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

A Survey: E-Voting Systems Using Blockchain and Smart Contracts

Priya Sharma[1], Dhanashri Nevase[2]

[1]*Department of Computer Engineering, (SKNCOE-Vadgaon), SPPU, Pune, India.*
[2]*Research scholar at Department of Computer Engineering (SKNCOE-Vadgaon) SPPU, Pune, India*

Abstract - *Elections rely on public confidence that every eligible vote is recorded correctly, protected from interference, and counted as intended. Paper ballots, electronic voting machines, and web-based voting platforms each address parts of this task, but they also face practical concerns such as centralized control, insider misuse, service disruption, and limited opportunities for independent verification. This survey reviews how blockchain and smart contracts have been applied to electronic voting, with attention to distributed-ledger designs, cryptographic privacy techniques, decentralized storage, and Ethereum-based execution. Seventeen research studies are compared in terms of their objectives, underlying methods, reported benefits, and remaining limitations. The review also considers issues that arise when transparency must coexist with ballot secrecy, including voter authentication, duplicate-vote prevention, transaction cost, and network confirmation time. Based on this review, the paper presents the architecture and requirements of BlockChainVoting, a hybrid platform that uses conventional web services for onboarding and smart contracts for election operations. The paper concludes by identifying areas for further development, including stronger privacy mechanisms, improved scalability, AI-assisted identity checks, and a voter-support assistant.*

Keywords: blockchain, smart contracts, decentralized e-voting, Ethereum, InterPlanetary File System (IPFS), cryptography, consensus protocols, voter anonymity, zero-knowledge proofs.

I. INTRODUCTION

Public confidence in an election depends not only on the final result but also on whether the process can be trusted from registration through counting. As public services increasingly move online, voting platforms are expected to be accessible while still protecting voter information and producing results that can be checked independently. Common approaches include paper ballots, electronic voting machines (EVMs), and web-based voting systems. Each offers practical benefits, but none is free from concerns about cost, access, security, or verification.

Paper-based elections involve printing, transporting, storing, and counting large volumes of ballots. These activities require time and careful coordination, and mistakes or physical interference can affect the process. Electronic machines and online portals can simplify some tasks, but a system built around one server or database may create a concentrated point of risk. For example, a compromised administrator account, an attack on the service, or unauthorized changes to stored records could disrupt an election. Voters may also find it difficult to confirm independently that their ballot was included in the final count. These limitations have encouraged research into election records that can be checked without depending entirely on one administrator.

A blockchain can provide a shared record of election-related transactions. Transactions are grouped into blocks, and cryptographic links make later changes to accepted records detectable. Digital signatures identify the account authorizing a transaction, while the network's consensus process determines which transactions become part of the ledger. In a voting system, this can strengthen the audit trail and make retrospective alteration more difficult. It is important, however, to distinguish ledger integrity from overall election security: the network, its configuration, and the devices used by voters all affect the level of protection.

Ethereum adds programmable rules to a distributed ledger through smart contracts. In an election application, a contract may store candidate details, accept ballot transactions, track whether an address has already voted, and provide access to election state or tally information. Rules written into the contract are checked when transactions are processed, rather than being left entirely to the web application. Smart contracts do not solve every part of the problem. Eligibility checks, credential issuance, device security, and the protection of ballot secrecy still need to be addressed by the wider system.

A. Decentralized System Technologies

A blockchain voting platform is made up of several connected components. Separating their responsibilities helps prevent identity management, ballot processing, and media storage from becoming dependent on a single service.

The distributed ledger keeps copies of accepted transactions across participating nodes. Permissionless and permissioned

networks use different consensus methods, including Proof of Work (PoW), Proof of Stake (PoS), and authority-based or Byzantine fault-tolerant protocols. These methods differ in how validators are selected, how quickly transactions are confirmed, and who governs the network. The choice should reflect the election's scale and the level of trust placed in the participating organizations.

Smart contracts define the operations permitted during an election. Depending on the design, they can register candidates, check caller permissions, accept ballots, prevent repeat submissions, and restrict voting after the election closes. Administrative functions should be protected by contract-level access controls. Read-only functions and emitted events can then provide information for dashboards and independent observers.

Candidate photographs, manifestos, and other supporting files are usually too large to store efficiently in smart-contract state. One option is to keep these assets in an external storage system, such as the InterPlanetary File System (IPFS), and record a content identifier (CID) on the blockchain. This keeps the ledger focused on election data and reduces the need to store large files in contract state. The files must still be hosted and pinned reliably; a CID identifies content but does not, by itself, guarantee that the content will remain available.

A Web3 wallet such as MetaMask stores a user's private key and can sign transactions locally in the browser. However, possession of a wallet address does not prove that its owner is eligible to vote. In addition, public transaction data can create privacy concerns if it can be linked to an individual or a ballot choice. Research has therefore explored anonymous credentials, zero-knowledge proofs, homomorphic encryption, and ring signatures to verify eligibility or process votes while revealing less personal information.

