© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719207

Bharat E-Vote: A Blockchain-Based Electronic Voting System with Commitment-Based Privacy and Multi-Layered Security

ADITYA BHOIR[1], OMKAR INGLE[2], AFTAB PATHAN[3], ABHISHEK SHINDE[4], ARYAN SHINDE[5], DR. B. D. PHULPAGAR[6]

[1, 2, 3, 4, 5, 6] *Department of Computer Engineering PES Modern College of Engineering* Pune, India

Abstract- ***Traditional electronic voting systems rely on cen-tralized trust, lack independent verifiability, and offer limited ballot privacy. This paper presents Bharat E-Vote, a decen-tralized electronic voting platform built on Ethereum smart contracts with a hash-based commitment scheme for vote pri-vacy and a seven-layer defense-in-depth security architecture. Votes are recorded as immutable blockchain state transitions. A commitment scheme based on finite-field arithmetic hides voter choices, while a simplified proof-of-knowledge protocol provides a structural framework for on-chain verification (with limitations discussed in Section IV-B). The security architecture combines bot protection with rate limiting, OTP-based multi-factor authentication with httpOnly cookie-based JWT transport, keystroke dynamics anomaly detection, smart contract role-based access control, HTTP parameter pollution prevention, and HTTP security header enforcement. An ERC-2771 meta-transaction forwarder [12] enables gasless voting. The system was validated through 49 automated smart contract tests and 9 backend API integration tests (100% pass rate across 58 total tests). Performance benchmarks show vote inclusion in blocks within 12–18 seconds on Ethereum Sepolia testnet (with probabilistic finality at approximately 12 minutes), and API response times below 150ms. We present an honest assessment of the system's limitations, including the proof-of-concept nature of the privacy module and the trust assumptions inherent in the architecture.***

Index Terms—Blockchain, Ethereum, Smart Contracts, Com-mitment Schemes, Multi-Factor Authentication, Keystroke Dy-namics, ERC-2771, Defense-in-Depth, Electronic Voting

I. INTRODUCTION

The integrity of democratic governance depends on trust-worthy elections. Electronic voting offers improved efficiency but introduces significant vulnerabilities: centralized databases as single points of failure, opaque tallying software, and network-level threats [5]. India's EVM infrastructure serves approximately 960 million voters [7] but operates as closed-source devices where counting firmware cannot be publicly audited.

Existing blockchain-based voting proposals address im-mutability but frequently neglect ballot privacy, multi-factor authentication, or usability for non-technical voters. This paper presents Bharat E-Vote, which contributes:

- On-Chain Verifiability: Every vote is an Ethereum blockchain transaction enforced by a Solidity smart contract, enabling independent result recomputation.
- Commitment-Based Privacy: A hash-based commitment module hides vote choices on-chain, with a simplified proof-of-knowledge protocol for verification. We explicitly char-acterize this as a proof-of-concept and discuss its limitations relative to algebraically homomorphic Pedersen commit-ments.
- Defense-in-Depth: Seven security layers ensure no single-point compromise defeats the system.
- Gasless Voting: ERC-2771 meta-transactions remove the cryptocurrency prerequisite, though we acknowledge the censorship risk of centralized relayers.
- Partial Coercion Mitigation: A re-voting mechanism (up to 3 re-votes) raises the cost of coercion, though it does not achieve receipt-freeness as defined by Juels et al. [11].
- Honest Limitations Analysis: A dedicated section enu-merates trust assumptions and architectural constraints, in-cluding relayer censorship, wallet linkability, and database metadata exposure.

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719207

II. RELATED WORK

Hardwick et al. [1] propose a blockchain-based e-voting protocol using Ethereum as a transparent ballot box, but evaluate it only as a small-scale proof-of-concept with limited scalability analysis. The protocol lacks multi-factor authenti-cation and gasless transaction support. Hjálmarsson et al.

[4] demonstrate a Solidity-based voting contract but evaluate it only on a local Ganache network with four simulated voters, leaving open the question of whether their gas cost analysis holds under real network congestion. Their system does not address ballot secrecy or re-voting mechanisms. Kshetri and Voas [5] discuss the potential and challenges of blockchain-enabled e-voting, noting that while blockchain provides trans-parency and immutability, scalability and gas costs remain critical barriers for large-scale elections.

