International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 11 | Nov 2026 www.irjet.net p-ISSN: 2395-0072

Secure Vote- A Blockchain - Inspired E-Voting System with Real Time Fraud Detection: A Review of Architecture, Security, Privacy, and Future Directions

Ananya Khadse[1], Manisha Biswas[2], Hrishikesh Gawhad[3], Shweta Meshram[4]

[1234]*Department of Computer Science and Engineering (Data Science), Guru Nanak Institute of Technology, Nagpur, Maharashtra, India*

Abstract - *Electronic voting can improve accessibility and reduce the time required for ballot collection and counting, but it introduces difficult requirements for voter authentication, ballot secrecy, integrity, auditability, availability, and public trust. Conventional centralized electronic voting architectures concentrate sensitive election functions in a small number of servers, creating high-value targets and making independent verification difficult. Blockchain provides a distributed and tamper-evident transaction layer that can record election events and enforce predefined rules through smart contracts. However, blockchain alone does not automatically guarantee voter privacy, coercion resistance, endpoint security, or correct election procedures. This review examines recent research on blockchain-based e-voting and organizes it around the requirements of a secure e-voting project: voter registration and eligibility, one-person-one-vote enforcement, ballot protection, immutable recording, transparent tallying, and independent audit. The paper compares public and permissioned blockchain approaches, discusses cryptographic techniques such as hashing, digital signatures, encryption, and zero-knowledge proofs, and presents a four-layer architecture suitable for an academic prototype. A smart-contract state model and security workflow are also described. Recent literature from 2020–2026 shows that blockchain can strengthen integrity and transparency, while privacy, scalability, usability, key management, coercion resistance, and legal governance remain open challenges. The review concludes with research directions involving zero-knowledge protocols, off-chain computation, decentralized identity, privacy-preserving tallying, and scalable Layer-2 architectures.*

Keywords - Blockchain, E-Voting, Electronic Voting, Smart Contracts, Voter Authentication, Ballot Integrity, Privacy, Transparency, End-to-End Verifiability, Zero-Knowledge Proofs.

I. INTRODUCTION

Voting is a fundamental democratic process in which the reliability of the mechanism used to record and count votes directly affects public confidence in the result. Electronic voting attempts to digitize parts of this process to improve convenience, accessibility, speed, and administrative efficiency. At the same time, moving elections into networked environments introduces new attack surfaces involving credentials, servers, software, communications, devices, and databases. A secure e-voting system therefore has to protect the vote while also allowing the election to be independently checked.

Traditional centralized e-voting architectures generally rely on a trusted authority to maintain voter records, receive ballots, store them, and calculate the final result. This model is operationally practical, but the central server becomes a critical trust and failure point. A compromise of the database or application layer can potentially affect large numbers of ballots. Even when the server is not compromised, voters may have limited ability to verify that their ballot was recorded and included correctly. Recent reviews consistently identify integrity, privacy, authentication, transparency, verifiability, scalability, and coercion resistance as central design requirements [1]–[6].

Blockchain changes the trust model by distributing a replicated ledger among participating nodes. Once a valid block is accepted according to the selected consensus mechanism, previously recorded transactions become difficult to alter without detection. Smart contracts can voting attempts. Research has demonstrated prototypes using Ethereum, Solidity, IPFS, MetaMask, and related Web3 components [7]–[10].

However, an important distinction must be maintained: blockchain can protect the integrity of information that is correctly submitted to it, but it cannot by itself prove that a voter's device was malware-free, that an identity was correctly established, or that a voter was not coerced. Similarly, storing a plain vote transaction on a public ledger may conflict with ballot secrecy. Therefore, a secure design must combine blockchain with cryptographic privacy mechanisms and carefully separated identity and ballot layers.

This review is aligned with the proposed academic project, “Secure E-Voting System Using Blockchain.” The project is treated as a secure prototype rather than as a claim that blockchain alone can replace the complete governance, legal, accessibility, and operational requirements of a national election. The review contributes: (1) a structured problem formulation for

secure e-voting; (2) a four-layer architecture; (3) a literature synthesis of recent blockchain voting research; (4) a smart-contract voting lifecycle; (5) a security and threat analysis; and (6) future directions for privacy-preserving and scalable implementations.

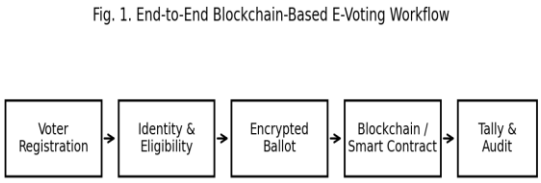

Fig. 1. End-to-End Blockchain-Based E-Voting Workflow.

II. MATHEMATICAL PROBLEM FORMULATION

A. System Entities and Voting Model

Let V = {v1, v2, …, vN} denote the set of registered voters and C = {c1, c2, …, cM} the set of candidates. Each voter vi is associated with an eligibility credential Ei, while the actual ballot should not expose the real-world identity of vi to the public ledger. Let B denote the set of valid ballots submitted during an election.

