International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 985-995
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0140

Analyzing The Use of Blockchain in E – Voting Systems

Chriso Christudhas[1], Rajkumar Shende[2],
[1]PG Student, Dept. of Computer Engineering, St. Francis Institute of Tech., Borivali, Maharashtra, India
[2]Associate Professor, Dept. of Computer Engineering, St. Francis Institute of Tech., Borivali, Maharashtra, India
Emails: *chriso2212@gmail.com[1],rajkumarshende@sfit.ac.in[2]*

Abstract
Modernization of electoral systems is essential as digital platforms increasingly replace traditional processes. This paper presents a secure blockchain-based e-voting system implemented using Ethereum smart contracts to address transparency, trust, and tamper-resistance concerns. A commit–reveal voting protocol preserves ballot secrecy by concealing voter choices during the voting phase, while a Merkle tree–based whitelist authenticates eligible voters efficiently without storing identities on-chain. Each voter casts exactly one vote using a unique blockchain address, enforced through smart-contract logic. The system ensures immutability, integrity, and verifiability of election results and was implemented and tested using the Ganache Ethereum local blockchain and MetaMask, demonstrating a scalable and privacy-preserving e-voting solution suitable for real-world deployment.

Keywords: *Blockchain, E-Voting, Commit–Reveal Protocol, Merkle Tree, Ethereum, Ganache*

1. Introduction
Electronic voting (e-voting) refers to the use of electronic systems to cast, transmit, and count votes. While e-voting improves efficiency, reduces human error, and accelerates result computation, it also raises significant concerns regarding transparency, security, and voter trust. Conventional electronic voting machines (EVMs) often rely on proprietary "black-box" software, limiting public verifiability and increasing skepticism regarding vote integrity and auditability. Additionally, centralized control in traditional systems introduces risks of tampering, limited transparency, and challenges in preserving voter anonymity. Blockchain technology offers a decentralized and immutable alternative capable of addressing these limitations. By recording transactions across distributed nodes, blockchain systems provide tamper resistance, transparency, and verifiable record-keeping without reliance on a single trusted authority. Leveraging these properties, this paper proposes a blockchain-based e-voting system implemented using Ethereum smart contracts that emphasizes privacy, integrity, and scalability. To preserve ballot secrecy and prevent coercion, the system employs a commit–reveal voting protocol, where votes are cryptographically committed before being revealed after the voting phase. To improve scalability and reduce on-chain storage overhead, a Merkle tree–based whitelist is used to verify voter eligibility through cryptographic proofs. The proposed design enforces one-vote-per-voter, eliminates centralized control, and ensures immutable election results. Implementation and testing on a private Ethereum network demonstrate that integrating commit–reveal voting with Merkle-based authorization provides a secure and scalable framework for modern electronic voting systems.

2. Literature Survey
Canidio and Danos [1], [2] analyze front-running attacks in blockchain systems and propose commit–reveal mechanisms as a mitigation strategy. Their work demonstrates that commit–reveal protocols prevent harmful transaction reordering while preserving fairness among honest participants, establishing commit–reveal as a practical cryptographic primitive for decentralized applications. Lee et al. [3] identify vulnerabilities in traditional commit–reveal schemes, particularly last-revealer attacks, and propose Commit-Reveal², a layered protocol with randomized reveal ordering and accountability mechanisms, improving robustness while reducing gas costs. Complementing application-layer solutions, Kelkar et al. [4] address

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 985-995
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0140

fairness at the consensus level through order-fair protocols that limit adversarial manipulation of transaction ordering. Merkle tree structures have been widely studied for secure and efficient data verification in blockchain systems. Kuznetsov et al. [5] analyse the collision resistance and security parameters of Merkle trees, emphasizing the importance of appropriate hash length and tree depth. Deng et al. [6] improve Ethereum's Merkle Patricia Trie performance using GPU acceleration and parallel hashing techniques, while Zhang et al. [7] propose QMDB, an optimized Merkle-based database that enhances throughput while preserving cryptographic verifiability. Oberst [8] explores stateless client architectures using Verkle trees and SNARK-integrated Merkle trees, demonstrating improved scalability and reduced state storage requirements. Security challenges in Ethereum ecosystems have also been extensively studied. He et al. [9] analyses transaction-based phishing attacks (TxPhish) and propose detection mechanisms. Guan and Li [10] investigate address poisoning attacks exploiting wallet interface vulnerabilities, while Yu et al. [11] examine user behaviour in wallet selection, highlighting trade-offs between usability and security. Yan et al. [12] identify weaknesses in Web3 authentication systems, particularly blind message attacks involving wallet-based authentication tools such as MetaMask. Private Ethereum environments and blockchain-based identity systems further demonstrate practical deployment frameworks. Mathur [13] presents GANACHE as a secure experimentation platform incorporating cryptographic protections such as encryption and zero-knowledge proofs. Lopes et al. [14] propose a blockchain-based digital identity management system implemented using Ethereum and Ganache, validating decentralized identity architectures. Finally, Hajian Berenjestanaki et al. [15] provide a comprehensive review of blockchain-based e-voting systems, identifying benefits such as transparency and integrity while highlighting persistent challenges in privacy, scalability, and usability. Collectively, these works establish the theoretical and technical foundations for fairness-preserving transaction protocols, scalable cryptographic data structures, and secure blockchain deployments. However, existing studies typically address these components in isolation—either focusing on transaction fairness, data structure optimization, or voting system design. The proposed system builds upon these foundations by integrating commit–reveal voting with a Merkle tree–based authorization mechanism within a unified smart-contract architecture. This combined approach simultaneously addresses privacy, integrity, and scalability in a practical blockchain e-voting implementation, thereby bridging the gap between theoretical security mechanisms and applied election system design.

