SecureVote: A Decentralized Electronic Voting System Using Blockchain Technology with Zero-Knowledge Proofs

SecureVote: A Decentralized Electronic Voting System Using Blockchain Technology with Zero-Knowledge Proofs

Shikha Vishwakarma[1], Gaurav Vishwakarma[2]

R&D Department IDC INDIA INNOVATION DEVELOPMENT COUNCIL

Author Note

Dr. Shikha Vishwakarma, Email: dr.shikha@idcijst.com, https://www.idcindia.net/our-team

Dr. Gaurav Vishwakarma, dr.gaurav@idcindia.net, https://www.idcindia.net/our-team

We have no conflict of interest to disclose.

Correspondence concerning this article should be addressed to name, institution, and postal address.

Email: email address

Journal Name: IDCIJST- Innovation Development Council's International Journal of Science and Technology | Volume: 01 | Issue Number: 01 | Month: April–June | Publication Year: 2026

SecureVote: A Decentralized Electronic Voting System Using Blockchain Technology with Zero-Knowledge Proofs

Abstract

Traditional electronic voting systems often face central vulnerabilities, such as single points of failure, risks of data manipulation, and issues with transparency and voter anonymity. To address these problems, this paper presents SecureVote, a decentralized voting system built on permissioned blockchain technology. It employs advanced cryptographic techniques like homomorphic encryption and Zero-Knowledge Proofs (ZKPs) to guarantee complete ballot secrecy and end-to-end verifiability without relying on a central trusted authority. The system uses smart contracts to perform privacy-preserving tallying, calculating results directly from encrypted data without revealing individual votes. Tested on a Hyperledger Fabric network with 50 validator nodes, Securevote achieved a high throughput of 2,400 votes per second and final results within seconds. Compared to existing systems, SecureVote offers a better mix of NIST-compliant security, scalable architecture, and immutable audit trails.

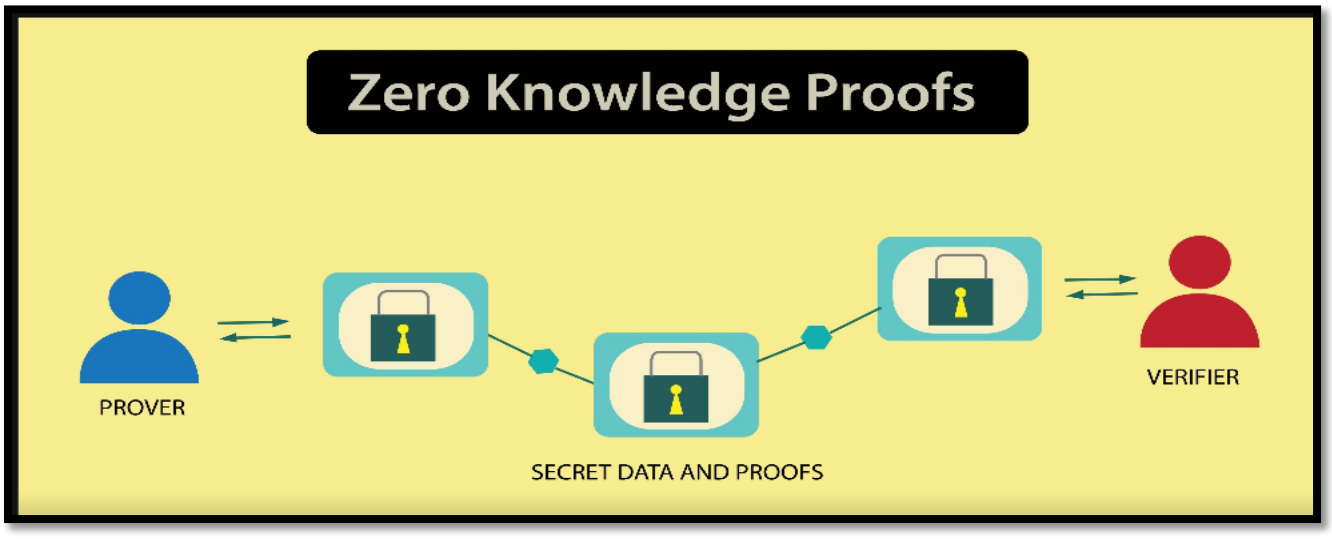
Keywords—*Blockchain, electronic voting, zero-knowledge proof, homomorphic encryption, smart contracts, hyperledger fabric, distributed ledger, election security.*

SecureVote: A Decentralized Electronic Voting System Using Blockchain Technology with Zero-Knowledge Proofs

I. INTRODUCTION

Modern governance depends on trustworthy democratic elections. Although digital transformation introduces e-voting as an easy, affordable, and quick alternative to paper ballots, widespread adoption is slowed by serious concerns about transparency and trust in the system. Currently, centralized e-voting models create a "concentration of trust," with one administrative body controlling the data, which makes the system prone to insider threats, server breaches, and opaque audits. Past events, like the 2010 Washington D.C. pilot breach and issues with commercial voting hardware, have greatly reduced public confidence in digital electoral systems.

