



International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 2(1) (2026)



**ijerst.editor@gmail.com
editor@ijerst.com**

Research Paper

Blockchain-Based Transparent and Tamper-Resistant Decentralized Digital Voting System

Dr. D. Sirisha¹, P. Santosh Kumar², S. Vignesh³, Md. Azad Ansari⁴, M. Jayaram⁵

¹Professor, Department of Computer Science and Engineering,

^{2,3,4,5} Student, Department of Computer Science and Engineering,

Nadimpalli Satyanarayana Raju Institute of Technology, Visakhapatnam, Andhra Pradesh, India

¹drdsirisha.cse@nsrit.edu.in, ²pallisantoshkumar8105@gmail.com, ³vigneshsampathirao@gmail.com, ⁴mdazadansari96180@gmail.com, ⁵mallajayaram22@gmail.com

Abstract

The electoral integrity is the keystone of democracy. The major drawback is that one can still tamper with the normal voting systems such as the paper ballots or server-based voting, by ballot tampering, more than once, or concealed tallying. In this paper, a novel blockchain-based digital voting system addressing these issues directly is designed. The system is built with Ethereum and Solidity with the Ganache and checked with MetaMask which offers a decentralized and immutable voting system whereby no repository is allowed to alter the documented votes. Another crucial feature of the proposed design is a dual registration scheme: using blockchain wallet addresses (with support to single and batch onboarding), admins can register voters, whereas non-technical users can register via an email-based flow of One-Time Password (OTP) that forwards the votes via a secure relayer contract. The admin dashboard enables users to take care of the whole election lifecycle, they can create elections, add candidates with logos, and track live results and they can give approvals to voters down to the last detail. The experimentations with a local area network there were none with double-vote incidents, and the updates published in sub-seconds time per mined transaction, and emails with registered voters could play without knowing anything about blockchain. The findings indicate that resistant to tampering smart-contract logic can be constructed to offer end-to-end verifiability and mass availability at the same time.

Keywords: Digital Voting, Blockchain, Ethereum, Solidity, Smart Contracts, Decentralised Application, Electronic Voting, Ganache, MetaMask, OTP Authentication, Tamper Resistance, Relayer Pattern.

I. Introduction

Democratic governance reduces to adequate measuring of every genuine vote that has been cast and ensuring that the entire process is subject to review by anyone. Paper balloting is not unheard of but it can easily be tampered with physically, there is miscounting and due to logistical hitches, which become increasingly problematic when the population increases. Centralized e-voting addresses a

group of problems but poses a new threat: in case of a server attack it may tamper or delete votes without leaving a footprint to the observers [1].

Blockchain technology provides a solution and an alternative that is cool. An encrypted ledger system based on cryptographic consensus does not rely on one party of trust, and once a record is inserted to it, it will be difficult to make any modifications at a later time, since it will require recalculation of the entire chain to the spot of modification which is practically impossible in case the consensus rules have been established correctly [6, 7]. On top of this, Ethereum smart contracts [8] enable you to code the election rules directly on chain, which removes the possibility of the administration tampering with the rules in the course of voting.

This is a culmination of the project that it is constructed as a complete and operational blockchain voting system. It is set up and runnable on an Ethereum testnet (Ganache) located on the local network (LAN) and can be accessed by any web browser. Developed two contracts to serve the various voter groups: Ballot.sol to vote during elections with everybody having a MetaMask wallet, and BallotEmail.sol to vote during elections where the voter registers via email, with a relayer contract sending his vote. Collectively, they contain three endemic literature problems (1) usability barrier since blockchain-native interfaces cannot cater to non-technical voters; (2) scalability in voter registration and identity verification; and (3) verifiable results in real-time and no central server.

The remaining part of the paper is arranged as such. Section II examines Literature Survey. Section III describes the system design in general. Section IV guides one through the implementation of every part. The results of the experiment and their comparison are presented in Section V. Section VI concludes and recommendations of further work.

