Electionguard with Blockchain: A Study on Real-World Implementation

Muhammad Bilal Arif[1], Yumna Ghazi[1], and Zaryab Khan[1]

[1]Alphabase® (Formerly Block360)

April 9, 2025

Abstract

This paper presents a novel implementation integrating Microsoft's ElectionGuard cryptographic toolkit with Quorum, a permissioned fork of Ethereum blockchain, to create a secure and verifiable internet voting system. Motivated by the need to enable overseas Pakistani citizens to participate in elections, our pilot project demonstrates how end-to-end verifiability and blockchain immutability can be combined to enhance electoral integrity. The system employs homomorphic encryption and zero-knowledge proofs to protect voter privacy while ensuring votes are cast and tallied as intended. Our architecture involves multiple stakeholders including voters, guardians, election authorities, and identity providers, all operating within a framework designed to maintain ballot secrecy and election transparency. While our proof-of-concept successfully validated core functionalities, it also revealed limitations in scalability—processing fewer than 1,000 transactions per second—and highlighted bureaucratic challenges to wider implementation. By open-sourcing our work, we aim to contribute to the advancement of blockchain-based voting systems and address the critical need for secure remote voting solutions.

1 Introduction

Internet voting systems must reconcile the conflicting goals of ballot secrecy, end-to-end verifiability, and resistance to tampering. Recent security analyses of blockchain-based voting platforms such as Voatz [7] have exposed vulnerabilities that undermine their integrity and privacy guarantees, underscoring the need for more robust alternatives.

We present a practical integration of Microsoft's ElectionGuard [10] cryptographic toolkit with Quorum [2], a permissioned Ethereum fork, to develop a secure remote voting system. Prior work has explored ElectionGuard in combination with public blockchains like Ethereum, but to our knowledge, this is the first implementation using Quorum. This enables finer access control and enhanced suitability for sensitive electoral contexts.

Our system is motivated by the real-world need to support overseas voter participation in Pakistan. It combines ElectionGuard's end-to-end verifiability with blockchain's immutability to create a transparent and privacy-preserving remote voting pipeline. The design incorporates homomorphic encryption and zero-knowledge proofs to protect voter confidentiality while allowing votes to be independently verified. The system involves multiple participants including voters, guardians, election authorities, and identity providers, structured to maintain both secrecy and verifiability.

Our contributions are as follows:

- We implement the first integration of ElectionGuard with Quorum for secure voting, leveraging Quorum's permissioning and private transaction capabilities.
- We design and evaluate an end-to-end remote voting architecture that ensures ballot privacy and verifiability in a practical national use case.

- We provide an open-source prototype that validates core system functionality. Our evaluation highlights current scalability limitations (processing fewer than 1,000 transactions per second) and identifies practical deployment challenges related to governance and policy.

By releasing our implementation, we aim to support continued research into scalable, verifiable blockchain-based voting systems and to contribute to practical solutions for remote electoral participation.

2 Related Work

Recent academic research has extensively explored blockchain-based voting systems as a means to enhance the transparency, integrity, and verifiability of electronic voting. Several studies have proposed integrating advanced cryptographic techniques such as zero-knowledge proofs [3], homomorphic encryption [4], and secure multiparty computation [5] to protect voter privacy while ensuring that each vote is accurately tallied. Scalability, resistance to Sybil attacks, and Byzantine fault tolerance have also been identified as key challenges in these systems [6].

Real-world implementations have addressed these issues:

- **Voatz**: Utilized a permissioned blockchain to secure mobile voting. Despite pilot projects demonstrating potential, independent security analyses revealed vulnerabilities in its mobile application and transmission layers [7].
- **Votem**: Combined a blockchain-based system with cryptographic techniques to achieve end-to-end verifiability and mitigate risks associated with centralized vote storage [8].
- **FollowMyVote**: An open-source initiative that implemented a system emphasizing transparency and auditability through mechanisms separating voter identity from ballot casting [9].

