Designing, Implementing, and Evaluating an Ethereum-Based Voting Platform for Campus Elections

Nour Al-Sheikh

April 30, 2025

Abstract

This proposal outlines a project for designing, implementing, and evaluating a permissioned Ethereum-based voting platform specifically tailored for campus-level elections. The project involves comprehensive literature synthesis, architecture design including authentication and privacy, risk mitigation, rigorous evaluation methods, anticipated contributions, required resources, and an implementation timeline. The deliverables include a functional prototype, open datasets, and a research paper prepared for submission to IEEE Blockchain.

1 Introduction & Motivation

Despite the historical reliability of paper ballots, traditional campus voting systems continue to face significant challenges in terms of auditability, efficiency, and transparency. Recent reports from various institutions highlight substantial issues around vote integrity and the practical difficulties of auditing election outcomes accurately and swiftly (Specter et al., 2020; Mannonov & Myeong, 2023). For example, manual recounts for student government elections at several U.S. universities have revealed discrepancies that are time-consuming and costly to resolve, typically requiring days or even weeks of verification and investigation.

Blockchain technology, and specifically Ethereum's permissioned blockchain models, presents a viable solution by providing a decentralized and cryptographically verifiable infrastructure. This technology ensures immutable and auditable election results, significantly enhancing trust and transparency compared to conventional centralized electronic voting systems (Kshetri & Voas, 2018; Dagher & et al., 2018). In Ethereum-based voting systems, cryptographic methods such as commit–reveal and homomorphic encryption have demonstrated substantial promise for preserving voter privacy while enabling public verifiability of the final election tally (Adida, 2008; Cortier et al., 2014; Dagher & et al., 2018).

Institutional goals within higher education further reinforce the adoption of blockchain technology. The U.S. Department of Education underscores the necessity for compliance with regulations like the Family Educational Rights and Privacy Act (FERPA), emphasizing transparency and protection of student data. Ethereum-based blockchain voting aligns strongly with these goals by providing cryptographic guarantees for voter privacy, integrity of electoral data, and facilitating transparent audits without compromising individual voter confidentiality (Sharma et al., 2022).

Moreover, the broader movement toward digital transformation in higher education institutions, documented extensively by EDUCAUSE and similar organizations, highlights the critical role of innovative digital solutions to streamline administrative operations, reduce costs, and improve student experiences. Blockchain-based voting not only aligns with these strategic digital transformation goals but also delivers measurable operational efficiencies. Comparative studies indicate that blockchain voting reduces the administrative overhead associated with traditional paper-based ballots, potentially cutting costs by up to 50% and decreasing result processing time from days to mere hours (Jafar et al., 2021; West Virginia Secretary of State, 2018).

This project proposes implementing a small-scale Ethereum blockchain voting system tailored specifically for campus elections at Earlham College. The primary motivations for this project include demonstrating tangible improvements in transparency, efficiency, and user trust, while also evaluating the practical scalability and security of blockchain technologies within an academic setting. Additionally, this system aims to serve as a proof-of-concept to encourage broader discussions about secure digital governance systems at academic institutions nationwide.

The following proposal will detail the comprehensive literature synthesis guiding this initiative, outline the intended system architecture—including voter authentication and privacy considerations—address critical risk mitigation strategies, and propose rigorous evaluation methods to assess both technical performance and user satisfaction.

2 Related Work

Blockchain-based voting systems have attracted significant scholarly attention as potential replacements for traditional voting methodologies, especially in contexts demanding transparency and auditability. Recent developments have demonstrated multiple architectures, each with distinct advantages and trade-offs regarding privacy, performance, and complexity.

2.1 Blockchain Voting Architectures

Blockchain voting schemes typically employ one of three main cryptographic tallying methods: **commit–reveal**, **homomorphic encryption**, and **zk-SNARKs**.

