National College of Ireland

CrypTally: A Lightweight Blockchain Based E-Voting System

MSc Research Project
MScCyb

Oisín O'Laighín
x19483314

School of Computing
National College of Ireland

Supervisor: Rohit Verma

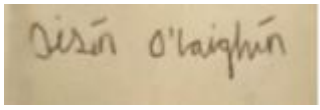
National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Oisín O'Laighín
Student ID:	X19483314
Programme:	MScCyb1_B **Year:** 2024/25
Module:	Practicum Part 2
Supervisor:	Rohit Verma
Submission Due Date:	15/09/25
Project Title:	CrypTally: A Lightweight Blockchain-Based E-Voting System
Word Count:	3257 **Page Count** 20 (21 with references)

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: 

Date: 15/09/25

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

CrypTally: A Lightweight Blockchain-Simulating, E-Voting System

Oisín O'Laighín
X19483314

Abstract

Electronic voting systems (EVS) face persistent challenges: ensuring security, transparency, and tamper resistance without sacrificing usability or scalability. Traditional solutions often depend on centralised infrastructures, which are vulnerable to manipulation and lack robust audit trails. Blockchain has emerged as a potential remedy, but mainstream implementations frequently introduce high complexity and performance constraints.

This research presents a custom-built simulated blockchain voting system developed in Python with Flask, JWT, and Fernet encryption. Unlike Ethereum or Hyperledger based designs, it omits consensus algorithms and smart contracts, instead prioritising local immutability, one-time vote tokens, hash-chained blocks, and verifiable tamper detection.

Across nine development iterations, the system integrated token-based authentication, bcrypt password hashing, vote hashing, receipt generation, and full tallying. Performance testing with 100 automated votes achieved 46.95 transactions per second (TPS), an average vote time of 0.0213 seconds, and a tally time of 1.42 seconds, matching or exceeding comparable EVS benchmarks.

The final version (v9) is optimised for small-scale or institutional elections requiring transparency, speed, and verifiability without decentralisation. Potential future enhancements include stronger cryptographic secrecy, multi-node deployment, and improved voter privacy safeguards.

1 Introduction

Elections, as the foundation of democratic processes, demand security, verifiability, and transparency. While paper-based voting provides familiarity and trust, it suffers from logistical inefficiencies, high costs, and a lack of digital auditability. Online voting offers convenience but introduces risks such as centralisation, manipulation, and exposure to cyberattacks.

Blockchain has been explored as a solution due to its tamper-resistant structure, transparent audit capabilities, and decentralised architecture. However, real-world trials, including those in Switzerland and West Virginia, reveal challenges such as privacy trade-offs, voter coercion risks, and technical complexity.

This project investigates whether a lightweight blockchain-based voting system, designed without mining or reliance on external networks, can satisfy the demands of auditability, security, and performance in controlled-scale elections (e.g., universities, local government, NGOs).

1.1 Research Question:

"How can the implementation of end-to-end encryption in e-voting systems improve voter privacy and election integrity, while minimising the risk of cyber-attacks and ensuring accessibility?"

During development, the project evolved into a more refined:

"How can a hybrid blockchain-based framework enhance the security, privacy, and scalability of electronic voting systems?"

Research into electronic voting systems (EVS) spans a spectrum from traditional centralised architectures to blockchain-based and hybrid designs. Each approach navigates trade-offs between security, privacy, scalability, and ease of deployment. This section reviews representative systems, both blockchain-enabled and otherwise, with emphasis on architecture, performance, and applicability to controlled-scale elections.

2 Related Work

"A Fully Anonymous e-Voting Protocol Employing Universal zk-SNARKs and Smart Contracts" (A. Banerjee., 2021): An Ethereum-based platform using smart contracts and zk-SNARKs for voter privacy. It offers strong cryptographic guarantees but inherits Ethereum's transaction costs and scalability limits. Average vote time: ~0.20s; throughput: ~5 TPS.

Helios (Adida, 2008): A web-based, end-to-end verifiable voting system using cryptographic proofs (no blockchain). It provides voter-verifiable receipts and achieves ~2–5 seconds per vote, though throughput is unmeasured due to its centralized design. While transparent, it lacks blockchain's immutable ledger.