Let the election state be $E = \langle V, C, B, L, SC, T \rangle$, where L is the blockchain ledger, SC is the smart-contract state, and T is the election time window. The system objective is to maximize election integrity and verifiability while preserving ballot secrecy and enforcing the one-person-one-vote constraint.

$$\text{ValidVote}(v_i) = \text{Eligible}(v_i) \wedge \text{Active}(E) \wedge \neg\text{Voted}(v_i)$$

$$\text{VoteRecord} = \text{Hash}(\text{Ballot}) + \text{Signature/Proof} + \text{Timestamp} + \text{ContractState}$$

$$\text{Result}(c) = \Sigma_i \, \mathbb{1}(B_i = c), \text{ subject to } \text{ValidVote}(v_i) = 1$$

B. Security Constraints

A secure transaction should satisfy three core conditions. First, a ballot must be accepted only from an eligible voter who has not already consumed the voting right. Second, the public ledger should allow integrity verification without revealing the mapping between voter identity and selected candidate. Third, after the election closes, the tally procedure should be reproducible from the committed ballot records or cryptographic proofs.

$$\Pr[\text{Accept}(B_i) \mid \neg\text{Eligible}(v_i) \vee \text{Voted}(v_i)] = 0$$

$$\text{Privacy}(B_i) : \text{Identity}(v_i) \nrightarrow \text{Choice}(B_i)$$

$$\text{Integrity}(B) : H(B_i) = H(B_i)\text{stored} \Rightarrow B_i \text{ is unmodified}$$

III. LITERATURE REVIEW & SYSTEMATIC SURVEY

The literature was organized around the evolution from conceptual blockchain voting to privacy-preserving, scalable, and implementation-oriented architectures. Earlier studies emphasized decentralization and immutability; subsequent work increasingly focused on privacy, smart-contract security, off-chain computation, performance, and formal verification. Recent surveys also show that no single architecture simultaneously solves all election requirements [1]–[6].

A. Early Research and Security Requirements (2020–2022)

Taş and Tanrıöver presented a systematic review of blockchain-based e-voting and highlighted both opportunities and unresolved challenges, including privacy, scalability, authentication, and the practical suitability of blockchain for elections [1]. The work is useful for defining the security requirements that should be evaluated before selecting a blockchain architecture.

Studies published in 2021 examined blockchain as a way to improve trust, privacy, and security in electronic voting. One IEEE survey evaluated blockchain-based e-voting architectures and emphasized decentralized storage and trustless operation [2]. Another implementation-oriented study used Solidity smart contracts and discussed zero-knowledge proofs as a path toward privacy-preserving voting [7].

Research in 2022 continued to compare voting frameworks and identify essential requirements such as voter anonymity, resistance to duplicate voting, verifiability, and decentralized counting. Benabdallah et al. provided a systematic literature review of blockchain solutions for e-voting and emphasized authentication, privacy, security, smart contracts, and transparency [3]. Other IEEE work explored public-blockchain voting and practical implementations using smart contracts [8], [9].

B. Blockchain Voting Architectures (2023–2024)

Abundant research in 2023 shifted toward complete system architectures. A major IEEE Access survey synthesized the past, present, and future of blockchain-based e-voting and reviewed systems deployed or proposed by governments, organizations, and researchers [4]. The survey also emphasized that blockchain improves

tamper resistance but does not remove the need for careful privacy and governance mechanisms.

Tanwar et al. investigated a blockchain-based voting mechanism that combines smart contracts with IPFS and a 5G communication environment, with attention to security, cost, storage, and response time [10]. Such hybrid architectures illustrate the practical value of keeping large or auxiliary data off-chain while recording verifiable commitments on-chain.

Sharp et al. compared blockchain-based voting mechanisms using blockchain models, cryptographic techniques, vote-counting approaches, and security requirements. Their 2024 study also proposed a flexible blockchain-based system and demonstrated feasibility on constrained hardware [5]. Ohize et al. provided a broad survey of architectures, trends, solutions, performance constraints, and security concerns, emphasizing scalability and infrastructure requirements [6].

C. Privacy, Scalability, and Recent Directions (2025–2026)

Recent work increasingly combines blockchain with privacy technologies rather than treating the ledger as the complete solution. BP-Vot, for example, combines smart contracts with differential privacy and self-sovereign identity concepts, illustrating a direction in which voter identity and election transactions are separated more carefully [11].

Zero-knowledge off-chain computation has also been investigated to reduce on-chain storage and improve scalability while retaining transparency and privacy properties [12]. This direction is important for large elections because directly storing every sensitive voting operation on a public chain can be expensive and can expose metadata.

Recent 2026 work continues to explore Ethereum-based implementations, smart-contract voting, and the practical trade-offs between immutability, anonymity, transaction cost, and usability [13]. These developments suggest that future secure e-voting systems are likely to be hybrid: blockchain for integrity and coordination, off-chain components for computation and storage, and advanced cryptography for privacy.

TABLE I: SYNTHESIS OF REPRESENTATIVE BLOCKCHAIN E-VOTING CONTRIBUTIONS

Ref./Year	Core Focus / Method	Key Contribution	Limitation / Open Issue
[1] 2020	Systematic review	Security requirements and research challenges	Scalability and practical deployment remain difficult
[2] 2021	Blockchain e-voting survey	Trust, privacy and decentralized voting	Broad survey; implementation varies by system
[7] 2021	Privacy-preserving e-voting	Solidity smart contract; ZKP direction	Prototype scale and deployment assumptions
[3] 2022	Systematic literature review	Authentication, privacy, smart contracts, transparency	No universal architecture
[8] 2022	Public blockchain voting	Public-chain election workflow	Public ledger metadata/privacy concerns
[9] 2022	Blockchain e-voting prototype	Application and smart-contract integration	Prototype rather than national-scale validation
[4] 2023	IEEE Access survey	Past, present and future of blockchain voting	Open challenges in trust and scalability
[10] 2023	5G + blockchain voting	IPFS, smart contracts, performance analysis	Infrastructure dependence
[12] 2023	ZK off-chain voting	Scalable privacy-preserving computation	Cryptographic complexity
[5] 2024	Survey + proposal	Cryptography, counting and flexible blockchain	Requires further large-scale validation
[6] 2024/25	Architecture survey	Trends, architectures, security and performance	Scalability and infrastructure remain open
[11] 2025	BP-Vot	Smart contracts + differential privacy + SSI	Complexity and deployment maturity
[13] 2026	Ethereum implementation	Smart-contract based voting prototype	Gas, usability and practical limitations

IV. ARCHITECTURAL TAXONOMY

For the proposed Secure E-Voting System Using Blockchain, the architecture is organized into four interconnected layers. The separation prevents voter identity, ballot content, application logic, and blockchain state from becoming a single monolithic component. It also supports later replacement of the consensus network or privacy mechanism without redesigning the complete voter interface.