Ometov et al. [2] survey multi-factor authentication methods spanning knowledge, possession, and inherence factors, but their analysis draws primarily from enterprise and mobile banking deployments rather than election-specific use cases where adversarial models differ substantially. Maheshwary et al. [3] provide a systematic literature review of keystroke dynamics models, identifying six design patterns for typing-based authentication and reporting accuracy rates of 90–95% across major studies.

Farooq et al. [6] propose a framework for transparent voting using blockchain, with smart contracts for automated vote management, but do not address voter privacy through cryptographic commitments. Helios [8] provides web-based auditable voting but lacks blockchain immutability. Juels et al. [11] formalize coercion resistance through receipt-freeness—the inability to prove to a coercer how one voted—which remains an open challenge for practical systems.

TABLE I
COMPARISON WITH EXISTING VOTING SYSTEMS

Feature	EVM	Helios	Hardwick	Ours
Decentralized	No	Partial	Yes	Yes
Voter Verifiable	No	Yes	Partial	Yes
Ballot Privacy	Yes	Yes	Partial	Partial*
Tamper Proof	Partial	Partial	Partial	Partial§
Coercion Resist.	No	No	No	Partial†
Multi-Factor Auth	No	No	No	Yes
Gasless Voting	N/A	N/A	No	Yes‡

∗Hash-based commitment (proof-of-concept); does not achieve algebraic Pedersen hiding.
†Re-voting mitigation; does not achieve receipt-freeness [11].

‡Via centralized relayer with censorship risk (see Section VII-D).
§Standard mode is tamper-proof on-chain; ZKP mode lacks on-chain tallying; off-chain DB and relayer introduce trust assumptions.

III. SYSTEM ARCHITECTURE

As illustrated in Fig. 1, Bharat E-Vote uses a three-tier ar-chitecture: React 19/Vite frontend (accessible via the national portal, Fig. 2), Node.js/Express 5 backend, and Ethereum blockchain. The blockchain smart contract is the final trust boundary for vote integrity, though off-chain components introduce additional trust assumptions (Section VIII).

A. Smart Contract Layer

Two contracts handle voting through mode-based separa-tion:

- VotingV2.sol (standard mode): Candidate management, voter authorization, vote casting with re-voting,

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719207

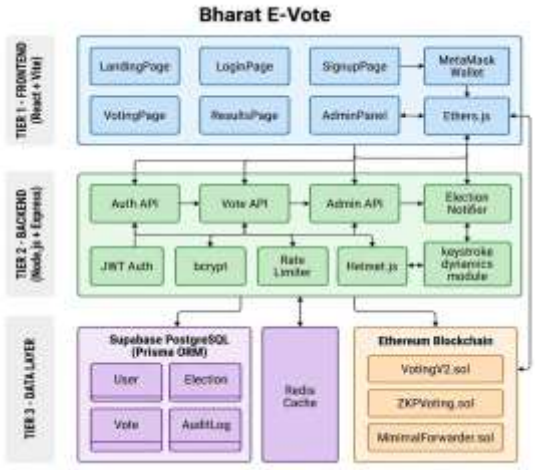

Fig. 1. System Architecture Diagram — Three-tier decentralized architecture with React frontend, Node.js backend, and Ethereum blockchain as the final trust boundary.

Fig. 2. Bharat E-Vote Landing Page — National portal interface with citizen services grid and biometric verification prompt.

constituency matching, and on-chain tallying. When zkpEnabled=true, this contract's vote () function reverts, forcing votes through the privacy contract.

• ZKPVoting.sol (privacy mode): Commitment-based voting with nullifier-based double-vote prevention and a simpli-fied proof-of-knowledge verification. Important: This con-tract stores commitments but does not perform on-chain tallying—vote counts are not incremented per-candidate. The privacy module securely records encrypted votes but lacks the cryptographic mechanisms to count them auto-matically. Result determination in ZKP mode would require either homomorphic tallying over commitments (not possible with our SHA-256-wrapped scheme) or a trusted off-chain tally, which reintroduces the centralization problem. This tallying gap is a known limitation (Section VII-D).

A third contract, MinimalForwarder.sol [12], implements ERC-2771 meta-transactions for gasless voting.