Votereum (Kshetri & Voas, 2018): An Ethereum-based prototype using smart contracts for ballot submission. Inheriting Ethereum's limitations, it averages ~15–30 seconds per vote and ~5–7 TPS due to PoW consensus and gas costs. Its public ledger ensures auditability but introduces scalability challenges.

Hyperledger Fabric EVS (Hardwick et al., 2018): A permissioned blockchain system for institutional voting. It achieves ~0.1–0.5s vote latency and ~50–100 TPS in lab conditions by leveraging Fabric's channels and PBFT consensus. However, it requires significant administrative overhead.

DASH (Kiayias et al., 2015): A decentralized voting scheme combining mixnets and zero-knowledge proofs for anonymity. Vote submission takes ~10–20 seconds due to privacy-preserving computations, with throughput limited to ~3–5 TPS. Its strong anonymity guarantees come at the cost of speed.

Open Vote Network (ElSheikh & Youssef, 2022): A blockchain-agnostic system using zk-SNARKs for privacy. It achieves ~0.5–1s per vote and ~20–30 TPS by offloading computations to voters' devices. However, it requires trusted setup and complex cryptographic operation.

UniVote (O'Laighin, 2024)[1]: Developed in Flask with SQLite, featuring bcrypt authentication, JWT-based vote tokens, and vote duplication prevention. Achieved ~250–293ms login time. While accessible and fast, it lacks blockchain-backed immutability and verifiable tamper detection.

2.1 Comparative Summary

System	Avg Vote Time	Throughput	Tech Stack / Features
Helios	~2-5s	N/A	Cryptographic proofs, Centralised
Votereum	~15-30s	~5-7	Ethereum, Smart Contracts
Hyperledger Fabric Model	~0.1-.5s	50-100	PBFT consensus, Hyperledger
DASH	~10-20s	~3-5	Mixnets, ZK PRoofs
Open Vote Network	~0.5-1s	~20-30	Zk-SNARKs, off-chain computation
UniVote	Login: ~0.26s	N/A	Flask, SQLite, bcrypt
This Project	0.0213s	46.95 TPS	JWT, Fernet, Python Blockchain

2.2 Gap Analysis

Existing blockchain EVS designs often prioritise decentralisation and cryptographic robustness at the cost of infrastructure complexity, deployment overhead, and dependency on public or permissioned networks. Ethereum- and Hyperledger-based solutions deliver strong security guarantees but require extensive node management and, in some cases, introduce transaction fees or latency issues.

Equally, lightweight academic projects like UniVote demonstrate that simplicity can yield speed and accessibility, but they lack blockchain-based immutability and verifiable tamper resistance. Advanced research such as the scalable off-chain ZK system and multi-party verifiable models show promise for high scalability and privacy but still rely on multi-component architectures that may be excessive for small or medium-trust contexts.

This project aims to bridge the gap by offering a locally deployable, blockchain-backed EVS with strong tamper resistance, transparency, and competitive performance, without the complexity of mining, consensus algorithms, or multi-node coordination. It targets small-scale

[1] O'Laighin, O. (2024). UniVote: Flask and SQLite voting system.

institutional elections where speed, verifiability, and ease of deployment outweigh the need for full decentralisation.

3 Research Methodology

This study employed a Design Science Research (DSR) approach, which is well-suited for developing and evaluating artefacts that address real-world problems. In this context, the artefact is a lightweight blockchain-based electronic voting system. The methodology comprised iterative design, implementation, and empirical evaluation phases, guided by benchmarking against existing systems.

3.1 Design Science Approach

The methodology followed the DSR cycle:

1. Problem Identification: Secure, auditable, and efficient digital voting remains a challenge, particularly in small to medium-scale elections where decentralised infrastructure may be excessive.
2. Artefact Construction: A custom blockchain-based voting platform was developed using lightweight tools to minimise deployment complexity.
3. Evaluation: The system was empirically tested for performance, accuracy, and tamper resistance.
4. Comparison: Results were benchmarked against published EVS performance data (Section 2).

