RESEARCH AND APPLICATION OF BLOCKCHAIN TECHNOLOGY TO ENHANCE TRANSPARENCY AND SECURITY IN ELECTRONIC VOTING SYSTEMS

Ngo Duc Canh*, Chu Thi Quyen, Le Thi Linh, Roan Van Quyen, Phung Thi Thuy

Ha Noi University of Industry - School of Information and Communication Technology, Vietnam

*Corresponding author: 2022605692@st.haui.edu.vn

(Received: February 02, 2026; Revised: March 12, 2026; Accepted: March 15, 2026)

DOI: 10.31130/ud-jst.2026.24(3).079E

Abstract - In the context of Vietnam's ongoing digital transformation in public governance, the development of electronic voting systems that ensure transparency, fairness, and independent verifiability has become a critical requirement. However, many existing solutions face difficulties in balancing voter anonymity with the prevention of electoral fraud. This paper proposes a decentralized, system-level voting architecture built on the Ethereum platform, in which voting rules and security mechanisms are enforced immutably through smart contracts. The proposed system employs a Commit–Reveal scheme to mitigate front-running attacks and prevent the premature disclosure of voting results; utilizes Merkle proofs to verify voter eligibility without revealing personal identities; and integrates a nullifier mechanism combined with encrypted Citizen Identification Numbers (CINs) to enforce the principle of "one citizen, one vote." The results indicate that, the proposed model enhances transparency at the execution level and strengthens security at the architectural level for blockchain-based electronic voting systems.

Key words - Blockchain; electronic voting; smart contract; security; transparency; fairness

1. Introduction

In the digital era, electronic voting (e-voting) has emerged as an inevitable trend aimed at optimizing operational costs and increasing voter participation rates [1]. However, traditional e-voting systems built on centralized client-server architectures face serious challenges, including cybersecurity vulnerabilities, risks arising from single points of failure (SPOF), and the lack of independent verification mechanisms, as vote-counting processes are often conducted within the "black box" of software [2].

The advent of blockchain technology, with its inherent properties of decentralization, immutability, and consensus mechanisms, has opened new pathways for establishing transparent and end-to-end verifiable voting systems [3]. In blockchain-based voting architectures, "Each vote is recorded as a transaction on the blockchain, and once added, it cannot be altered or deleted. This ensures the integrity of the voting process and provides a high level of trust in the results" [4]. Nevertheless, the transition from theoretical models to large-scale practical development remains fraught with obstacles. Representative studies since 2016 have revealed significant trade-offs: the model proposed by Lee et al. still relies on a trusted third party (TTP), while the solutions introduced by McCorry et al. and by Liu and Wang, although enhancing voter privacy, suffer from limitations in scalability and operational complexity [3], [5]. A comprehensive review of the literature indicates that many current models have yet to optimize smart contract execution costs and fail to adequately address performance constraints when deployed at a national scale [6]. More recent studies have explored zero-knowledge proof–based voting, rollup architectures, and decentralized identity frameworks to improve scalability and privacy; however, these approaches often introduce additional cryptographic or infrastructural complexity that may hinder practical deployment on public blockchains.

Based on the above analysis, a key research gap can be identified at the system architecture level. Existing blockchain-based e-voting studies either focus primarily on cryptographic protocol design or remain at a conceptual stage, while practical and deployable architectures that integrate voter eligibility verification, identity uniqueness, and auditability on public blockchains at a national scale remain insufficiently explored.

Motivated by this gap, this study investigates the integration of blockchain and smart contracts into electronic voting systems to evaluate how decentralized execution can enhance transparency, data integrity, and rule enforcement. Unlike prior work such as McCorry et al., which emphasizes end-to-end verifiability through cryptographic protocol construction, this paper adopts a system-oriented and applied perspective, focusing on architectural integration and practical feasibility on public blockchains rather than introducing new cryptographic primitives or formal security proofs [3]. Recent advances in electronic voting, including zk-based and Helios-like systems, offer strong privacy and verifiability guarantees but often face challenges related to computational overhead, centralized trust assumptions, and large-scale deployability. Recent implementation-oriented studies further emphasize that "by using privacy preserving solutions, decentralized protocols and smart contracts we can ensure transparency in vote counting and tamper-proof election results, thus ultimately mitigating the opportunity for voter fraud" [7]. In contrast, the proposed approach emphasizes a blockchain-native architecture that prioritizes transparency, auditability, and practical deployment under real-world governance constraints. Accordingly, this study analyzes how blockchain and smart contracts can be