3. Methodology

This section describes the design and implementation methodology of the proposed blockchain-based e-voting system. The system was developed incrementally in three stages: (i) a baseline blockchain voting system, (ii) an enhanced privacy-preserving voting system using a commit–reveal protocol, and (iii) a scalable authorization mechanism using a Merkle tree–based voter whitelist. Each stage builds upon the previous one to address specific limitations related to transparency, privacy, and scalability.

3.1. Applications and Tools Used

The proposed e-voting system was implemented using the Ethereum blockchain framework and supporting development tools. The following applications were used throughout the development and testing process:

- **Solidity** was used as the smart contract programming language to implement election logic, access control, and voting rules.
- **Ganache** was used to create a local private Ethereum blockchain, providing a deterministic environment for deploying and testing smart contracts.
- **MetaMask** was used as a blockchain wallet for managing accounts, signing transactions, and simulating voters and administrators.
- **Remix IDE** was used for writing, compiling, deploying, and interacting with smart contracts.
- **Node.js** was used to generate Merkle trees, Merkle roots, and cryptographic proofs for voter authorization in the final system.

These tools collectively enabled secure, repeatable,

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 985-995
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0140

and transparent testing of the voting protocol without relying on external infrastructure.

3.2.Baseline Blockchain Voting System (Simple Voting)

The first stage of the methodology involved implementing a basic blockchain-based e-voting system to establish core functionality and correctness.

3.2.1. Election Setup

In this phase, an administrator deploys a smart contract to the blockchain. The administrator defines the list of candidates participating in the election. Each candidate is assigned a unique identifier within the smart contract. The contract also initializes the election phase and enforces role-based access control, ensuring that only the administrator can configure election parameters.

3.2.2. Voter Authorization

In the baseline system, eligible voters are explicitly authorized by the administrator using an on-chain whitelist. Each voter is identified by a unique Ethereum wallet address. The smart contract maintains a mapping of authorized addresses, ensuring that only eligible voters can participate in the election.

3.2.3. Vote Casting

Each voter casts a vote by submitting a transaction to the blockchain specifying the chosen candidate. The smart contract verifies that:

a) The voter is authorized,
b) The voter has not voted previously, and
c) The vote is submitted during the correct election phase.

Upon successful validation, the vote count for the selected candidate is incremented immediately.

3.2.4. Vote Counting and Verification

Votes are counted in real time as transactions are confirmed on the blockchain. Since all votes are stored as immutable blockchain transactions, the results can be publicly verified. However, in this baseline approach, vote choices are visible on-chain as soon as votes are cast, which raises privacy concerns.

3.3.Privacy Enhancement Using Commit–Reveal Protocol

To address the lack of vote privacy in the baseline system, a commit–reveal protocol was introduced in the second stage of the methodology.

3.3.1. Commit Phase

During the commit phase, each voter submits a cryptographic commitment instead of the actual vote. The commitment is generated by hashing the selected candidate identifier, a secret value chosen by the voter, and the voter's blockchain address. This hash is stored on-chain, while the actual vote choice remains hidden. The smart contract verifies voter eligibility and ensures that each voter can submit only one commitment.

3.3.2. Reveal Phase

After the commit phase ends, the election transitions to the reveal phase. In this phase, voters reveal their original vote choice and secret. The smart contract recomputes the hash using the revealed data and compares it with the previously stored commitment. If the hashes match, the vote is accepted and counted. This two-phase approach ensures that vote choices remain confidential during the election while still allowing correct and verifiable vote counting after the reveal phase.

3.3.3. Privacy and Integrity Guarantees

The commit–reveal protocol prevents early disclosure of voting trends, mitigates vote coercion, and ensures that voters cannot change their vote after committing. At the same time, cryptographic verification guarantees the integrity of the election results.

3.4.Scalable Voter Authorization Using Merkle List

The third stage of the methodology focuses on improving scalability and gas efficiency by replacing explicit on-chain voter whitelists with a Merkle tree–based authorization mechanism.

3.4.1. Merkle Tree Construction

A list of eligible voter addresses is prepared off-chain by the administrator. These addresses are hashed and organized into a Merkle tree using a cryptographic hash function. The resulting Merkle root uniquely represents the entire voter list and is stored on-chain in the smart contract.

3.4.2. Merkle Proof-Based Authentication

Instead of storing all voter addresses on-chain, each voter provides a Merkle proof during the commit phase. The smart contract verifies the proof against the stored Merkle root to confirm voter eligibility.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 985-995
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0140

This approach allows voters to prove authorization without revealing or storing the complete voter list on the blockchain.

