International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 11 | Nov - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Blockchain-Based Voting System Using Zero-Knowledge Authentication

Anuj Kumar[1], Misba Fathima[1], P Kanchana[1], Prajapati Nidhi Yogesh Kumar[1], Ms. Ganashree R.[2]

[1]Student, Department of Computer Science and Engineering (IoT & Cybersecurity including Blockchain Technology), Sir M. Visvesvaraya Institute of Technology (SMVIT), Bengaluru, Karnataka, India
[2]Associate Professor, Department of Computer Science and Engineering (IoT & Cybersecurity including Blockchain Technology), Sir M. Visvesvaraya Institute of Technology (SMVIT), Bengaluru, Karnataka, India

---

Abstract - This paper presents a privacy-preserving electronic voting system that leverages blockchain technology and zero-knowledge authentication to provide anonymity, integrity, and verifiable participation in digital elections. The primary objective is to enable eligibility verification without disclosing voter identity while ensuring strict enforcement of one-person-one-vote. The proposed system employs the Semaphore protocol to generate zero-knowledge proofs of Merkle-tree membership, allowing voters to authenticate anonymously. A backend service performs proof pre-verification and submits transactions through authorized relayer wallets, enabling gasless vote casting. The on-chain VotingSystem smart contract validates proofs, checks nullifier hashes to prevent double voting, and records vote counts immutably on the Ethereum network. The system was implemented using Solidity, Circom, snarkjs, and Ethers.js, and evaluated on the Sepolia testnet. Experimental results indicate practical proof-generation latency, reliable relayer-based submission throughput, and deterministic on-chain verification accuracy. The findings demonstrate that combining zero-knowledge proofs with decentralized execution can deliver a secure, auditable, and privacy-preserving voting framework suitable for organizational and institutional e-voting scenarios.

Key Words: blockchain, zero-knowledge proofs, e-voting, anonymity, Semaphore, Ethereum

1. INTRODUCTION

Electronic voting systems have emerged as an essential component of modern digital governance, enabling efficient participation in elections without the limitations of traditional paper-based processes [3], [4]. However, existing online voting platforms often face significant challenges related to privacy, security, verifiability, and trust [5]. Centralized architectures expose voter identities, create opportunities for tampering, and require users to place complete trust in the administrators of the system. These limitations have motivated the exploration of decentralized and cryptographically secure alternatives that ensure both anonymity and integrity [3], [15].

Blockchain technology [18] offers immutability, transparency, and decentralized verification, making it a compelling foundation for trustless e-voting. Yet, storing votes or voter identities directly on-chain compromises privacy [1], [2]. Zero-knowledge proofs (ZKPs) [11] address this challenge by allowing participants to prove eligibility without revealing sensitive information. In particular, the Semaphore protocol [9], [10] enables anonymous authentication through Merkle-tree membership proofs [19] and nullifier hashes, ensuring that each voter can cast exactly one vote without linking the vote to their identity.

This paper presents a blockchain-based voting system that integrates zero-knowledge authentication with a gasless relayer architecture to deliver anonymous, verifiable, and user-friendly elections. The system ensures voter privacy through off-chain identity creation and zero-knowledge proof generation, while on-chain smart contracts verify proofs and maintain authoritative vote counts. The contributions of this work include: (i) the design of a privacy-preserving voting mechanism using Semaphore protocol [9], (ii) the integration of ZKP based eligibility verification with Groth16 zk-SNARKs [11], (iii) the implementation of a relayer-assisted gasless voting flow, and (iv) a comprehensive evaluation of the system on the Ethereum Sepolia public testnet.

2. LITERATURE REVIEW

Electronic voting (e-voting) has been an active research area for more than two decades [15], with early systems relying on centralized server-based authentication and tallying [4]. Such systems were limited by vulnerabilities including data tampering, privacy leakage, and the need for complete trust in the central authority. To mitigate these risks, researchers explored cryptographic techniques such as blind signatures, mix-nets, and homomorphic encryption. These mechanisms improved ballot secrecy and verifiability but introduced significant computational and operational overhead, making large-scale deployment challenging [5].