II. Literature Review

Since the genesis of Bitcoin in 2008, there has been ongoing scholarly interest in evaluating electronic voting by the means of blockchain. The work of Nakamoto [6] had shown that a decentralised, self-sovereign network of distrustful nodes could come to consensus on a shared state without an overall authority, and new work by Buterin [8] in the form of his Ethereum white paper extended this model to allow Turing-complete programmability and complex electoral logic, encoded as smart contracts. Probably one of the first conceptual frameworks of a e-voting system based on the blockchain and using the infrastructure of Bitcoin was offered by Ayed [2] and focused on preserving the anonymity and avoiding civil disenfranchisement. Zheng et al. [3] then provided an optimistic overview of blockchain-based architectures and consensus mechanisms - Proof of Work, Proof of Stake, and Byzantine Fault Tolerant mechanisms - thus defining the theoretical background, in which domain-specific mechanisms like those of our system are executed.

A self-tallying boardroom voting protocol on Ethereum introduced by McCorry, Shahandashti, and Hao [4] ensured the privacy of voters by preserving homomorphic commitments, which proved that strong cryptographic guarantees could be fully realized on-chain. Hardwick, Akram and Markantonakis [5] came up with a model that permits blockchain voting system, striking a balance between the requirement of transparency and privacy, specifically noting that real-life examples will require the ability to attend to those voters who do not have cryptocurrency wallets, which is precisely the reason our system needs to be dual-registration.

Voting implementations not just in theory but also in practice with the blockchain have provided some key empirical insights. As an example of audit-grade transparency at the national level, Agora released a permissioned blockchain to tally the 2018 Sierra Leone presidential election [9]. Voatz, a voting app tested in municipal elections in the US, received a large amount of security attention: Specter et al. [10] performed a penetration test which showed vulnerabilities in the mobile client and server-side applications, and demonstrates the importance of the security of surrounding infrastructure in liberalizing blockchain usage.

A parallel cryptographic line of research has been caused by voter privacy. Benaloh and Tuinstra [11] proposed the idea of receipt-free voting so as to eliminate the sale of votes and coercion. As it has been shown by Crammer, Gennaro and Schoenmakers [12], homomorphic encryption can be used to implement tallying privately without disclosing ballots on an individual basis. More recently, privacy-preserving on-chain voting via zero-knowledge proof systems (also known as zk-SNARKs used in Zcash [13]) are also proposed and will be incorporated in our future work.

The trade-off between the decentralisation, the security, and throughput is formalised as the blockchain trilemma of Buterin [14]. Each of those can be scaled up to large populations of transactions through solutions like Optimistic Rollups or zk-Rollups, which, in addition to micropinning cryptographic commitments to the main chain, are a logical next step towards large-scale elections [15]. The relayer model used in BallotEmail.sol is based on EIP-2771 [18] that standardises an implementation of a trusted forwarder scheme allowing, despite their ineligibility to interact with an ETH-based system, users to engage in on-chain transactions.

One of the research problems still under study is voter authentication and identity verification. Hacking Fujioka, Okamoto and Ohta, [16] developed a workable secret vote system that combined the use of blind signatures and anonymous channels. More recently, alternative reviews have suggested W3C Verifiable Credentials and Decentralised Identifiers (DIDs) [17] as a blockchain-compliant platform based on identity providers that are not centralised. Five canonical assessment dimensions of the e-voting system indicated by Kulyk et al. [19] as well as formal protocol analyses by Cortier, Galindo, and Turuani [20], namely correctness, privacy, verifiability, robustness, and usability are the evaluation framework used in Section V of this paper.

The blockchain system proposed herein contributes concretely to the usability and registration dimension and clearly gives out the privacy and scalability constraints which will be addressed in the further iterations.

III. Methodology

A. System Architecture and Design Philosophy

The proposed system architecture for Blockchain-based Digital Voting System maintains an inflexible separation between policy, defined in unchangeable smart contracts and the front end, defined by standard web interface. This division indicates that the election logic remains intact and open to all in the case of the web server or the admin workstation being hijacked. The architecture depicted in Figure 1 is the five-layer architecture that we constructed in the system.