Commit–Reveal Schemes. This method involves voters submitting cryptographic commitments of their votes, later revealing them to be verified and counted. BroncoVote, an Ethereum-based campus election system piloted at Santa Clara University, utilized this approach due to its straightforward implementation and low computational overhead (Dagher & et al., 2018). While commit–reveal is relatively easy to implement and highly scalable, it provides moderate privacy protections, as votes become public after the reveal phase (McCorry et al., 2017).

Homomorphic Tallying Schemes. Homomorphic encryption allows vote aggregation directly on encrypted ballots, maintaining voter privacy throughout the tallying process. Helios pioneered this approach in electronic voting, significantly improving voter privacy and auditability but with higher computational overhead, limiting its scalability in large elections (Adida, 2008; Cortier et al., 2014). Homomorphic schemes typically require careful cryptographic management, including key ceremonies, to ensure security and integrity.

zk-SNARK Schemes. Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs) provide the strongest available privacy guarantees by allowing voters to prove the validity of their ballots without revealing their choices or identities. While promising, the complexity and computational demands associated with zk-SNARK schemes make them less suitable for large-scale elections without considerable infrastructure investment. Academic experiments like OzKoin and Hawk have demonstrated theoretical feasibility but have highlighted substantial practical challenges (Tsang et al., 2016; Kosba et al., 2016).

Table 1 summarizes key comparisons among these methodologies:

2.2 Permissioned Ethereum Pilots in Campus Elections

Numerous academic institutions have explored permissioned Ethereum blockchain systems for campus elections, providing valuable insights into their practical implementation. For example, BroncoVote at Santa Clara University successfully handled approximately 1,200 student votes, demonstrating the practical applicability of blockchain for campus-scale elections, despite some initial friction with wallet management and voter onboarding (Dagher & et al., 2018). Similarly, Tsukuba City conducted a blockchain-based election for social programs using Ethereum on AWS infrastructure, achieving robust voter turnout and demonstrating blockchain's applicability beyond purely academic environments (Tsukuba City & AWS, 2018).

Pilots such as these highlight critical factors for successful deployment, including clear voter education, streamlined authentication workflows, and proactive trust-building measures. Empirical data suggest that user adoption significantly increases when voters perceive transparency and trustworthiness in the voting process, even if the onboarding experience requires additional initial effort (Mannonov & Myeong, 2023; Schiarelli & Dupuis, 2023).

2.3 User Experience and Onboarding Challenges

Ethereum-based decentralized applications (DApps) typically require users to interact through wallet interfaces, such as MetaMask. Studies indicate significant user drop-off rates during the initial onboarding process—specifically, wallet installation and private-key management phases. For instance, an empirical usability study conducted across several universities reported a 30–40% drop-off rate during initial wallet setup stages, primarily due to complexity and uncertainty about security practices (Mannonov & Myeong, 2023).

To mitigate this friction, streamlined UI/UX design patterns have emerged, advocating wizard-style, step-by-step workflows, embedded wallet solutions, and contextual help throughout the onboarding process. For example, Polys, a blockchain voting application from Kaspersky Lab, introduced user-friendly interfaces with clear, step-by-step guidance, significantly reducing setup time and enhancing user completion rates (Lab, 2019).

2.4 Security and Vulnerability Lessons

Security audits of early blockchain voting platforms, including notable examples such as Voatz and Polys, have uncovered vulnerabilities that underscore the importance of rigorous security practices. Voatz's mobile voting pilot during West Virginia elections

Table 1: Comparison of Blockchain Voting Tallying Methods

Method	Privacy	Scalability	Complexity
Commit–Reveal	Moderate	High	Low
Homomorphic Tallying	High	Moderate	High
zk-SNARK	Very High	Low	Very High

revealed substantial vulnerabilities stemming from insufficient cryptographic validation and weak endpoint security, resulting in significant media coverage and reduced public trust (Specter et al., 2020; West Virginia Secretary of State, 2018).

Furthermore, static and dynamic analysis of Ethereum smart contracts for voting applications has identified common vulnerabilities, including reentrancy attacks, improper access control, and integer overflow errors. Tools such as MythX, Slither, and Echidna have become standard practice for Ethereum contract auditing, providing automated detection of common smart-contract vulnerabilities and significantly reducing manual analysis overhead (Moreno-Sánchez et al., 2018; Koch et al., 2018).