3.4.3. Integration with Commit–Reveal Voting

Merkle proof verification is performed during the commit phase to ensure that only eligible voters can submit commitments. Once verified, the commit–reveal process proceeds as described earlier. This integration combines scalable authorization with privacy-preserving voting.

3.5.Final Vote Counting and Result Publication

After all eligible voters have revealed their votes, the administrator finalizes the election. The smart contract prevents any further voting or modifications. Final vote counts for each candidate are stored on-chain and can be accessed by any participant. Since all votes are validated through cryptographic commitments and Merkle proofs, the published results are immutable, verifiable, and resistant to tampering.

3.6.Summary of Methodology Evolution

The methodology progresses from a simple blockchain voting system to a robust, privacy-preserving, and scalable e-voting protocol. The baseline system establishes correctness and transparency, the commit–reveal protocol enhances voter privacy and fairness, and the Merkle list significantly improves scalability and gas efficiency. Together, these components form a comprehensive blockchain-based e-voting system suitable for real-world deployment.

4. Results

The results show progressive improvements in the proposed e-voting system. While the baseline blockchain approach ensured vote integrity and transparency, it lacked privacy and scalability. The commit–reveal protocol preserved ballot secrecy, and the Merkle tree–based whitelist further enhanced scalability and efficiency. Together, these mechanisms provide a secure and practical blockchain-based e-voting solution.

4.1.Simple Blockchain Voting

4.1.1. System Initialization and Deployment

The voting smart contract was successfully compiled and deployed on a local Ethereum blockchain created using the Ganache environment. Remix IDE was used for compilation and deployment, with MetaMask acting as the transaction signer. The administrator account was used to deploy the contract and initiate the election setup process.

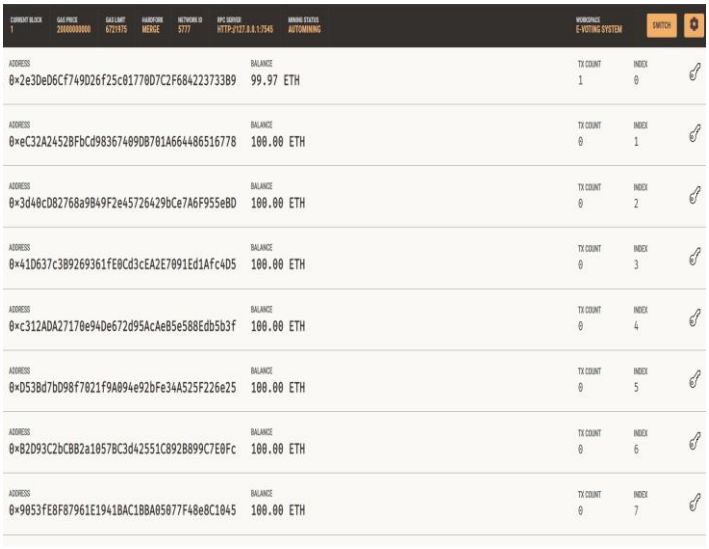

Figure 1 Ganache Network

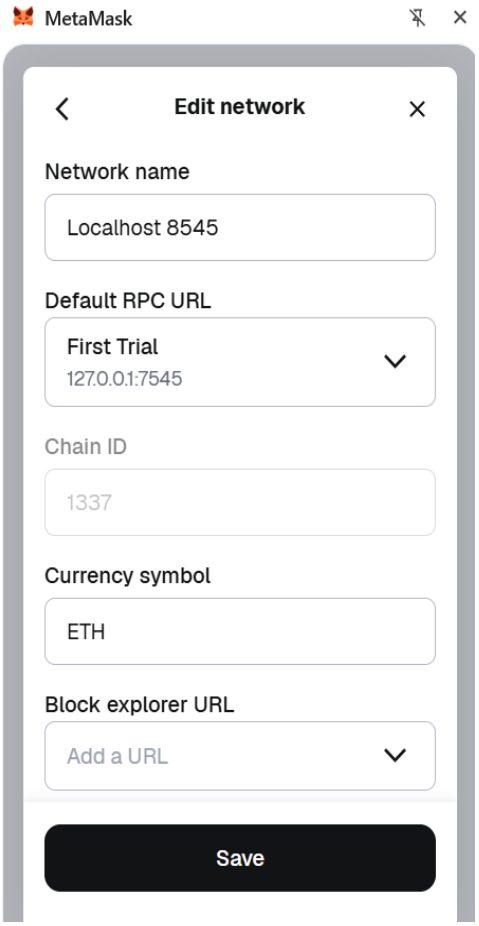

Figure 2 MetaMask Connected to Ganache

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 985-995
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0140

Figure 3 Imported Accounts

These results confirm that the blockchain environment was correctly configured and that the smart contract was deployed under administrator control.

4.1.2. Candidate Registration and Voter Whitelisting

Following deployment, the administrator successfully registered multiple candidates using the smart contract's candidate management functions. Eligible voters were explicitly authorized by adding their blockchain wallet addresses to an on-chain whitelist maintained by the contract.