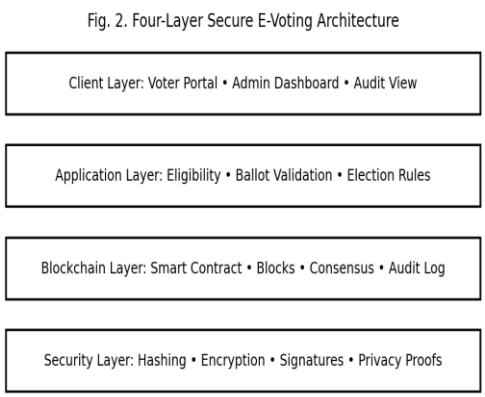

Fig. 2. Four-Layer Secure E-Voting Architecture.

1. Client and Voter Interface Layer

The client layer provides voter registration, login or wallet authentication, candidate information, ballot selection, vote confirmation, and an audit interface. The interface should clearly distinguish identity verification from ballot submission. Accessibility, error prevention, and confirmation messages are critical because a technically secure backend can still produce an unreliable election if users cannot understand or correctly operate the interface.

2. Application and Eligibility Layer

The application layer validates election configuration, voter eligibility, candidate status, election timing, and one-person-one-vote rules. A conventional database may maintain administrative records, while the blockchain stores verifiable commitments and election events. In a stronger design, a decentralized identity or privacy-preserving credential can prove eligibility without exposing unnecessary personal information.

3. Blockchain and Smart-Contract Layer

The blockchain layer provides replicated state, transaction ordering, tamper-evident records, and programmable election rules. Smart contracts can create an election, register candidates, authorize voting windows, accept valid ballots or ballot commitments, prevent repeated voting, close the election, and publish the final tally. The choice between public and permissioned blockchain should depend on the threat model, governance requirements, transaction cost, and expected voter population.

4. Security and Cryptographic Layer

The security layer provides hashing, digital signatures, encryption, key management, and—where required—zero-knowledge proofs. A hash can detect modification, a digital signature can authenticate a transaction source, and encryption can protect ballot secrecy. Zero-knowledge mechanisms can prove eligibility or correctness without revealing sensitive information. These controls should be applied according to the required election property rather than simply added as independent features.

V. SECURITY AND ALGORITHMIC FRAMEWORKS

A. Voter Registration and Eligibility

A voter should be registered once and associated with a unique eligibility record. The system can maintain an eligibility flag or cryptographic credential that is consumed when a vote is accepted. Importantly, the identifier used for eligibility should not become a public identifier for the ballot. This separation reduces the risk that observers can correlate a voter's identity with the selected candidate.

$$\text{Eligible}(v_i) = \text{IdentityVerified}(v_i) \wedge \text{RegistrationActive}(v_i) \wedge \text{ElectionActive}$$

For an academic prototype, eligibility may be implemented using an administrator-controlled registration list. A more advanced system can replace this with decentralized identity or anonymous credentials.

B. Ballot Integrity and Hashing

Before recording a ballot commitment, the application can compute a cryptographic digest of the ballot representation. If H denotes a collision-resistant hash function, then a small change to the ballot produces a different digest with overwhelming probability. The hash should be treated as an integrity mechanism rather than as a confidentiality mechanism because hashing alone does not hide a low-entropy candidate choice.

$$h_i = \text{SHA-256}(\text{Encode}(B_i \,||\, \text{nonce}_i))$$

C. Digital Signatures and Authentication

Digital signatures can establish that a transaction was authorized by the holder of a private key. In a voter-facing application, the signing key may belong to a wallet or

authentication service. Signature verification protects the transaction against unauthorized modification after signing. Key loss, however, creates an operational problem; recovery mechanisms must not permit an attacker to obtain a replacement credential without strong identity assurance.

D. One-Person-One-Vote Algorithm

The smart contract maintains a spent-vote state for each eligible voting credential. When a ballot is submitted, the contract checks election status, voter eligibility, and whether the voting right has already been consumed. The state is updated atomically with the ballot commitment so that two competing transactions cannot both succeed under a properly designed contract.

$$\text{if } \text{Eligible}(v_i) \wedge \neg\text{Spent}(v_i) \wedge \text{Now} \in [t_{start}, t_{end}] \rightarrow \text{Accept};$$
$$\text{Spent}(v_i) \leftarrow \text{True}$$

else → Reject

E. Privacy-Preserving Tallying

A naïve blockchain implementation that writes the candidate choice in plaintext can provide integrity but destroy ballot secrecy. Therefore, the proposed design should preferably store encrypted ballots or commitments and perform tallying only after the election closes. Homomorphic encryption, mix networks, threshold decryption, or zero-knowledge proofs can be introduced as the project evolves. For a student prototype, a controlled encrypted-ballot workflow may be sufficient to demonstrate the principle, while advanced cryptography can be presented as future work.

Algorithm 1: Secure Ballot Submission

Input: voter credential V, candidate choice C, election state E. Output: accepted or rejected ballot.