Overall, these studies underscore the necessity of a multi-layered approach to blockchain security that encompasses code audits, threat modeling, proactive vulnerability management, and continuous user education.

The insights from existing literature strongly suggest that while blockchain voting systems present significant advantages over traditional methods, they require careful balancing of privacy, usability, and security considerations. This proposal draws on these lessons to design a system that addresses these trade-offs explicitly, optimizing for transparency, voter trust, and operational efficiency within a manageable institutional scope.

3 System Design

This section describes the end-to-end architecture for my permissioned Ethereum voting system, including the blockchain network setup, smart contracts, authentication flow, privacy safeguards, user interface design, and performance objectives.

3.1 Permissioned Ethereum Network

I will deploy a small permissioned Ethereum network using the Clique Proof-of-Authority (PoA) consensus protocol. Clique offers fast block times and low computational overhead, ideal for a campus election with under 1 000 transactions per day.

- **Consensus Mechanism:** Clique PoA configured for a 1-second target block time.
- **Validator Nodes:** Three authority nodes operated by:
  1. The Computer Science Department
  2. The Student Government
  3. A designated Faculty Auditor (Charlie? Yunting?)
- **Network Topology:**
  - *Validator nodes* produce and validate blocks.
  - An *archival node* provides a public JSON-RPC endpoint for auditors and researchers.
  - *Light clients* run in users' browsers via MetaMask or an embedded wallet.

- **Hosting Options:**
  - On-premises VMs (4 vCPU, 8 GB RAM, 100 GB SSD) managed by campus IT.
  - AWS Educate t3.medium instances (2 vCPU, 4 GB RAM, elastic storage), covered by student credits.

3.2 Smart-Contract Suite

The core functionality is implemented in three Solidity contracts:

RegistryContract
: Maintains a Merkle root of eligible voter addresses. Eligibility is enforced by verifying inclusion proofs on-chain.

BallotContract
: Implements a two-phase *commit–reveal* protocol:

  1. *Commit phase:* Voter sends commitHash $= H(\text{vote} \,\|\, \text{nonce})$.
  2. *Reveal phase:* Voter submits $(\text{vote}, \text{nonce})$ which the contract verifies against the stored hash.

TallyContract
: Aggregates revealed ballots using off-chain Paillier encryption. After all reveals, validators jointly decrypt the sum via a 2-of-3 threshold scheme.

Contracts will be written in Solidity 0.8, tested with Hardhat, and leverage OpenZeppelin's libraries for access control and safe arithmetic.

3.3 Authentication Flow

User authentication proceeds in three steps:

1. **Email Verification:** Voter enters their campus email; backend sends a one-time link (Magic Link) containing a signed JSON Web Token (JWT) valid for ten minutes.
2. **Wallet Binding:** Clicking the Magic Link opens the DApp. The client wallet (MetaMask or embedded) signs a challenge containing the JWT. Upon verification, the backend calls `RegistryContract.register(address)` to whitelist that address.
3. **Vote Casting:** The whitelisted address may send one commit transaction to `BallotContract`. Attempts to commit more than once are rejected on-chain.

A recovery workflow allows voters who lose access to their wallet keys to request re- issuance via email; the system invalidates the old address and registers a new one, logging all changes for audit.

3.4 Privacy and Auditability

To protect voter privacy while enabling transparency:

- *Commit–reveal* hides vote choices until the reveal phase.
- Ballots are Paillier-encrypted off-chain; on-chain storage contains only ciphertexts and commitment hashes.
- A threshold decryption ceremony among validators yields the final tally, ensuring no single authority can decrypt alone.
- Future extension: integrate a zk-SNARK circuit to prove validity of each commitment without revealing the vote or nonce, though this is scoped as future work.

All on-chain events (`Registered`, `Committed`, `Revealed`, `Outcome`) are public, enabling third-party auditors to verify the complete voting process.