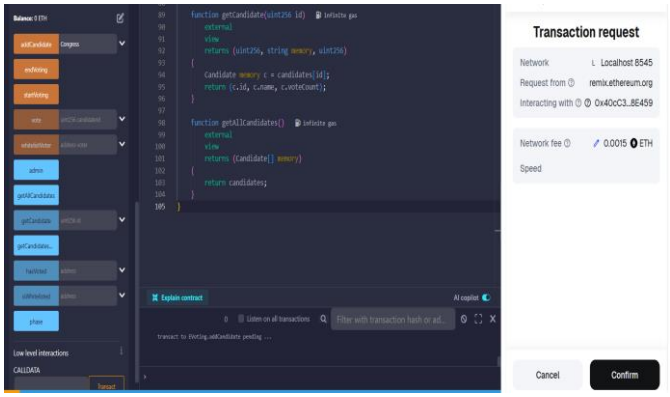

Figure 4 Addition Transaction

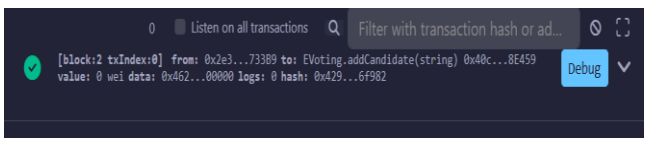

Figure 5 Addition Success

This step ensured that only pre-approved voter addresses could participate in the election, enforcing access control through smart-contract logic.

4.1.3. Vote Casting and Transaction Validation

Voting was carried out by switching MetaMask accounts from the administrator to individual voter accounts. Each voter submitted a vote as a blockchain transaction. The smart contract validated each transaction by checking voter eligibility, ensuring that the voter had not previously voted, and confirming that voting was performed during the active election phase.

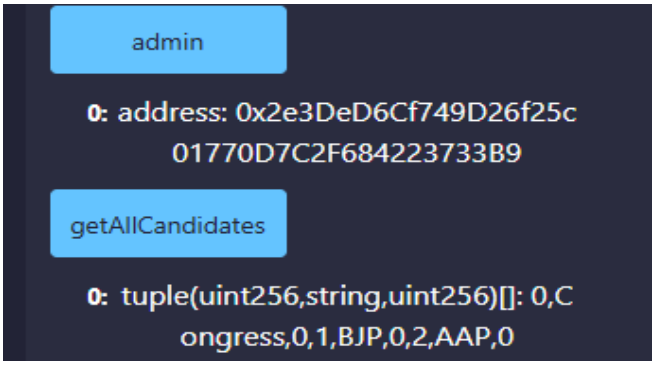

Figure 6 Information Utility

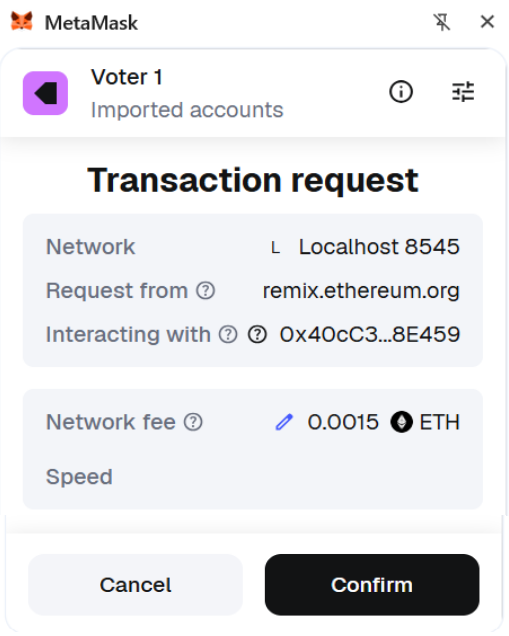

Figure 7 Vote Confirmation Transaction

The results show that each valid vote was successfully recorded on-chain, and the vote count for the selected candidate was incremented immediately after transaction confirmation.

4.1.4. Enforcement of One-Vote-Per-Voter

The system correctly prevented duplicate voting

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 985-995
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0140

attempts. When a voter attempted to cast a second vote, the smart contract rejected the transaction, demonstrating enforcement of the one-vote-per-voter rule through on-chain state variables.

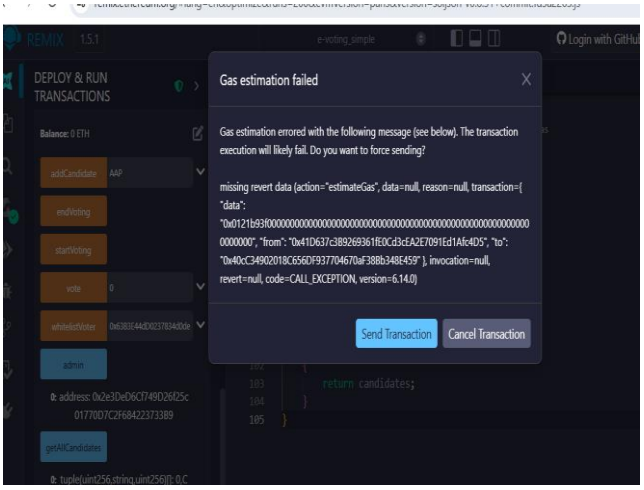

Figure 8 Duplicate Votes

This confirms the correctness of vote integrity enforcement in the baseline system.