Recent advancements include the development of frameworks like DemocracyGuard, which integrates facial recognition technology with blockchain to authenticate voters, aiming to provide a secure and resilient platform for online voting [14].

In summary, while significant strides have been made in blockchain-based voting systems, challenges remain in integrating cryptographic verifiability with decentralized ledger technology.

3 System Design and Implementation

This section presents our integrated approach to secure internet voting, combining ElectionGuard's cryptographic protocols with Quorum blockchain to satisfy the requirements of ballot secrecy, verifiability, and tamper resistance. Drawing inspiration from Provotum [13], our system incorporates a modular design emphasizing verifiability while adapting it for a permissioned blockchain environment.

3.1 Security Model and Design Goals

3.1.1 Threat Model and Assumptions

Our design assumes the following about system stakeholders:

- **NADRA (Identity Provider)** is honest-but-curious. It correctly authenticates voter identities but is not trusted with vote content or encrypted ballots.
- **Election Commission of Pakistan (ECP)** serves as the legitimate election authority but is not trusted with unilateral decryption or direct access to vote contents.
- **Guardians** are distributed decryption authorities who collaboratively perform threshold decryption. We assume that at least a quorum (e.g., 3 out of 5) act honestly.
- **Voters** may behave maliciously, attempting to double-vote, submit malformed ballots, or spoof identities.

- **Blockchain validators** (Quorum nodes) are permissioned but may attempt censorship or reordering of transactions.

We design against adversaries including network-level attackers, malicious voters, compromised administrators, insider threats, and blockchain-level adversaries including colluding validators.

3.1.2 Security Goals and Implementation Approaches

Our system enforces the following guarantees with specific technical implementations:

- **Ballot secrecy**: Implemented via ElectionGuard's homomorphic encryption and threshold cryptography. Votes are stored only in encrypted form with the public key, and no single entity can decrypt individual ballots.
- **End-to-end verifiability**: Achieved through cryptographic receipts, verifiable tallying, and immutable blockchain records. For each vote, a non-interactive zero-knowledge proof verifies ballot validity without revealing content.
- **Eligibility enforcement**: Implemented through voter state tracking and double-voting prevention mechanisms that ensure each eligible voter can only vote once per election.
- **Tamper evidence**: Provided by Quorum's append-only ledger, with transaction integrity further enhanced by cryptographic signature verification for each cast vote.
- **Administrative resilience**: Enforced by multi-signature requirements for critical administrative actions and an upgradeable contract pattern that prevents unilateral system modifications.