systematically incorporated into electronic voting systems and proposes architectural design guidelines aimed at enforcing voting rules consistently, minimizing subjective intervention, and strengthening independent verifiability to contribute to fairer and more trustworthy electronic voting systems.The main contributions of this study are summarized as follows: (1) A system-level blockchain-native voting architecture integrating eligibility verification, nullifier-based uniqueness enforcement, and commit–reveal fairness control; (2) An architectural security model in which core election properties are enforced directly at the smart contract layer; (3) An analytical evaluation of scalability and gas-related constraints under public Ethereum deployment assumptions.

2. Blockchain and Smart Contracts in Voting System

2.1. Blockchain

The blockchain concept was the first introduced by Haber and Stornetta in 1991. The primary objective of designing a timestamp for digital documents to prevent tampering [8]. From a technical perspective, a blockchain is a distributed ledger in which transactions are grouped into blocks and sequentially linked through cryptographic hash functions, forming a data chain in which any modification can be readily detected. Two fundamental properties of blockchain are decentralization and immutability. Instead of being stored by a central authority, data are replicated and synchronized across multiple peer-to-peer nodes. A new block is accepted only after network consensus is achieved, ensuring that confirmed transactions cannot be altered or deleted.

Recording votes as blockchain transactions guarantees that each ballot is registered exactly once and cannot be modified after confirmation. The immutability of the distributed ledger enables permanent storage of the entire voting history, while the public nature of the blockchain facilitates independent auditing and verification of election results without reliance on a trusted intermediary.

2.2. Smart Contracts

Smart contracts are programs that are deployed and executed directly on the blockchain, in which application logic is encoded as immutable rules and automatically triggered by valid transactions. This approach eliminates reliance on trusted intermediaries while ensuring consistency and verifiability of the execution process.

Within electronic voting systems, smart contracts function as the central execution mechanism, ensuring that electoral rules are applied automatically and uniformly to all transactions. Due to the immutability of the blockchain, these rules cannot be modified after deployment, thereby guaranteeing transparency and fairness throughout the electoral process.

2.3. Recommendation for applying Smart Contract in Voting System

The electronic voting system is designed following a decentralized three layer architecture comprising the User Layer, the Application Layer, and the Blockchain & Storage Layer. This architecture combines smart contracts deployed on the Ethereum platform with the InterPlanetary File System (IPFS) to simultaneously ensure data integrity, transparency, and cost efficiency.

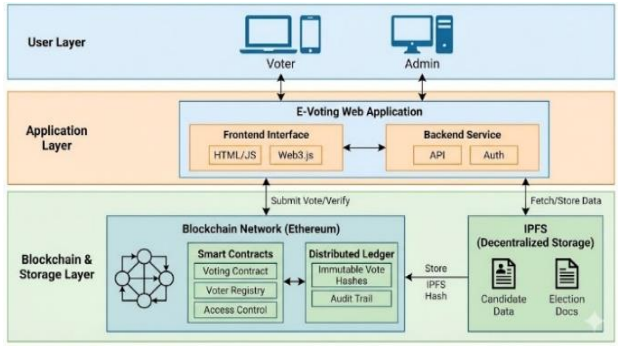

Figure 1. *Overall architecture of the electronic voting system integrating blockchain and IPFS*

In addition to security and anonymity mechanisms, transparency and auditability are achieved through the immutable recording of critical operations directly on the blockchain. Due to the tamper-resistant nature of the distributed ledger, the entire operational history can be independently accessed and verified. To mitigate the high cost of on-chain storage, a hybrid storage model is adopted, in which smart contracts store only core data and cryptographic hash values, while large-volume data are maintained off-chain on IPFS. This design approach ensures transparency while maintaining economic feasibility for practical deployment.