3.5 User Interface

A wizard-style DApp provides a streamlined experience:

1. **Login Screen:** Collects email and triggers Magic Link flow.
2. **Wallet Link Screen:** Guides users to sign the JWT challenge via MetaMask or embedded wallet.
3. **Voting Screen:** Presents ballot options; upon selection, automatically sends the commit transaction.
4. **Verification Screen:** After reveal phase, users enter their receipt (commit hash) to confirm inclusion in the tally.

Contextual tooltips, a progress bar, and an optional tutorial video ensure that average end-to-end voting time remains under 120 seconds. The interface is fully responsive for desktop and mobile browsers.

3.6 Performance Objectives

The system is designed to meet the following targets on a campus network:

Block Time ≤ 1 s (Clique default).

Throughput $\geq$ 10 transactions/s (enough for 2 000 voters in under 4 minutes at peak).

Transaction Latency Commit plus reveal round-trip < 5 s in 95^{th} percentile.

Gas Cost Under \$0.01 per vote on the PoA network.

Monitoring via Prometheus and Grafana will capture these metrics during pilot elections.

3.7 Extensibility

The architecture is modular:

- Contracts use a proxy pattern for upgradeability.
- Authentication can pivot from email-based tokens to campus SSO if needed.
- Privacy layer can be enhanced with zk-SNARK proofs once performance evaluations are complete.
- RESTful APIs expose raw and aggregated data for third-party dashboards or mobile clients.

This design balances security, performance, and usability, while providing clear paths for future enhancements.

4 Security & Threat Modeling

Ensuring the security of a blockchain voting system requires a comprehensive threat analysis and rigorous testing regimen. I will apply the STRIDE framework (Shostack, 2014) to identify and mitigate threats specific to my architecture.

4.1 STRIDE Threat Analysis

4.1.1 Spoofing

Threat: An attacker impersonates a legitimate voter or node authority.

Mitigation:

- Email-based Magic Links tied to campus addresses for initial authentication.
- Wallet signature verification of JWTs ensures only the owner of a private key can register a vote.
- Validator nodes' PoA keys stored in secure HSM or locked-down campus servers.

4.1.2 Tampering

Threat: Unauthorized alteration of ballots, smart contracts, or transaction data.

Mitigation:

- All ballots stored immutably on-chain using commit–reveal; modifications reject.
- Smart contracts undergo static analysis (Slither) and symbolic execution (MythX) before deployment.
- Continuous integration with Echidna fuzz tests to detect logic flaws.

4.1.3 Repudiation

Threat: Voters or administrators deny having performed an action (e.g., casting a vote).

Mitigation:

- On-chain event logs (`Registered`, `Committed`, `Revealed`) provide an immutable audit trail.
- Voter-held receipts (commit hashes) enable individual non-repudiation.

4.1.4 Information Disclosure

Threat: Leak of sensitive data, such as vote choices before tally or private keys.

Mitigation:

- Commit–reveal protocol hides vote contents until the reveal phase.
- Off-chain Paillier encryption and threshold decryption prevent early tally decryption.
- Secure storage of private keys; recommend using hardware wallets for high-assurance setups.

4.1.5 Denial of Service

Threat: Flooding the network with bogus transactions or overwhelming servers.

Mitigation:

- Permissioned network restricts who can send transactions.
- Rate limiting at the API gateway; CAPTCHA or email rate limits for registration.
- Monitoring and alerting via Prometheus/Grafana for unusual traffic spikes.

4.1.6 Elevation of Privilege

Threat: An attacker gains administrative control of contracts or validator nodes.

Mitigation:

- Multi-signature (2-of-3) for critical administrative functions.
- Access controls in smart contracts via OpenZeppelin's `Ownable` and `AccessControl`.
- Regular audits of validator node configurations and key management procedures.