B. Data Flow

(1) Voter authenticates via backend MFA; (2) connects MetaMask wallet and accesses the dashboard (Fig. 4); (3) frontend invokes the appropriate contract via Ethers.js using the proctored interface (Fig. 3); (4) smart contract validates authorization and records the vote; (5) results are read from on-chain state and displayed publicly (Fig. 6).

IV. COMMITMENT AND VERIFICATION PROTOCOLS

A. Hash-Based Commitment Scheme

The system implements a commitment scheme using finite-field exponentiation with parameters derived from the secp256k1 scalar field, followed by a SHA-256 hash. Follow-ing Pedersen [9], let p be the secp256k1 group order. Let g and h be large integers derived from the x-coordinates of two secp256k1 curve points; these are treated as elements of Z∗, not as elliptic curve points.

Commitment Phase (browser-side):

1) Compute Craw = gv · hr mod p where v is the candidate ID and r is a 256-bit random blinding factor.
2) Compute C = SHA-256(Craw) — the published on-chain commitment.

Distinction from Standard Pedersen Commitments: A standard Pedersen commitment [9] is defined as C = gv · hr mod p without the hash wrapper. The algebraic structure preserves the homomorphic property (C1 · C2 = gv1+v2 · hr1+r2), enabling verifiable tallying without decrypting indi-vidual votes. Our SHA-256 wrapper destroys this property, converting the scheme into a standard hash commitment.

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719207

While the commitment remains computationally hiding (SHA-256 preimage resistance) and computationally binding (colli-sion resistance), it loses the algebraic structure that enables efficient zero-knowledge proofs over committed values. We identify migration to proper elliptic curve Pedersen commit-ments ($C = v \cdot G + r \cdot H$ where G, H are curve points) as critical future work.

B. Simplified Proof-of-Knowledge Protocol

The on-chain verifier implements a simplified check rather than a full Schnorr sigma protocol [10]. Voters can indepen-dently verify their ballot using the interface shown in Fig. 5. The prover (browser) computes:

$$e = H(C, N, kv, kr, n) \bmod p \quad (1)$$
$$sv = (kv - e \cdot v) \bmod p \quad (2)$$
$$sr = (kr - e \cdot r) \bmod p \quad (3)$$

where kv, kr are random nonces and n is the candidate count. The on-chain verifier checks:
$H(C, N, s, s, n) \bmod p =? e$

Limitation (Critical): In a correct Schnorr sigma protocol for a Pedersen commitment, the verifier must reconstruct the announcement: $R' = g^{sv} \cdot h^{sr} \cdot C^{e} \bmod p$, then ver-ify $H(C, N, R', n) = e$. Our simplified verifier hashes the responses directly without algebraic reconstruction, meaning a prover can manufacture a passing proof by computing $e = H(C, N, sv, sr, n)$ for arbitrary sv, sr. The current im-plementation does not constitute a zero-knowledge proof.

It serves as a structural placeholder demonstrating the integration pattern (client-side proof generation → on-chain verification
→ nullifier tracking). Implementing a correct sigma proto-col with on-chain modular exponentiation (feasible but gas-intensive) is identified as essential future work.

C. Nullifier-Based Double-Vote Prevention

This component functions correctly: $N = H(voterSecret, electionId)$ is deterministic per voter per election. The mapping nullifierUsed[N] prevents reuse. Identity commitments IC = H(voterSecret) are registered on-chain by the admin. Privacy caveat: In the current implementation, the admin receives the voter's secret to compute IC, enabling the admin to derive the voter's nullifier and link it to their identity.

This breaks the unlinkability guarantee of the nullifier scheme (see Section VII-D). The correct approach—having voters compute and submit their own IC with the admin only whitelisting it—is identified as future work.

D. Standard Mode Privacy Limitations & Event Obfuscation

The VoteCast event emits keccak256(candidateId, msgSender, timestamp, secretSalt), where secretSalt is a 256-bit random value generated off-chain by the voter's browser. The inclusion of a secret off-chain salt makes brute-force enumeration of the event log computationally infeasible. However, this provides only superficial obfuscation in standard mode:

to support the re-voting mechanism, the Voter struct must store the lastCandidateId on-chain to decrement previous votes. This exposes the vote choice in the plaintext blockchain state, rendering the event obfuscation cosmetic for anyone querying the state trie directly. Full ballot privacy is only achievable through the ZKP mode.