2.3.1. Threat Model and Security Requirements in Blockchain-Based Electronic Voting

In a public blockchain environment, all transactions are observable, exposing electronic voting systems to risks such as premature disclosure of ballot contents before election closure, front-running attacks, and double voting. Therefore, the proposed system is designed to simultaneously satisfy core security requirements, including ballot confidentiality, voter eligibility verification, enforcement of the “one voter, one vote” principle, protection of voter anonymity, and support for independent auditability. These requirements are enforced directly at the smart contract layer and serve as the foundation for the security mechanisms discussed in the subsequent sections.

In addition to classical security threats, blockchain-based voting systems also face challenges related to coercion resistance and vote selling. In a public and verifiable environment, preventing voters from being influenced or coerced by external parties remains a non-trivial problem. While full coercion resistance typically requires advanced cryptographic techniques or specialized voting devices, the proposed system focuses on mitigating coercion risks at the architectural level by avoiding the generation of transferable voting receipts and by enforcing receipt-free voting semantics. Methodologically, this study adopts a system-oriented analytical approach rather than a formal cryptographic proof model. The proposed voting system is evaluated through architectural decomposition and threat-

based analysis, in which security and transparency properties are examined at the smart contract execution level. This approach enables the assessment of practical enforceability, auditability, and scalability constraints in real-world blockchain environments, complementing existing formal cryptographic voting research.

2.3.2. Commit–Reveal Mechanism for Preserving Ballot Secrecy and Fairness

In a public blockchain environment, all transactions can be observed before being confirmed on-chain, including those residing in the mempool. If ballot contents are submitted in plaintext, voting information may be disclosed prematurely, leading to herd effects and enabling front-running attacks. To address this issue, the proposed system adopts a Commit-Reveal mechanism, which strictly separates the time of ballot commitment from the time of vote disclosure.

During the commitment phase, voters do not submit their candidate choices directly but instead transmit a commitment value computed off-chain. This value is generated using a one-way cryptographic hash function with strong security properties, ensuring that the original ballot content cannot be inferred. The commitment process is formally modeled as:

$$\mathbf{C} = \mathbf{H}\left(\mathbf{v} \,||\, \mathbf{r}\right) \tag{1}$$

Where, $\mathbf{C}$ denotes the commitment value, $\mathbf{H}(.)$ represents the Keccak-256 hash function, $\mathbf{v}$ is the voting choice, and $\mathbf{r}$ is a voter-generated secret random value. Each voter is allowed to generate only a single commitment per election, preventing the submission of multiple commitments intended to disrupt the voting process.

After the commitment phase concludes, the system enters the disclosure phase. In this stage, voters disclose their voting choice $\mathbf{v}$ together with the corresponding secret value $\mathbf{r}$. The smart contract verifies the validity of the vote by recomputing the hash and checking the following condition:

$$\mathbf{H}\left(\mathbf{v} \,||\, \mathbf{r}\right) = \mathbf{C} \tag{2}$$

The ballot is accepted and included in the election results only when this condition is satisfied. Each commitment may be revealed only once, thereby preventing any modification of the ballot after the commitment.

By clearly separating the commitment and disclosure phases, the Commit–Reveal mechanism eliminates the possibility of inferring intermediate voting outcomes during the election, preventing front-running attacks, and ensures the secrecy, integrity, and fairness of ballots in Blockchain-based electronic voting systems. It should be noted that while the Commit–Reveal mechanism effectively prevents front-running and premature result disclosure, it does not provide full protection against vote-selling or coercion in adversarial environments. However, by ensuring that no verifiable linkage between a voter and a revealed choice exists on-chain, the mechanism significantly reduces the practical feasibility of vote trading under standard threat assumptions.

2.3.3. Voter Eligibility Verification with Privacy Preservation

In electronic voting systems deployed on public blockchains, voter eligibility verification must simultaneously satisfy two inherently conflicting requirements: (i) ensuring that only eligible voters are allowed to participate in the election, and (ii) preventing the disclosure of voters' identities and the voter registry on the public ledger. Storing identification data or explicit voter lists directly on the blockchain not only exposes significant risks of personal data leakage but also contradicts the fundamental principles of privacy protection in decentralized systems.