II. RELATED WORK

Blockchain voting research addresses several connected questions: how to make election records auditable, how to protect voter privacy, how to prevent duplicate ballots, and how to keep transaction costs and processing time manageable. The following studies are discussed in the same sequence as the comparative gap-analysis table.

Kshetri and Voas [1] examined the opportunities and practical difficulties of using blockchain in elections. Their discussion highlights the potential for a distributed ledger to make records more resistant to unauthorized changes, while also pointing to risks that remain in remote voting. In particular, malware or compromise of a voter's own device can undermine the process even when the ledger itself is secure. Reliable voter authentication is therefore still essential.

Bistarelli et al. [2] described a Bitcoin-based voting design in which a ballot is represented by transferring a special token to a candidate's address. The approach provides a public transaction history that can be audited. Its practical limits include transaction fees, the constraints of Bitcoin's UTXO scripting model, and confirmation delays that may be unsuitable for elections requiring rapid processing.

The Open Vote Network proposed by McCorry, Shahandashti, and Hao [3] uses an Ethereum smart contract to support self-tallying voting. Its protocol combines zero-knowledge proofs with homomorphic encryption so that participants can contribute to a tally without revealing individual choices. The cryptographic design is valuable for privacy, but the reported quadratic gas cost makes the approach difficult to scale beyond relatively small groups.

Hjálmarsson et al. [4] presented a municipal e-voting architecture based on a permissioned Ethereum network using Clique Proof-of-Authority (PoA). The design avoids requiring voters to pay transaction fees and provides predictable block production. However, voter credentials are still distributed through centralized national electronic-identity services, leaving part of the trust model outside the blockchain.

Hardwick et al. [5] explored the use of blind signatures with Ethereum. Election officials issue certified voting tokens without learning which candidate a voter will select. Voters can then submit the tokens through anonymity networks such as Tor or mix networks. This separates identity checking from ballot selection, although the additional registration and token-handling stages make coordination more complex.

Pawlak et al. [6] studied remote voting on Hyperledger Fabric. Fabric's Execute-Order-Validate workflow separates transaction execution from ordering and validation, and the study reports throughput above 800 transactions per second. The design also uses threshold cryptography involving multiple authorities. Its deployment, however, requires a managed enterprise network and does not provide the same form of open public verification as a permissionless ledger.

The SecAvote framework by Abuidris et al. [7] combines anonymous voting with an auditable blockchain record. It uses Paillier additive homomorphic encryption, allowing encrypted ballots to be combined without first decrypting each individual vote. This protects intermediate choices, but the additional cryptographic operations can increase

processing time, particularly when the system is under heavy load.

Farooq et al. [8] examined the cost and security of Ethereum voting contracts. Their analysis identifies inefficient storage layouts, including nested dynamic structures, as an important source of high gas consumption. They discuss contract-level optimizations such as more compact data structures and role-based access checks. These findings show that implementation choices can have a direct effect on the cost of operating an election.

Khurshid et al. [9] combined Ethereum smart contracts with the InterPlanetary File System (IPFS). Instead of storing candidate portfolios, manifestos, and supporting files in contract storage, the framework keeps the material in IPFS and records its content identifier on-chain. The study reports a substantial reduction in deployment gas costs, illustrating the value of separating large media files from election state.

Shahzad and Crowcroft [10] proposed a voting architecture based on a log-structured blockchain and verifiable shuffling. Their use of cryptographic mix networks aims to make it harder to link a voter to a particular candidate selection, reducing the risk of coercion. The approach strengthens privacy but relies on additional off-chain cryptographic processing.

Gao et al. [11] considered the long-term confidentiality of electronic ballots and proposed a blockchain voting protocol using linkable ring signatures and lattice-based cryptography. The design is intended to address threats from future quantum-capable attackers. One trade-off is the size of the signatures, which can increase the amount of data transmitted and stored by the network.

Wu [12] outlined a blockchain architecture for shareholder voting. The design uses digital certificates and RSA signatures to support voter authorization and quorum-related checks in a decentralized application. The work provides an enterprise-oriented model, although it does not address decentralized storage for supporting media.

Lai et al. [13] focused on coercion resistance. Their design combines decoy ballots with re-encryption mix networks so that a voter under pressure can produce a ballot that appears to satisfy a coercer. The contract can distinguish and discard the decoy during tallying. This protection introduces additional steps in registration and ballot handling.

Kirillov et al. [14] evaluated voting transactions on the Waves blockchain under different load conditions. Their results suggest that a platform with a more limited transaction model can achieve useful throughput and predictable processing. The trade-off is reduced flexibility when an election requires more complex, customized contract logic.