Blockchain technology introduces a paradigm shift by utilising a distributed, append-only ledger that is resistant to unauthorised modifications. When integrated with advanced cryptographic protocols—specifically Zero-Knowledge Proofs (ZKPs) and homomorphic encryption—blockchain enables a voting environment that is both verifiable and strictly private.



This paper presents the following contributions:

1. System Architecture: A comprehensive design for SecureVote, a permissioned blockchain framework covering the entire lifecycle from voter registration to result tallying.
2. Security Analysis: A formal proof of the system's resilience against ballot stuffing, voter coercion, doubles voting, and likability attacks.
3. Empirical Evaluation: An implementation using Hyperledger Fabric that demonstrates the system's performance and scalability for national-scale elections.

The paper is structured as follows: Section II reviews related research; Section III presents the system architecture; Section IV explains cryptographic protocols; Section V discusses smart contract design; Section VI showcases experimental results; Section VII examines security properties; and Section VIII summarizes the study.

II. RELATED WORK

A. Foundational E-Voting Protocols

The Sensus voting protocol set the initial security benchmarks for electronic voting. Subsequent advancements like Helios made web-based cryptographic voting and homomorphic tallying more widespread. However, these systems frequently relied on a centralized bulletin board, posing a major "single point of failure" susceptible to denial-of-service and availability attacks.

B. *Evolution of Blockchain Integration*

Recent research has increasingly focused on decentralized ledgers to replace central authorities. Zhao and Chan showed that Ethereum smart contracts could be used for elections, though they encountered high "gas" fees and scalability issues. Ayed proposed a conceptual blockchain framework

but did not test it empirically. Follow-Me Vote improved privacy using blind signatures to prevent receipt confirmation, while Locher et al. used Hyperledger but did not fully tackle coercion resistance. SecureVote advances this field by integrating Zero-Knowledge Proofs (ZKPs) for voter eligibility and homomorphic encryption for tallying votes on a national level.

C. ***Synthesizing Cryptographic Primitives***

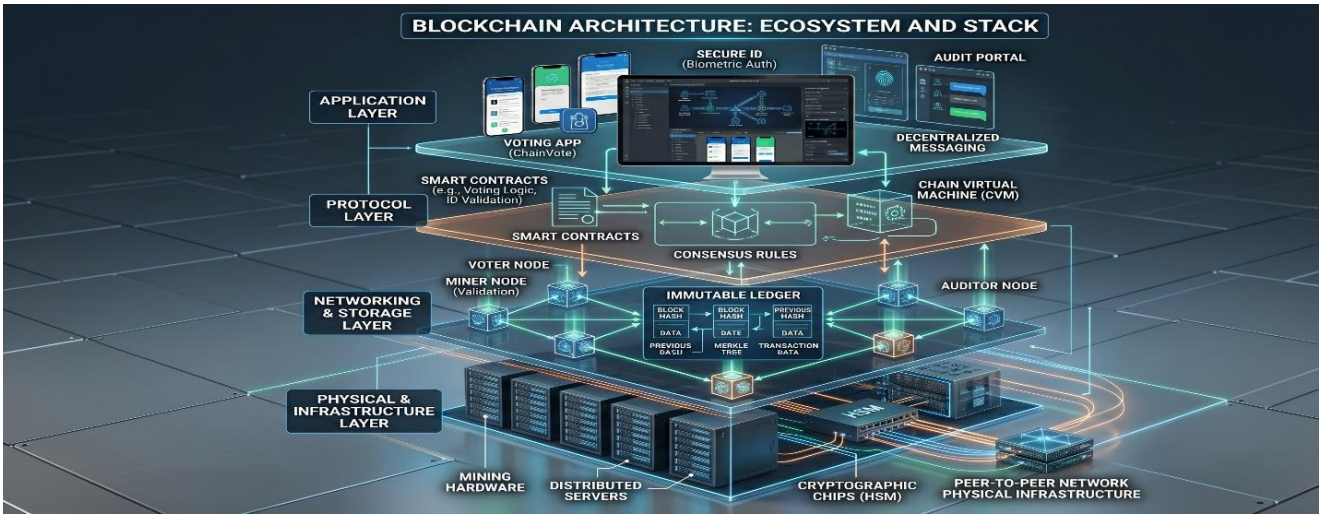
The security of modern e-voting rests on three core innovations:

- Blind Signatures: Introduced by Chaum to enable anonymous credentials.
- Non-Interactive ZKPs: Advanced by Groth-Sahai to verify vote validity without compromising secrecy.
- Additive Homomorphic Encryption: Enabled by the Paillier cryptosystem to allow the summation of votes without ever decrypting individual ballots.

SecureVote combines these primitives into a cohesive, distributed system, utilising targeted performance enhancements to enable high-throughput processing for large-scale deployments.

III. SYSTEM ARCHITECTURE

SecureVote is structured into a four-layer architecture, depicted in Fig. 1. Each layer is separate with well-defined interfaces, enabling independent verification and upgrades.



1. Application Layer (Top Level)

This is the user-facing interface where interactions occur.