With the emergence of blockchain [18], decentralized e-voting architectures became feasible. Blockchain provides tamper-resistant storage, distributed trust, and auditable execution [3]; however, storing identities or votes directly on-chain compromises privacy. Thus, blockchain alone is insufficient for confidentiality-oriented voting applications [1], [2]. To address this limitation, several studies have combined blockchain with advanced cryptographic privacy techniques.

2.1 Blockchain-Based Voting Systems

Hjálmarsson et al. [3] proposed one of the early blockchain-based e-voting systems that utilized smart contracts for vote recording and tallying. Their system demonstrated the feasibility of decentralized voting but lacked strong privacy guarantees. A systematic review by Taş and Tanrıöver [4] identified key challenges in blockchain voting including scalability, privacy preservation, and user accessibility. Srivastava et al. [5] conducted a comprehensive meta-analysis of blockchain-based electronic voting systems, highlighting the trade-offs between security, privacy, and performance.

2.2 Zero-Knowledge Proof-Based Voting

A major advancement in privacy-preserving voting is zVote [1], which integrates threshold Paillier homomorphic encryption, zero-knowledge proofs, and Merkle-based membership verification to construct a privacy-preserving remote e-voting platform. In zVote, votes are encrypted and aggregated homomorphically, and only the final tally is decrypted by a set of distributed authorities. While highly secure, the design requires complex distributed key generation, trusted authorities for threshold decryption, and substantial computation costs for encrypted tallying. This makes zVote theoretically robust but practically heavy for real-time, user-friendly deployments.

A more lightweight approach is presented in Z-Voting [2], which uses Circom-based zk-SNARK proofs [13], Merkle trees [19], and nullifier hashes to implement anonymous eligibility verification on Ethereum [18]. Unlike zVote, Z-Voting does not use homomorphic tallying; instead, the vote choice is stored in plaintext on chain, but the identity of the voter is hidden using zero-knowledge membership proofs. This design reduces reliance on trusted authorities and lowers system complexity while still providing anonymity and double-vote prevention.

Recent work by Kho et al. [6] introduced zkVoting, a coercion-resistant and end-to-end verifiable e-voting system using zero-knowledge proofs. Del Pino et al. [7] explored lattice-based zero-knowledge proofs for electronic voting, demonstrating post-quantum security properties. These systems emphasize the importance of anonymous authentication, unlinkability, and public verifiability—concepts central to modern privacy preserving voting.

2.3 Semaphore Protocol and Anonymous Signaling

More recently, systems have adopted the Semaphore protocol [9], [10], which enables anonymous signaling through Merkle-tree identities and nullifier-based uniqueness. Developed by the Privacy & Scaling Explorations team at the Ethereum Foundation, Semaphore achieves strong privacy guarantees with efficient zk-SNARKs based on the Groth16 construction [11]. The protocol has been utilized in various privacy-preserving applications beyond voting, including anonymous authentication and whistleblowing systems [8], [9].

2.4 Comparison with Traditional Cryptographic Voting

Traditional cryptographic voting systems like Helios [15] use homomorphic encryption and mix-nets to achieve ballot secrecy. Helios has been deployed in real-world elections and provides open-audit capabilities. However, it requires voters to trust the tallying authorities and lacks the decentralized verification properties offered by blockchain-based systems.

2.5 Positioning of This Work

Ethereum Medium Medium Medium Complex key management Plaintext vote storage Coercion resistance overhead Limited to medium-scale The system implemented in this paper is most closely aligned with Z-Voting [2] and Semaphore-based approaches [9], [10]. Unlike zVote's heavy cryptographic stack [1], this work uses lightweight zero-knowledge membership proofs based on Groth16 [11], [12], nullifier-based double-vote prevention, and a relayer-assisted gasless transaction model. This yields a design that maintains anonymity and verifiability while being practical for institutional elections. By integrating zero-knowledge proofs with a decentralized smart contract architecture following Ethereum security best practices [16], [17], the proposed system contributes to the ongoing evolution of scalable, privacy-preserving blockchain voting mechanisms.