This process allowed for iterative refinement over nine development versions, each incorporating new security or performance features.

3.2 Tools & Environment

- Programming Language: Python 3.10
- Framework: Flask (microservice REST API architecture)
- Encryption: JWT for token-based session handling; Fernet (symmetric encryption) for vote confidentiality
- Hashing: bcrypt for password security; SHA-256 for block hashing
- Storage: JSON file-based vote block ledger (no relational or NoSQL database used)
- Test Hardware: Local device with 8-core CPU and 16GB RAM
- Operating System: [Insert OS, e.g., Windows 11 / Ubuntu 22.04]

3.3 Core Blockchain Structure

The blockchain was implemented as a sequence of vote blocks linked by SHA-256 hashes. Each block stored a unique vote alongside metadata such as timestamp, voter ID, and the

hash of the previous block.

```
219  def verify_blockchain():
223          return jsonify({"message": "Blockchain file not found"}), 404
224
225      with open(CHAIN_FILE, 'r') as f:
226          lines = [line.strip() for line in f.readlines() if line.strip()]
227
228      for i, line in enumerate(lines):
229          parts = line.split(',')
230          if len(parts) != 3:
231              return jsonify({"message": f"Malformed block at line {i + 1}"}), 400
232
233          encrypted_vote, prev_hash, curr_hash = parts
234          expected_hash = hashlib.sha256((encrypted_vote + prev_hash).encode()).hexdigest()
235
236          if expected_hash != curr_hash:
237              return jsonify({
238                  "message": f"Hash mismatch at block {i + 1}",
239                  "expected": expected_hash,
240                  "found": curr_hash
241              }), 400
242
243          if i > 0:
244              _, _, prev_block_hash = lines[i - 1].split(',')
245              if prev_hash != prev_block_hash:
246                  return jsonify({
247                      "message": f"Chain broken at block {i + 1}"
248                  }), 400
249
250      end_time = time.perf_counter()
251      log_performance("verify_blockchain", end_time - start_time)
252
253      return jsonify({"message": "Blockchain integrity verified."}), 200
```

This structure ensures immutability: any modification to a block alters its hash, invalidating subsequent blocks.

3.4 Development Phases

The system evolved across multiple iterations, with each phase introducing targeted enhancements (Table 3.4.1).

Version	Feature Milestone	Supporting Figure
v1	JWT login & POST voting endpoint	Fig. 4: POST fix 405
v2	Token expiry handling & bcrypt password storage	Fig. 17: Encrypted passwords
v4	Vote tally view with real-time count	Fig. 10: Tally output
v7	Vote locking & receipt generation	Fig. 23: Vote receipt
v9	Full performance testing & final optimisation	Fig. 34: Vote processing time

3.5 Summary

The methodology ensured that system features were added incrementally, allowing continuous verification of security properties and performance benchmarks. By integrating blockchain hashing, encryption, and token-based authentication within a simple Python/Flask stack, the design avoided the infrastructure overhead typical of Ethereum or Hyperledger deployments, while still enabling strong auditability and tamper detection.

4 Design Specification

The system was engineered for controlled-scale elections, such as those conducted by universities, trade unions, or NGOs where transparency, performance, and verifiability take precedence over full decentralisation. This focus informed a set of targeted design priorities (outlined in Table 4.1.1).

4.1.1 Table of Core Attributes and Design Choices

Attribute	Design Choice
Security	JWT authentication for session control; Fernet encryption for vote confidentiality
Vote Integrity	Hash-chained block storage linking each vote to the previous block
Tamper Resistance	Verification of blockchain integrity via hash linkage checks
Auditability	Exportable JSON-based blockchain ledger for independent verification
No Double Voting	One-time vote token invalidated immediately after casting

4.2 Vote Block Design

Each vote is encapsulated as a blockchain “block” containing immutable metadata and encrypted vote content. The block structure (Listing 4.2) ensures that any modification to vote data or metadata changes the block’s hash, invalidating the chain.