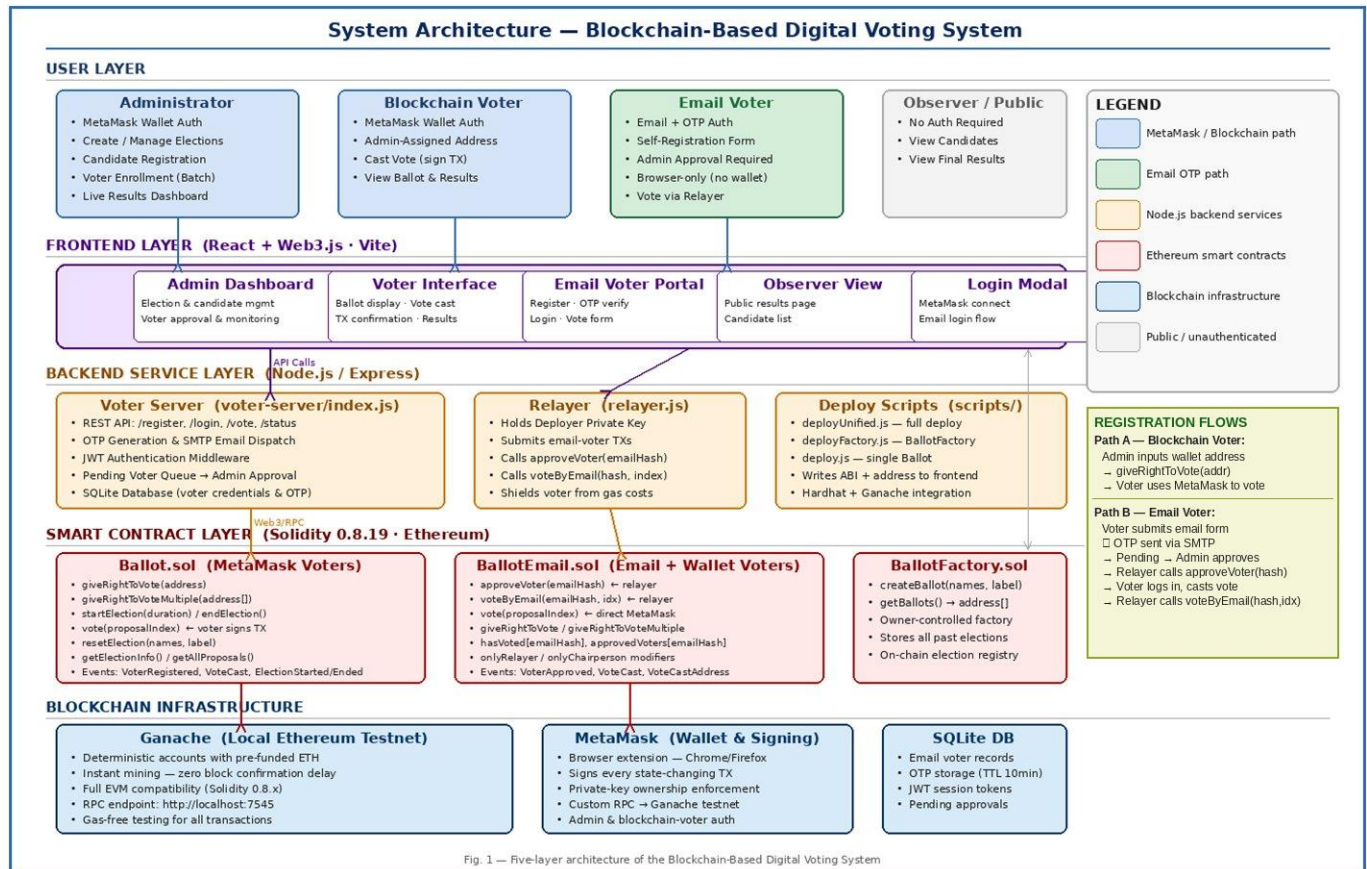


Figure 1. Five-layer architecture of the Blockchain-Based Digital Voting System.