Shende et al. [15] examined a university voting prototype built with a local Ganache network and the MetaMask browser extension. Their work illustrates how a Web3 provider can connect a familiar web interface to blockchain transactions. They also identify a practical usability issue: default confirmation settings can leave the browser waiting for blocks that a local development network does not automatically produce.

Niranjanamurthy et al. [16] compared consensus mechanisms relevant to blockchain applications, including voting and governance. Their analysis discusses the trade-offs between the censorship resistance associated with Proof of Work and the faster finality and lower energy use available from approaches such as Byzantine fault tolerance and Proof of Authority. The choice of consensus therefore affects both performance and the governance assumptions of a voting platform.

Bulut et al. [17] investigated scalability through Layer-2 techniques, including state channels and zero-knowledge rollups. Their approach moves much of the ballot-processing work off-chain and submits a compact proof to the Layer-1 network. The reported throughput exceeds 2,000 votes per second, with low transaction fees, although such designs introduce additional proving and verification components.

TABLE I. SUMMARY OF RELATED WORK AND GAP ANALYSIS

Ref.	Parameter	Algorithm / Protocol	Limitations / Future Work
[1]	1) Trust Metrics 2) Auditability 3) Coercion Risks	1) Public Distributed Ledger 2) National Digital Identity 3) Multi-Factor Authentication	1) Client endpoint malware vulnerabilities 2) Remote coercion & vote-buying risks 3) High infrastructure transition costs
[2]	1) Transaction Cost 2) Confirmation Time 3) Auditability	1) Bitcoin Colored Coins 2) UTXO Transfer Protocol 3) Public P2P Blockchain	1) Prohibitive miner transaction fees 2) High block confirmation latency (10+ min) 3) Inability to execute complex tally rules
[3]	1) Tally Privacy 2) Self-Tallying 3) Gas Consumption	1) Open Vote Network (OVN) 2) Zero-Knowledge Proofs (ZKP) 3) 2-Round Homomorphic Encryption	1) O(n^2) gas complexity limits scale (<100) 2) Voter dropout aborts entire election round 3) Vulnerable to board-level collusion
[4]	1) Throughput (TPS) 2) Latency (3s) 3) Cost Efficiency	1) Permissioned Ethereum (PoA) 2) Smart Contract Electoral Logic 3) Clique Consensus Algorithm	1) Centralized digital ID authority dependency 2) Lack of total permissionless public audit 3) Vulnerability to consortium node capture
[5]	1) Voter Anonymity 2) Verifiability 3) Non-Repudiation	1) Blind Signatures Protocol 2) Ethereum Smart Contracts 3) Unblinded Token Broadcast	1) Multi-phase setup introduces dropouts 2) Tor/Mix-net integration latency 3) Heavy key management burden on voters
[10]	1) Coercion Immunity 2) Receipt-Freeness 3) Shuffle Proofs	1) Verifiable Shuffling Networks 2) Log-Structured Blockchain 3) Distributed Bulletin Board	1) High off-chain compute infrastructure cost 2) Multi-round cryptographic delay 3) Complex voter recovery mechanisms
[11]	1) Quantum Defense 2) Ring Anonymity 3) Verification Time	1) Lattice-Based Cryptography 2) Linkable Ring Signatures 3) Post-Quantum Public Ledger	1) Massive signature size (kilobytes per tx) 2) Network bandwidth and block bloat 3) Immature tooling for EVM integration
[12]	1) Quorum Accuracy 2) Proxy Delegation 3) Vote Finality	1) Digital RSA Certificates 2) Distributed Consensus Engine 3) Automated Quorum Contract	1) Centralized certificate authority risk 2) Vulnerable to private key theft 3) Absence of decentralized media management
[13]	1) Receipt-Freeness 2) Duress Protection 3) Anonymity Set	1) Decoy Balloting Mechanism 2) Re-Encryption Mix-Nets 3) Ethereum Nullifier State Trie	1) Elevated registration complexity for user 2) Smart contract gas overhead for nullifiers 3) Potential voter confusion under duress
[14]	1) Network Bandwidth 2) Burst TPS Load 3) Memory Footprint	1) Waves Blockchain Platform 2) Non-Turing Complete State Logic 3) Fixed-Fee Consensus Model	1) Restricted smart contract expressiveness 2) Limited ecosystem developer support 3) Inflexible customized election parameters