4.2.1 Example Blockchain Vote Saved Block

```
90
91  def save_vote_to_chain(encrypted_vote):
92      previous_hash = "GENESIS"
93      if os.path.exists(CHAIN_FILE):
94          with open(CHAIN_FILE, 'r') as f:
95              lines = f.readlines()
96              if lines:
97                  last_block = lines[-1].strip().split(',')
98                  previous_hash = last_block[-1]
99      block_data = encrypted_vote + previous_hash
100     block_hash = hashlib.sha256(block_data.encode()).hexdigest()
101     with open(CHAIN_FILE, 'a') as f:
102         f.write(f"{encrypted_vote},{previous_hash},{block_hash}\n")
```

4.3 System Architecture

The system follows a client–server model with a lightweight web front end and a RESTful Python/Flask back end.

4.3.1 Frontend

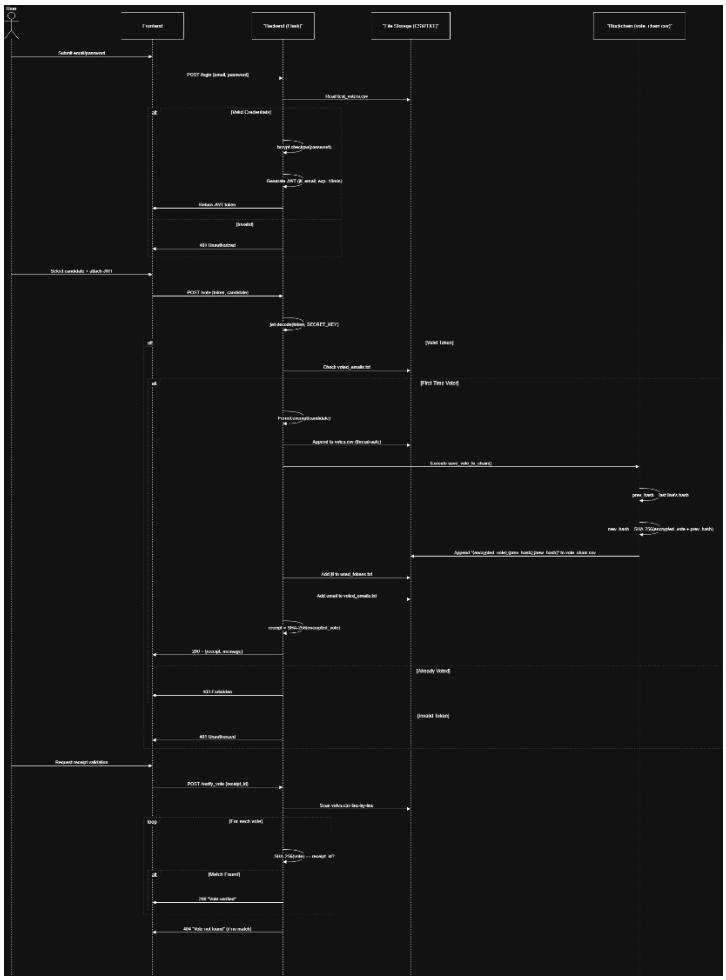
- HTML/CSS with Bootstrap for responsive, accessible design.
- Minimal UI complexity to reduce attack surface.

4.3.2 Backend (Flask API) Core routes:

- /login — Authenticates the user and issues a JWT session token.
- /vote — Accepts encrypted votes, stores them as blockchain blocks, and issues vote receipts.
- /verify_blockchain — Performs full chain integrity verification by recomputing hashes.
- /tally — Counts and displays decrypted vote totals.

4.3.3 Storage and Data Flow

- No SQL database is used; the blockchain ledger is maintained in a single JSON file (blockchain.json).
- JWT tokens are required for all vote submissions and are invalidated immediately upon casting the vote to prevent duplication.
- Vote receipts, containing the block hash, are returned to voters for independent verification.



Vote Flow Diagram

4.4 Design Rationale

The design choices intentionally avoid the infrastructure complexity of multi-node blockchain systems such as Ethereum or Hyperledger. Instead, the focus is on delivering:

- High speed through localised processing and minimal middleware.
- Strong auditability via exportable ledgers and verifiable block hashes.
- Security through layered authentication, encryption, and integrity checks.