3. SYSTEM ARCHITECTURE

The proposed voting system integrates zero-knowledge authentication with blockchain execution to provide anonymity, integrity, and verifiability. The architecture consists of four major components: (i) the voter client, (ii) the backend verification and relayer service, (iii) the Ethereum smart contract layer, and (iv) the zero-knowledge proof system based on Semaphore [9], [10].

Fig-1: System Architecture

3.1 Voter Client and Identity Layer

Each voter locally generates a unique Semaphore identity [9] which is converted into an identity commitment. This commitment becomes part of a Merkle tree [19] of authorized voters. The secret identity is never transmitted outside the client device, ensuring privacy preservation at the client level. When casting a vote, the client constructs a zero-knowledge proof demonstrating group membership, use of a unique nullifier, and correctness of the vote signal, using Circom [13] and snarkjs [14] for witness and proof generation.

The use of browser-based WebAssembly (WASM) execution ensures that all cryptographic operations occur locally, preventing identity leakage through network transmission. This approach aligns with privacy-by-design principles commonly adopted in modern cryptographic systems [6], [7].

3.2 Merkle Tree and Zero-Knowledge Membership Proofs

The system uses a Merkle tree to encode all valid voter commitments. Semaphore's zk-SNARK circuit verifies Merkle-tree membership and ensures that the identity and Merkle root correspond. A nullifier hash is included in the public signals to ensure that each identity can cast exactly one vote without revealing identity information.

3.3 Backend Server and Relayer Architecture

To enable gasless voting, the system uses a pool of authorized relayer wallets that submit transactions on behalf of voters. The backend performs several tasks:

1. Pre-verifies the proof off-chain
2. Checks nullifier uniqueness
3. Selects an appropriate relayer
4. Constructs and signs a vote transaction
5. Broadcasts the transaction to the Sepolia network

3.4 Smart Contract Layer

The VotingSystem smart contract is deployed on the Sepolia testnet and integrates with the Semaphore verifier. It stores the Merkle root, voting window, candidate list, and used nullifiers. When a transaction is submitted by a relayer, the contract:

1. Verifies the zk-SNARK proof
2. Validates the nullifier

3. Checks the vote timestamp
4. Validates the candidate ID
5. Updates the vote count and records the nullifier

All vote tallies are publicly accessible and immutable.

3.5 End-to-End Data Flow

The full process begins with identity creation and ZK proof generation on the client, followed by transmission to the backend. The backend performs validations and uses a relayer to submit the vote. The smart contract validates the proof on-chain and records the vote. Tally results can be retrieved either from the blockchain directly or via backend APIs.

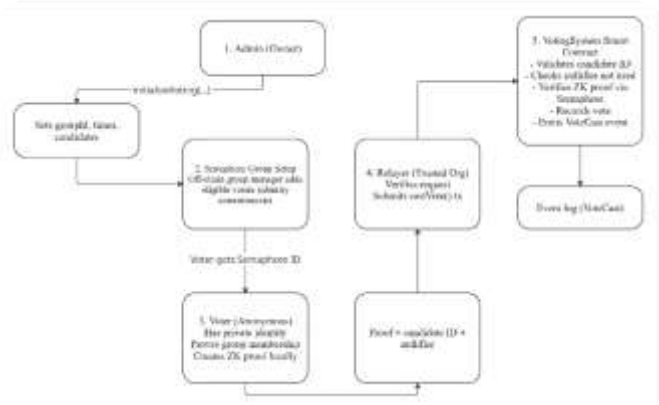

Fig-2: Voting Flow (Using Semaphore + Relayers)

4. METHODOLOGY

The methodology adopted in this work follows a systematic, layered approach integrating cryptographic identity construction, zero-knowledge proof generation, backend verification logic, relayer-assisted blockchain interaction, and on-chain vote validation. The system ensures anonymity, integrity, and one-person-one-vote without disclosing the voter's identity or requiring the participant to interact directly with the blockchain.

