration (ICTSA)*, pages 1–6, 2021.

[13] Jaewon Bae and Hyuk Lim. Random mining group selection to prevent 51 In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 81–82, 2018.

[14] Shiyao Gao, Dong Zheng, Rui Guo, Chunming Jing, and Chencheng Hu. An anti-quantum e-voting protocol in blockchain with audit function. *IEEE Access*, 7:115304–115316, 2019.

[15] Sankarshan Damle, Sujit Gujar, and Moin Hussain Moti. Fasten: Fair and secure distributed voting using smart contracts. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–3, 2021.

[16] Abhishek Kaudare, Milan Hazra, Anurag Shelar, and Manoj Sabnis. Implementing electronic voting system with blockchain technology. In *2020 International Conference for Emerging Technology (INCET)*, pages 1–9, 2020.

[17] Emre Yavuz, Ali Kaan Koç, Umut Can Çabuk, and Gökhan Dalkılıç. Towards secure e-voting using ethereum blockchain. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, pages 1–7, 2018.

[18] Wei Cui, Tong Dou, and Shilu Yan. Threats and opportunities: Blockchain meets quantum computation. In *2020 39th Chinese Control Conference (CCC)*, pages 5822–5824, 2020.