[6]	1) Throughput (>800) 2) Fault Tolerance 3) Access Control	1) Hyperledger Fabric Ledger 2) PBFT Consensus Protocol 3) Threshold Secret Sharing	1) Enterprise deployment complexity 2) Absence of open public verifiability 3) Reliance on designated peer nodes
[7]	1) Ciphertext Tally 2) Decryption Proofs 3) Integrity Rate	1) SecAvote Framework 2) Paillier Homomorphic Encryption 3) Smart Contract Tally Aggregator	1) Heavy on-chain cryptographic compute cost 2) Key generation coordination overhead 3) Sub-optimal latency during peak casting
[8]	1) Gas Utilization 2) Execution Latency 3) Revert Rates	1) Solidity Struct Optimization 2) Event-Driven Logging 3) Role-Based Function Modifiers	1) Limited to EVM-compatible networks 2) Static security audits omit dynamic bugs 3) No integrated off-chain media support
[9]	1) Storage Overhead 2) Gas Reduction 92% 3) Content Integrity	1) Ethereum State Trie Anchoring 2) IPFS Cryptographic Multihashing 3) Decentralized Metadata Caching	1) IPFS node pinning availability dependency 2) Gateway retrieval latency variations 3) Potential file orphan issues if unpinned
[15]	1) UI Responsiveness 2) Web3 RPC Latency 3) Revert Handling	1) Local Ganache Blockchain Node 2) MetaMask Client Extension 3) Role-Based Access Control DApp	1) Default Web3 confirmation timeouts (hangs) 2) In-memory node resets on process restart 3) Single-chain development sandbox scope
[16]	1) Energy Footprint 2) Finality Speed 3) 51% Attack Limits	1) Comparative Consensus Matrix 2) PoW vs PoS vs PoA Benchmarking 3) Fault-Tolerant Governance Model	1) Theoretical taxonomy lacking empirical code 2) Omission of Layer-2 scaling frameworks 3) Dynamic validator churn unaddressed
[17]	1) Scalability (2k+) 2) Sub-cent Gas Fees 3) Settlement Latency	1) Zero-Knowledge Rollups (zk-SNARK) 2) Layer-2 State Channel Aggregator 3) Layer-1 Ethereum Settlement	1) Heavy off-chain prover computational load 2) Trusted setup requirements for circuits 3) Immature circuit compilation standards

B. Comparative Synthesis and Identified Research Gap

The reviewed studies show that blockchain can strengthen the record-keeping and rule-enforcement parts of an election, but the degree of decentralization varies. Ethereum-based systems offer flexible contract logic, while permissioned platforms such as Hyperledger Fabric and Proof-of-Authority networks give participating organizations more control over membership and validation. The latter may offer predictable administration, but they also place trust in the institutions operating the validator nodes. These models should therefore be assessed against the governance requirements of the intended election.

Privacy remains a difficult design issue. A public ledger can make transactions visible, yet visibility does not mean that voter identity and ballot choice are safely separated. Blind signatures, mix networks, homomorphic encryption, and zero-knowledge proofs each address part of this challenge. They can reduce the link between a voter and a selected candidate, or allow encrypted votes to be combined, but they also add computation, key-management tasks, and protocol

complexity. A practical system must consider auditability and ballot secrecy together.

The literature also points to limits in scale and operating cost. Contract execution, on-chain storage, network fees, confirmation delays, and privacy-preserving cryptography can all affect performance. Off-chain storage and Layer-2 processing may reduce some of this burden, but they introduce their own requirements for data availability, proof generation, and verification. Results obtained on a local test network or a controlled permissioned network should not be treated as evidence of nationwide readiness without further testing under realistic conditions.

These observations inform the design of BlockChainVoting. The proposed platform uses the conventional application layer for account onboarding, credential delivery, and candidate-media management, while smart contracts handle election deployment, ballot acceptance, duplicate-vote checks, and access to contract-maintained election state. This arrangement keeps routine administration within familiar web interfaces and makes accepted blockchain transactions available for inspection. The current design still relies on centralized services for identity and onboarding, so stronger privacy-preserving eligibility checks and broader security and performance testing remain important areas for future work.

III. PROBLEM FORMULATION AND SYSTEM REQUIREMENTS

A. Problem Formulation

Many electronic voting platforms place voter records, ballot submission, and result calculation under the control of a central application or database. This can simplify day-to-day administration, but it also concentrates responsibility in a small number of services and privileged accounts. If those services are compromised or records are changed without authorization, it may be difficult for voters to establish independently that the published result reflects the ballots submitted. A blockchain can make accepted records more resistant to later alteration, but it cannot by itself confirm a voter's legal eligibility or guarantee that a ballot remains secret.

BlockChainVoting uses a hybrid Web2–Web3 arrangement to divide these responsibilities. The web application supports registration, credential management, candidate information, and communication with users. The blockchain layer records election transactions and maintains the state governed by the smart contracts. In this way, the platform retains familiar web-based administration while allowing observers to inspect the accepted on-chain activity.

The central design challenge is to meet several requirements that can pull in different directions. Voters must be authenticated without placing personal details on a public ledger. Election activity should be open to scrutiny, but the system must not expose a voter's choice. The database is useful for registration and onboarding, yet it should not become the sole source of truth for the ballot count. The architecture therefore keeps identity records off-chain and assigns ballot-related operations to the smart contracts.