1: Verify voter eligibility and election status.
2: Check that the voting credential has not already been consumed.
3: Encode the selected candidate and generate a random nonce.
4: Encrypt or commit the ballot using the selected privacy mechanism.
5: Generate the required signature/proof.
6: Submit the transaction to the smart contract.
7: If all contract conditions are valid, record the ballot commitment and mark the voting right as spent.
8: Return a transaction identifier for later verification; otherwise reject the transaction.

VI. SMART CONTRACT ELECTION EXECUTION

The smart contract functions as the deterministic election controller. Its responsibilities should be limited to operations that benefit from verifiable shared state. Large documents, personal records, and heavy cryptographic computation should normally remain off-chain. The contract should expose a small set of auditable functions such as createElection(), registerCandidate(), authorizeVoter(), castVote(), closeElection(), and publishResult().

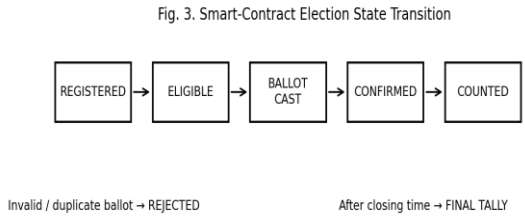

Fig. 3. Smart-Contract Election State Transition.

A. State Transition Model

The election lifecycle can be represented by six conceptual states: CREATED, OPEN, VOTING, CLOSED, TALLIED, and AUDITED. A voter moves from REGISTERED to ELIGIBLE and then to VOTED only after the contract accepts a valid transaction. Invalid or duplicate transactions are rejected without changing the successful-vote state.

CREATED → OPEN → VOTING → CLOSED → TALLIED → AUDITED

REGISTERED → ELIGIBLE → VOTED ; duplicate/invalid → REJECTED

B. Smart Contract Pseudocode

```
contract SecureVoting:
  createElection(start, end, candidateSet)
  require(admin == msg.sender)
  require(start < end)
  castVote(credential, ballotCommitment, proof):
    require(now >= start && now <= end)
    require(eligible[credential] == true)
    require(spent[credential] == false)
    require(verify(proof, ballotCommitment))
    spent[credential] = true
    store(ballotCommitment)
  closeElection():
    require(now > end)
```

```
  electionClosed = true
publishResult(resultProof):
  require(electionClosed == true)
  require(verifyTally(resultProof))
  resultPublished = true
```

C. Auditability

Each successful ballot transaction can return a transaction identifier or commitment reference. Auditors can verify that the number of accepted transactions matches the contract state, that election timestamps are within the defined window, and that no credential was consumed more than once. Auditability should not be confused with exposing individual vote choices; a good design enables public verification while protecting the secrecy of the ballot.

VII. COMPARATIVE ANALYSIS

The following comparison positions the proposed project against common e-voting architectures. It focuses on technical properties rather than claiming that one platform is universally superior. Centralized systems can provide strong operational control, while blockchain architectures can improve distributed auditability; the final choice depends on election scale, governance, legal requirements, and threat model.

Dimension	Centralized E-Voting	Public Blockchain	Permissioned Blockchain	Proposed Prototype
Trust model	Central authority	Distributed/public	Governed consortium	Controlled blockchain network
Vote integrity	Server/database controls	Ledger + consensus	Ledger + governed consensus	Hash + smart contract
Auditability	Limited/administrator dependent	High ledger transparency	High within consortium	High for recorded events
Ballot privacy	Depends on server design	Requires encryption/ZKP	Requires encryption/ZKP	Encrypted/committed ballot
One-person-one-vote	Central database	Credential + contract	Credential + contract	Eligibility + spent state
Scalability	Generally high	Variable; congestion possible	Potentially higher	Prototype-scale
Transaction cost	Operational cost	Gas/fees possible	Controlled cost	Test-network/low-scale
Main weakness	Single trust point	Privacy, metadata, throughput	Governance/trust assumptions	Prototype and key-management limits

VIII. OPEN SECURITY & ENGINEERING CHALLENGES

A. Endpoint and Malware Risk

Blockchain protects the ledger after a transaction is accepted, but a compromised voter device can manipulate what the voter sees or what is submitted. A malicious browser extension, malware, or phishing attack can therefore bypass ledger-level guarantees. Secure software updates, device hardening, transaction confirmation, and independent client verification are required.

B. Privacy and Metadata Leakage

Even when the candidate choice is encrypted, public transaction metadata may reveal timing, wallet relationships, transaction frequency, or other correlations. Therefore, privacy analysis must consider the complete transaction graph rather than only the ballot ciphertext. Mix networks, anonymous credentials, batching, and zero-knowledge proofs can reduce such leakage.

C. Coercion and Vote Buying

A remote voter may be pressured to reveal a credential, screen, or receipt. A simple blockchain transaction receipt can accidentally become evidence of how a person voted. Coercion resistance therefore requires specialized protocol design such as receipt-freeness, deniable credentials, or mechanisms that allow a voter to prove participation without proving the selected candidate.

D. Smart-Contract Vulnerabilities

Election contracts are high-value targets because a programming error can affect eligibility, voting windows, candidate registration, or tallying. Common software risks include incorrect access control, state-transition bugs, reentrancy where applicable, arithmetic errors, and flawed validation. Formal verification, independent code review, unit testing, and testnet deployment should precede production use.

E. Scalability and Transaction Fees

Writing one blockchain transaction per voter may become expensive or slow on congested public networks. Layer-2 networks, batching, rollups, off-chain commitments, and specialized permissioned consensus can reduce the burden. A scalable design must still preserve independent verification and avoid moving critical trust back to a single server.

F. Key Management and Recovery

Cryptographic systems are only as secure as their key management. Lost keys, stolen credentials, insecure administrator accounts, or weak recovery processes can undermine the election. Key rotation, multi-signature administration, hardware-backed storage, threshold authorization, and carefully audited recovery procedures are possible controls.