4.1.5. Election Finalization and Result Retrieval

After all votes were cast, the administrator ended the election using the designated smart contract function. Final results were retrieved directly from the blockchain using read-only contract calls, displaying both individual candidate results and aggregated vote counts.

Figure 9 Voting End via Admin

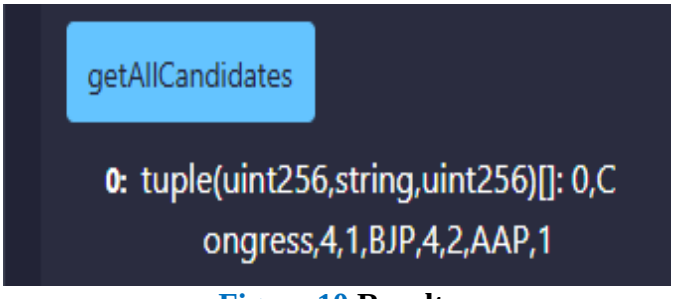

Figure 10 Results

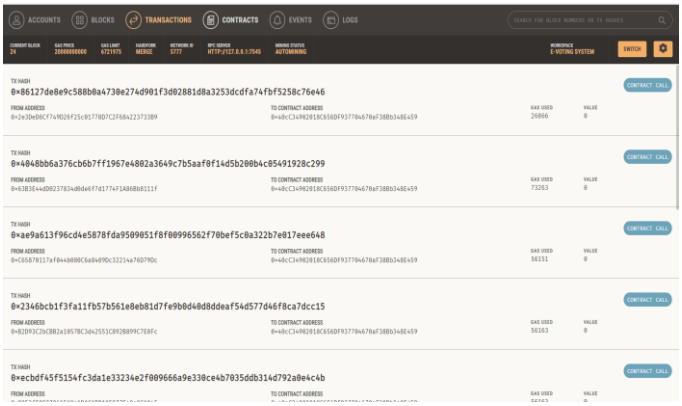

Figure 11 Ganache Transactions

Since votes are stored as immutable blockchain transactions, the final election outcome could be independently verified by inspecting the blockchain ledger.

4.1.6. Observations and Limitations

The baseline blockchain voting system successfully demonstrated:

- Correct vote recording and counting
- Transparency and auditability of election results
- Immutability of votes once recorded
- However, the results also revealed key limitations:
- Vote choices were publicly visible on-chain during the election
- Voter identities were explicitly stored in an on-chain whitelist
- Administrative overhead increased linearly with the number of voters

These limitations motivated the introduction of privacy-preserving and scalable mechanisms in subsequent system iterations.

4.2.Results: Commit–Reveal Blockchain Voting

This section presents the experimental results obtained from the commit–reveal–based blockchain voting system. The objective of this phase was to evaluate the effectiveness of the commit–reveal protocol in preserving ballot secrecy during the voting process while maintaining vote integrity and verifiability.

4.2.1. Vote Commitment Phase

During the commit phase, voters did not submit their actual vote choices directly to the blockchain. Instead, each voter generated a cryptographic

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 985-995
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0140

commitment by hashing three parameters: the selected candidate identifier, a secret value known only to the voter, and the voter's blockchain address. This hash was submitted to the smart contract as a commitment.

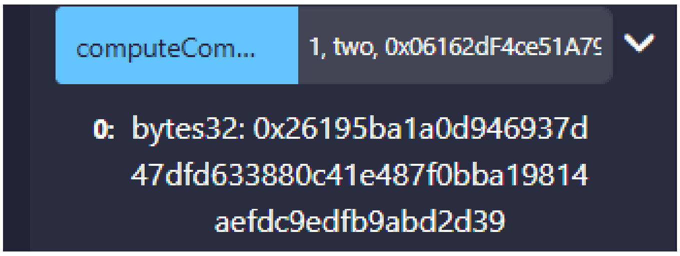

Figure 12 Computing Hash Code

These results confirm that vote commitments were successfully generated off-chain and stored on-chain without revealing the voter's actual choice.

4.2.2. Commitment Verification and Storage

Once the commitment hash was submitted, the smart contract verified that the voter was eligible and had not previously committed a vote. The hash was then stored on-chain, and the voter's committed status was updated accordingly.

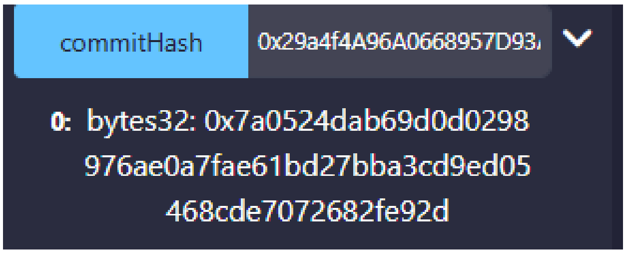

Figure 13 Hash Verification

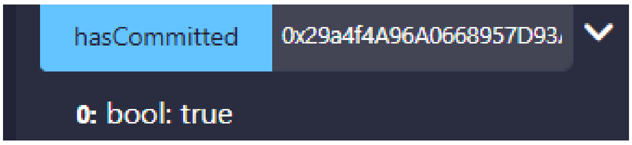

Figure 14 Vote Confirmation

This demonstrates that the system correctly tracked voter participation while keeping vote choices confidential during the commit phase.