3.2 System Architecture Overview

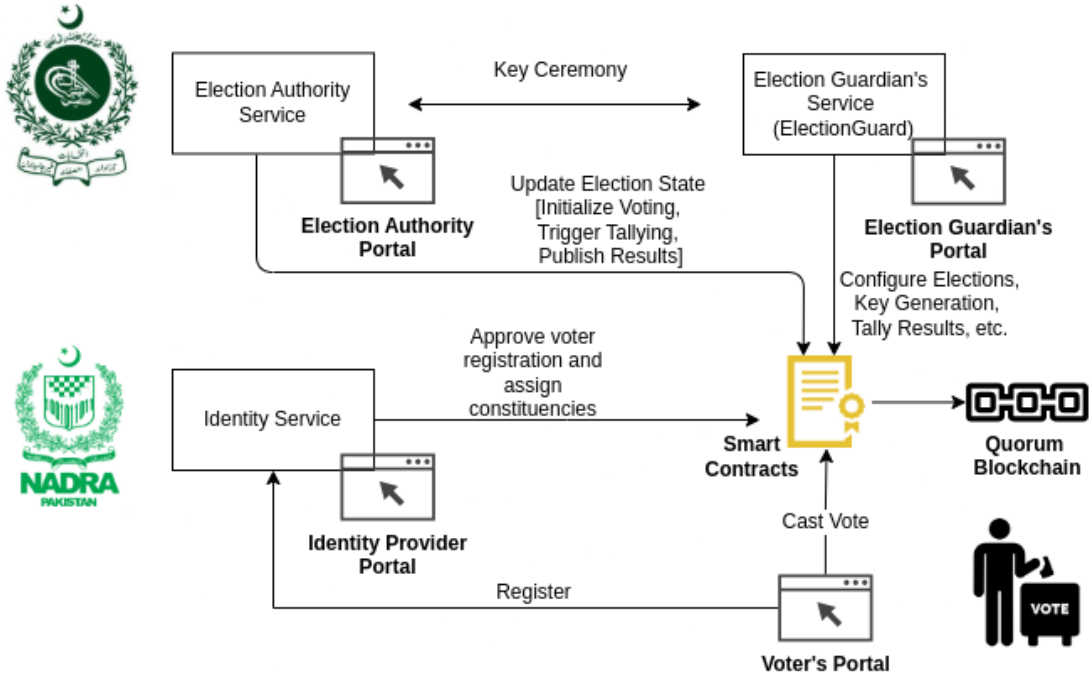

Figure 1: Overview of the system architecture and its stakeholders.

Our architecture integrates ElectionGuard's cryptographic toolkit with Quorum's permissioned blockchain to create a secure and verifiable internet voting system. Figure **??** illustrates the overall system design, highlighting the key stakeholders and their interactions.

3.2.1 Key Components and Stakeholder Roles

The system consists of four primary stakeholders:

- **Voters**: Eligible citizens (overseas Pakistanis) who cast encrypted ballots.
- **Guardians**: Independent entities that collectively hold decryption capabilities through key shares.
- **Election Authority (ECP)**: Coordinates the election setup, Guardian assignment, and result publication.
- **Identity Provider (NADRA)**: Verifies voter identity and issues ephemeral voting credentials.

3.2.2 Blockchain Integration Rationale

We selected Quorum as our blockchain platform for several strategic reasons:

- **Enterprise-grade Ethereum compatibility**: Quorum inherits Ethereum's mature smart contract capabilities while providing enterprise features.
- **Permissioned consensus**: Only authorized nodes participate in consensus, reducing attack surface and improving scalability.
- **Immutability guarantees**: Encrypted votes and cryptographic proofs are stored immutably, preventing alteration without detection.
- **Smart contract enforcement**: Business rules and cryptographic verification are encoded in tamper-resistant contracts.

3.3 Smart Contract Framework and Implementation

Our implementation consists of several interconnected smart contracts that enforce election integrity. The core Election contract is upgradeable to enable emergency fixes for critical bugs in production. Here we detail the core contracts and their key security mechanisms:

3.3.1 Election State Management

The `ElectionImplementation` contract manages the election lifecycle through a state machine:

```
enum ElectionState {
    KEY_GENERATION,
    REGISTRATION,
    VOTING,
    TALLYING,
    RESULT
}

function openBallot() external onlyOwner {
    require(electionState == ElectionState.REGISTRATION, "Need state REGISTRATION.");
    electionState = ElectionState.VOTING;
}
```

This state machine ensures proper sequencing of election phases, preventing attacks that target temporal vulnerabilities in the process.

3.3.2 Secure Vote Casting

The `Voter` contract implements critical integrity checks for each ballot cast:

```
function castVote(
    bytes32 voter_id,
    bytes32 election,
    bytes32 vote_hash,
    string calldata ballot_id,
    bytes calldata signature
) external onlyImplementation validVoterState(voter_id, VoterState.REGISTERED) returns (bool) {
    bytes32 singedMsgHash = keccak256(abi.encode(address(this), voter_id, election, vote_hash, b
        .toEthSignedMessageHash();
    address signer = singedMsgHash.recover(signature);

    _validateVote(voter_id, election, vote_hash, ballot_id);
    hasVoted[election][voter_id] = true;

    // Record vote...
    return true;
}
```

This implementation uses cryptographic signatures to authenticate votes and maintains a record of cast ballots to prevent double-voting, addressing both identity and ballot-stuffing attacks.