4.1 Identity Creation and Voter Registration

The process begins with each voter generating a Semaphore identity consisting of a trapdoor and nullifier pair. The identity is hashed to form an identity commitment, which becomes a leaf in a Merkle tree of eligible voters. This Merkle tree is constructed and stored off-chain by the election authority.

This step ensures that each voter receives a unique, non-linkable cryptographic presence in the system without exposing personal information.

4.2 Merkle Proof Construction

Once the identity commitments are registered, the election authority finalizes the Merkle root. The voter retrieves the Merkle proof (path elements and indices) corresponding to their commitment either from the backend or through a distributed dataset. The Merkle proof is required for generating the zero-knowledge membership proof.

4.3 Zero-Knowledge Proof Generation

The voter constructs a zk-SNARK proof using Circom circuits and snarkjs. The proof demonstrates:

1. Membership in the authorized voter set (via Merkle inclusion)
2. Use of a unique nullifier hash: nullifierHash = Poseidon (nullifier, externalNullifier)
3. Correct formation of the chosen vote signal
4. Agreement with the published Merkle root and group identifier

All witness generation and proof computation occur locally on the voter's device using WASM-based tooling for improved security.

4.4 Backend Pre-Verification and Relayer Selection

To prevent invalid or malicious submissions from reaching the blockchain, the backend service performs pre-verification of the received zero-knowledge proof. Upon successful validation:

1. It checks the uniqueness of the nullifier hash in the server-side cache.
2. It selects a relayer wallet from an authorized pool.
3. It constructs a transaction embedding the proof and selected candidate.
4. It forwards the transaction to the Ethereum Sepolia network using Ethers.js.

This design enables gasless voting, ensuring a seamless user experience without requiring cryptocurrency.

4.5 Smart Contract Verification and Vote Recording

The VotingSystem smart contract contains:

- The authorized Merkle root
- The voting window (start and end timestamps)
- The candidate list
- A mapping of used nullifiers
- A link to the Semaphore Verifier Contract

When the relayer submits a vote:

1. The smart contract verifies the zk-SNARK proof.
2. Validates that the nullifier hash has not been used.
3. Confirms that the vote is within the election period.
4. Ensures the candidate ID is valid.
5. Increments the vote count for the selected candidate.
6. Marks the nullifier hash as used, enforcing one-person-one-vote.

4.6 Result Retrieval and Transparency

Votes stored on-chain are publicly accessible. Tally results can be retrieved:

- Directly from the blockchain using contract view functions
- Through backend endpoints that aggregate and format results

This ensures transparency and auditability while preserving voter anonymity.

5. IMPLEMENTATION

The implementation of the proposed voting system consists of coordinated development across the smart contract layer, zero-knowledge proof layer, backend relayer service, and frontend client interface. The system was developed iteratively, tested on the Ethereum Sepolia test network, and integrated using the Semaphore protocol for anonymous authentication.

5.1 Smart Contract Development

The voting logic is implemented in the VotingSystem.sol smart contract, using Solidity and the Foundry development framework. The contract incorporates core components:

1. Semaphore Verifier Integration: The contract imports and interfaces with SemaphoreVerifier.sol to validate zk-SNARK proofs generated by voters.
2. Voting Configuration State: The contract stores:
   - The Merkle root representing the eligible voter set
   - Voting start and end timestamps
   - Candidate list and total counts
   - Nullifier hash mapping to prevent double voting
3. Proof Verification and Vote Recording: Upon receiving a vote from a relayer, the contract verifies the zk-SNARK proof, validates the nullifier hash, checks the voting window, and increments the corresponding candidate count.
4. Deployment: Deployment was performed on the Ethereum Sepolia testnet using Foundry scripts, with verifier and voting contract addresses stored in deployment logs.