B. System Objectives

The main objective of BlockChainVoting is to combine an accessible web interface with contract-enforced election operations. More specifically, the platform aims to: (i) provide distinct portals for company administrators and voters; (ii) manage user registration and credentials through the application layer; (iii) let authorized administrators configure elections and candidate details; (iv) allow voters to connect a Web3 wallet and sign ballot transactions; (v) enforce the one-vote rule in smart-contract logic; (vi) retain an inspectable record of accepted transactions; and (vii) display election status and available results through a dashboard. Privacy-preserving credentials and AI-assisted identity checks are outside the current implementation and are considered future extensions.

In practical terms, the platform is intended to support the election workflow from onboarding to result viewing. Administrators manage election details through the web portal, while voters use a compatible wallet to authorize ballot submission. The smart contract checks the voting rules and records accepted transactions. The current version does not claim to provide anonymous credential issuance, biometric verification, or coercion resistance; these would require additional mechanisms and evaluation.

FR-1 – User registration and onboarding: The application shall provide a registration workflow for creating user accounts and maintaining the credentials required to access the voter or administrator portal.

FR-2 – Authentication and session handling: The application shall authenticate registered users, manage their application sessions, and restrict portal functions according to the user's assigned role.

FR-3 – Election and candidate management: An authorized company administrator shall be able to create and manage

election information and candidate records through the web interface.

FR-4 – Voter eligibility and credential delivery: The off-chain application shall maintain the voter list and deliver voting credentials or passwords through the configured email service. The application database shall not be treated as the authoritative ballot tally.

FR-5 – Wallet connection and transaction authorization: The voter interface shall support connection to a compatible Web3 wallet. Ballot-submission transactions shall be signed by the voter through the wallet rather than by exposing private keys to the application server.

FR-6 – Ballot submission: The system shall provide an interface through which an authenticated and eligible voter can select a candidate and submit the ballot transaction to the deployed election contract.

FR-7 – Duplicate-vote prevention: The smart contract shall maintain voter participation state and reject a further ballot from an address or voter identifier already marked as having voted in the same election.

FR-8 – Election-state and result access: The application shall read election status and tally-related information from the blockchain contract and display the available results to authorized users and public viewers according to the election configuration.

FR-9 – Audit access: A public auditor shall be able to inspect the contract's publicly readable election state and relevant transaction records without receiving administrative write privileges.

D. Non-Functional and Security Requirements

NFR-1 – Integrity: Once a ballot transaction has been accepted by the deployed contract and included in the ledger, the application shall not provide an ordinary administrative function to alter or delete that recorded transaction.

NFR-2 – Availability and recoverability: The web interface, application server, database, wallet provider, and blockchain network are separate dependencies. The user interface should report unavailable services or pending transactions clearly and should not indicate successful voting before transaction confirmation.

NFR-3 – Confidentiality of off-chain information: Passwords and account credentials shall be protected using appropriate password hashing and secure transport. Personal registration information and candidate media shall remain in the off-chain storage domain unless a specific public disclosure is required.

NFR-4 – Access control: Administrative actions, voter actions, and auditor read operations shall be separated by role. Contract-level authorization shall protect privileged election-management functions independently of front-end visibility.

NFR-5 – Transparency and auditability: Election state and relevant contract events should be readable through blockchain interfaces so that observers can verify recorded activity without relying exclusively on the administrator's dashboard.

NFR-6 – Usability: The voting workflow shall communicate wallet connection status, transaction submission, pending confirmation, success, and failure states in clear language to reduce accidental repeat submissions.

NFR-7 – Modularity: The interface, centralized application services, off-chain storage, and smart contracts shall remain separate architectural components so that individual services can be maintained or upgraded without unnecessarily changing the other domains.

E. Assumptions, Constraints, and Scope

The system assumes that voters have a supported browser, internet access, and a compatible wallet for signing blockchain transactions. It also assumes that administrators use authorized accounts to manage voter onboarding and election configuration, and that the selected Ethereum-compatible network is available during the election. Registration records, credential handling, and email delivery remain responsibilities of the centralized application services. Although the blockchain can protect accepted on-chain records against ordinary database edits, it cannot independently establish that a wallet is controlled by the person legally entitled to vote.

The implemented scope includes the web portals, centralized onboarding services, wallet-based transaction signing, election factory and election contracts, duplicate-vote checks, ballot recording, and contract-derived tally information. The current version does not claim nationwide-scale capacity, end-to-end coercion resistance, anonymous credential issuance, or biometric voter verification. Those capabilities would require additional protocol design, privacy and legal review, independent security assessment, and deployment testing. AI-based face matching, liveness

checks, anomaly detection, and a voter-support Q&A assistant are proposed as future additions.