4.2 Lessons from Early Blockchain Voting Pilots

Security audits of real-world blockchain voting applications have revealed critical vulnerabilities:

- **Voatz Mobile Voting Pilot** (West Virginia, 2018): A security analysis uncovered flaws in the mobile client that allowed tampering with vote casting and potential privacy breaches (Specter et al., 2020). Key lessons include the necessity of end-to-end cryptographic proofs on the client and rigorous vetting of all client-side code.
- **Polys Platform** (Kaspersky Lab, 2019): While offering user-friendly interfaces, Polys suffered from insufficient hardening of its web endpoints and lacked comprehensive API rate limiting, leaving it exposed to denial-of-service and injection attacks (Lab, 2019).

These case studies underscore the importance of both smart-contract security and robust client–server safeguards.

4.3 Ethereum-Specific Pen-Testing and Tools

For smart-contract analysis and DApp security testing, I will employ a layered toolchain suitable for beginners and scalable to professional standards:

- **Slither** (Trail of Bits): Static analysis to catch common Solidity vulnerabilities (reentrancy, uninitialized state, etc.) (Moreno-Sánchez et al., 2018).
- **MythX** (ConsenSys): Cloud-based symbolic execution and fuzzing scans. Free trial tier allows up to 5 analyses/month for prototyping (Koch et al., 2018).
- **Echidna** (Trail of Bits): Property-based fuzz testing of smart contracts, generating sequences of transactions to break invariants (Moreno-Sánchez et al., 2018).
- **Mythril** (MythX's open-source engine): Local symbolic analysis for deeper inspection without API limits.

Additional best practices:

- Conduct a STRIDE-based threat modeling workshop early in development.
- Integrate security tests into CI/CD pipelines (GitHub Actions or Jenkins).
- Allocate dedicated time in the timeline for external penetration tests by CS security-club volunteers.

Combining STRIDE analysis, lessons from prior pilots, and a robust toolchain ensures that my system's security posture is rigorously validated before any campus deployment.

5 Risk & Mitigation

6 Evaluation Plan

My evaluation combines user-centered studies with rigorous technical benchmarks across three campus-wide pilot trials.

6.1 Pilot Trials Overview

I will conduct *three* successive pilot elections, each open for one week, to iteratively refine and validate the system:

1. **Pilot 1 (Alpha):** Small-scale test with 15–20 CS students on a mock election.
2. **Pilot 2 (Beta):** Medium-scale test with 50–75 mixed students/faculty, real campus SGA election.
3. **Pilot 3 (Release):** Full-scale test with up to 200 participants (open call to campus community).

After each pilot, I will incorporate feedback and fix issues before the next trial.

6.2 User Experience Study

I will assess usability and trust via standard instruments:

System Usability Scale (SUS)
: Ten-item questionnaire yielding a 0-100 usability score (Brooke, 1996).

Perceived Trust
: Seven-point Likert scale adapted from Mannonov and Myeong's TAM study (Mannonov & Myeong, 2023).

Task Completion Time
: Time from login to successful vote commit (target $<$120 s).

Post-Task Interviews
Semi-structured feedback on pain points, e.g. wallet linking or Magic Link delays.

I will recruit 15–20 participants per pilot, ensuring diverse representation (majors, tech backgrounds). Hypotheses include:

- H1: Mean SUS score $\geq$70 (good usability).
- H2: Average trust rating increases by $\geq$1 point post-vote.
- H3: 80% of participants complete voting in under 2 minutes.

6.3 Technical Benchmarks

Performance will be measured on my PoA network using Prometheus + Grafana:

Each pilot will generate logs for offline analysis and comparison against the above targets.

6.4 Security and Penetration Testing

To validate robustness, I will perform:

- **Smart-Contract Audits** using Slither, MythX, and Echidna (covering reentrancy, access control, overflow).
- **Client-Side Penetration Tests** simulating MITM and malicious browser extensions on a campus lab machine.
- **STRIDE Review** update post-pilot to capture any emergent threats.

A summary report of findings and mitigations will be produced after each pilot, ensuring continuous security hardening.

7 Resources & Infrastructure

This project leverages both campus-managed hardware and cloud resources for development, testing, and pilot deployments, alongside freely available security-analysis tooling under academic licenses.