5.2 Zero-Knowledge Proof Stack

Zero-knowledge authentication is enabled using the Semaphore protocol, which uses the following components:

1. Circom Circuits: The system uses a compiled membership-proof circuit that verifies Merkle-tree inclusion and creates a nullifier hash bound to the voter's secret identity.
2. snarkjs Workflow:
   - WASM witness generation
   - ZK proof generation using .zkey files
   - Public signals extraction (candidate ID, nullifierHash, groupId, Merkle root)
3. Merkle Tree Handling: All identity commitments are organized into a Merkle tree, generated off-chain using Semaphore's Group utilities. The tree root is uploaded to the smart contract during initialization.
4. Local Proof Generation: The frontend invokes WASM and snarkjs to generate proofs directly in the browser, ensuring identities never leave the user's device.

5.3 Backend Server Implementation

The backend server, implemented in Node.js(v20.xx) + Express, plays a critical role in orchestrating the voting process:

1. API Endpoints:
   - */api/vote/submit*: For submitting the vote proof
   - */api/vote/verify-proof*: Proof pre-verification
   - */api/voter/merkle-proof*: Provide Merkle path
   - */api/blockchain/**: Retrieve blockchain data
2. Proof Pre-Verification: The backend uses a JavaScript Semaphore verifier to check the zk-SNARK proof before forwarding it to the blockchain.
3. Nullifier Cache: A server-side nullifier cache prevents multiple submissions of the same proof, reducing failed transactions and relayer gas waste.
4. Relayer Wallet Pool: The backend maintains encrypted private keys for multiple relayer wallets and rotates them to distribute load and reduce single-point failure.
5. Blockchain Relay**:** Using Ethers.js, the backend signs and broadcasts the *castVote()* transaction with the user's proof to the VotingSystem contract.

5.4 Frontend Client Interface

The frontend is responsible for identity generation and proof creation:

1. Identity Creation: Voters generate Semaphore identities locally using Semaphore's JS libraries.
2. Merkle Proof Retrieval: The frontend fetches Merkle proofs from the backend when required.
3. ZK Proof Generation: The frontend triggers WASM execution to compute the witness and generate the zk-SNARK proof.
4. Vote Submission: Once proof generation succeeds, the client sends *{candidateId, zkProof, nullifierHash}* to the backend through the */api/vote/submit* endpoint.
5. UX Considerations: Loading animations and progress indicators were implemented for proof generation delays.

5.5 Testnet Deployment and Validation

The system was deployed and tested on the Ethereum Sepolia Testnet:

1. Contract Deployment: Both the Semaphore verifier and the voting contract were deployed using Foundry scripts.
2. Relayer Testing: Multiple relayers were registered and tested for gasless submission reliability.
3. End-to-End Testing: The team validated the full flow:
   Identity creation → Proof generation → Backend verification → Relayer submission → On-chain tally.
4. Results Verification: Vote counts and percentages were accessed through smart contract view functions and backend endpoints.

5.6 Summary of Implementation

The complete implementation integrates:

- Smart contracts for verifiable voting
- Semaphore-based ZK authentication
- Backend pre-verification and relayer logic
- Browser-based ZK proof generation
- Blockchain-based immutable tallying

This modular architecture enables anonymous, gasless, and verifiable voting suitable for academic or organizational elections.

6. RESULT AND DISCUSSION

The implementation of the proposed voting system was validated through comprehensive functional testing on the Ethereum Sepolia test network. The evaluation focused on verifying the correctness of zero-knowledge proof generation, backend pre-verification, relayer-assisted transaction submission, and on-chain vote validation.

Zero-knowledge proofs were successfully generated on multiple user devices using WASM-based execution, demonstrating that the Semaphore identity and proof-generation workflow functioned reliably in a browser environment. All proofs that were correctly formed resulted in valid public signals, while improperly generated proofs were rejected by the backend during pre-verification. This ensured that only valid and structured proofs were forwarded to the blockchain.