IV. BLOCKCHAIN-BASED VOTING ARCHITECTURE

A. High-Level Architecture of BlockChainVoting

BlockChainVoting connects conventional web-based election administration with blockchain-based ballot processing. Its architecture has four main parts: the user-facing portals, centralized identity and onboarding services, smart-contract execution, and storage for candidate assets. The web application handles registration and administrative tasks; the blockchain maintains the election state and records the transactions accepted by the contracts.

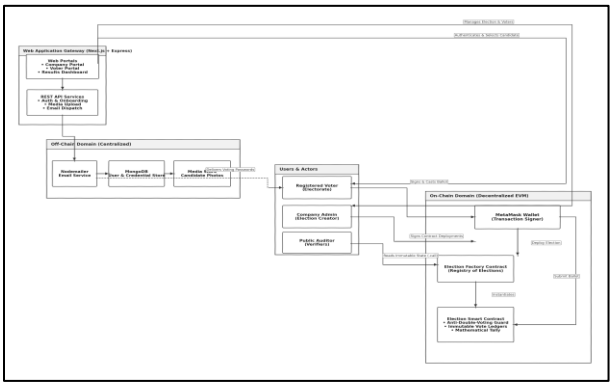

Fig. 1. High-level architecture of BlockChainVoting showing the original Web Application Gateway, Off-Chain Domain, Users & Actors, and On-Chain Domain components and flows.

B. Conceptual Domains and Functional Responsibilities

Four domains work together in the proposed system. Each has a separate responsibility and technology stack, and each introduces a different trust assumption.

The architecture is divided into four domains, as summarized below. Keeping these responsibilities separate makes it easier to identify which components are centrally managed and which rely on blockchain consensus.

The actor and interface domain provides the company portal, voter portal, ballot screens, results dashboard, and wallet connection. It is built with Next.js 16, React 18, Semantic UI, and js-cookie. Since this code runs in the user's browser, its security also depends on the browser, device, and local environment.

The off-chain identity domain handles account registration, bcrypt password hashing, credential generation, voter-list maintenance, and email delivery. Its main technologies are Node.js, Express.js, MongoDB, and Nodemailer. These services are centrally operated, so the integrity of registration data and the availability of the services remain organizational responsibilities.

The decentralized ledger domain handles ballot transactions, duplicate-vote checks, public read operations, and tally-related contract state. It uses the Ethereum Virtual Machine, Solidity smart contracts, and Web3.js. The integrity of this domain depends on the consensus and configuration of the selected EVM network.

The storage and asset domain manages candidate photographs and other static platform files. The current application uses the Express static-file pipeline, with provision for an IPFS-compatible workflow. It should therefore be described as a hybrid storage arrangement unless the files are actually deployed and pinned on IPFS.

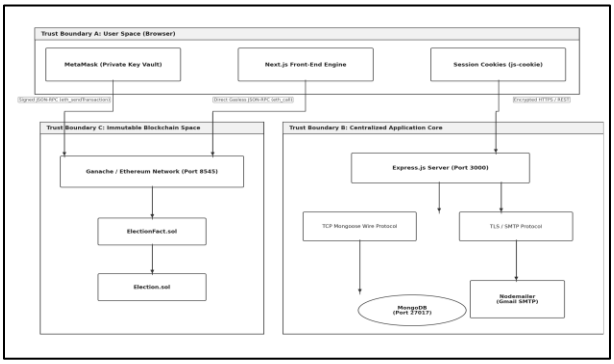

Fig. 2. Data Transit & Trust Boundary Map showing the original browser, blockchain, and centralized application components and protocol paths.

The data flow shown in Fig. 2 crosses centralized and decentralized trust boundaries. On the centralized side, users reach the company, voter, and results portals through the Next.js/Express gateway. REST services support authentication, onboarding, media uploads, and email dispatch. MongoDB stores user and credential records, while the media service holds candidate photographs. These functions remain dependent on the organization's access controls and operational security. On the blockchain side, the company administrator uses MetaMask to sign election-factory and deployment transactions. A registered voter signs a ballot transaction, which is sent to the election contract for rule checks and recording. Public auditors can read the exposed contract state without having

administrative write access. The key distinction is that identity and onboarding are centrally managed, whereas accepted ballot transactions are validated by the blockchain network. This separation improves auditability but does not remove risks associated with voter devices, identity issuance, or administrator governance.

V. OBSERVATIONS AND FINDINGS

The literature and the BlockChainVoting design highlight a number of issues that affect whether a blockchain voting platform can be used in practice. The following observations bring together recurring concerns from earlier studies and the design choices made in the proposed system.