7.1 Compute Infrastructure Options

On-prem VMs suffice for up to three validator nodes plus one archival node. AWS Educate, however, simplifies provisioning and maintenance—ideal for short-term pilot trials and load testing.

7.2 Security-Analysis Tooling

Under academic and open-source programs, I have free or low-cost access to:

- **Slither** (Trail of Bits) – Static analysis for common Solidity issues.
- **MythX** (ConsenSys) – Cloud-based symbolic execution; academic tier includes 50 analyses/month.
- **Echidna** (Trail of Bits) – Property-based fuzzer for smart-contract invariants.
- **Mythril** – Local symbolic analysis engine; fully open-source.

CI pipelines will invoke these tools on every commit, covering early static checks, automated fuzzing, and final audits without extra cost.

7.3 Additional Resources

- **Campus CS Lab Access:** Dedicated machines with hardware wallets for integration and user support.

Metric	Target	Tooling
Block time (mean)	$\leq$1 s	Built-in JSON-RPC, Grafana dashboard
Throughput	$\geq$10 tx/s	Eth-bench
Commit+Reveal Latency	$\leq$5 s (95th)	Custom JS script
Gas usage per vote	<60 k gas	Hardhat gas reporter
Node recovery time	$\leq$2 min	Simulated crash/restart

Table 3: Key Technical Performance Metrics

- **Writing Center:** Assistance refining technical sections and overall prose.
- **Disability Services:** Guidance ensuring UI accessibility meets WCAG 2.1 standards.

8 Deliverables & Dissemination

Deliverables include a deployable Ethereum voting DApp prototype, structured open datasets following FAIR principles, and a research paper formatted for submission to IEEE Blockchain. Open repositories (e.g., GitHub, Figshare) will be used to host datasets, ensuring reproducibility.

9 IRB Review and Compliance

Because the user-experience study involves human subjects, Institutional Review Board (IRB) approval is required before any pilot trial. Earlham College's IRB ensures that all research involving human participants adheres to ethical principles and federal regulations (45 CFR 46) and college policies.

Key IRB considerations for this project include:

- **Submission Type:** The proposed user surveys, interviews, and usage logging qualify as *expedited review* or possibly *exempt* (minimal risk) under categories such as "surveys and interviews" and "benign behavioral interventions."
- **Required Documentation:**
  - Completed IRB *Exemption Request Form* or *Expedited Request Form*.
  - Informed Consent Form outlining the purpose, procedures, risks, benefits, confidentiality, and voluntary nature of participation.
  - Debriefing Statement template for post-pilot communications.
- **Timeline:**
  - *Expedited/Exempt* proposals are typically reviewed within one week.
  - *Full* proposals require submission by Tuesday for review at the Friday board meeting (allow two weeks before the pilot start).
- **Data Retention and Confidentiality:**
  - All survey responses and usage logs will be stored on encrypted campus servers and retained only for the duration of the three pilot trials plus a three-month archival period, after which raw identifiers will be purged.
  - Only aggregated, anonymized metrics will be published; no personally identifiable information (PII) will appear in any reports or datasets.

I will submit the IRB application no later than six weeks before Pilot 1 and schedule

any required ethics-training modules. Close coordination with the IRB convener (Dr. Kyle Henning) will ensure timely approval and compliance.

10 Conclusion

This proposal has outlined a comprehensive plan to design, implement, and evaluate a permissioned Ethereum voting platform tailored for campus elections. By integrating robust authentication via email-issued tokens, a commit–reveal voting workflow, and a permissioned PoA network, the system addresses key challenges in integrity, privacy, and scalability. My multi-phase pilot trials, coupled with mixed-methods evaluation (usability surveys, trust metrics, performance benchmarks, and security audits), will yield actionable data on real-world deployment at Earlham College.

The anticipated deliverables include:

- A fully functional DApp prototype with smart contracts and React front-end.
- An end-to-end evaluation framework and open dataset of performance and user-trust metrics.
- Reusable security modules (Slither/MythX/Echidna configurations) and CI pipelines for future blockchain projects.
- Documentation of best practices for campus-scale blockchain elections, including governance recommendations and contingency plans.