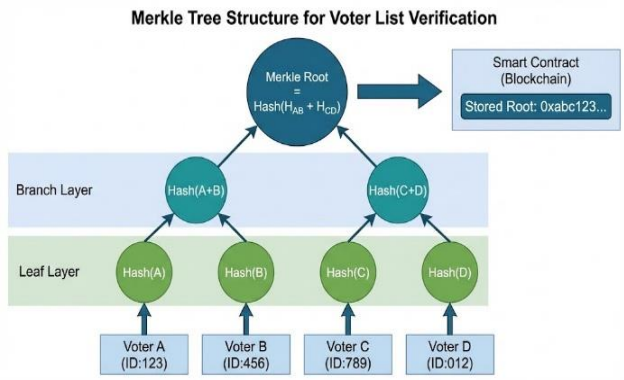

Figure 2. *Merkle tree structure for the voter registry*

To address this challenge, the proposed system employs a Merkle proof based mechanism as a cryptographic method for verifying voter eligibility without revealing the underlying dataset. Instead of storing the complete voter list on-chain, the smart contract maintains only a single representative value the Merkle root which cryptographically commits to the entire set of eligible voters verified off-chain.

The voter registry is managed off-chain and organized in the form of a Merkle tree, where each leaf node corresponds to the hash of a voter's normalized and preprocessed identification data. The Merkle tree is constructed using a recursive hashing process, in which each internal node is computed as:

$$\mathbf{H}_{\mathbf{parent}} = \mathbf{H}\left(\mathbf{H}_{\mathbf{left}} \,||\, \mathbf{H}_{\mathbf{right}}\right) \tag{3}$$

where $\mathbf{H}(.)$ denotes a one-way cryptographic hash function and $||$ represents concatenation. The resulting Merkle Root $\mathbf{M}_{\mathbf{root}}$ serves as a cryptographic commitment to the complete set of eligible voters and is immutably stored on the Blockchain.

During the voting process, voters do not submit any identifying information. Instead, they provide a Merkle Proof consisting of the set of intermediate hash values required to demonstrate that their corresponding leaf belongs to the Merkle tree rooted at $\mathbf{M}_{\mathbf{root}}$. The Smart Contract verifies the proof by reconstructing the hash path from the leaf to the root and checking the following condition:

$$\mathbf{Verify\,Proof}\left(\mathbf{leaf}, \mathbf{proof}, \mathbf{M}_{\mathbf{root}}\right) = \mathbf{true} \tag{4}$$

If the condition is satisfied, the voter is recognized as eligible and allowed to proceed with the voting process;

otherwise, the transaction is rejected without altering the system state. Throughout this process, the Smart Contract functions solely as a cryptographic verifier and has no ability to access or infer any personal identification information. In terms of performance and scalability, the Merkle Proof mechanism offers significant advantages over storing voter lists linearly on-chain. For $\mathbf{n}$ eligible voters, the verification complexity is reduced from $\mathbf{O(n)}$ to $\mathbf{O(\log_n)}$, while on-chain storage costs remain constant, limited to a single fixed-size Merkle root. This design enables the system to scale efficiently to large scale elections without incurring prohibitive Blockchain costs.

Overall, The Merkle proofs approach achieves a significant balance between voter eligibility verification, anonymity preservation, and operational cost efficiency. This approach is well suited to decentralized voting models, where transparency and verifiability are ensured without compromising voter privacy.

2.3.4. Access Control and Secure System Governance via RBAC

In traditional electronic voting systems, administrative authority is often centralized in a single entity, leading to the risk of a single point of failure. When this entity is compromised or subject to internal abuse, the entire election process from system configuration to final results may be manipulated. To mitigate this risk, the proposed system adopts a role based access control (RBAC) mechanism implemented directly at the smart contract layer.