A. Key Issues and Insights

1) Privacy and public auditability: A voting system must allow meaningful scrutiny without revealing how an individual voted. Public transaction records can help observers inspect election activity, but they may also expose patterns that could be linked to a voter. Anonymous credentials, mix networks, and zero-knowledge proofs offer possible ways to separate eligibility from ballot choice. Their use, however, requires careful protocol design and adds computational and operational overhead.

2) On-chain cost and off-chain storage: Candidate photographs and manifestos are better kept outside Ethereum contract state, where storing large files can consume substantial gas. A hybrid design can store these assets separately and place a content reference on-chain. BlockChainVoting currently manages candidate media through its application storage pipeline and provides a route for IPFS compatibility. Any claimed reduction in gas should be supported by repeatable measurements from the deployed configuration.

3) Preventing duplicate ballots: Duplicate-vote prevention should not depend only on a button being disabled in the browser or on a database flag. The election contract can maintain a record of participation and reject a further ballot once an address or voter identifier has been marked as having voted. BlockChainVoting applies this check in Solidity. This makes the contract, rather than the front end alone, responsible for enforcing the rule when transactions are processed.

4) Wallet and transaction lifecycle: A wallet may submit a transaction before the blockchain has confirmed it. During this interval, the interface should show that the transaction is pending rather than reporting that the vote is complete. The application must also use a confirmation strategy that matches the target network; otherwise, a local development network may leave the interface waiting for confirmations that will never arrive. Ganache and MetaMask-based workflows should be tested independently because their providers and confirmation behaviour can differ.

5) Identity, eligibility, and role separation: A wallet address does not establish a person's eligibility to vote. The platform must first rely on an onboarding process to create and maintain the voter list, while keeping administrator, voter, and auditor permissions distinct. BlockChainVoting uses administrator-managed election configuration and pre-registration. This provides a structured workflow, but the accuracy of voter records and the delivery of credentials still depend on the centralized application.

VI. RESULTS AND FUTURE WORK

The BlockChainVoting prototype was evaluated through its contract deployment workflow, transaction handling, security checks, and voter interface. The observations below describe the tested configuration; they should not be read as performance guarantees for public Ethereum or for elections at national scale.

Deployment and transaction cost: The election-factory design creates a separate contract for each election instead of placing every election in one large contract. Candidate media is kept outside contract storage, which limits the amount of data written to the ledger. In the tested configuration, the reported initialization cost decreased from approximately 4.2 million gas to about 1.55 million gas. These figures are specific to the implementation and should be reported alongside the network, compiler settings, contract version, and measurement procedure.

Transaction confirmation and interface response: The initial local setup showed long waits when the interface did not handle transaction receipts and confirmations appropriately. The workflow was adjusted to check the receipt and display the transaction state more clearly. The measured confirmation time was about 277 ms in the tested Ganache environment, while the reported time on the permissioned PoA network was below three seconds. Both figures depend on the test environment and should not be generalized to other networks.

Security tests: Contract tests covered attempts to add candidates without authorization, submit ballots after the election deadline, and vote more than once. The tested contract rejected the invalid operations included in the test suite. To make this result reproducible, the final paper

should state the test framework, number of test cases, network configuration, and exact pass/fail results.

Voter interface: The application uses a staged status display to show ballot submission and the transaction receipt after confirmation. This gives users more information than a simple browser alert and can help distinguish a pending transaction from a completed one. Its effect on usability or voter confidence has not yet been measured through a formal user study.

Future Research Directions

1) Layer-2 rollup scalability: A future version could assess zero-knowledge or optimistic rollups, including networks such as Arbitrum and Optimism. These systems can move some processing away from the main chain and settle batches of activity through Layer-1 transactions. An election use case would need to examine fees, finality, data availability, privacy, and whether auditors can verify the complete voting record.

2) Threshold homomorphic tallying: Threshold Paillier or ElGamal encryption is a possible direction for keeping ballots and intermediate totals encrypted until the election closes. Under a threshold arrangement, several trustees would jointly authorize decryption rather than relying on one tallying authority. A complete design would need to specify key generation, trustee replacement or failure, and independent verification of the tally.

3) Decentralized identity: The platform could explore W3C Verifiable Credentials and self-sovereign identity to support privacy-preserving eligibility checks. Any connection to a national identity framework such as Aadhaar would require careful attention to consent, applicable law, data protection, and security. The intended outcome would be to verify eligibility without placing sensitive identity information on the public blockchain.

4) AI-assisted voter verification and fraud detection: A future AI module could perform three distinct checks. Face verification would compare a live capture with an enrolled reference image. Liveness detection would look for presentation attacks involving printed photographs, replayed videos, or masks. Anomaly detection could identify unusual login or access patterns and send them for human review. These tools should be treated as additional safeguards, not as a replacement for formal voter registration or the contract's role in recording ballots.