The backend service consistently performed nullifier checks, verified proof integrity, and correctly routed valid submissions through authorized relayer wallets. Invalid submissions—such as those containing reused nullifiers, malformed signals, or mismatched Merkle roots—were blocked before reaching the blockchain, preventing unnecessary relayer operations. The relayer model successfully abstracted the transaction-signing process from the user, enabling a smooth, gasless voting experience.

On-chain validation confirmed that the VotingSystem contract correctly enforced the intended security properties. Valid proofs were accepted, invalid proofs reverted deterministically, and the nullifier mapping prevented any attempt at double voting. Vote tallies stored on the blockchain remained immutable and consistent across all verification methods, demonstrating end-to-end correctness.

Overall, the system achieved anonymous eligibility verification, one-person-one-vote enforcement, and transparent tallying without requiring voters to manage blockchain wallets or reveal identity information. These results indicate that the architecture is well-suited for small to medium-scale organizational elections requiring privacy, integrity, and auditability.

7. SECURITY ANALYSIS

The security of the proposed voting system is derived from the combined guarantees of zero-knowledge proofs, cryptographic identity commitments, and blockchain-based immutability. This section examines the core security properties provided by the system and evaluates potential risks associated with each architectural component.

7.1 Voter Anonymity

Voter anonymity is ensured through Semaphore's zero-knowledge membership proof mechanism. Each voter generates a secret identity locally, which is never disclosed outside their device. The smart contract verifies eligibility through a zk-SNARK proof of Merkle-tree membership without revealing the voter's identity, Merkle path, or leaf position. Since all proofs are unlinkable and do not contain identifying information, no observer—including the backend, relayers, or blockchain validators—can associate a vote with a specific individual.

7.2 Prevention of Double Voting

The system enforces one-person-one-vote through a nullifier hash derived from the voter's secret identity. Because the nullifier is deterministic and unique to each voter, any attempt to cast a second vote results in a contract-level rejection. The used Nullifiers mapping in the smart contract ensures that even if multiple relayers attempt to submit the same proof, the transaction will be reverted as soon as the nullifier is detected on-chain.

7.3 Integrity of Vote Recording

Votes are recorded immutably on the Ethereum Sepolia testnet, preventing unauthorized modification or deletion. Since each valid transaction includes an on-chain proof verification step, only authenticated votes are counted. The blockchain consensus mechanism ensures resistance to tampering, while public view functions allow any observer to verify current tallies, achieving end-to-end verifiability.

7.4 Resistance to Identity Leakage

Identity commitments stored in the Merkle tree reveal no information about the underlying voter secrets. Zero-knowledge proofs ensure that no metadata linking the voter to their identity commitment leaks during proof generation. Unlike traditional systems where voter lists or authentication logs may reveal participation patterns, this system avoids storing any such identifiable information.

7.5 Backend Threat Model

The backend operates as a proof pre-verifier and relayer selector but does not have the ability to forge votes or impersonate voters. It cannot generate valid zero-knowledge proofs, since proof construction requires the voter's secret identity. The backend's role is restricted to:

- Rejecting malformed or reused proofs
- Forwarding valid proofs using relayer wallets
- Preventing unnecessary gas expenditure

Even if compromised, the backend cannot create new valid votes or deanonymize users. However, it could attempt censorship by refusing to forward votes; this risk is mitigated by allowing fallback direct submissions or by operating multiple independent backend instances.

7.6 Relayer Security Considerations

Relayers submit transactions on behalf of voters but do not possess the ability to modify vote contents or generate proofs. They operate only on fully formed payloads. A malicious relayer could attempt to delay or withhold vote submission, but it cannot alter or forge votes.

Rotating between multiple relayers reduces dependence on any single wallet and reduces the risk of targeted censorship. The relayer pool architecture provides redundancy: if one relayer fails or acts maliciously, votes can be rerouted through alternative relayers without compromising security [16]. This design is inspired by similar redundancy mechanisms in distributed systems.