Instead of centralized administration, system privileges are decomposed into independent roles with well-defined functional scopes. Each role is permitted to perform only a restricted set of actions, in accordance with the principles of least privilege and separation of duties. Roles are identified by fixed cryptographic hash values to ensure uniqueness and resistance to forgery, and are formally modeled as:

$$\mathbf{R_i = H(Label_i)} \tag{5}$$

Where $\mathbf{H(.)}$ denotes a one-way cryptographic hash function and $\boldsymbol{Label_i}$ represents the logical identifier of the corresponding role. The set of roles defined in the system is given by:

$$\mathbf{R = \{R_{Manager}, R_{Registrar}, R_{Auditor}\}} \tag{6}$$

The **Election Manager** role ($\mathbf{R_{Manager}}$) is responsible for initializing and configuring election parameters, including the voting period, candidate list, and related system attributes. This role is explicitly prohibited from participating in voter registration or performing any operations on ballots.

The **Voter Registrar** role ($\mathbf{R_{Registrar}}$) is fully isolated and is authorized to verify, add, and manage the list of eligible voters. Separating this role from election configuration prevents a single entity from both creating an election and arbitrarily registering illegitimate voters.

The **Auditor** role ($\mathbf{R_{Auditor}}$) is a special read-only role that is permitted to access system data for monitoring and verification purposes but is strictly prohibited from modifying the system state. This design enables independent auditing without compromising the integrity of the election process.

Access control is formally enforced through a permission-checking function defined as:

$$\mathbf{CheckAccess(u,f)} = \begin{cases} \mathbf{Allow, \exists r \in R: HasRole(u,r) \wedge r \in AllowedRoles(f)} \\ \mathbf{Deny, otherwise} \end{cases} \tag{7}$$

where, $\mathbf{u}$ denotes the calling entity and $\mathbf{f}$ represents the target function. Under this mechanism, no single account can obtain full control over the system; even in the worst-case scenario where a role is compromised, the impact remains strictly confined.

Since the RBAC mechanism is embedded within immutable smart contracts after deployment, all authorization rules are publicly verifiable and cannot be unilaterally modified. This property significantly enhances resistance to administrative abuse, improves transparency, and strengthens the overall trustworthiness of blockchain based electronic voting systems.

2.3.5. Ensuring Fairness and Resistance to Manipulation During the Voting Process

In blockchain-based electronic voting systems, fairness is enforced at the smart contract layer through deterministic and uniform execution of predefined election rules. By embedding these rules directly into immutable code, the system eliminates discretionary human intervention during the voting process.

A voter's right to cast a ballot is restricted to a predefined time window, formally expressed as:

$$\mathbf{T_{start} \le T_{vote} \le T_{end}} \tag{8}$$

where $\mathbf{T_{start}}$ and $\mathbf{T_{end}}$ denote the start and end times of the election, respectively, and represents the timestamp of the voting transaction. Transactions submitted outside this interval are automatically rejected, preventing early, late, or time-based manipulation.

Furthermore, intermediate results are not disclosed during the voting period. Ballots are aggregated and revealed only after the election concludes, reducing herd effects and minimizing strategic or psychological influence on voters.

Because smart contracts are immutable once deployed, election rules cannot be altered unilaterally, including by administrative accounts. This immutability ensures consistency and resistance to manipulation, shifting trust from centralized authorities to publicly verifiable and auditable code.

2.3.6. Preventing Double Voting While Preserving Practical Anonymity Using Nullifiers and Hashed Citizen Identification

In anonymous electronic voting systems, enforcing the principle of *"one citizen, one vote"* is inherently challenging, as voter identities are deliberately decoupled from individual ballots. To address this challenge without

introducing direct identity–ballot linkage, the proposed system adopts a two-layer mechanism that combines nullifiers and hashed Citizen Identification Numbers (CINs), where each mechanism serves a limited and clearly separated purpose.

At the first layer, each voter generates a nullifier, which functions as an anonymous, single-use representation of voting eligibility within a specific election context. The nullifier is defined as:

$$\mathbf{N} = \mathbf{H}\left(\mathbf{S_{id}} \,||\, \mathbf{E_{ctx}}\right) \tag{9}$$

where, $\mathbf{S_{id}}$ denotes a voter-held secret identifier and $\mathbf{E_{ctx}}$ represents the election context. Each nullifier is strictly single-use; once a nullifier has been recorded on-chain, any attempt to reuse the same value is automatically rejected by the smart contract. This mechanism guarantees that each eligible participant can submit at most one ballot per election, while avoiding the disclosure of personal identity information.