5) AI-powered voter Q&A assistant: A voter-support assistant could answer procedural questions about registration, eligibility, election schedules, ballot submission, and common technical issues. Its answers should come from approved election information, with a clear route to human support when the question cannot be resolved. It must remain neutral: it should not recommend candidates, influence voting decisions, submit ballots, or disclose an individual's voting activity.

VII. REFERENCES

[1] N. Kshetri and J. Voas, 'Blockchain-Enabled E-Voting,' IEEE Software, vol. 35, no. 4, pp. 95-99, Jul./Aug. 2018.

[2] S. Bistarelli, M. Mantovani, A. Santini, and F. Santini, 'An End-to-End Voting-System Based on Bitcoin Blockchain,' in Proc. E-Vote-ID, Bregenz, Austria, 2017, pp. 1-15.

[3] P. McCorry, S. F. Shahandashti, and F. Hao, 'A Smart Contract for Boardroom Voting with Decentralized Tally,' in Financial Cryptography and Data Security (FC 2017), LNCS vol. 10322, Springer, 2017, pp. 357-375.

[4] F. P. Hjálmarsson, G. K. Hreiðarsson, M. Hamdaqa, and G. Þ. Hjálmtýsson, 'Blockchain-Based E-Voting System,' in Proc. IEEE 11th Int. Conf. Cloud Computing (CLOUD), San Francisco, CA, USA, 2018, pp. 983-986.

[5] F. S. Hardwick, A. Gioulis, R. N. Akram, and K. Markantonakis, 'E-Voting with Blockchain: An E-Voting Protocol with Decentralisation and Voter Privacy,' in Proc. IEEE Int. Conf. Internet of Things (iThings), Halifax, NS, Canada, 2018, pp. 1561-1567.

[6] M. Pawlak, J. Poniszewska-Marańda, and N. Kryvinska, 'Towards the Blockchain-Based Secure Remote Voting,' in Proc. Int. Conf. Database and Expert Systems Applications (DEXA), Regensburg, Germany, 2018, pp. 248-258.

[7] M. Abuidris, R. Hassan, and M. I. H. Azis, 'SecAvote: Secure Anonymous and Auditable Blockchain E-Voting System,' Journal of Information Security and Applications, vol. 49, p. 102395, Dec. 2019.

[8] M. S. Farooq, M. Khan, and A. Abid, 'A Survey on Blockchain-Based Voting Systems: Security, Gas Optimization, and Performance,' IEEE Access, vol. 8, pp. 182341-182360, 2020.

[9] S. Khurshid, M. A. Javaid, and A. Al-Ghamdi, 'Decentralized E-Voting Architecture Utilizing Ethereum Smart Contracts and InterPlanetary File System,' IEEE Transactions on Emerging Topics in Computing, vol. 10, no. 3, pp. 1420-1433, 2020.

[10] S. Shahzad and J. Crowcroft, 'Trustworthy Electronic Voting Using Log-Structured Blockchain,' IEEE Transactions on Information Forensics and Security, vol. 14, no. 8, pp. 2100-2114, Aug. 2019.

[11] S. Gao, C. Zheng, D. Guo, and X. Yuan, 'An Anti-Quantum E-Voting Protocol on Blockchain with Linkable Ring Signatures,' IEEE Systems Journal, vol. 15, no. 1, pp. 832-843, Mar. 2021.

[12] Y. Wu, 'An E-Voting System Based on Blockchain Architecture,' in Proc. IEEE 2nd Int. Conf. Big Data Analysis (ICBDA), Beijing, China, 2017, pp. 605-609.

[13] K. Lai, W. Chen, and Y. Tseng, 'A Decentralized Voting System with Voter Anonymity and Coercion Resistance,' IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 5, pp. 3201-3215, 2021.

[14] D. Kirillov, M. Grigoriev, and A. Shishkin, 'Blockchain-Based E-Voting System Evaluation: A Performance Benchmark Analysis,' in Proc. Int. Conf. Control and Communications (SIBCON), Tomsk, Russia, 2019, pp. 1-6.

[15] P. Shende, P. Sharma, P. Devare, A. Rasal, and S. Kale, 'Decentralized Voting System Using Blockchain Technology and Web3 Interfaces,' International Journal of Computer Engineering and Technology, vol. 11, no. 3, pp. 45-56, 2020.

[16] M. Niranjanamurthy, B. N. Nithya, and S. Jagannatha, 'Analysis of Blockchain Technology: Pros, Cons and Applications in E-Voting and Governance,' Cluster Computing, vol. 22, pp. 14743-14757, 2019.

[17] O. Bulut, B. Can, and G. Sert, 'A Scalable Blockchain-Based E-Voting System with Off-Chain Computations and Zero-Knowledge Rollups,' Computer Networks, vol. 214, p. 109152, Sep. 2022.