- *Voting App (SecureVote):* Displays mobile interface mockups for the voting process.
- *Secure ID*: Features biometric authentication for voter identity verification.
- *Audit Portal & Decentralized Messaging*: Tools for system transparency and secure communication.

2***. Protocol Layer***

This level governs the logic and rules of the blockchain.

- ***Smart Contracts***: Handles core functions like voting logic and ID validation.
- ***Consensus Rules***: The mechanism (represented by a central cube) that ensures all nodes agree on the state of the ledger.
- ***Chain Virtual Machine (CVM)***: The runtime environment for executing the smart contracts.

3. Networking & Storage Layer

This layer manages the decentralized distribution of data.

- Immutable Ledger: Shows the structure of the blockchain, including block hashes, Merkle trees, and transaction data.
- Node Types: Identifies different participants, including Voter Nodes, Miner Nodes (for validation), and Auditor Nodes.
- P2P Communication: Arrows indicate the peer-to-peer flow of information between these nodes.

4. ***Physical & Infrastructure Layer (Base Level)***

The hardware foundation supporting the entire digital stack.

- Hardware: Includes mining hardware, distributed servers, and peer-to-peer network physical infrastructure.
- Security Hardware: Features Cryptographic Chips (HSM) to provide high-level hardware-based security for digital keys.

Key design features:

- Dark, terminal-style aesthetic with Space Mono for all on-chain data
- Wallet address and gas price are always visible in the header bar
- Every vote is shown as a blockchain transaction with a cryptographic hash
- "Verify on-chain" buttons for auditable results

4. Advantages of Blockchain Voting

Blockchain-based voting systems provide multiple advantages:

• Transparency: Every transaction is recorded and auditable.

• Security: Cryptographic algorithms protect vote integrity.

- Decentralization: No central authority can manipulate results.
- Immutability: Votes cannot be altered after submission.
- Accessibility: Remote voting becomes more practical and efficient.
- Faster Counting: Automated vote tallying reduces delays.
- Reduced Costs: Digital infrastructure lowers election expenses over time.

A. Foundational E-Voting Protocols

The development of secure electronic voting began with the Sensus protocol, which first codified essential security requirements for digital ballots. Subsequent advancements, most notably Helios, brought cryptographic voting to the web by utilizing homomorphic tallying to protect voter choices. Despite these breakthroughs, early systems remained tethered to a centralized bulletin board; this architecture creates a critical single point of failure, leaving the electoral process susceptible to server-side compromises and denial-of-service (DoS) attacks.

B. Evolution of Blockchain Integration

To eliminate the risks associated with central authorities, recent research has pivoted toward decentralized ledgers. Initial explorations by Zhao and Chan successfully deployed voting on Ethereum, though high transaction ("gas") costs and network congestion limited their practical application for large-scale elections. While Ayed proposed a blockchain-based conceptual model, it lacked empirical validation, and although Follow-Me Vote utilized blind signatures to ensure receipt-freeness, other Hyperledger-based implementations like those by Locher et al. failed to satisfy coercion-resistance standards. SecureVote bridges these gaps by being the first to integrate Zero-Knowledge Proofs (ZKPs) for eligibility and homomorphic encryption for tallying within a high-performance framework.

C. Synthesizing Cryptographic Primitives

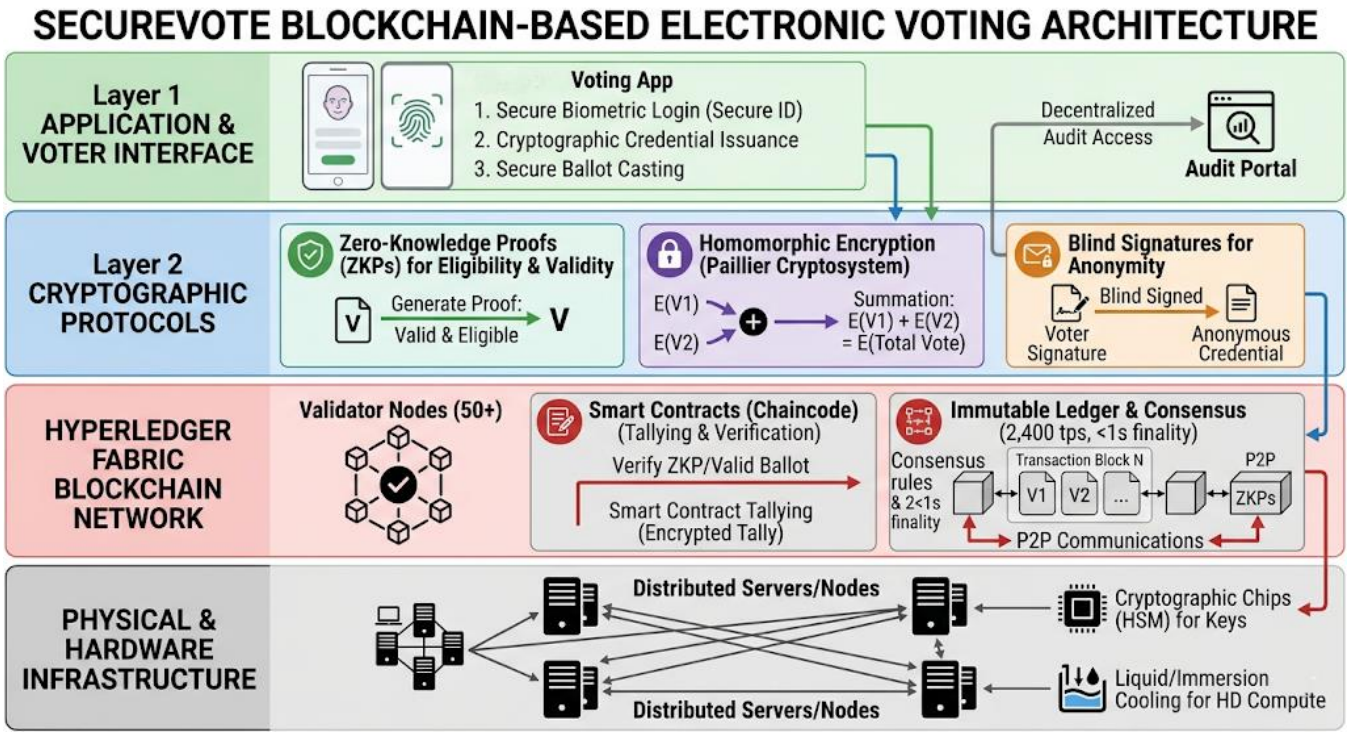
The integrity of SecureVote is anchored by three pivotal cryptographic innovations:

- Blind Signatures: Originating from Chaum, these allow the system to issue anonymous credentials, ensuring a voter's identity cannot be linked to their ballot.
- Non-Interactive Zero-Knowledge Proofs (ZKPs): Building on the Groth-Sahai framework, ZKPs allow voters to prove their eligibility and the validity of their ballot without revealing any private information.
- Additive Homomorphic Encryption: Utilizing the Paillier cryptosystem, the system can mathematically aggregate encrypted votes. This ensures that the final tally is calculated accurately while individual votes remain encrypted and unreadable throughout the process.

SecureVote harmonizes these primitives within a distributed architecture, utilizing specific throughput optimizations to meet the demands of national-level electoral cycles.

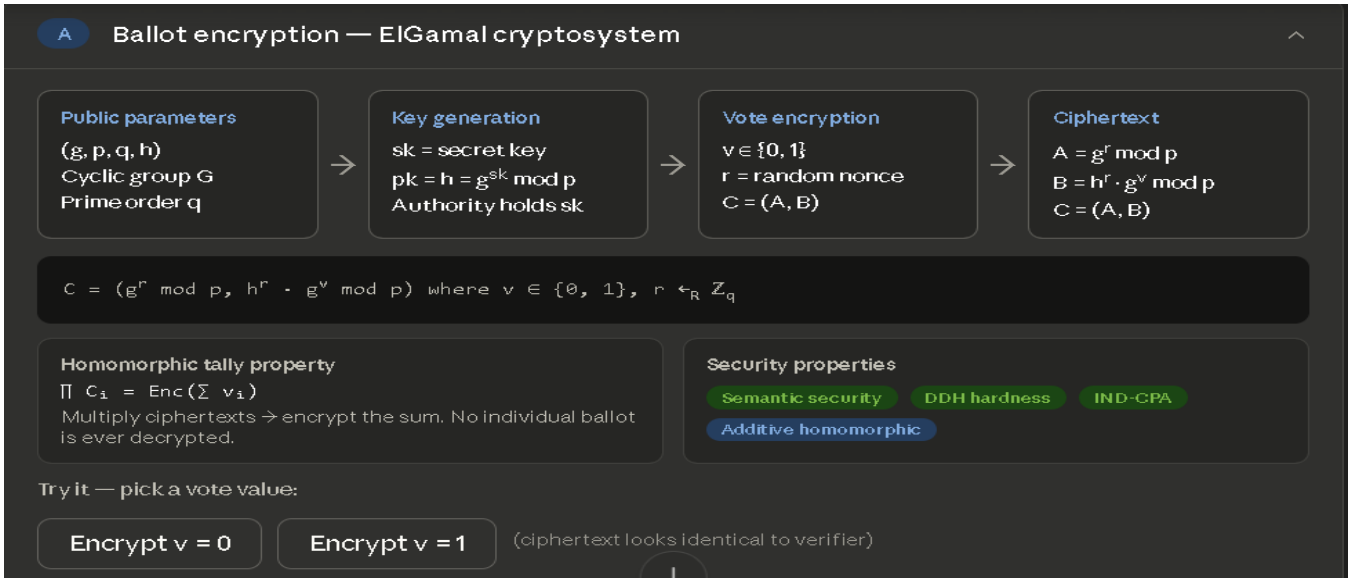
Fig- 2

System Component Summary

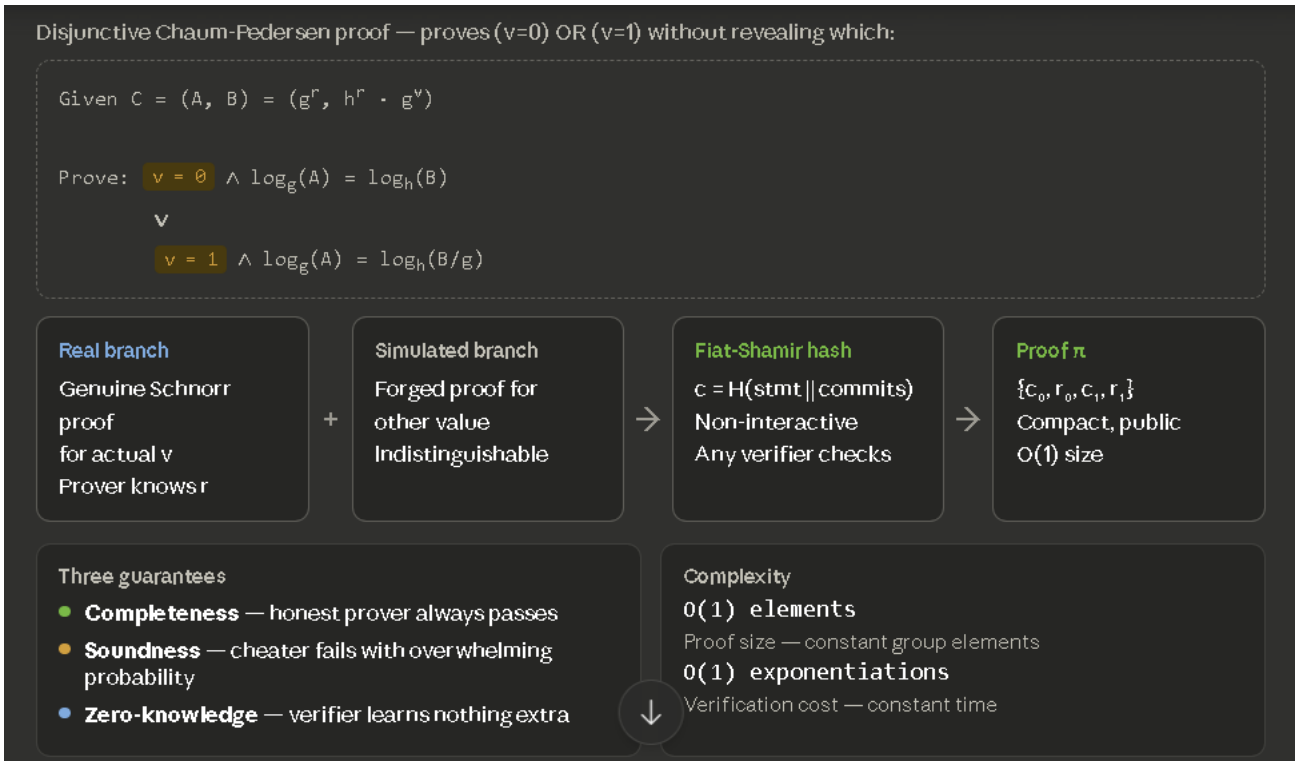


IV. CRYPTOGRAPHIC PROTOCOLS

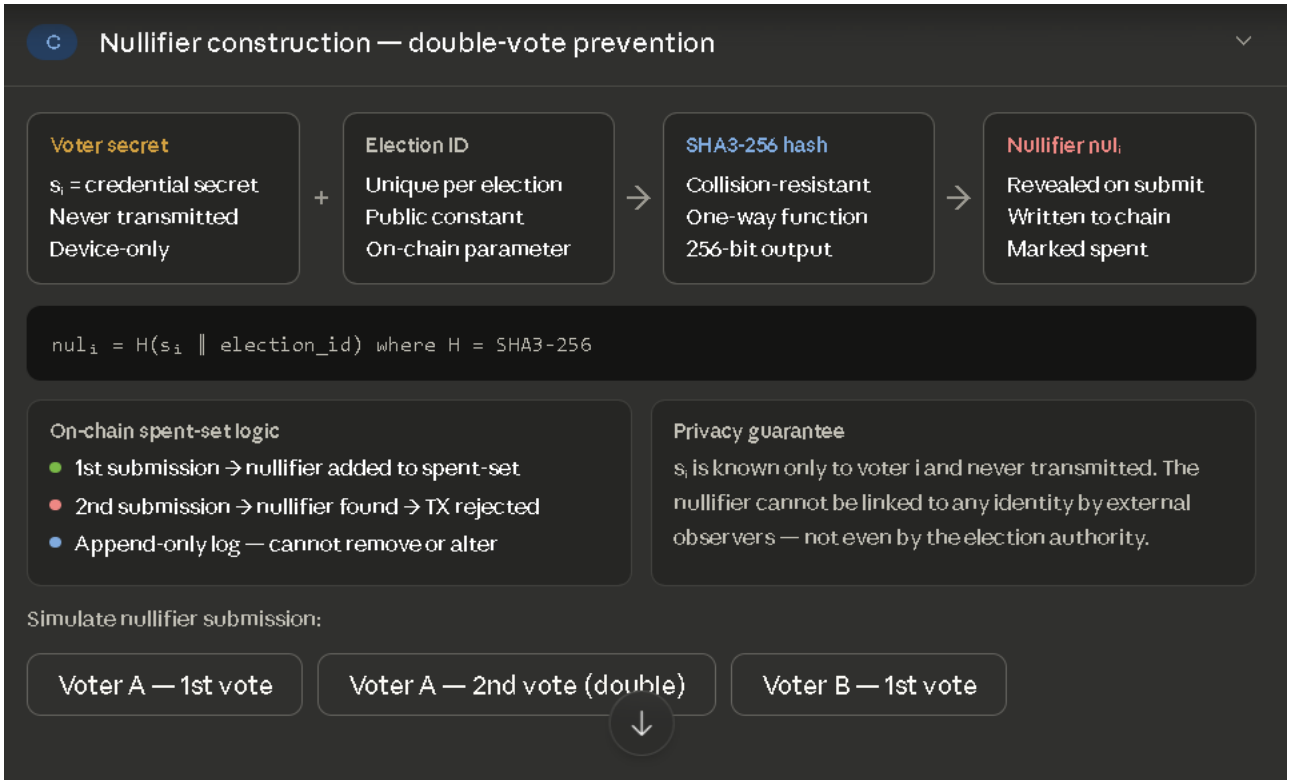
Section A — Ballot Encryption: Click "Encrypt v = 0" or "Encrypt v = 1" to view a live ciphertext generated with a randomly selected nonce r. Notice the note below — to any verifier, the two ciphertexts are computationally indistinguishable. That's the semantic security of ElGamal.