B. Technology Stack

A modern Decentralised Application stack is selected, which aligns with the best practices in Ethereum developer. Ganache v7 is the built-in local blockchain, and therefore facilitates deterministic accounts that are initialized with pre-test ETH and instant mining which is really convenient. All smart contracts are compiled to Solidity 0.8.19, which already has inbuilt protection against overflow and added two of our personal modifiers just to keep the access control clean. Frontend is developed using the React 18 and the Vite build system, which is connected to the Web3.js to ensure that the interface is able to interact to the Ethereum RPC. MetaMask is the wallet of choice; it protects all transactions and adds the appropriate cryptographic ownership of all wallets to each user. The backend deploys an instance of a Node.js/Express application to do the OTP generation and SMTP email transmissions with Nodemailer is developed which deals with JWT and keeping the election data in a small SQLite database to ensure that databases are not lost during the email-voter interactions. A separate relayer module keeps the private key of the deployer secure and gives on-chain transactions on behalf of the email-registered voters and keeps them off the gase of paying gas fees and the administrative burden of configuring a wallet.

Table 1. Technology Stack of the Blockchain-Based Digital Voting System.

Layer	Technology	Purpose
Blockchain	Ethereum / Ganache v7	Immutable distributed ledger
Smart Contracts	Solidity 0.8.19	Voting rules and access control
Wallet / Auth	MetaMask	Transaction signing, admin/voter login
Frontend	React 18 + Web3.js	Voter and admin user interfaces
Backend	Node.js + Express	OTP, JWT, SMTP, relay
Database	SQLite	Email voter credentials and OTP storage
Network	LAN / IP addressing	Multi-device access without internet

C. Role Separation and Access Control

The user roles that are built around consist of three which have strict privileges that are enforced both on a smart contract level, as well as app level.

Administrator: uses MetaMask authentication, the address of the chairperson of the Ganache. Admin receive full Power-point control where he can create and administer elections, register voters and approve voters, cycle through the election cycle and get the real-time result. The smart contracts go to the extent of ensuring that the chairperson is not able to vote - just to be on the safe side.

Blockchain Voter: are whitelisted by the administrator which gives them a wallet address. Voter use MetaMask to log in and personal key to sign out every vote. It is the most powerful authentication, though it assumes that voter is already familiar with the blockchain.

Email Voter: These individuals register themselves through a self-registered form in a browser, authenticate themselves using OTP and are adjudicated by an administrative system. The relay makes the vote on its behalf. The vote integrity is also concrete by ensuring that the email address is never used due to the emailHash mapping of Ballotemail.sol that combat the chance of a single email address to receive a vote.

D. Network Architecture

Ganache is directly running on the laptop of the administrator exposing RPC endpoint on the localhost, port 7545. The React front end and the Express back end both would be served over the same host but this time over the LAN IP instead of the 127.0.0.1 address and thus anyone on campus could just open a browser and begin voting on the front end. Users of MetaMask configure a custom RPC to the that Ganache, and email voters communicate to the Express API. The relay is a kind of middle-man; it serves all the on-chain live traffic of the email voters.

E. Smart Contract Design

Ballot.sol is for election creation as well as vote casting. The state of address (weight, voted, vote) and name (voteCount ol) is a Voter struct entry which the store in a dynamic array, and Proposal struct (name, voteCount) is stored in a dynamic array. the onlyChairperson modifier checks the functions of the admin and the electionIsActive checks the functions of the election within its authorized time frame when timed. It includes giveRightToVote and giveRightToVoteMultiple to whitelist, startElection and endElection to manage the life cycle and vote enables a user to cast a single, atomic vote. resetElection is to clear the state between polls within the same contract. Each of the view is a pure function equivalent to getElectionInfo, and getAllProposals, therefore, there is no cost to getting the results live.

BallotEmail.sol is an extension of non-wallet users. Along with a standard voter whitelist, added a relayerAddress that can call approveVoter(bytes32 emailHash) and voteByEmail(bytes32 emailHash, uint proposal index) which is also keccak256 of the hash of the email, so it provide some email privacy on-chain and yet not compromising the rule that it is unlearned-by-anyone-at-once of a single voter.

Ballotfactory.sol is a registry helper. This one enables the owner of the factory to generate new Ballot and keep on chain a collection of all prior elections. The getBallots() view also provides the array of the previous election addresses in the past so that the archive is fully available.