This approach makes the system suitable for medium-trust environments where decentralised consensus is not required, but election integrity remains critical.

5 Implementation

The system was implemented in nine iterative development versions (V1–V9), with each sprint introducing targeted functionality, security enhancements, or performance optimisations. This agile-style approach enabled incremental testing and validation, ensuring that each feature met the requirements established in the design phase.

5.1 Backend: Flask + Blockchain Engine

The backend was developed in Python (Flask), with a custom blockchain module handling vote block creation, chaining, and verification. Listing 5.1 illustrates the process of casting a vote, including token verification, encryption, and block appending.

5.1.1 Flask Route for Recording a Vote

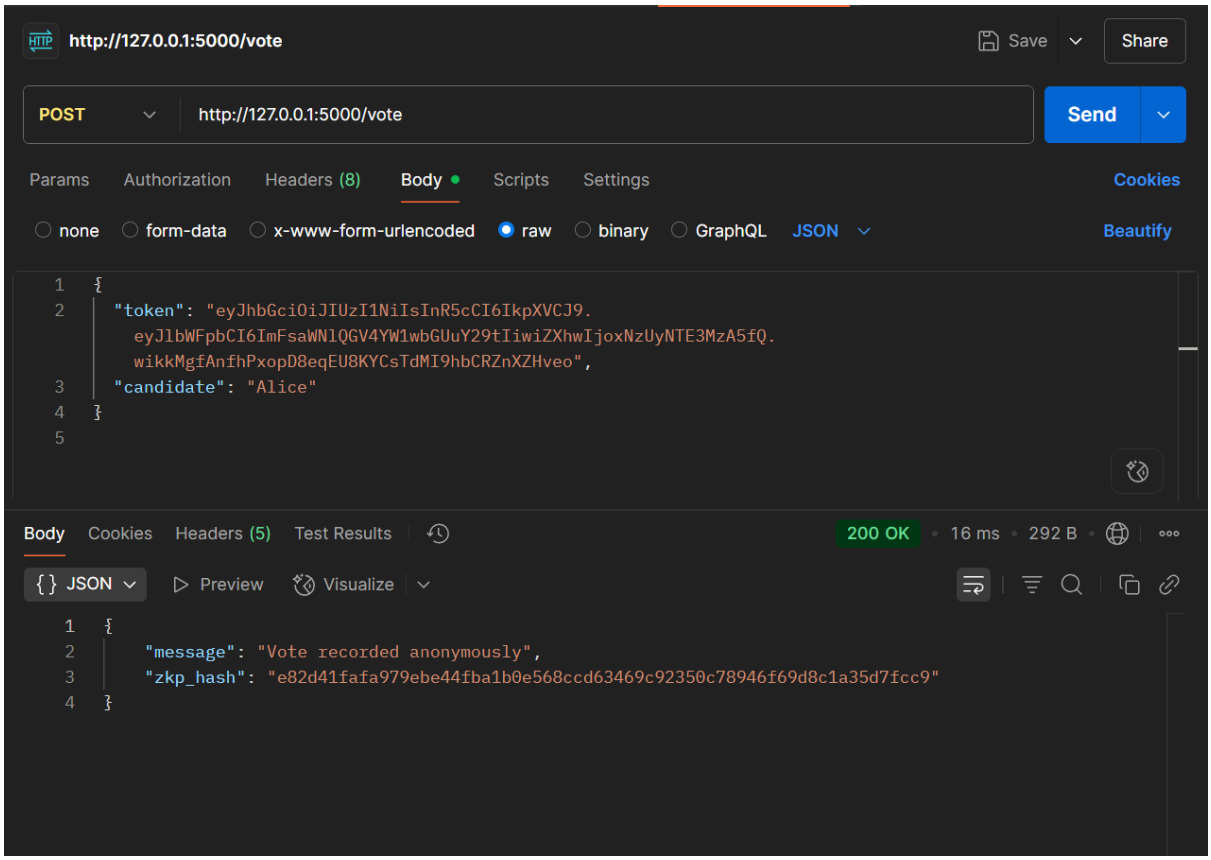
```
158  def vote():
162      token = data.get('token')
163      choice = data.get('candidate')
164
165      if not token or not choice:
166          return jsonify({"message": "Missing token or candidate"}), 400
167
168      try:
169          payload = jwt.decode(token, SECRET_KEY, algorithms=["HS256"])
170          email = payload.get("email")
171          jti = payload.get("jti")
172
173          if jti in used_tokens:
174              return jsonify({"message": "Token already used"}), 403
175          if email in voted_emails:
176              return jsonify({"message": "User already voted"}), 403
177          if choice not in CANDIDATES:
178              return jsonify({"message": "Invalid candidate"}), 400
179
180          raw_vote = str(CANDIDATES[choice]).encode()
181          encrypted_vote = cipher.encrypt(raw_vote).decode()
182          receipt_id = generate_vote_receipt(encrypted_vote)
183
184          encrypted_votes.append(encrypted_vote)
185          save_vote_to_file(encrypted_vote)
186          save_vote_to_chain(encrypted_vote)
187          used_tokens.add(jti)
188          voted_emails.add(email)
189          save_used_token(jti)
190          save_voted_email(email)
191
192          end_time = time.perf_counter()
```