At the second layer, to prevent a single citizen from casting multiple votes through the use of multiple blockchain wallets, the system introduces a hashed CIN as a uniqueness constraint at the citizen level. The hashed CIN is used exclusively for duplicate detection and is not linked to ballot contents, voter addresses, or nullifiers. The smart contract stores only the hash value for one-time uniqueness verification and never retains the original identifier.

In practical deployment, the CIN hashing process is assumed to be performed off-chain, where entropy-enhancing measures such as salting can be applied to mitigate dictionary-based inference risks. Consequently, the proposed approach does not claim cryptographic anonymity against adversaries with full access to external population databases. Instead, it aims to prevent direct on-chain identity linkage while ensuring voting uniqueness under a realistic threat model. Due to the immutability of the blockchain, the usage states of both nullifiers and hashed CIN values cannot be altered once recorded. Furthermore, in anonymous voting mode, no association between ballots and voter wallet addresses is stored on-chain, significantly reducing the feasibility of post-election deanonymization via blockchain analysis. Overall, the proposed two-layer design balances practical deployability with privacy preservation, while enforcing voting uniqueness at both the ballot and citizen levels.

From a usability perspective, voting eligibility in the proposed system is not permanently bound to a specific blockchain address. Instead, eligibility is enforced through nullifiers and hashed citizen identifiers. As a result, the loss of a private key does not inherently imply the loss of voting rights. While a full on-chain recovery mechanism is not implemented, the architecture supports off-chain re-verification and controlled re-issuance of voting credentials prior to election finalization.

3. Evaluation and survey results

3.1. Performance Evaluation and Gas Cost Analysis

As summarized in Table 1, the array-based approach exhibits linear growth in on-chain storage and verification cost as the number of voters increases, which can lead to practical limitations under Ethereum's block gas constraints. In contrast, the Merkle Proof-based approach achieves constant on-chain storage and logarithmic verification complexity, providing improved architectural scalability for voter eligibility verification. Based on typical Ethereum gas costs, where a Keccak-256 hash operation consumes approximately 30 gas per 32-byte word, verification of a Merkle tree with depth 20 would require roughly 20 hash computations, resulting in only a few hundred gas units for hashing operations per voter, excluding transaction overhead.

Table 1. *Analytical comparison of on-chain storage and verification complexity between array-based voter lists and Merkle Proof-based approaches*

Comparison Criteria	Array-based Storage	Merkle Proposed-based Approach
On-chain Storage Complexity	($\mathbf{O(n)}$) elements	($\mathbf{O(1)}$) root hash
Initialization Cost (Asymptotic)	($\mathbf{O(n)}$)	($\mathbf{O(1)}$)
Verification Cost	($\mathbf{O(n)}$)	($\mathbf{O(\log_n)}$)
Scalability (Analytical)	Limited by block gas constraints	Improved scalability under gas constraints
Privacy Exposure	Voter list visible on-chain	Voter list hidden off-chain

It should be emphasized that the comparison presented in Table 1 is based on analytical complexity analysis of smart contract operations rather than empirical measurements. The evaluation focuses on on-chain storage requirements and computational characteristics at the contract level, without claiming real-time performance guarantees. In practice, when deployed on public Ethereum networks, system throughput remains constrained by factors such as transaction processing capacity, block confirmation time, and network congestion. Consequently, scalability claims in this study should be interpreted in terms of architectural and computational feasibility, rather than as an assertion of unrestricted large-scale transaction throughput on Layer-1 Ethereum. Potential integration with Layer-2 solutions or alternative blockchain infrastructures is therefore considered an important direction for future work

3.2. Security Threat Model and Attack Resistance Analysis

The security of an electronic voting system deployed on a public blockchain depends on its ability to withstand realistic adversarial behaviors. In this study, we consider a standard public blockchain threat model in which adversaries can observe on-chain data and pending transactions, submit arbitrary transactions, and control multiple blockchain accounts. It is assumed that the underlying cryptographic primitives satisfy standard security properties such as collision resistance and preimage resistance.