G. Legal, Accessibility, and Governance Issues

A technically secure voting system may still be unsuitable for a real election if it fails accessibility, legal, audit, or public-governance requirements. Voters with limited connectivity or disabilities must be supported, and election authorities must define who controls upgrades, emergency stops, candidate registration, dispute resolution, and final certification.

IX. EMERGING RESEARCH FRONTIERS

1. Zero-Knowledge Voting Proofs

Zero-knowledge protocols can allow the system to prove eligibility and ballot validity without exposing sensitive voter information. This can strengthen end-to-end verifiability while reducing the need to publish private election data.

2. Decentralized Identity and Anonymous Credentials

A future architecture can separate the proof that a person is eligible from the public identity of the ballot transaction. Anonymous credentials can reduce the need to place personally identifiable information on the blockchain.

3. Off-Chain Computation and Layer-2 Settlement

Large elections can move expensive computation or transaction aggregation off-chain while retaining compact commitments on-chain. Layer-2 systems can lower cost and improve throughput while preserving a verifiable settlement layer.

4. Formal Verification and Automated Contract Auditing

Election contracts should be analyzed as security-critical software. Formal models can verify invariants such as one-person-one-vote, correct election timing, and authorized administrative actions.

5. Privacy-Preserving Tallying

Threshold decryption, homomorphic encryption, and verifiable mixnets can allow ballots to remain hidden while enabling the public to verify that the final tally follows the committed election rules.

6. Post-Quantum Readiness

Long-lived election records may require cryptographic migration planning as quantum computing develops. Future systems should consider post-quantum signatures and key-establishment methods for voter credentials and administrative keys.

X. DOMAIN-SPECIFIC CASE STUDIES

A. University and Student Elections

University elections are a practical environment for evaluating blockchain voting because the voter population, candidate set, election duration, and governance authority are bounded. The proposed system can be demonstrated using student registration, candidate approval, secure vote casting, automatic counting, and an audit dashboard. Such a deployment can validate the complete workflow without making claims about national election readiness.

B. Organizational and Cooperative Elections

Organizations can use blockchain-based voting for shareholder, committee, or membership elections where participants are known and governance rules are explicit. Permissioned networks may be appropriate when membership is controlled and transaction privacy is important.

C. Public-Election Research Prototypes

For large public elections, the prototype architecture provides a research baseline rather than a direct deployment blueprint. The main research questions become scalability, accessibility, coercion resistance, anonymous eligibility, independent auditing, legal compliance, and recovery from infrastructure failures. Recent surveys emphasize that these issues remain active even when blockchain provides an immutable ledger [4]–[6].

XI. CONCLUSION

The review shows that blockchain can provide a strong foundation for a secure e-voting prototype by distributing election state, recording tamper-evident transactions, and allowing smart contracts to enforce deterministic voting rules. For the proposed Secure E-Voting System Using Blockchain, the most important architectural principle is separation: voter identity and eligibility should be handled separately from ballot secrecy, while the blockchain should maintain only the verifiable state necessary for election integrity and audit.

Recent literature from 2020 to 2026 demonstrates meaningful progress in blockchain-based e-voting, including smart-contract implementations, public and permissioned architectures, IPFS-assisted systems, privacy-preserving approaches, zero-knowledge computation, and decentralized identity concepts [1]–[13]. At the same time, blockchain should not be presented as a complete security solution. Endpoint compromise, coercion, metadata leakage, key loss, contract vulnerabilities, scalability, accessibility, and legal governance remain significant barriers.

The proposed academic architecture therefore uses blockchain as a verifiable coordination and integrity layer, complemented by cryptographic protection and off-chain application services. Future development can extend the prototype with zero-knowledge proofs, anonymous credentials, threshold tallying, formal smart-contract verification, and Layer-2 scalability. This direction offers a practical pathway from a demonstrable student project toward a research-grade secure e-voting framework.

A third limitation concerns the blockchain trust model. Permissioned networks still require trust in organizations controlling validator nodes, while public blockchains introduce transaction fees, congestion, and metadata exposure. The appropriate architecture depends on the election context. Blockchain should therefore be treated as one component of a broader security architecture rather than as an automatic replacement for institutional trust.

A second limitation is cryptographic complexity. Advanced privacy-preserving methods such as zero-knowledge proofs, homomorphic encryption, and verifiable mixnets can substantially increase implementation and verification complexity. A student project should clearly distinguish implemented features from proposed future enhancements. This prevents overclaiming security properties that have not been experimentally validated.

The proposed project is an academic prototype. It can demonstrate secure workflow, smart-contract enforcement, tamper-evident records, and auditability, but it cannot by itself establish suitability for government or national elections. Real elections require independent certification, accessibility compliance, legal authorization, operational continuity, voter education, dispute procedures, and long-term governance.

The reviewed literature contains many prototypes and experimental systems, but these studies differ substantially in consensus mechanisms, cryptographic assumptions, voter populations, evaluation datasets, and threat models. Direct numerical comparison across all papers is therefore inappropriate. The review synthesizes design patterns and security requirements rather than claiming that one reported performance value represents all blockchain voting systems.

XIV. LIMITATIONS OF THE REVIEW AND PROPOSED PROTOTYPE

Deployment Security: Private keys used by administrators should never be hard-coded into application source files. Configuration secrets should be kept outside the public repository, and production deployment should use protected key storage. Smart-contract source code should be version-controlled, tested, and reviewed before deployment.

Counting and Audit: At the end of the voting window, the administrator closes the election. The tally component processes valid ballot records or encrypted commitments and publishes a verifiable result. Auditors can inspect the number of accepted votes, rejected attempts, election timestamps, and state transitions. The audit view should avoid exposing voter identities or individual choices.