4.2.3. Transition to Reveal Phase

After all eligible voters submitted their commitments, the election administrator transitioned the system from the commit phase to the reveal phase. No additional commitments were accepted after this transition. This step ensured that vote revelation could only occur after all commitments were finalized.

4.2.4. Vote Reveal and Validation

In the reveal phase, voters submitted their original candidate identifier and secret value. The smart contract recomputed the hash using the revealed information and compared it with the previously stored commitment. Votes were accepted only if the recomputed hash matched the original commitment.

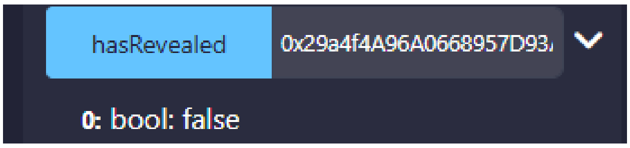

Figure 15 Revealed or Not

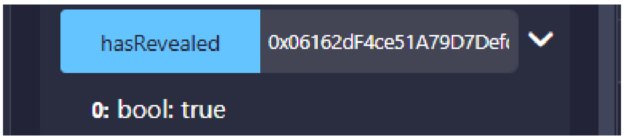

Figure 16 After Reveal

The results show that valid reveals were accepted and counted correctly, while the system prevented vote modification after commitment.

4.2.5. Detection of Invalid Reveal Attempts

The system successfully rejected invalid reveal attempts where the revealed candidate or secret did not match the original commitment. Such attempts resulted in transaction reversion, demonstrating the binding property of the commit–reveal protocol.

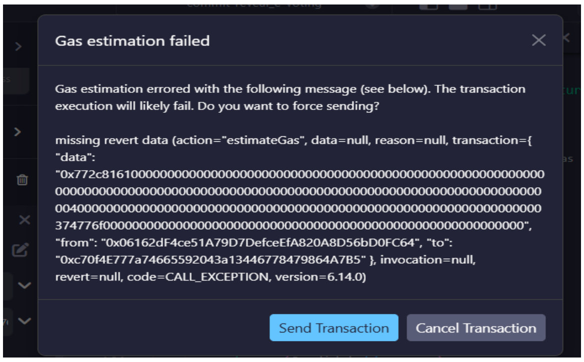

Figure 17 Gas Estimation

Voting during Reveal Phase not actionable This confirms that voters cannot alter their vote after committing, ensuring election integrity.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 985-995
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0140

4.2.6. Final Vote Counting and Result Verification

After all valid reveals were processed, final vote counts were retrieved directly from the blockchain. The results reflected only correctly revealed votes and were immutable once recorded.

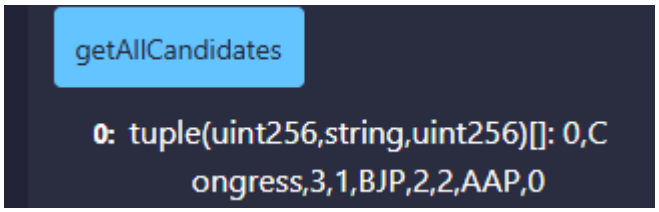

Figure 18 Results

These results demonstrate that the commit–reveal protocol successfully balances privacy and verifiability by hiding votes during the election and revealing them only after the voting phase concludes.

4.2.7. Observations

The commit–reveal voting system achieved the following:

- Preserved ballot secrecy during the voting phase
- Prevented early disclosure of voting trends
- Ensured cryptographic binding between commitment and revealed vote
- Maintained transparency and immutability of final results

However, the approach required multiple transactions per voter and introduced additional protocol complexity, motivating the use of scalable authorization techniques in the final system.

4.3.Results: Merkle List–Based Commit–Reveal Blockchain Voting

This section presents the experimental results obtained from the final version of the proposed e-voting system, which integrates a Merkle tree–based voter whitelist with the commit–reveal voting protocol. The objective of this phase was to evaluate improvements in **authorization efficiency and scalability**, while preserving the **privacy and integrity guarantees** established by the commit–reveal mechanism.

4.3.1. Merkle-Based Voter Authorization

In the proposed system, explicit on-chain voter whitelists were replaced with a Merkle tree–based authorization mechanism. Eligible voter addresses were processed off-chain to construct a Merkle tree, and the resulting Merkle root was stored on-chain. During the commit phase, voters proved their eligibility by submitting a Merkle proof along with their vote commitment.

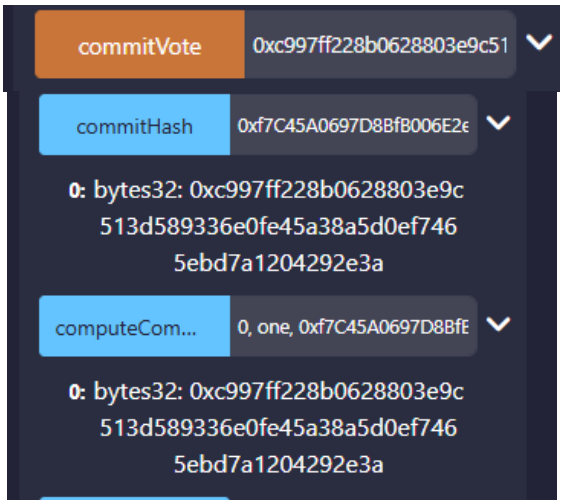

Figure 19 Merkle Proof Verification During Vote Commitment

These results confirm that Merkle proofs were successfully verified by the smart contract, allowing only eligible voters to participate without storing individual voter identities on-chain.