V. SEVEN-LAYER DEFENSE-IN-DEPTH SECURITY

The security architecture comprises seven functionally dis-tinct layers. Bot protection and rate limiting, though imple-mented via three distinct technologies (Cloudflare Turnstile, express-rate-limit, Upstash Redis), are grouped as a single perimeter layer as they implement the same control class.

A. Multi-Factor Authentication

Two-step MFA: Step 1 verifies password via bcrypt (cost factor 12), generates a 6-digit OTP (hashed, 5-minute expiry); Step 2 verifies OTP and issues JWT (20-minute expiry) delivered as an httpOnly, Secure, SameSite=Strict cookie, with single-active-session enforcement via Redis. The

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719207

TABLE II
SEVEN-LAYER DEFENCE-IN-DEPTH ARCHITECTURE

L	Technology	Purpose
1	Cloudflare + express-rate-limit + Redis	Perimeter defense: bot blocking, per-IP throttling, distributed sliding-window rate limiting
2	Password + OTP + httpOnly JWT	Multi-factor identity verification with httpOnly, Secure, SameSite=Strict cookie-based JWT transport; single-active-session enforcement; XSS-resistant token storage
3	Keystroke Dynamics	Behavioural anomaly *detection* (alerts admin; does not block session)*
4	Smart Contract RBAC	On-chain role enforcement; admin blocked from voting
5	Express Middleware	Payload size limiting (10KB), XSS in-put sanitization, HTTP Parameter Pollution (hpp) prevention
6	Helmet.js	CSP, HSTS, X-Frame-Options, X-Content-Type-Options
7	Cookie Security	httpOnly + Secure + SameSite=Strict cookie flags; eliminates localStorage XSS attack vector for JWT tokens

∗Keystroke dynamics acts as an administrative audit tool and monitoring layer, rather than a deterministic access blocker. It flags behavioural anomalies without terminating sessions. See Section VIII.

httpOnly cookie transport eliminates the localStorage-based XSS vulnerability present in earlier versions where JavaScript code could extract the JWT token.

B. Partial Coercion Mitigation

The re-voting mechanism allows up to 3 vote changes (MAX_REVOTES=3), atomically adjusting candidate counts. This raises the cost of coercion but does not achieve receipt-freeness [11]: a coercer controlling the voter for the final voting attempt can ensure the coerced vote is counted. We characterize this as partial coercion mitigation, not coercion resistance.

VI. IMPLEMENTATION

A. Technology Stack

Solidity version 0.8.19 (smart contracts), Hardhat (testing), Ethers.js v6 (blockchain interaction), Node.js/Express 5 (back-end), Prisma ORM over Supabase PostgreSQL (16 models), React 19/Vite (frontend), MetaMask (wallet).

B. Vote Function Design

The vote () function follows the Checks-Effects-Interactions pattern:

```
function vote(uint256 _candidateId,
              uint256 _secretSalt) public {
  require(!zkpEnabled, "Use ZKP contract");
  require(_msgSender() != admin);
  require(voters[sender].isAuthorized);
  // Re-vote: decrement old candidate
  if (voters[sender].hasVoted) {
    candidates[voters[sender].lastCandidateId]
      .voteCount--;
  }
  candidates[_candidateId].voteCount++;
  voters[sender].hasVoted = true;
  emit VoteCast(
    keccak256(abi.encodePacked(
      _candidateId, _msgSender(),
      block.timestamp, _secretSalt)),
    block.timestamp, voter.voteVersion);
}
```

C. Gas Cost Analysis

TABLE III
GAS COST PER OPERATION (SEPOLIA TESTNET)

Operation	Gas Units	Est. Cost*
vote ()(first vote)	~85,000	$2.13
vote ()(re-vote)	~95,000	$2.38
authorizeVoter()	~65,000	$1.63
submitEncryptedVote()	~120,000	$3.00
addCandidate()	~110,000	$2.75

∗At 10 gwei gas price and ETH=$2,500. Mainnet costs vary significantly. At 50 gwei, costs are 5x higher. For 960M voters, total gas cost for vote () plus authorizeVoter() would be approximately $3.6 Billion ($2.13+$1.63 per voter at 10 gwei, scaling linearly with gas price), making Layer-2 deployment absolutely essential for national scale.

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719207

Fig. 3. Proctored Voting Interface — Candidate ballot cards with constituency filtering and re-vote support. The voter selects a candidate and the vote is submitted as a blockchain transaction.