F. Voter Registration Subsystem

The proposed system also has two major registration processes in order to meet various institutional configurations.

Direct blockchain registration: Each of the above steps will be done by the admin picking one or more Ganache wallet addresses on the dashboard and giveRightToVote or giveRightToVoteMultiple. To carry out the transaction on a class project it takes milliseconds on the local testnet- a transaction. This would work well in the scenario when the institution has a pool of the wallet addresses already.

Email self-registration: Voter fill in a form, including name, and email. System creates a six-digit OTP with an expiry of ten minutes, digest the password using bcrypt and store in SQLite and send the OTP using SMTP. Upon verification of the OTP, the account is kept in a pending status. After the approval of the thumbs-up by the admin, the relayer initiates approveVoter(emailHash) of BallotEmail.sol, and the voter receives a confirmation email. This solution has the effect that the on-chain record does not come into existence until a human verifies the legitimacy of the voter.

G. Voter Interface

Voter react components are designed to self-adapt to the state of the election. In case it is open, the names, logos, and voting buttons of every candidate are displayed by Voterinterface and VoterEmailinterface. After a vote has been hit, the component communicates the transaction hash or the acknowledgement of the server, and it blocks any further communication of that session. When the election ends, ObserverView just pops out presenting the final scores and declaring the winner in a read-only view which is accessible to all.

H. Security Analysis

Some of the threat vectors that are considered during the design phase and which might be used to compromise e-voting involve a variety. The proposed system ensures the vote cannot be altered in

future owing to the hash -chained transaction structure of the blockchain- changing a vote would be almost impossible against a properly configured blockchain. In order to preclude the chance of voting twice, system need to do atomic check in a single transaction: a voted boolean and a hasVoted mapping, whose property is that no two transactions may vote, and Ethereum order is used to order transactions who attempt to vote non-concurrently in such a way that they are mutually non-conflicting. The last step is to ensure that the admin is not able to vote by reversing any move made by the address of the chairperson. OTP will last ten minutes and save it using bcrypt, thus it is difficult to repeat or brutally attack a register endpoint. Our primary resource is the relayer: those who sign up using email are left with no choice but to believe the backend that their vote will be delivered properly. This is a well-known weakness that will be resolved in further work with the commit-reveal and zero-knowledge technology.

High-Level Design

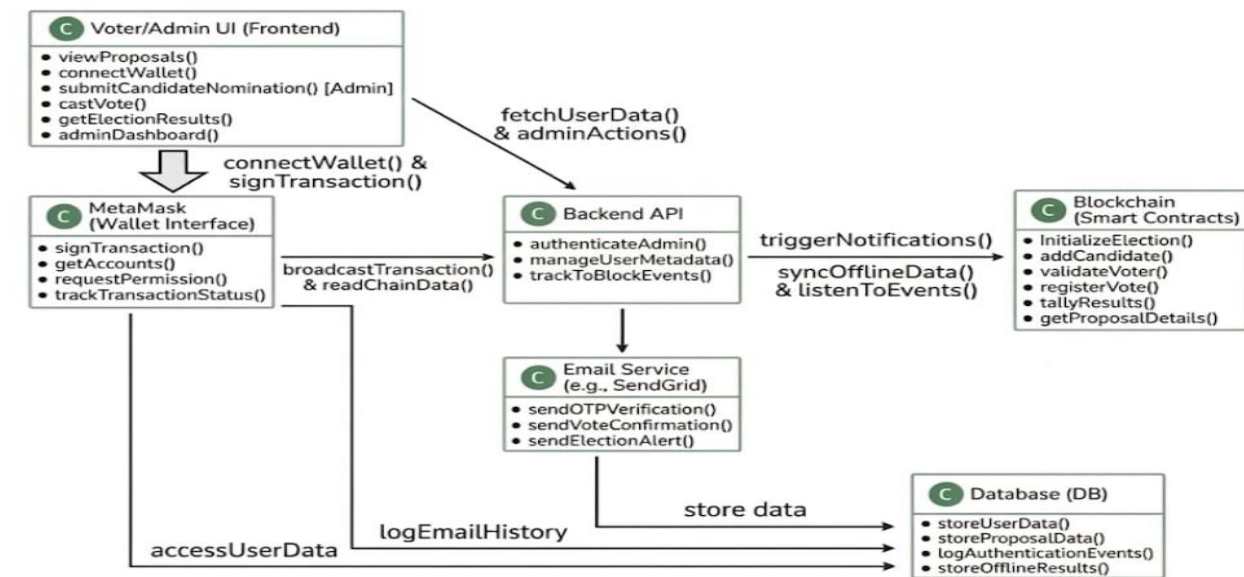


Figure 2. High-Level Design of the Blockchain-Based Digital Voting System.