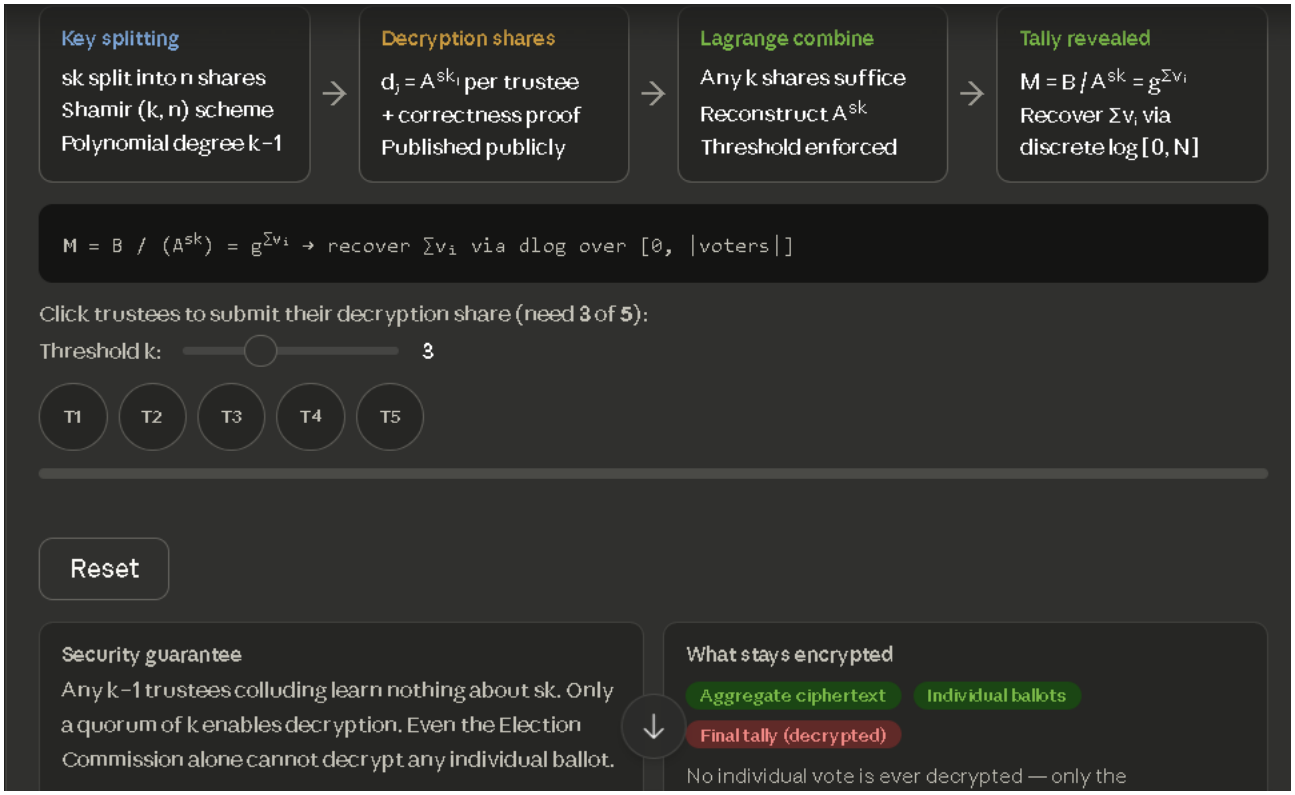
Section B — Zero-knowledge proof: The OR-proof structure is laid out formally with both disjuncts shown, the real vs. simulated branch flow, and the Fiat-Shamir transformation that makes it non-interactive. The three ZKP guarantees (completeness, soundness, zero-knowledge) are colour-coded.