4.3.2. Vote Commitment with Merkle Proof Verification

During the commit phase, each voter generated a cryptographic commitment hash using their selected candidate identifier, a secret value, and their blockchain address. The commitment was submitted together with the corresponding Merkle proof. The smart contract verified both voter eligibility and commitment uniqueness before accepting the vote.

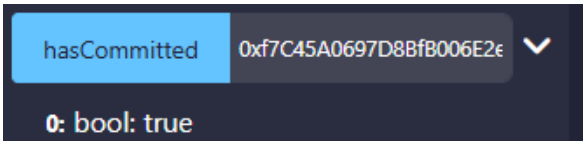

Figure 20 Authorized Voter

This demonstrates that only authorized voters could successfully commit their votes and that duplicate commitments were prevented without requiring per-

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 985-995
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0140

voter on-chain authorization.

4.3.3. Final Results and Election Integrity

After all valid commitments were revealed using the commit–reveal protocol, the administrator finalized the election. Final vote counts were retrieved directly from the blockchain, reflecting only successfully verified and authorized votes. Since all results are recorded on-chain, they remain immutable and publicly verifiable.

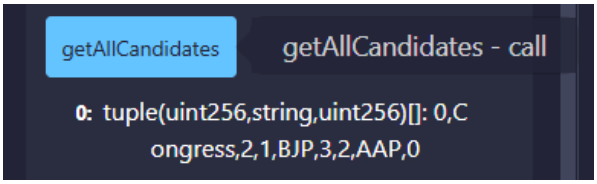

Figure 21 Results

This confirms that integrating Merkle-based authorization does not affect vote correctness or result integrity.

4.3.4. Efficiency and Scalability Analysis

An analysis of blockchain transaction activity revealed significant efficiency improvements over previous system versions. The administrator performed only a limited number of setup and phase-transition transactions, while each voter required exactly two transactions—one for vote commitment and one for vote revelation—regardless of the total number of voters. Notably, no per-voter authorization transactions were required from the administrator.

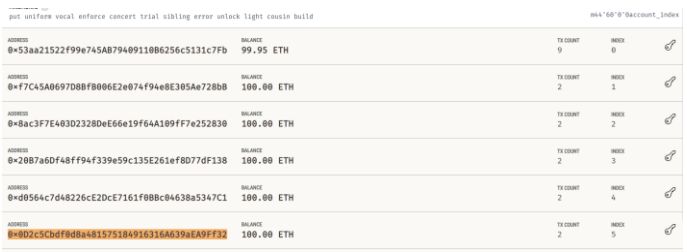

Figure 22 Ganache Transaction Overview Showing Reduced Administrative Overhead

The Ganache account summary demonstrates a constant transaction count per voter and minimal administrative involvement, validating the scalability benefits of using a Merkle tree–based whitelist.

4.3.5. Observations

The integrated Merkle list and commit–reveal system achieved the following:

- Scalable voter authorization with constant on-chain storage
- Preservation of voter privacy during the election
- Prevention of unauthorized and duplicate voting
- Immutable and verifiable election results
- Reduced administrative overhead compared to explicit whitelisting

These results confirm that the final system effectively addresses the limitations identified in earlier implementations and provides a secure, efficient, and privacy-preserving blockchain-based e-voting solution.

4.4.Comparative Analysis

Table 1 Feature Comparison

Feature	Simple	Commit-Reveal	Merkle+Commit-Reveal
Vote privacy during election	No	Yes	Yes
One-vote-per-voter	Yes	Yes	Yes
Early result visibility	Yes	No	No
On-chain voter whitelist	Yes	No	No
Scalability	Low	Medium	High
Gas-efficient authorization	No	No	Yes

Table 1 highlights the functional evolution of the e-voting system across the three implementations. While the baseline blockchain approach ensures vote integrity and transparency, it lacks privacy and scalability. The commit–reveal protocol addresses vote confidentiality and fairness by hiding voter choices during the election. The final system further improves scalability and authorization efficiency by

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 985-995
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0140

introducing a Merkle tree–based whitelist, completing the functional requirements of a secure e-voting system.

Table 2 Transaction Cost Comparison

Aspect	Simple	Commit-Reveal	Merkle+Commit-Reveal
Admin setup transactions	O(n)	O(n)	O(1)
Voter transactions	1	2	2
On-chain whitelist storage	Yes	Yes	No
Proof verification	No	No	Yes (O(log n))

Table 2 summarizes the security guarantees provided by each system version. The baseline implementation ensures immutability but exposes voter behavior on-chain. The commit–reveal protocol significantly strengthens security by providing ballot secrecy and resistance to vote coercion. Integrating Merkle-based authorization further enhances privacy by preventing public enumeration of eligible voters, resulting in a system that satisfies both integrity and confidentiality requirements.