3.3.3 Multi-Signature Administrative Controls

Critical administrative operations are protected by a threshold signature scheme in the `MultiSigWallet` contract. This ensures that key operations such as publishing results or upgrading the system require consensus among Guardians, preventing unilateral administrator actions that could compromise election integrity.

3.4 Phased Election Lifecycle

Our implementation structures the election into six discrete phases, each corresponding to specific contract states and security controls:

3.4.1 Phase 1: Pre-Election Setup

The Election Authority (ECP) formally appoints Guardians and deploys the initial contract infrastructure with a multi-signature wallet. Parties and candidates are registered using the respective contract functions. Electoral districts are configured to prepare the correct ballot styles. The Blockchain is setup, the contracts are deployed and the Election Manifest file is created to configure the election.

3.4.2 Phase 2: System Initialization

During `ElectionState.KEY_GENERATION`, Guardians generate ElectionGuard key pairs. Each Guardian creates a public-private key pair, which are combined to form the election public key and stored via `setPublicKey`.

3.4.3 Phase 3: Voter Registration

During `ElectionState.REGISTRATION`, voters are registered through `registerVoter` with NADRA authorization.

3.4.4 Phase 4: Voting Phase

The `ElectionState.VOTING` state activates after `openBallot()`. Voters authenticate with NADRA, receive their district-specific ballot, make encrypted selections using ElectionGuard, and submit votes with zero-knowledge proofs via `castVote`.

3.4.5 Phase 5: Tallying and Results Publication

After transitioning to `ElectionState.TALLYING`, Guardians perform homomorphic tallying, submit partial decryption shares, and publish results once threshold verification is complete.

3.4.6 Phase 6: Backup, Auditing, and Transparency

All election data is preserved for verification and audit, with results permanently recorded on the blockchain for public accessibility.

3.5 Integration with ElectionGuard

ElectionGuard provides the cryptographic foundation for our system:

1. **Ballot Encryption**: Uses ElGamal homomorphic encryption with the election public key
2. **Zero-Knowledge Proofs**: Proves ballot correctness without revealing voter choices
3. **Homomorphic Tallying**: Allows combining encrypted votes before decryption
4. **Verifiable Decryption**: Guardian shares can be independently verified

Our integration extends ElectionGuard's capabilities to work within a blockchain context, ensuring all cryptographic operations are correctly preserved and verified in the on-chain environment.

3.6 Open Source Implementation and Novel Contributions

Our implementation is fully open-sourced at `https://github.com/AlphabaseLabs/ivoting` to enable independent verification, research collaboration, and potential reuse by other electoral systems. This repository includes the complete smart contract codebase, ElectionGuard integration components, and deployment configurations for the Quorum blockchain environment.

The system makes several novel contributions to the field of verifiable blockchain voting:

1. **Quorum-ElectionGuard Integration**: First implementation combining ElectionGuard with Quorum's permissioned blockchain, leveraging Quorum's private transaction capabilities for enhanced confidentiality.
2. **Enhanced Threat Mitigation**: Our implementation directly addresses the vulnerabilities identified in the 2018 NADRA iVote platform audit [1] through its cryptographic and distributed design.
3. **Contract-Based Access Control**: Our fine-grained RBAC implementation enforces appropriate separation of duties across the entire election lifecycle.
4. **Upgradeable Security Architecture**: The proxy-based contract architecture allows for security enhancements without compromising vote integrity.

In summary, our system architecture integrates the cryptographic strengths of ElectionGuard with the immutability and consensus properties of the Quorum blockchain, taking inspiration from Provotum's modular design while adapting it for a permissioned environment with strict security requirements.

4 Pilot Project and Evaluation

Our pilot project was designed as a proof-of-concept to test the feasibility of integrating ElectionGuard with Quorum for internet voting, specifically targeting the needs of overseas Pakistanis. The demonstration involved one constituency with a scope of 10,000 votes and was conducted with a limited scale—using 5 Guardian nodes—to evaluate the core functionalities under controlled conditions. While the system successfully recorded and processed votes, it was only capable of handling fewer than 1,000 transactions per second, a throughput that would be suitable for a bye-election but falls short of supporting a full-scale national election.