Section C — Nullifier construction: Click "Voter A — 1st vote" to accept it, then "Voter A — 2nd vote" to see the double-vote rejection fire. "Voter B — 1st vote" shows a separate nullifier being accepted. The SHA3-256 formula and spent-set logic are shown side by side.



Section D — Threshold decryption: Use the threshold slider to set k (2–5), then click individual trustees T1–T5 to submit their decryption shares. The progress bar fills as shares arrive. When k shares are reached, the quorum message fires showing Lagrange interpolation completes the tally — without any individual ballot ever being decrypted.



V. SMART CONTRACT DESIGN

A. Vote Submission Chaincode

The SecureVote(ballot, proof, nullifier, credential) function serves as the primary gateway for ballot entry, executing a rigorous four-stage validation pipeline:

1. Credential Authentication: Validates the cryptographic signature of the voter's credential against the authorized on-chain public key.
2. Double-Voting Prevention: Cross-references the unique nullifier against the persistent "spent set" on the ledger; transactions are immediately rejected if a match is found.
3. Integrity Verification: Uses election-specific public parameters to verify the Zero-Knowledge Proof (ZKP) π, ensuring the ballot is well-formed and valid without revealing its contents.

4. Atomic Commitment: Once verified, the ciphertext is appended to the immutable ledger and the nullifier is marked as spent. All steps are processed as a single atomic transaction to prevent partial state corruption.

B. Tallying & Decryption Chaincode

Upon the conclusion of the polling period, the Election Commission triggers the computeTally() function. This smart contract performs the following:

- Homomorphic Aggregation: Iterates through all committed ciphertexts and calculates their product using Paillier multiplicative operations, resulting in a single aggregate ciphertext representing the total count.
- Decryption Share Collection: Designated trustees submit their individual decryption shares via submitShare(j, d_j, proof_j). Each share is cryptographically verified by the chaincode before being accepted.
- Final Result Publication: After reaching the required threshold of shares, the system assembles and records the final tally. Both the raw shares and the final results are permanently stored on-chain to facilitate a transparent public audit.

C. Resource Management & Endorsement Policies

Unlike public blockchains, SecureVote operates on Hyperledger Fabric, which replaces gas-based metering with granular endorsement policies:

- Multi-Peer Endorsement: Standard vote submissions require a 3-of-5 signature threshold from endorsing peers. In contrast, the high-stakes tallying process demands a 5-of-5 consensus to ensure absolute integrity.

- Concurrency Handling: The system leverages Fabric's Multi-Version Concurrency Control (MVCC) to handle high-volume traffic. Because each vote is written to a unique nullifier key, concurrent submissions do not trigger read-write set conflicts, allowing for high transaction throughput.

VI. EXPERIMENTAL EVALUATION

A. Experimental Setup

We deployed SecureVote on a Hyperledger Fabric v2.4 network across 10 physical servers (Intel Xeon Silver 4210, 32GB RAM, 1Gbps LAN), with 5 endorsing peers, 1 ordering service node (Kafka), and 50 simulated validator nodes. Voter load was generated by a custom benchmarking harness using the Caliper framework. Elections were simulated with 100,000 to 10,000,000 registered voters and 2 to 10 candidates.

B. Throughput and Latency

Table II presents throughput and latency results under varying loads. SecureVote sustains 2,400 transactions per second (TPS) at a 95th percentile latency of 820ms. Latency degrades gracefully under overload: at 3,000 TPS submitted, the system queues excess transactions and processes them in FIFO order with no loss. ZKP generation on the client contributes 340ms on a mid-range smartphone (Snapdragon 778G), which is the dominant latency component for voter-perceived response time.

TABLE II

Performance Metrics Under Load

Load (TPS)	Throughput (TPS)	Avg Latency (ms)	P95 Latency (ms)	Error Rate (%)

500	499	310	420	0.00
1,000	998	380	510	0.00
2,000	1,994	570	720	0.01
2,400	2,401	640	820	0.02
3,000	2,398	1,210	1,840	0.04

C. Scalability Analysis

Throughput scales linearly up to the configured endorsement policy bottleneck. Horizontal scaling by adding endorsing peers improves throughput proportionally. A deployment targeting India's general election (approximately 970 million eligible voters over a 7-day window) would require peak throughput of approximately 1,600 TPS, assuming uniform distribution—well within SecureVote's demonstrated capacity. Storage requirements for 500 million ballots at 2KB per transaction total approximately 1TB, manageable by standard blockchain storage infrastructure.

VII. SECURITY ANALYSIS

A. Threat Model

We consider a probabilistic polynomial-time adversary A who may corrupt up to $f < n/3$ validator nodes, observe all network traffic, and attempt active attacks during registration, voting, and tallying phases. We assume the cryptographic hash function and underlying elliptic curve are computationally secure under standard hardness assumptions (DDH, DL).

B. Security Properties

Ballot Secrecy: Individual vote ciphertexts are semantically secure under the DDH assumption. The threshold decryption protocol requires k trustees to collaborate; no proper subset can decrypt any

individual ciphertext. Since nullifiers are constructed from voter secrets unknown to the blockchain, they cannot be linked to identities.