IV. Results and Discussion

The proposed system is configured on a local network and it had four devices one being a local network -admin and the other voters. To ensure that this is in proper working order, system conducts some mock elections to make sure that it is accessible, resistant to manipulation and is quick to offer the outcomes and also user friendly.

A. Functional Correctness

All major functions were tested on all occasions: creating a new election, setting it up, creating the candidates, enrolling the voters either via the email or blockchain channel, voting, and retrieving the results. No multiple votes ever occurred, regardless of whether somebody attempted to send the form again, or replay a stolen HTTP message, or attempted to send two successive votes. Voted flag and

hasVoted mapping in the contract never permitted duplicates, and returned typical errors of revert of EVM.

Table 2. Functional details of the Blockchain-Based Digital Voting System.

Test Case	Method	Expected Result	Actual Result	Status
Admin login	MetaMask+chairperson address	Dashboard access granted	Dashboard displayed	Pass
Voter registration (blockchain)	giveRightToVote(addr)	Address whitelisted on-chain	Whitelisted in < 1 s	Pass
Voteregistration email + OTP)	SMTP OTP → admin approval	Voter approved, notified	Approved, email sent	Pass
Castvote(blockchain voter)	MetaMask-signed TX	Vote recorded on-chain	TX confirmed in < 1 s	Pass
Castvote(emailvoter)	Relayer voteByEmail()	Vote recorded via relayer	TX confirmed in < 1 s	Pass
Double-vote attempt	Re-submit vote form	Contract reverts TX	EVM revert: already voted	Pass
Admin vote attempt	Chairperson calls vote()	Contract reverts TX	EVM revert: not authorised	Pass
Batchvoter registration	giveRightToVoteMultiple()	10 addresses whitelisted	All registered registered in 1 TX	Pass
End election + view results	endElection()→ObserverView	Results visible to all	Final result displayed	Pass
Offline result query	getElectionInfo() view call	Gasless result retrieval	Returned in < 200 ms	Pass

B. Tamper Resistance and Immutability

Transactions explorer was applied to the raw blockchain state, which was inspected following the completion of every test election. Every vote was represented by a separate transaction, marked by its hash, a definite block time and a record of events. The contract returned an error at the EVM level whenever any other party (or a script) attempted to interfere with the vote counts of an address that had not been approved, regardless of whether or not there has been a change.

C. Real-Time Result Monitoring

Every two seconds, the dashboard of the admin made `getElectionInfo` and `getAllProposals` call. The display of the new count took between one and two seconds after hitting a vote. Since the system do not use an extra tallying server, there exist no single point of failure risks and this demonstrates that native blockchain aggregation can be used to perform real-time monitoring at the scale.

D. Usability

Individuals register through their respective emails, including those who had never used blockchain before, could register, pass the OTP, log-in, and vote with only a simple instruction sheet that no additional assistance was required. Individuals who had directly registered on the chain had a single MetaMask set up, after which it was identical. This demonstrates that providing the two approaches really strike the right balance between high security and convenience.

E. Comparative Analysis

Table 1 is a comparison between the system and three previous methods and evaluates it in eight metrics. The proposed system match or surpass the previous blockchain prototypes by every security feature and taken an actual leap forward in ease of use with the email registration feature. The actual disadvantage of centralized e-voting that does manifest itself is scaling over on big chains, which address in subsequent iterations.