Voting Interface: After authentication, the voter receives only the elections for which the credential is eligible. The interface displays candidates, collects the selected option, and provides a confirmation screen. The client then generates the ballot commitment or encrypted ballot and submits it to the smart contract. A transaction identifier is returned as evidence that the request was accepted by the blockchain layer.

Candidate and Election Management: The administrator creates an election with a start time, closing time, candidate list, and authorized voter set. Candidate identifiers should remain stable during the election so that the tally cannot be changed by modifying display names after voting begins. Once the election opens, configuration functions should be restricted or disabled until the election closes.

A practical academic implementation can use a web-based voter interface connected to a blockchain test environment through a wallet or Web3 provider. The application layer can use a conventional backend for registration and administrative records, while Solidity smart contracts manage election state. A lightweight database can maintain

non-sensitive application metadata. This hybrid design demonstrates the central idea of blockchain-based integrity without unnecessarily placing every application record on-chain.

XIII. IMPLEMENTATION CONSIDERATIONS FOR THE PROPOSED PROJECT

Usability Evaluation: Security mechanisms are useful only when voters can operate them correctly. A small controlled user study can record completion time, error rate, failed authentication attempts, and user understanding of confirmation messages. The interface should clearly indicate whether a ballot was accepted, rejected, or still pending. The system should not require voters to understand blockchain terminology.

Privacy Evaluation: Privacy should be evaluated by inspecting which voter attributes appear in application logs, blockchain events, transaction payloads, and audit reports. Personally identifiable information should not be directly associated with a public ballot. If encryption or zero-knowledge proofs are implemented, the evaluation should verify that a verifier can establish validity without learning the protected choice.

Integrity Evaluation: A test set of valid and deliberately modified ballots can verify that the stored cryptographic commitment changes when ballot data changes. The audit module should independently recompute the commitment and compare it with the recorded value. Any mismatch should be reported as a verification failure rather than silently corrected.

Throughput can be measured as the number of successfully confirmed voting transactions per unit time. A useful experiment gradually increases the number of concurrent simulated voters and records successful confirmations, rejected transactions, average response time, and peak resource consumption. For a student prototype, the objective is not to prove national-scale capacity but to characterize the behavior of the selected test network under controlled load.

Performance is an important research dimension because a voting system must remain usable when many voters submit transactions during a limited election window. The proposed evaluation should distinguish application latency from blockchain confirmation latency. Application latency measures the time required to validate a request and construct the transaction, whereas confirmation latency measures the time until the transaction is included in an accepted block. Separating these values helps identify whether delays originate in the application, network, or blockchain.

XII. PERFORMANCE AND EVALUATION METRICS

Phase 5—Evaluation and Documentation: The final stage compares the prototype with the requirements defined at the beginning. Metrics should include successful transaction rate, average transaction confirmation time, duplicate-vote rejection rate, tally correctness, contract execution cost, and usability observations. Security claims should be limited to properties actually demonstrated by the implementation.

Phase 4—Security Testing: Testing should include functional tests, negative tests, access-control tests, transaction replay tests, and basic smart-contract vulnerability analysis. The objective is to demonstrate that invalid operations fail and that valid operations produce consistent state transitions. Where cryptographic proofs are used, verification failures should be tested independently from application-level failures.

Phase 3—Smart-Contract Development: The contract should implement election creation, candidate registration, voter authorization, vote submission, election closure, and result publication. Each function requires access-control checks and state validation. Test cases should include invalid candidates, unauthorized voters, duplicate ballots, voting before the start time, voting after the closing time, and unauthorized result modification. Contract events can provide a compact audit trail for external monitoring.

Phase 2—Architecture Design: The architecture separates the public-facing application from the ledger. Voter information can remain in an administrative database or identity service, while the blockchain stores only election state, commitments, and verifiable events. This separation reduces the amount of personal information placed on-chain and makes the system easier to upgrade. The architecture also permits a permissioned blockchain to be substituted for a public chain if governance or cost requirements demand it.

Phase 1—Requirement Analysis: Functional requirements include voter registration, eligibility verification, candidate management, ballot casting, vote confirmation, election closure, counting, and audit reporting. Non-functional requirements include integrity, confidentiality, availability, usability, transparency, and verifiability. The requirements should be mapped to measurable test cases so that a successful demonstration is based on evidence rather than only on screenshots.

The reviewed literature supports a layered methodology for the proposed Secure E-Voting System. The first phase is requirement analysis, in which election actors, trust assumptions, voting rules, privacy requirements, and audit requirements are documented. The second phase is architecture design, where identity, application logic,

blockchain state, and cryptographic services are separated. The third phase is implementation, in which the voter interface and smart contract are developed and connected through a controlled test environment. The fourth phase is security testing, followed by functional evaluation and documentation of limitations.

XI. PROPOSED RESEARCH METHODOLOGY

Coercion Resistance: Remote voting creates a unique coercion problem because a voter may be observed by another person while casting a ballot. A normal blockchain receipt can unintentionally become proof of a particular choice. Consequently, transaction confirmation should prove that a valid ballot was accepted without creating a transferable proof of the candidate selected. Receipt-free or deniable voting mechanisms remain an advanced research requirement.

Ballot Privacy and Correlation: Publicly visible transactions can reveal timing and behavioral metadata even when the vote itself is encrypted. A privacy-preserving architecture should therefore avoid publishing personally identifiable information, direct candidate choices, or unnecessary wallet relationships. Anonymous credentials, encrypted ballots, mixing, batching, and zero-knowledge proofs can reduce the correlation between a real-world voter and an on-chain ballot commitment.