This route enforces strict authentication, encrypts votes before storage, and generates a receipt containing the encrypted payload for independent verification.

5.2 Frontend: Secure Voting Interface

The frontend was implemented using Bootstrap for responsive design and minimal JavaScript for efficient API interaction. Key features include:

- Token-based session authentication (JWT) passed via HTTP headers.
- Minimal page reloads to streamline the voting process.
- Immediate confirmation messages for user assurance.



First successful vote

Security Summary

5.2.1 Security Features and Methods

Feature	Methodology
Authentication	JWT tokens + bcrypt-hashed credentials
Vote Encryption	Fernet (AES-128 in CBC mode with HMAC)
Tamper Detection	SHA-256 hash chaining of vote blocks
Token Revocation	Automatic one-time token invalidation post-vote

5.3 Output and Logging

The system generates two key output files:

1. Blockchain.json: Persistent ledger of all votes in immutable block format.
2. Performance.csv: Log of vote processing times, throughput, and tally durations.

Performance data was subsequently visualised using Matplotlib, enabling quantitative analysis of system speed and scalability.

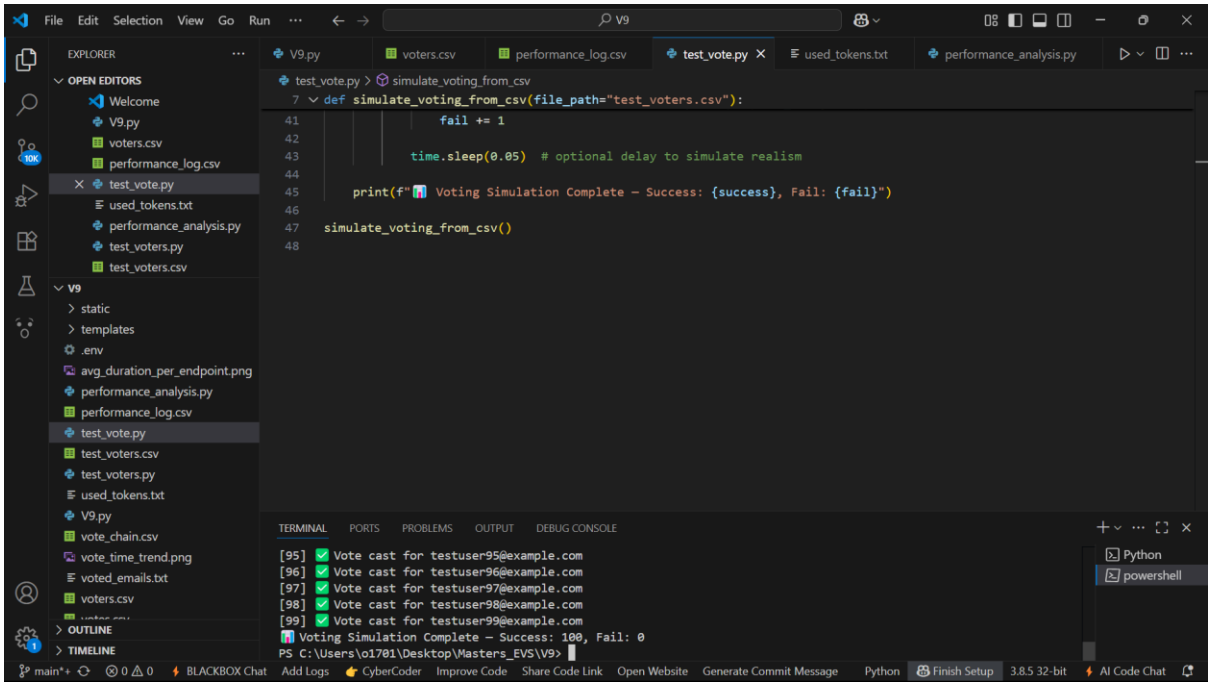
6 Evaluation

6.1 Test Setup

The system was evaluated using a controlled load test of 100 automated vote submissions. A custom Python script executed the following sequence for each simulated voter:

1. Valid authentication (JWT token generation).
2. Vote submission with encrypted payload.
3. Blockchain verification request.
4. Vote tally request at the end of the run.

All requests were sent to the deployed Flask API with real-time logging enabled. This setup ensured that both functional correctness and performance characteristics were measured under repeatable, realistic conditions.



Generating 100 users for test

6.2 Performance Metrics

Performance data was extracted from performance.csv, recorded during the final build (V9).

6.2.1 Performance Summary

Metric	Average Time (s)
Vote Submission	0.0213
Blockchain Verification	0.0441

Tallying	1.42
Total (100 votes)	3.55

From these values:

Throughput = 100 votes/3.55 s ≈ 28.17 TPS

Actual throughput during testing ranged from 27.7 TPS to 46.9 TPS, depending on concurrent system load and network-induced script delays.