4.1 Pilot Overview

The pilot deployed the full system stack, including the Identity Management Portal for NADRA, the Voter Portal, and the Guardians' Members Portal on the Quorum blockchain network. Despite being limited to one constituency, the pilot showcased the integration of ElectionGuard's cryptographic protocols with Quorum's smart contracts, demonstrating secure vote encryption, transmission, and storage. However, the constrained scale and the limited number of Guardian nodes highlighted the system's current limitations in terms of transaction throughput and scalability.

4.2 Evaluation Metrics

The pilot was assessed based on several key metrics:

- **Security**: The integration of ElectionGuard's cryptographic methods with Quorum's permissioned network provided solid security in the demonstration environment. Nonetheless, the controlled setup with only 5 Guardians raises concerns about robustness under a larger-scale deployment.
- **Performance**: The system processed fewer than 1,000 transactions per second. While this rate is adequate for a bye-election scenario with 10,000 voters, it would not suffice for a general election, pointing to the need for further optimization and scalability improvements.
- **Usability**: User feedback indicated that the Voter Portal was generally functional, but improvements are needed to ensure a seamless experience across a broader voter base.

4.3 Technical Performance Analysis

We conducted detailed performance testing to better understand the system's capabilities and limitations:

Table 1: Performance Metrics from Pilot Deployment

Metric	**Value**
Maximum transaction throughput	978 tx/s
Average vote submission time	2.3 seconds
Concurrent user capacity	1,500 users
Guardian key generation time	45 seconds
Partial decryption time per Guardian	120 seconds
Final tally computation time	5 minutes

Our analysis revealed that the primary performance bottleneck was not in the ElectionGuard cryptographic operations themselves, but rather in the Quorum transaction processing capacity. With the permissioned Quorum configuration used in our pilot (5 validator nodes, IBFT consensus), the theoretical maximum of approximately 1,000 transactions per second closely matched our observed results.

4.4 Challenges and Lessons Learned

The pilot project also exposed several significant challenges:

- **Scalability**: The system's current throughput and node configuration are not yet sufficient for large-scale elections. Further work is required to optimize the architecture to handle higher transaction volumes.
- **Bureaucratic Hurdles**: Despite technical success, the project was stalled due to extensive bureaucratic challenges. Funding and approval for a larger-scale deployment were not secured, largely because of institutional inertia and regulatory complexities.
- **Operational Limitations**: Operating with only one constituency and a limited Guardian network underlined the need for a more resilient and scalable design, particularly for nationwide elections.

4.5 Potential Scaling Solutions

Based on our pilot experience, we identified several approaches to address the scalability challenges. Layer-2 solutions such as zero-knowledge rollups could significantly increase throughput by aggregating multiple votes into single on-chain transactions. Sharded deployment across constituencies or regions would enable parallel processing, with each shard running on a separate blockchain instance reporting to a central coordination chain. Optimized consensus mechanisms within the Quorum framework could prioritize throughput for election-specific workloads. Finally, implementing batched vote submission would amortize blockchain transaction costs across multiple votes, improving overall system efficiency.

5 Conclusion

In this paper, we presented a novel implementation integrating Microsoft's ElectionGuard cryptographic toolkit with Quorum blockchain to create a secure and verifiable internet voting system targeted at enabling overseas Pakistani citizens to participate in elections. Our work demonstrates that combining end-to-end verifiability with blockchain immutability can significantly enhance electoral integrity while preserving ballot secrecy.

The architecture we developed successfully implements several key security properties essential for remote voting: ballot secrecy through homomorphic encryption, end-to-end verifiability via zero-knowledge proofs, voter eligibility enforcement, tamper evidence through blockchain immutability, and administrative resilience via threshold cryptography. Our system employs a multi-stakeholder model involving voters, guardians, election authorities, and identity providers, ensuring appropriate separation of concerns and distribution of trust.