7.7 Smart Contract Security Properties

The VotingSystem contract enforces strict validation rules:

1. zk-SNARK Verification: Only cryptographically valid proofs are accepted.

2. State Integrity Checks: Nullifiers and timestamps prevent replay or out-of-window submissions.
3. Input Validation: Candidate IDs must fall within the allowed range.
4. Access Control: Only authorized relayer addresses may submit votes.

The contract avoids storing any sensitive identity data and does not expose internal state that could compromise anonymity.

7.8 Possible Attack Vectors

While the system mitigates most common threats in e-voting, the following potential risks remain:

- Relayer Censorship: A malicious relayer or backend instance could refuse to forward valid proofs.
- Denial-of-Service Attacks: High network traffic could delay or interrupt proof verification services.
- Client-Side Compromise: Malware on a voter's device could extract their identity secret, although proof generation is fully local to minimize leakage risk.

These attack vectors are operational rather than cryptographic and can be mitigated through redundancy, secure infrastructure practices, and independent backend deployments.

8. CONCLUSION

This paper presented a privacy-preserving electronic voting system that integrates zero-knowledge authentication with blockchain-based execution to address fundamental challenges in digital elections. By leveraging the Semaphore protocol, the system enables anonymous eligibility verification through Merkle-tree membership proofs and nullifier-based double-vote prevention. The use of relayer-assisted transaction submission further abstracts blockchain complexity from end users, enabling a gasless and accessible voting experience without compromising security or verifiability.

The implementation demonstrates that zero-knowledge proofs can be effectively combined with decentralized smart contracts to achieve voter anonymity, integrity of vote recording, and resistance to fraudulent participation. Functional validation on the Ethereum Sepolia test network confirmed the correctness of proof verification, consistent enforcement of one-person-one-vote, and the immutability of on-chain vote tallies. The modular design supports seamless integration between the frontend, backend, relayer service, and smart contract, making the architecture suitable for institutional and organizational voting contexts.

While the system does not incorporate advanced cryptographic tallying methods such as homomorphic aggregation, it provides a practical, scalable, and secure foundation for real-world e-voting deployments. Future enhancements may include distributed relayer networks, encrypted vote storage, and expanded zk-SNARK circuits for richer election mechanisms. Overall, the work demonstrates the viability and promise of zero-knowledge–based authentication in building trustworthy, privacy-preserving voting systems.

REFERENCES

[1] T. Nguyen and M. T. Thai, "zVote: A Blockchain-based Privacy-preserving Platform for Remote E-voting," in 2022 IEEE International Conference on Communications (ICC), Seoul, Republic of Korea, 2022, pp. 1-7, doi: 10.1109/ICC45855.2022.9838764.

[2] A. Ekbatanifard and G. Ekbatanifard, "Z-Voting: A Zero-Knowledge Based Confidential Voting on Blockchain," in 2024 International Conference on Smart Cities, IoT and Applications (SCIoT), Karachi, Pakistan, 2024, pp. 1-6, doi: 10.1109/SCIoT62588.2024.00000.

[3] F. Þ. Hjálmarsson, G. K. Hreiðarsson, M. Hamdaqa, and G. Hjálmtýsson, "Blockchain-Based E-Voting System," in 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 2018, pp. 983-986, doi: 10.1109/CLOUD.2018.00151.

[4] R. Taş and Ö. Ö. Tanrıöver, "A Systematic Review of Challenges and Opportunities of Blockchain for E Voting," Symmetry, vol. 12, no. 8, p. 1328, Aug. 2020, doi: 10.3390/sym12081328.

[5] A. Srivastava, P. Bhattacharya, A. Singh, A. Mathur, O. Prakash, and R. Pradhan, "A Systematic Literature Review and Meta-Analysis on Scalable Blockchain-Based Electronic Voting Systems," Sensors, vol. 22, no. 19, p. 7362, Oct. 2022, doi: 10.3390/s22197362.