Compromised Client Endpoint: A blockchain cannot determine whether a voter's phone or computer is trustworthy. Malware could replace the selected candidate, display false information, or steal credentials before the transaction reaches the chain. Therefore, the user interface should show a final confirmation step, use secure authentication, minimize locally stored secrets, and avoid exposing sensitive private keys. For high-assurance elections, independent verification clients and secure hardware can be considered.

Malicious Voter and Duplicate Voting: The one-person-one-vote rule is implemented as a state invariant. Once a valid voting credential is consumed, the corresponding state must become permanently unavailable for another successful ballot during the election. Atomic contract execution is important because two requests arriving close together should not both observe the credential as unused. Unit tests should explicitly cover duplicate, concurrent, expired, and unauthorized submissions.

Administrator Threats: Election administrators require strong role separation. Functions such as candidate registration, election creation, election closure, and result publication should not be controlled by a single unrestricted account in a production-oriented design. Multi-signature administration or threshold authorization can reduce the risk that one compromised private key changes the election state. The academic prototype can demonstrate this principle by separating voter, administrator, and auditor roles.

Network Attacks and Replay: Every accepted voting transaction should contain a unique credential, nonce, or transaction identifier. The smart contract should reject a previously consumed credential and validate that the transaction belongs to the current election. Transport-layer security protects the client-server communication path, while digital signatures provide transaction-level authenticity. A blockchain record is useful only after the transaction has successfully passed these checks.

A secure e-voting system should begin with an explicit threat model rather than assuming that the blockchain itself is the attacker boundary. The proposed system considers four broad adversary classes: an external network attacker, a malicious or compromised administrator, a compromised voter endpoint, and a dishonest participant attempting duplicate or invalid voting. Each adversary has a different capability and therefore requires different controls. A network attacker may attempt to intercept or replay transactions, while a compromised administrator may try to alter election configuration. A malicious voter may attempt to vote more than once, and a compromised device may alter the ballot before submission. The security architecture must therefore combine cryptography, access control, transaction validation, audit logs, and operational controls.

X. THREAT MODEL AND SECURITY ANALYSIS

Finally, future work should evaluate the complete voter experience. A secure protocol that is difficult to understand can increase operational error and reduce trust. The project can therefore include accessible interface design, clear transaction status messages, recovery procedures, and an independent audit dashboard. The final research contribution should report not only successful blockchain transactions but also failed attacks, usability observations, limitations, and the assumptions under which each security property holds.

A fourth gap concerns formal security verification. Smart contracts that control election state should satisfy explicit invariants: an ineligible voter can never obtain a successful ballot; a voting credential cannot be consumed twice; voting is impossible outside the configured window; unauthorized administrators cannot modify a closed election; and a published result cannot be replaced without detection. These properties can be expressed as formal assertions and checked using automated testing or formal verification tools.

A third gap is scalable deployment. Recording every operation directly on a public blockchain may create cost

and latency problems. Future work can investigate Layer-2 rollups, batching, off-chain commitments, and permissioned consensus. The design should preserve a publicly verifiable commitment to the election state even when detailed computation occurs outside the base blockchain. Performance experiments should compare transaction throughput, confirmation delay, storage growth, and cost under increasing simulated voter loads.

A second gap is independently verifiable privacy-preserving tallying. The simplest prototype counts visible candidate transactions, which is easy to understand but unsuitable when ballot secrecy is required. A stronger design can encrypt each ballot and use threshold decryption so that no single administrator can decrypt the complete election. A zero-knowledge proof can accompany the tally to demonstrate that only valid ballots were included. This creates a verifiable chain from eligibility to final result without publishing individual choices.

A first research gap is the separation of identity from voting action. A voter must prove eligibility, but the proof should not become a permanent public identifier for the ballot. Future versions of the project can investigate anonymous credentials in which a trusted registration authority issues a cryptographic credential that proves eligibility without exposing the underlying identity. The blockchain can then verify that the credential has not previously been spent while the public ledger remains unable to directly map a ballot to a voter.

The literature indicates that the next generation of blockchain-based e-voting systems should move from simple immutable vote ledgers toward complete end-to-end verifiable protocols. Many prototypes demonstrate that a smart contract can accept a ballot and prevent duplicate submissions, but fewer systems provide strong evidence for ballot secrecy, coercion resistance, independent tally verification, accessibility, and large-scale performance at the same time. This gap is important for the proposed project because the objective should be to demonstrate a coherent security architecture rather than to equate blockchain immutability with complete election security.

XV. RESEARCH GAPS AND FUTURE WORK

The final contribution is a research roadmap. The prototype establishes a foundation on which more advanced privacy and scalability mechanisms can be tested. Future versions can replace the simplified ballot mechanism with encrypted or zero-knowledge voting, replace a public test network with a permissioned consortium network, or introduce Layer-2 settlement. In this way, the project can evolve from a classroom demonstration into a structured research platform.

The fourth contribution is a measurable experimental framework. Rather than evaluating the project only through screenshots, the implementation can report duplicate-vote rejection, invalid-transaction rejection, tally correctness, transaction confirmation time, and basic resource consumption. These metrics provide evidence for the security and performance claims made in the final report.

The third contribution is audit-oriented design. The project can provide an administrator interface for election configuration and an auditor interface for checking transaction counts, election status, and result consistency. This demonstrates that transparency does not necessarily mean exposing confidential ballot information. The distinction between public audit evidence and private voting content should remain a design principle throughout implementation.

The second contribution is the explicit separation of voter eligibility from ballot content. This separation is important because a secure election must know who is allowed to vote without publishing how each person voted. Even if the initial implementation uses a simplified credential model, the architecture provides a clear extension point for anonymous credentials, zero-knowledge proofs, or decentralized identity in later versions.