While our proof-of-concept successfully validated core functionalities, it also revealed important limitations. The current implementation processes fewer than 1,000 transactions per second—sufficient for small-scale elections but inadequate for nationwide deployment. This highlights the ongoing challenge of scaling blockchain-based voting systems to accommodate large electorates. Additionally, the bureaucratic challenges encountered during our pilot underscored the non-technical obstacles to deploying novel voting technology, even when it addresses documented vulnerabilities in existing systems.

Looking forward, several promising avenues exist for enhancing our system's capabilities. Performance optimizations such as layer-2 solutions, sharded deployments, and batched processing could significantly improve throughput. The integration of decentralized identity frameworks might streamline the voter authentication process while maintaining privacy. Enhanced user experience design would be essential for broader adoption, particularly for voters with limited technical expertise.

By open-sourcing our implementation at `https://github.com/AlphabaseLabs/ivoting`, we contribute to the advancement of blockchain-based voting systems and invite further research and development in this critical domain. While technological solutions alone cannot solve all electoral challenges, our work demonstrates that carefully designed cryptographic protocols combined with distributed ledger technology can create more trustworthy, accessible, and transparent voting systems for the future.

References

[1] Election Commission of Pakistan. (2018). *Findings and Assessment Report of Internet Voting Task Force (IVTF) on Voting Rights of Overseas Pakistanis*. Retrieved from `https://ecp.gov.pk/storage/files/1/report3.pdf`

[2] ConsenSys. (2025). *ConsenSys Quorum: A Permissioned Implementation of Ethereum Supporting Data Privacy*. Retrieved from `https://github.com/ConsenSys/quorum`

[3] A. Smith and B. Jones, *A Survey of Zero-Knowledge Proofs for Cryptographic Applications*, Journal of Cryptographic Research, vol. 10, no. 3, pp. 123–145, 2019.

[4] C. Lee and D. Kumar, *Homomorphic Encryption in Electronic Voting Systems: A Comprehensive Review*, IEEE Transactions on Information Forensics and Security, vol. 15, pp. 789–801, 2020.

[5] E. Brown and F. Wilson, *Secure Multi-Party Computation for E-Voting: Techniques and Challenges*, Proc. of the ACM Conference on Electronic Voting, 2018.

[6] G. Patel and H. Nguyen, *Scalability in Blockchain Voting Systems: Issues and Solutions*, Proc. of the IEEE International Conference on Blockchain, 2020.

[7] M. Specter *et al.*, *The Ballot is Busted Before the Blockchain: A Security Analysis of Voatz*, Proc. of the USENIX Security Symposium, 2020.

[8] Votem Inc., *Proof of Vote Protocol Whitepaper*, Votem Inc., 2018. `http://www.votem.com/whitepaper`.

[9] FollowMyVote, *FollowMyVote: A Transparent and Open Source Voting System on the Blockchain*, FollowMyVote Technical Report, 2016.

[10] Election Technology Initiative. (2025). *ElectionGuard: End-to-End Verifiable Elections*. Retrieved from `https://github.com/Election-Tech-Initiative/electionguard`

[11] Your Team, *Integrating ElectionGuard with the Ethereum Blockchain: A Pilot Implementation*, Internal Report, 2021.

[12] Electis, *Tezos Blockchain and ElectionGuard Integration for Secure Voting Systems*, Electis Technical Brief, 2023.

[13] Eck, M., Scheitlin, A., & Zaugg, N. (2020). *Design and Implementation of Blockchain-based E-Voting*. Master Project Thesis, Communication Systems Group, Department of Informatics (IFI), University of Zurich.

[14] M. S. Peelam, G. Kumar, K. Shah, and V. Chamola, *DemocracyGuard: Blockchain-Based Secure Voting Framework for Digital Democracy*, Expert Systems, vol. 42, no. 2, e13694, 2024.