Beyond the campus pilot, this work will contribute to the broader field by providing:

1. Empirical evidence on trade-offs between usability and security in blockchain voting.
2. A blueprint for other institutions to launch secure, transparent, and auditable digital elections.
3. Open-source artifacts (code, data, reports) that adhere to open-science principles, facilitating replication and further research.

In closing, this project not only aims to modernize campus governance but also to advance scholarly understanding of blockchain in voting applications—paving the way for more trustworthy and user-friendly election systems in higher education and beyond.

References

Adida, B. (2008). Helios: A feasible secure remote voting system. In *Proceedings of the usenix security symposium.*

Brooke, J. (1996). Sus: A "quick and dirty" usability scale. *Usability Evaluation in Industry*, 189–194.

Cortier, V., Galindo, D., Glondu, S., & Izabachène, M. (2014). Election verifiability for helios under weaker trust assumptions. In *Proceedings of the european symposium on research in computer security (esorics)* (pp. 327–344).

Dagher, G., & et al. (2018). Broncovote: Secure voting system using ethereum's blockchain. In *Proceedings of the international conference on information systems security and privacy (icissp)* (pp. 221–230).

Earlham College Institutional Review Board. (2025). *Institutional review board guidelines & procedures.* `https://earlham.edu/academics/specialized-programs/collaborative-research/institutional-review-board/`. (Accessed May 2025)

Jafar, U., Haider, A., & Ali, R. (2021). Blockchain for electronic voting sys-

tem—review and open research challenges. *Sensors*, *21*(17), 5874.

Koch, C., Wachsmann, S., & Boneh, D. (2018). Zokrates: A toolbox for zksnarks on ethereum. In *Proceedings of the 2018 ieee european symposium on security and privacy workshops* (pp. 47–54).

Kosba, A., Miller, A., Shi, E., Wen, Z., & Papamanthou, C. (2016). Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. *Proceedings of the IEEE Symposium on Security and Privacy*, 839–858.

Kshetri, N., & Voas, J. (2018). Blockchain-enabled e-voting. *IEEE Software*, *35*(4), 95–99.

Lab, K. (2019). *Polys: Blockchain-based voting platform for online and offline elections.* `https://polys.me/assets/Polys_Whitepaper.pdf`.

Mannonov, K. M., & Myeong, S. (2023). Citizens' perception of blockchain-based e-voting systems: A tam approach. *Sustainability*, *16*(11), 4387.

McCorry, P., Shahandashti, S. F., & Hao, F. (2017). A smart contract for boardroom voting with maximum voter privacy. In *Proceedings of the 2017 acm workshop on privacy in the electronic society* (pp. 65–75).

Moreno-Sánchez, J., Atlas, K., & Kiayias, A. (2018). Slate: Statically analyzable smart contracts. In *Proceedings of the 2018 ieee symposium on security and privacy* (pp. 395–414).

Schiarelli, V., & Dupuis, M. (2023). Evaluating the public perception of a blockchain-based election. In *Proceedings of the acm sigite conference on information technology education* (pp. 121–130).

Sharma, T., Gupta, R., & Patel, S. (2022). Blockchain-based e-voting systems: A technology review. *Electronics*, *13*(1), 17.

Shostack, A. (2014). *Threat modeling: Designing for security.* Wiley.

Specter, M. A., Koppel, J., & Weitzner, D. (2020). The ballot is busted before the blockchain: A security analysis of voatz, the first internet voting application used in u.s. federal elections. In *Proceedings of the 29th usenix security symposium* (pp. 335–348).

Tsang, P., Quinlivan, S., & Li, X.-Y. (2016). Ozkoin: A zero knowledge ethereum voting system. In *Proceedings of the 2016 acm ccs workshop on blockchain security and privacy* (pp. 19–26).

Tsukuba City & AWS. (2018). *Blockchain and personal id-based online voting in tsukuba's society 5.0 trials.* Case Study.

West Virginia Secretary of State. (2018). *West virginia mobile voting pilot.* Press Release and Reports.