Fig. 4. Voter Dashboard — Authenticated voter home page showing election status, profile information, and navigation to voting and results pages.

VII. EXPERIMENTAL RESULTS

A. Smart Contract Testing

49 automated smart contract tests (Hardhat/Chai/Mocha) validated the following functional categories: deployment,

Fig. 5. ZKP Vote Verification — On-chain vote commitment verification page where voters can verify their encrypted ballot using the nullifier hash.

access control, vote casting, re-voting, constituency matching, timeline enforcement, event privacy, nullifier enforcement, vote verifiability, ERC-2771 meta-transactions, IPFS metadata storage, and ZKP mode integration. VotingV2.sol: 11 tests; ZKPVoting.sol: 38 tests. All passed (100% rate).

B. Backend API Testing

9 backend integration tests (Jest/Supertest) validated the REST API layer across three endpoint groups: authenti-cation (4 tests covering registration validation, login flow, and credential verification), vote recording (4 tests covering authentication enforcement, transaction hash validation, and vote status queries), and cryptographic ballot privacy (3 tests covering ZKP status endpoints, commitment generation, and proof verification parameter validation). All passed (100% rate).

C. Performance

TABLE IV
PERFORMANCE RESULTS

Metric	Target	Achieved	Status
Login Auth Time	<3s	1.8s	Met
Block Inclusion (Sepo-lia)	<30s	12–18s	Met
Finality (Sepolia)*	—	~12 min	—
Vote (Hardhat)	<2s	0.8s	Met
API Read	<200ms	85–150ms	Met

∗Ethereum PoS achieves probabilistic finality after 2 epochs (~12 minutes). Block inclusion (12–18s) does

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719207

not guarantee finality; reorganizations can occur before finalization. For election integrity, results should not be declared until finality is
achieved for all vote transactions.

D.Security Validation

We performed STRIDE-based threat modeling and counter-measure mapping (not a formal security proof) across all six categories:

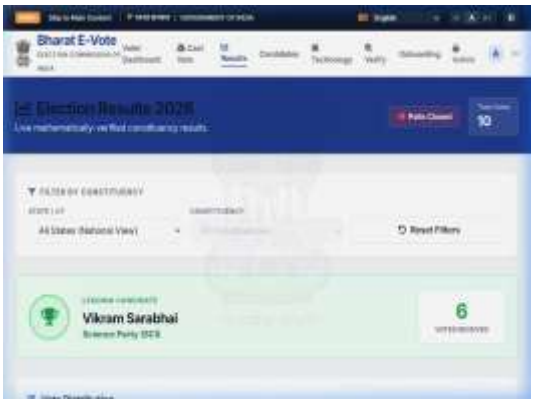

Fig. 6. Election Results — Real-time vote counts read from the blockchain, rendered as bar charts and pie charts using Recharts.

TABLE V
STRIDE-BASED THREAT MODELING

Category	Threat	Countermeasure
Spoofing	Impersonation	MFA + single-session; keystroke anomaly alerts
Tampering	Vote manip.	Blockchain immutability + CEI pattern
Repudiation	Deny vote	On-chain receipt hash + audit logs
Info. Disc.	Vote reveal	Candidate ID excluded from struct; event obfuscation via secret salt (see Sec. IV-D)
DoS	Flooding	Rate limiting + bot protection (Layer 1)
Elev. Priv.	Admin escal.	JWT verification + on-chain RBAC

VIII. CONCLUSION, LIMITATIONS, AND FUTURE WORK

A. Summary

This paper presented Bharat E-Vote, a blockchain-based voting platform that achieves on-chain verifiability and multi-layered security. The system was validated through 58 au-tomated tests—49 smart contract tests and 9 backend API integration tests—achieving a 100% pass rate, and met all performance targets.