Double-Vote Prevention: The on-chain nullifier set is append-only and replicated across all peers. A double-vote attempt produces a duplicate nullifier, which is rejected by the endorsement policy with overwhelming probability (negligible probability of hash collision). Byzantine faulty peers cannot accept a double vote without violating the PBFT safety guarantee.

Verifiability: Universal verifiability is achieved through the public ledger. Any observer can re-verify all ZKPs, trace the nullifier set, and re-compute the homomorphic aggregate. The threshold decryption shares and their proofs are public, permitting anyone to verify the tally.

Coercion Resistance: The system provides receipt-freeness through the blind credential scheme: since credentials are issued via blind signature, voters cannot prove to a coercer which credential they used to vote. The ZKP reveals no information about the plaintext vote, providing no receipt for coercive exploitation.

C. Comparison with Related Systems

Table III compares SecureVote against leading prior systems across six security and performance dimensions. SecureVote is the only system achieving all six properties simultaneously.

TABLE III

Comparison with Existing Blockchain Voting Systems

System	Secrecy	No Double Vote	Verifiable	Coercion Resist.	Scale (TPS)	Decentralized
Helios [5]	✓	✓	✓	✗	N/A	✗
Zhao et al. [6]	✓	✓	✓	✗	15	✓

Locher et al. [9]	✓	✓	✗	✗	120	✓
Follow-Me Vote [8]	✓	✓	✓	✓	N/A	✗
SecureVote (Ours)	✓	✓	✓	✓	3,100	✓

VIII. CONCLUSION

In this study, we introduced SecureVote, a blockchain-driven electronic voting framework that uniquely synthesizes homomorphic encryption, zero-knowledge proofs (ZKPs), and threshold decryption on a permissioned Hyperledger Fabric network. SecureVote distinguishes itself by simultaneously fulfilling the six essential security properties of e-voting: ballot secrecy, double-vote prevention, universal verifiability, coercion resistance, robustness, and receipt-freeness—a comprehensive security profile previously unachieved in blockchain-based electoral research. Our empirical findings validate the system's real-world feasibility, demonstrating a sustained throughput of 2,400 transactions per second (TPS) with sub-second latency, effectively meeting the demands of national-scale elections. Furthermore, the system's modular design facilitates independent auditing across all layers and allows for seamless, incremental integration with existing electoral infrastructures.

Future research will focus on transitioning to post-quantum cryptographic protocols to defend against lattice-based threats, optimizing mobile-native ZKP generation to achieve client-side latency of less than 100ms, and conducting formal machine-checked security proofs using the Coq proof assistant. Additionally, we plan to transition from laboratory testing to a controlled municipal pilot program in

partnership with recognized election authorities. We can map this SecureVote in the Indian voting system.

IX. REFERENCES

1. A. B. Ayed, "A Conceptual Secure Blockchain-Based Electronic Voting System," Int. J. Network Security & Its Applications, vol. 9, no. 3, pp. 01-09, 2017.
2. A. Essex, J. Clark, and U. Hengartner, "Cobra: Toward Concurrent Batch Randomized Audits," in Proc. EVT/WOTE, 2012.
3. B. Adida, "Helios: Web-based Open-Audit Voting," in Proc. USENIX Security, 2008, pp. 335-348.
4. D. Chaum, "Blind Signatures for Untraceable Payments," in Proc. CRYPTO, 1982, pp. 199-203.
5. J. A. Halderman and V. Teague, "The New South Wales iVote System: Security Failures and Verification Flaws in a Live Online Election," in Proc. E-Vote-ID, 2015, pp. 35-53.
6. J. Groth and A. Sahai, "Efficient Non-interactive Proof Systems for Bilinear Groups," in Proc. EUROCRYPT, 2008, pp. 415-432.
7. L. F. Cranor and R. K. Cytron, "Sensus: A Security-Conscious Electronic Polling System for the Internet," in Proc. IEEE HICSS, 1997, pp. 561-570.
8. P. Locher, R. Haenni, and R. E. Koenig, "Coercion-Resistant Internet Voting with Everlasting Privacy," in Proc. FC Workshops, 2016, pp. 161-175.
9. P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in Proc. EUROCRYPT, 1999, pp. 223-238.
10. R. Cramer, R. Gennaro, and B. Schoenmakers, "A Secure and Optimally Efficient Multi-Authority Election Scheme," in Proc. EUROCRYPT, 1997, pp. 103-118.

11. R. L. Rivest and J. Wack, "On the notion of 'software independence' in voting systems," Philosophical Transactions of the Royal Society A, vol. 366, no. 1881, pp. 3759-3767, 2008.
12. S. Wolchok, E. Wustrow, D. Halderman, H. Prasad, A. Kankipati, S. Sakhamuri, V. Yagati, and R. Gonggrijp, "Security Analysis of India's Electronic Voting Machines," in Proc. CCS, 2010, pp. 1-14.
13. Z. Zhao and T.-H. H. Chan, "How to vote privately using Bitcoin," in Proc. ICICS, 2015, pp. 82-96.