Table 3. Comparative Analysis of Voting System Approaches

Criterion	Paper Voting	Central e-Vote	Blockchain (Prior Work)	Proposed System
Tamper Resistance	Low	Moderate	High	High
Transparency	Low	Low	High	High
Double-Vote Prevention	Manual	Software	Contract	Contract
Voter Privacy	Physical	DB-based	Pseudonymous	ZKP (future)
Auditability	Limited	Partial	Full	Full
Decentralisation	None	None	Full	Full
Ease of Use	High	High	Moderate	High
Scalability	High	High	Limited	Limited

F. Limitations

The proposed system is tested on the local Ganache test network in the experiments. In the real Ethereum mainnet, gas charges may arise to high levels particularly in the process of an election involving a large number of voters. These costs may be alleviated with the help of Layer-2 roll-ups. Voter privacy is now based on pseudonymous addresses and hashed emails which is less secure than

actual cryptographic anonymity. Inclusion of zero-knowledge proof systems would help in better reassurances. The system is also still based on a relayer to perform a subset of the work; the next generation will be improved to encompass a bit of work to support commit-reveal protocols or be replaced with ZKP-based voting flows.

V. Future Work

Depending on limitations identified in the discussion and evaluation sections, four major directions of the future work have been identified.

- **Public / Layer-2 Testnet Deployment:** The system will be launched to Sepolia testnet on Ethereum and subsequently to some layer-two test network, such as Polygon zkEVM or Optimism, to check the financial states of realistic gas price cap, delay caused by congestion and network adversarial behaviour. This will generate empirical information on the cost per voter and transaction throughput restraint limits of the existing contract design.
- **Zero-Knowledge Proof Integration:** relying on zk-SNARK Voter anonymity shall be implemented using Groth16 proof system, as imported in the snarkjs library. This will give voters a way to authenticate their eligibility and uniqueness of their vote without letting anybody know who they are or their vote, which will ensure formal anonymity; a feature the current pseudonymity model cannot provide.
- **Email Voter commit-reveal Scheme:** In order to eliminate the relayer trust dependency on email voters, commit-reveal protocol will be utilized. The voters make a blinded commitment that they will follow on their promise during the commit stage; during the reveal stage, the relayer provides the opening that is verified on-chain. This creates a situation in which the relayer has to be honest about the choice made by the voter, and would create a verifiable on-chain discrepancy should he be untruthful.
- **Formal Smart Contract Security Audit:** The symbolic execution tools of Mythril and Slither will be used to perform formal security analysis on all three Solidity contracts, and Echidna will be used to fuzz property-based on the contracts. Vulnerabilities to be targeted by the audit will include reentrancy vulnerabilities, integer overflow/underflow, access control bypasses and denial-of-service vectors before any actual implementation.
- **Decentralised Identity Integration** W3C Decentralised Identifiers (DIDs) and Verifiable Credentials is an alternative to the current email-based user registration route to self-sovereign voter identity, which is able to store credentials and dispatch OTPs by a trusted backend server.
- **Mobile Application:** To enhance the presence of the system among blockchain-registered voters, a specialized mobile app based on React Native will be created that will lessen the reliance on the MetaMask browser add-on and allow biometric authentication as a signing proxy.

VI. Conclusion

In the current work, a digital voting system is developed, which leverages blockchains, and that completely overcomes the most significant limitations of the traditional paper-based voting system and non-decentralized electronic ones. The design prevents tampering and verifiability without any trust in one administrator, by coding all election rules in Ethereum smart contracts and enforcing these codes by the EVM transaction model. The dual-purpose model of integrating the voting of the blockchain with MetaMask and the OTP-authenticated email voting with the relayer maintains a low barrier of entry to non-technical voters and high levels of authentication to those voters who are communicating with the blockchain directly.

Empirical testing established that the system functions in practice in every test case: receipt of duplicate votes by a user was impossible; outcomes are disseminated in real time, with a delay of less than two seconds per vote by the user; and users with no background in blockchain usage could succeed. These results justify the fact that properly designed smart-contract architectures can serve the competing needs of security, transparency and usability which define trustful election systems.

The proposed work can further be developed in four directions: implementation on an actual testnet or Ethereum-based Layer-2 network with realistic gas-cost profiling; implementing a scheme that temporarily relies on commit-reveal voting, an implementation that does not depend on the relayer; and formal security audits of Solidity codebase, and then no-real-world implementations of the Solidity ahead system.