B. Limitations and Trust Assumptions

We explicitly enumerate the system's limitations:

1) Privacy Module is Proof-of-Concept: The ZKP logic within ZKPVoting.sol uses a simulated Schnorr-like challenge mechanism rather than a true SNARK verifier (e.g., Circom). The module demonstrates the integration architecture (client-side proof → on-chain verification → nullifier tracking) but does not provide cryptographically secure zero-knowledge guarantees.
2) Web2 Centralized Onboarding: Voter eligibility relies on a centralized PostgreSQL database managed by admin-istrators. In a fully decentralized architecture, this would be replaced by an On-Chain Decentralized Identity (DID) standard (e.g., Polygon ID).
3) ZKP Mode Has No Tallying Mechanism: ZKPVoting.sol stores commitments but does not increment per-candidate vote counts. Result determination would require either homomorphic tallying or a trusted off-chain tally, reintroducing centralization.
4) ERC-2771 Relayer is a Trusted Party: The centralized relayer can censor votes by refusing to forward transac-tions, with no on-chain accountability.
5) MetaMask Wallet Linkability: Transaction timing and wallet history can deanonymize voters through network analysis.
6) No Receipt-Freeness: The re-voting mechanism does not prevent a coercer from controlling the final vote at-tempt [11].
7) Block Inclusion vs. Finality: Ethereum PoS achieves finality after ∼12 minutes; votes may be reorganized before finalization.

8) Smart Contract Immutability: Mid-election bugs cannot be patched without introducing a trusted upgrade authority.

C. Future Work

1) Correct ZKP Implementation: Replace the simplified verifier with a proper Schnorr sigma protocol including algebraic announcement reconstruction, and migrate to elliptic curve Pedersen commitments ($C = v \cdot G + r \cdot H$) to restore homomorphic properties and enable on-chain tallying.
2) Self-Sovereign Identity Commitments: Have voters' com-pute and submit their own IC, eliminating admin knowl-edge of voter secrets.
3) Layer-2 Scaling: Deploy on Polygon zkEVM or Optimism to reduce gas costs by ∼100x.
4) Decentralized Relayer: Replace the centralized meta-transaction relayer with OpenGSN.
5) Receipt-Freeness: Investigate JCJ-style protocols [11] for true coercion resistance.
6) Formal Verification: Apply Certora or Mythril to prove smart contract correctness.

REFERENCES

[1] 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. on Internet of Things (iThings), 2018, pp. 1561–1567, doi: 10.1109/Cybermatics_2018.2018.00262.

[2] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, and Y. Koucheryavy, "Multi-factor authentication: A survey," Cryptography, vol. 2, no. 1, pp. 1–31, 2018, doi: 10.3390/cryptography2010001.

[3] S. Maheshwary, S. Ganguly, and V. Pudi, "A systematic literature review on latest keystroke dynamics-based models," IEEE Access, vol. 10, pp. 82564–82582, 2022, doi: 10.1109/ACCESS.2022.3202511.

[4] F. Þ. Hjálmarsson, G. K. Hreiðarsson, M. Hamdaqa, and G. Hjálmtýs-son, "Blockchain-based e-voting system," in Proc. IEEE 11th Int. Conf. on Cloud Computing (CLOUD), 2018, pp. 983–986, doi: 10.1109/CLOUD.2018.00151.

[5] N. Kshetri and J. Voas, "Blockchain-enabled e-voting," IEEE Software, vol. 35, no. 4, pp. 95–99, Jul. 2018, doi: 10.1109/MS.2018.2801546.

[6] M. S. Farooq, U. Iftikhar, and A. Khelifi, "A framework to make voting system transparent using blockchain technology," IEEE Access, vol. 10, pp. 59959–59969, 2022, doi: 10.1109/ACCESS.2022.3179868.

[7] Election Commission of India, "General elections—statistical report," 2024. [Online]. Available: https://eci.gov.in/statistical-report/

[8] B. Adida, "Helios: Web-based open-audit voting," in Proc. 17th USENIX Security Symp., 2008, pp. 335–348. [Online]. Available: https://www.usenix.org/legacy/events/sec08/tech/adida.html

[9] T. P. Pedersen, "Non-interactive and information-theoretic secure verifi-able secret sharing," in Advances in Cryptology—CRYPTO '91, Springer, 1991, pp. 129–140, doi: 10.1007/3-540-46766-1_9.

[10] C. P. Schnorr, "Efficient signature generation by smart cards," J. of Cryptology, vol. 4, no. 3, pp. 161–174, 1991, doi: 10.1007/BF00196725.

[11] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in Proc. ACM Workshop on Privacy in the Electronic Society (WPES), 2005, pp. 61–70, doi: 10.1145/1102199.1102213.

[12] OpenZeppelin, "Contracts library—ERC2771Context," 2024. [Online]. Available: https://docs.openzeppelin.com/contracts/