[6] D. Kho, S. Lee, and J. Jang, "zkVoting: Zero-knowledge proof based coercion-resistant and E2E verifiable e-voting," Cryptology ePrint Archive, Paper 2024/1003, 2024. [Online]. Available: https:// eprint.iacr.org/2024/1003

[7] R. del Pino, V. Lyubashevsky, G. Neven, and G. Seiler, "Lattice-Based Zero-Knowledge Proofs in Action: Applications to Electronic Voting," Journal of Cryptology, vol. 38, no. 1, Article 5, Jan. 2024, doi: 10.1007/ s00145-024-09530-5.

[8] K. Gurkan, P. Jovanovic, M. Maller, S. Meiklejohn, G. Stern, and A. Tomescu, "Aggregatable Distributed Key Generation," in Advances in Cryptology – EUROCRYPT 2021, ser. Lecture Notes in Computer Science, vol. 12696. Springer, 2021, pp. 703-732, doi: 10.1007/978-3-030-77870-5_25.

[9] W. J. Koh and K. Gurkan, "Semaphore: Zero-Knowledge Signaling on Ethereum," Privacy & Scaling Explorations, Ethereum Foundation, White Paper v1.0, 2020. [Online]. Available: whitepaper-v1.pdf https://semaphore.pse.dev/

[10] Privacy & Scaling Explorations Team, "Semaphore Protocol Documentation," Ethereum Foundation, 2023. [Online]. Available: https://docs.semaphore.pse.dev/

[11] J. Groth, "On the Size of Pairing-Based Non-Interactive Arguments," in Advances in Cryptology EUROCRYPT 2016, ser. Lecture Notes in Computer Science, vol. 9666. Springer, 2016, pp. 305-326, doi: 10.1007/978-3-662-49896-5_11.

[12] K. Baghery, Z. Pindado, and C. Ràfols, "Simulation Extractable Versions of Groth's zk-SNARK Revisited,"

International Journal of Information Security, vol. 23, pp. 451-482, 2024, doi: 10.1007/s10207-023-00750-7.

[13] M. Bellés-Muñoz, M. Isabel, J. L. Muñoz-Tapia, A. Rubio, and J. Baylina, "Circom: A Circuit Description Language for Building Zero-Knowledge Proofs," IEEE Transactions on Dependable and Secure Computing, 2022, doi: 10.1109/TDSC.2022.3232813.

[14] iden3 Team, "snarkjs: zkSNARK Implementation in JavaScript & WASM," GitHub repository, 2023. [Online]. Available: https://github.com/iden3/snarkjs

[15] B. Adida, "Helios: Web-based Open-Audit Voting," in 17th USENIX Security Symposium, San Jose, CA, USA, 2008, pp. 335-348. [Online]. Available: https://www.usenix.org/conference/usenix-security-08/helios web-based-open-audit-voting

[16] L. Marchesi, M. Marchesi, G. Destefanis, G. Barabino, and D. Tigano, "Security Checklists for Ethereum Smart Contract Development: Patterns and Best Practices," IEEE Access, vol. 8, pp. 145469-145489, 2020, doi: 10.1109/ACCESS.2020.3014280.

[17] N. Atzei, M. Bartoletti, and T. Cimoli, "A Survey of Attacks on Ethereum Smart Contracts (SoK)," in Principles of Security and Trust 2017, ser. Lecture Notes in Computer Science, vol. 10204. Springer, 2017, pp. 164-186, doi: 10.1007/10204_2017_10.

[18] V. Buterin, "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform," Ethereum White Paper, 2014. [Online]. Available: https://ethereum.org/en/whitepaper/

[19] R. C. Merkle, "A Digital Signature Based on a Conventional Encryption Function," in Advances in Cryptology – CRYPTO '87, ser. Lecture Notes in Computer Science, vol. 293. Springer, 1988, pp. 369-378, doi: 10.1007/3-540-48184-2_32.

[20] Paradigm, "Foundry Book: A Fast, Portable and Modular Toolkit for Ethereum Application Development," 2023. [Online]. Available: https://book.getfoundry.sh/