V. References

- [1] Rivest, R. L. and Wack, J. P. (2006). On the notion of software independence in voting systems. *Philosophical Transactions of the Royal Society A*, 366(1881), pp. 3759–3767.
- [2] Ayed, A. B. (2017). A conceptual secure blockchain-based electronic voting system. *International Journal of Network Security and Its Applications (IJNSA)*, 9(3), pp. 1–9.
- [3] Zheng, Z., Xie, S., Dai, H., Chen, X. and Wang, H. (2017). An overview of blockchain technology: Architecture, consensus, and future trends. *Proceedings of the IEEE International Congress on Big Data*, pp. 557–564.
- [4] McCorry, P., Shahandashti, S. F. and Hao, F. (2017). A smart contract for boardroom voting with maximum voter privacy. *Proceedings of the 21st International Conference on Financial Cryptography and Data Security (FC 2017)*, *Lecture Notes in Computer Science*, vol. 10322, pp. 357–375.
- [5] Hardwick, F. S., Akram, R. N. and Markantonakis, K. (2018). E-voting with blockchain: An e-voting protocol with decentralisation and voter privacy. *Proceedings of the IEEE International Conference on Internet of Things (iThings-2018)*, pp. 1561–1567.
- [6] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. [online] Bitcoin.org. Available at: <https://bitcoin.org/bitcoin.pdf>
- [7] Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, pp. 1–32.
- [8] Buterin, V. (2013). Ethereum white paper: A next-generation smart contract and decentralised application platform. *Ethereum Foundation*.
- [9] Agora. (2018). Agora blockchain voting in Sierra Leone. *Agora Foundation*. Available at: <https://www.agora.vote>

- [10] Specter, M., Koppel, J. and Weitzner, D. (2020). The ballot is busted before the blockchain: A security analysis of Voatz. Proceedings of the 29th USENIX Security Symposium, pp. 1535–1553.
- [11] Benaloh, J. and Tuinstra, D. (1994). Receipt-free secret-ballot elections. Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC), pp. 544–553.
- [12] Cramer, R., Gennaro, R. and Schoenmakers, B. (1997). A secure and optimally efficient multi-authority election scheme. Advances in Cryptology — EUROCRYPT 1997, Lecture Notes in Computer Science, vol. 1233, pp. 103–118.
- [13] Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E. and Virza, M. (2014). Zerocash: Decentralized anonymous payments from Bitcoin. Proceedings of the IEEE Symposium on Security and Privacy, pp. 459–474.
- [14] Buterin, V. (2021). Why sharding is great: Demystifying the technical properties. Ethereum Foundation Blog. Available at: <https://vitalik.ca/general/2021/04/07/sharding.html>
- [15] Thibault, L. T., Sarry, T. and Hafid, A. S. (2022). Blockchain scaling using rollups: A comprehensive survey. IEEE Access, 10, pp. 93039–93054.
- [16] Fujioka, A., Okamoto, T. and Ohta, K. (1992). A practical secret voting scheme for large scale elections. Advances in Cryptology — AUSCRYPT 1992, Lecture Notes in Computer Science, vol. 718, pp. 244–251.
- [17] Sporny, M., Longley, D. and Chadwick, D. (2022). Verifiable Credentials Data Model v1.1. W3C Recommendation. Available at: <https://www.w3.org/TR/vc-data-model/>
- [18] Dafflon, R., Nazar, J. and Rochat, T. (2021). EIP-2771: Secure protocol for native meta transactions. Ethereum Improvement Proposals. Available at: <https://eips.ethereum.org/EIPS/eip-2771>
- [19] Kulyk, O., Neumann, S., Koleber, E. and Volkamer, M. (2015). Towards usability of automated verification: The case of electronic voting. Proceedings of the International Conference on Availability, Reliability and Security (ARES), pp. 225–232.
- [20] Cortier, V., Galindo, D. and Turuani, M. (2016). A formal analysis of the Helios voting protocol. Proceedings of the IEEE 29th Computer Security Foundations Symposium (CSF), pp. 482–496.