Table 3 Security Properties

Property	Simple	Commit-Reveal	Merkle+Commit-Reveal
Immutability	Yes	Yes	Yes
Ballot secrecy	No	Yes	Yes
Coercion resistance	No	Yes	Yes
Eligibility privacy	No	No	Yes

Table 3 compares the efficiency and scalability characteristics of the three approaches. Explicit on-chain voter whitelisting introduces linear administrative overhead in the baseline and commit–reveal systems. In contrast, the Merkle tree–based approach replaces per-voter authorization with a single on-chain root, achieving constant storage complexity and significantly reducing administrative transactions. This makes the final system suitable for large-scale elections.

Conclusion

This paper presented a secure and scalable blockchain-based electronic voting system implemented using Ethereum smart contracts. A baseline model ensured vote integrity and immutability but revealed privacy limitations. To address this, a commit–reveal protocol was introduced to preserve ballot secrecy and prevent vote manipulation. Scalability was enhanced through a Merkle tree–based voter authorization mechanism, reducing on-chain storage and administrative overhead. The system was deployed and tested on a private Ethereum network using Ganache and MetaMask, confirming enforcement of one-vote-per-voter and immutable result recording. Overall, the proposed design demonstrates a practical and secure framework for blockchain-based electronic voting systems.

Acknowledgements

I would like to express sincere gratitude to St. Francis Institute of Technology for providing the institutional support and academic environment necessary to carry out this research. Special thanks are extended to Mr. Rajkumar Shende for their invaluable guidance, constructive feedback, and continuous encouragement throughout the development of this project. Their mentorship and support were instrumental in the successful completion of this work.

References

[1]. A. Canidio, V. Danos, Commitment Against Front-Running Attacks, arXiv preprint arXiv:2301.13785, 2023. https://arxiv.org/abs/2301.13785

[2]. A. Canidio, V. Danos, Commit-Reveal Schemes Against Front-Running Attacks, Proceedings of the 4th International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2022), 2023.https://doi.org/10.4230/OASIcs.Tokenomics.2022.7

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 985-995
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0140

[3]. S. Lee, E. Gee, N. Soroush, M.A. Bingol, K. Huang, Commit-Reveal²: Securing Randomness Beacons with Randomized Reveal Order in Smart Contracts, arXiv preprint, 2025.
[4]. M. Kelkar, S. Deb, S. Kannan, Order-Fair Consensus in the Permissionless Setting, Proceedings of the ACM Conference on Computer and Communications Security, 2021.
[5]. O. Kuznetsov, A. Rusnak, A. Yezhov, K. Kuznetsova, D. Kanonik, O. Domin, Merkle Trees in Blockchain: A Study of Collision Probability and Security Implications, Blockchain: Research and Applications, 2024.
[6]. Y. Deng, M. Yan, B. Tang, Accelerating Merkle Patricia Trie with GPU, Proceedings of the VLDB Endowment 17 (2024) 1856–1869.https://doi.org/10.14778/3659437.3659443
[7]. I. Zhang, R. Zarick, D. Wong, T. Kim, B. Pellegrino, M. Li, K. Wong, QMDB: Quick Merkle Database, LayerZero Labs Technical Report, 2024.
[8]. J. Oberst, Towards Stateless Clients in Ethereum: Benchmarking Verkle Trees and Binary Merkle Trees with SNARKs, arXiv preprint, 2024.
[9]. B. He, Y. Chen, Z. Chen, X. Hu, Y. Hu, L. Wu, R. Chang, H. Wang, Y. Zhou, TxPhishScope: Towards Detecting and Understanding Transaction-Based Phishing on Ethereum, Proceedings of the ACM Conference on Computer and Communications Security (CCS), 2023. https://doi.org/10.1145/3576915.3623210
[10]. S. Guan, K. Li, Characterizing Ethereum Address Poisoning Attacks, Proceedings of the ACM Conference on Computer and Communications Security (CCS), 2024. https://doi.org/10.1145/3658644.3690277
[11]. Y. Yu, T. Sharma, S. Das, Y. Wang, How Cryptocurrency Users Choose and Secure Their Wallets, Proceedings of the CHI Conference on Human Factors in Computing Systems, 2024. https:/ /doi.org/ 10.1145/3613904.3642534
[12]. K. Yan, X. Zhang, W. Diao, Stealing Trust: Unraveling Blind Message Attacks in Web3 Authentication, Proceedings of the ACM Conference on Computer and Communications Security (CCS), 2024. https://doi.org/10.1145/3658644.3670323
[13]. G. Mathur, GANACHE: A Robust Framework for Efficient and Secure Storage of Data on Private Ethereum Blockchains, Research Square, 2023. https:// doi.org/ 10.21203/rs.3.rs-3495549/v1
[14]. A.D. Lopes, T. Mello, W.R. Bezerra, Digital Identity Management System with Blockchain: An Implementation with Ethereum and Ganache, arXiv preprint arXiv:2507.21398, 2025.
[15]. M. Hajian Berenjestanaki, H.R. Barzegar, N. El Ioini, C. Pahl, Blockchain-Based E-Voting Systems: A Technology Review, Electronics 13 (2024) 17. https:// doi.org/ 10.3390/ electronics13010017