The proposed Secure E-Voting System Using Blockchain contributes primarily as an academic prototype that integrates several security concepts into one demonstrable workflow. Its first contribution is the use of blockchain as a tamper-evident election record. Instead of relying entirely on a centralized database, the system distributes the relevant election state across the selected blockchain network and uses smart-contract rules to make important transitions deterministic.

XVII. EXPECTED CONTRIBUTION OF THE PROPOSED PROJECT

This workflow also supports reproducibility. An auditor who receives the election configuration, public verification parameters, transaction identifiers, and final tally proof can repeat the verification process without trusting the original application server. Such independent verification is one of the strongest reasons to combine blockchain with cryptographic election protocols. The blockchain provides an ordered and tamper-evident record, while cryptographic proofs establish the validity of protected operations.

After the voting period closes, the contract transitions to a final state that prevents new ballots and restricts configuration changes. The tally service then processes only accepted ballots. An audit process can independently count accepted records and compare the result with the

published tally. If a mismatch occurs, the system should expose the inconsistency rather than automatically overwrite the ledger or database.

During ballot submission, the application creates a protected ballot representation. The protected representation may be an encrypted ballot, a commitment, or another cryptographic object appropriate to the selected protocol. The voter receives a confirmation identifier, but the identifier should not reveal the selected candidate. The blockchain records the commitment and election state, allowing an auditor to establish that a valid transaction exists without learning the private content of the ballot.

At the registration stage, the election authority establishes the set of eligible participants. The registration database or credential service should record only the information necessary to establish eligibility. When the voter enters the election, the application proves that the credential is valid. The blockchain then records a non-identifying state transition indicating that the credential is eligible. This approach is preferable to publishing a list of names and associated wallet addresses because it reduces unnecessary identity exposure.

End-to-end verifiability is a central objective for the proposed architecture because it changes the question from whether a server claims that an election is correct to whether independent participants can check the important stages of the election. The workflow begins with eligibility verification and continues through ballot creation, ballot acceptance, ledger confirmation, election closure, tally generation, and result publication. Each stage should expose sufficient evidence for verification while avoiding disclosure of the voter's identity or candidate choice.

XVI. END-TO-END VERIFIABILITY AND AUDIT WORKFLOW

ACKNOWLEDGMENT

The author expresses gratitude to the Department of Computer Science and Engineering (Data Science), Guru Nanak Institute of Technology, Nagpur, for academic support and project facilities. The author also acknowledges the researchers whose published studies provided the foundation for this review.

REFERENCES

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

[2] "E-Voting System Based on Blockchain Technology: A Survey," in Proc. 2021 Int. Conf. Information Technology (ICIT), 2021, doi: 10.1109/ICIT52682.2021.9491734.

[3] A. Benabdallah, A. Audras, L. Coudert, N. El Madhoun, and M. Badra, "Analysis of Blockchain Solutions for E-Voting: A Systematic Literature Review," IEEE Access, vol. 10, pp. 70746–70759, 2022, doi: 10.1109/ACCESS.2022.3187688.

[4] "E-Voting Meets Blockchain: A Survey," IEEE Access, vol. 11, pp. 23293–23308, 2023, doi: 10.1109/ACCESS.2023.3253682.

[5] M. Sharp, L. Njilla, C.-T. Huang, and T. Geng, "Blockchain-Based E-Voting Mechanisms: A Survey and a Proposal," Network, vol. 4, no. 4, pp. 426–442, 2024, doi: 10.3390/network4040021.

[6] H. O. Ohize et al., "Blockchain for Securing Electronic Voting Systems: A Survey of Architectures, Trends, Solutions, and Challenges," Cluster Computing, vol. 28, Art. no. 132, 2025, doi: 10.1007/s10586-024-04709-8.

[7] "Privacy-Preserving E-Voting System through Blockchain Technology," in Proc. IEEE Int. Conf. Technology, Research, and Innovation for Betterment of Society (TRIBES), 2021, doi: 10.1109/TRIBES52498.2021.9751618.

[8] S. S. and K. K. T. G., "E-Voting System Using Public Blockchain," in Proc. 2022 IEEE 2nd Mysore Sub Section Int. Conf. (MysuruCon), 2022, doi: 10.1109/MysuruCon55714.2022.9972479.

[9] N. H. R., G. P. M. S., S. B. G., D. Jain, et al., "E-Voting System Using Blockchain Technology," in Proc. 2022 4th Int. Conf. Advances in Computing, Communication Control and Networking (ICAC3N), 2022, doi: 10.1109/ICAC3N56670.2022.10074164.

[10] S. Tanwar, G. Sharma, and P. N. Bokoro, "Blockchain-Based Secure Voting Mechanism Underlying 5G Network: A Smart Contract Approach," IEEE Access, vol. 11, pp. 76537–76550, 2023, doi: 10.1109/ACCESS.2023.3297492.

[11] H. Baniata and G. Caluna, "BP-Vot: Blockchain-Based e-Voting Using Smart Contracts, Differential Privacy, and Self-Sovereign Identities," IEEE Access, vol. 13, pp. 46106–46123, 2025, doi: 10.1109/ACCESS.2025.3548404.

[12] A. Emami Neishabouri, "A Scalable Decentralized Privacy-Preserving E-Voting System Based on Zero-Knowledge Off-Chain Computations," Journal of

Information Security and Applications, vol. 79, Art. no. 103645, 2023, doi: 10.1016/j.jisa.2023.103645.

[13] "Comprehensive Analysis and Implementation Framework for Decentralized Voting System Using Ethereum Blockchain," in Proc. 2026 IEEE Int. Conf. on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI), 2026, doi: 10.1109/IATMSI68868.2026.11465385.

[14] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.