Double voting attacks are mitigated through a two-layer

enforcement mechanism. At the ballot level, each voter generates a unique nullifier bound to a specific election context. The usage of each nullifier is immutably recorded on-chain, ensuring that any attempt to reuse a nullifier is automatically rejected. At the citizen level, a hashed representation of the citizen identification number is used to enforce voter uniqueness across multiple wallets. Under standard hash security assumptions, this approach enforces the principle of "one citizen, one vote" without revealing voter identity. Front-running and vote manipulation attacks are addressed using a Commit–Reveal scheme. During the commitment phase, votes are represented solely by cryptographic hash values, making inference of voting choices from the mempool computationally infeasible. Vote disclosure is permitted only after the voting period ends, preventing early result exposure and preserving fairness throughout the election.

Insider threats are mitigated through role-based access control (RBAC) enforced directly at the smart contract layer. Administrative privileges are partitioned into distinct roles following the principles of least privilege and separation of duties. All sensitive administrative actions are recorded on-chain, enabling transparent auditability and preventing undetected post hoc manipulation due to blockchain immutability.

Finally, denial-of-service attacks based on transaction spamming are constrained by the economic cost imposed by transaction fees and by early validation checks implemented in smart contracts. While such attacks cannot be fully eliminated in public blockchain environments, these mechanisms significantly raise the cost of attack under practical conditions.

4. Conclusion

This paper presents a blockchain-based electronic voting architecture designed to enhance transparency, fairness, and execution-level security in public election environments. By enforcing election rules through immutable smart contracts, the proposed model ensures uniform rule application without reliance on centralized intermediaries. The integration of Commit–Reveal, Merkle proofs, nullifier mechanisms, and hashed CIN enables eligibility verification and double-voting prevention while preserving practical voter anonymity. Rather than introducing new cryptographic primitives or formal proofs, this work emphasizes system-level feasibility and architectural enforcement under the practical constraints of public blockchains. In contrast to zk-based and fully cryptographic end-to-end voting systems that prioritize strong theoretical guarantees, the proposed approach focuses on smart contract enforceability, auditability, and deployability in real-world governance contexts. While the architecture minimizes on-chain exposure of personal data and supports independent verification, its practical adoption requires careful alignment with national election regulations and data protection frameworks. Future work may explore integration with advanced privacy-preserving techniques and Layer-2 scalability solutions, as well as interdisciplinary evaluation of legal and ethical implications in large-scale deployments.

REFERENCES

[1] R. Krimmer, D. Duenas-Cid, and I. Krivonosova, "Debunking Myths about Electronic Voting: Lessons from Twenty Years of E-Voting in Estonia," *Government Information Quarterly*, vol. 38, no. 4, 2021.

[2] L. Xu, C. Chen, Y. Liu, and Y. Wang, "A multi-candidate voting model based on blockchain," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 12, pp. 1858-1870, 2021.

[3] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *Proc. Financial Cryptography and Data Security (FC)*, 2017, pp. 357-375.

[4] Stevena, F. H. Hadiprodjoa, I. N. Alama, and L. A. Wulandhari, "Implementation of blockchain-based electronic voting system: A case study in Indonesia," *Procedia Computer Science*, vol. 269, pp. 1–12, 2025.

[5] Y. Liu and Q. Wang, "An e-voting protocol based on blockchain," *IACR Cryptology ePrint Archive*, Report 2017/1043, 2017.

[6] M.-V. Vladucu, "E-voting meets blockchain: A survey," *IEEE Access*, vol. 11, pp. 24874-24896, 2023.

[7] D. D. Bhavani, G. R. Gayathri, B. M. Bhagavanthu, A. Sheeba, M. S. M. Maria, and B. P. Bhuvaneshwari, "Blockchain-based voting systems enhancing transparency and security in electoral processes," *ITM Web of Conferences*, vol. 76, p. 02004, 2025.

[8] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," *Journal of Cryptology*, vol. 3, no. 2, pp. 99–111, 1991.