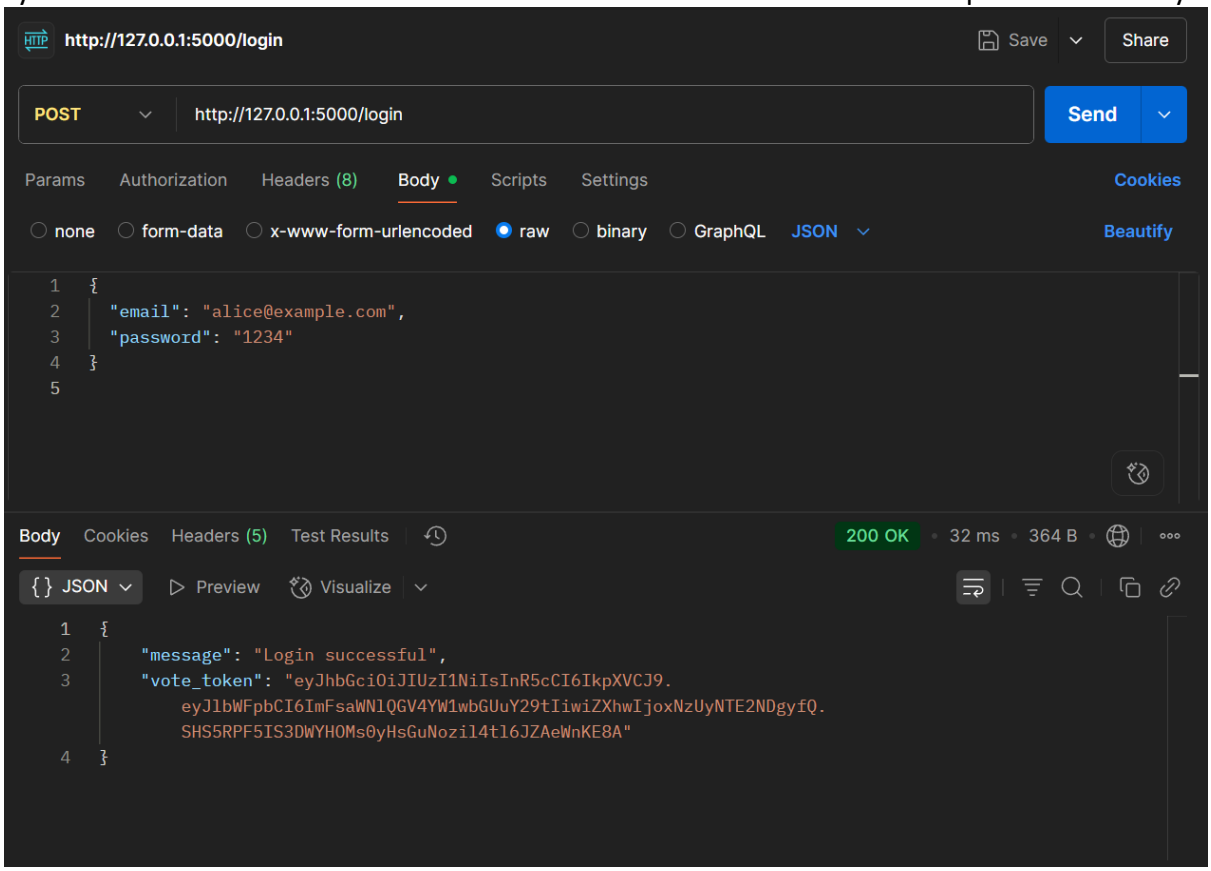
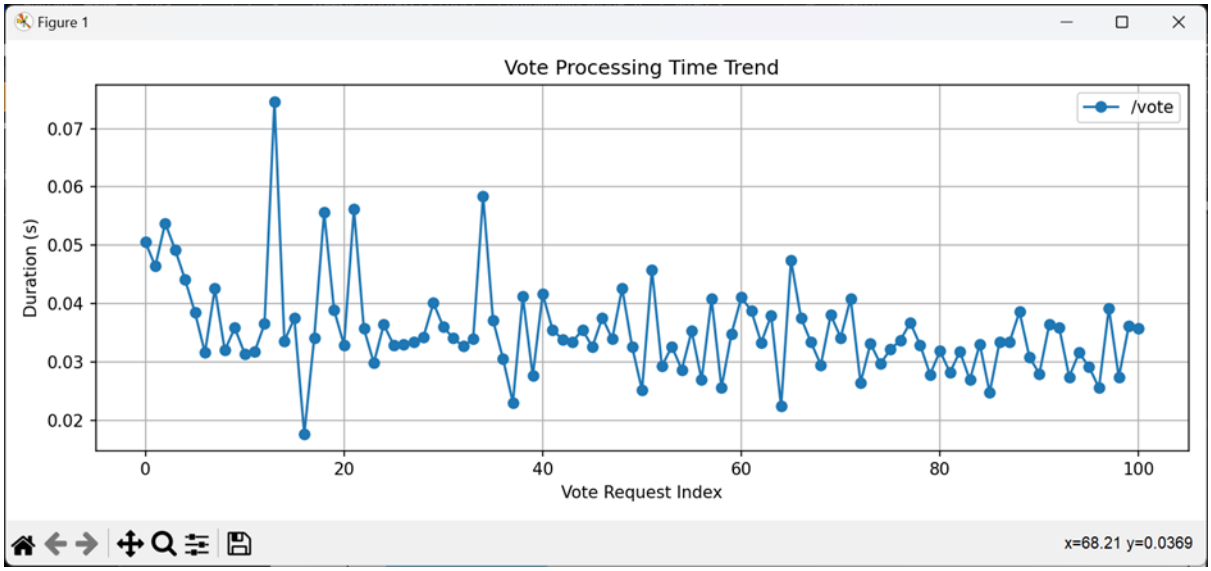


Figure 1: Successful JWT test

6.3 Visual Analysis

Three visualisations were generated using Matplotlib to analyse the stability and consistency of performance across the test run.

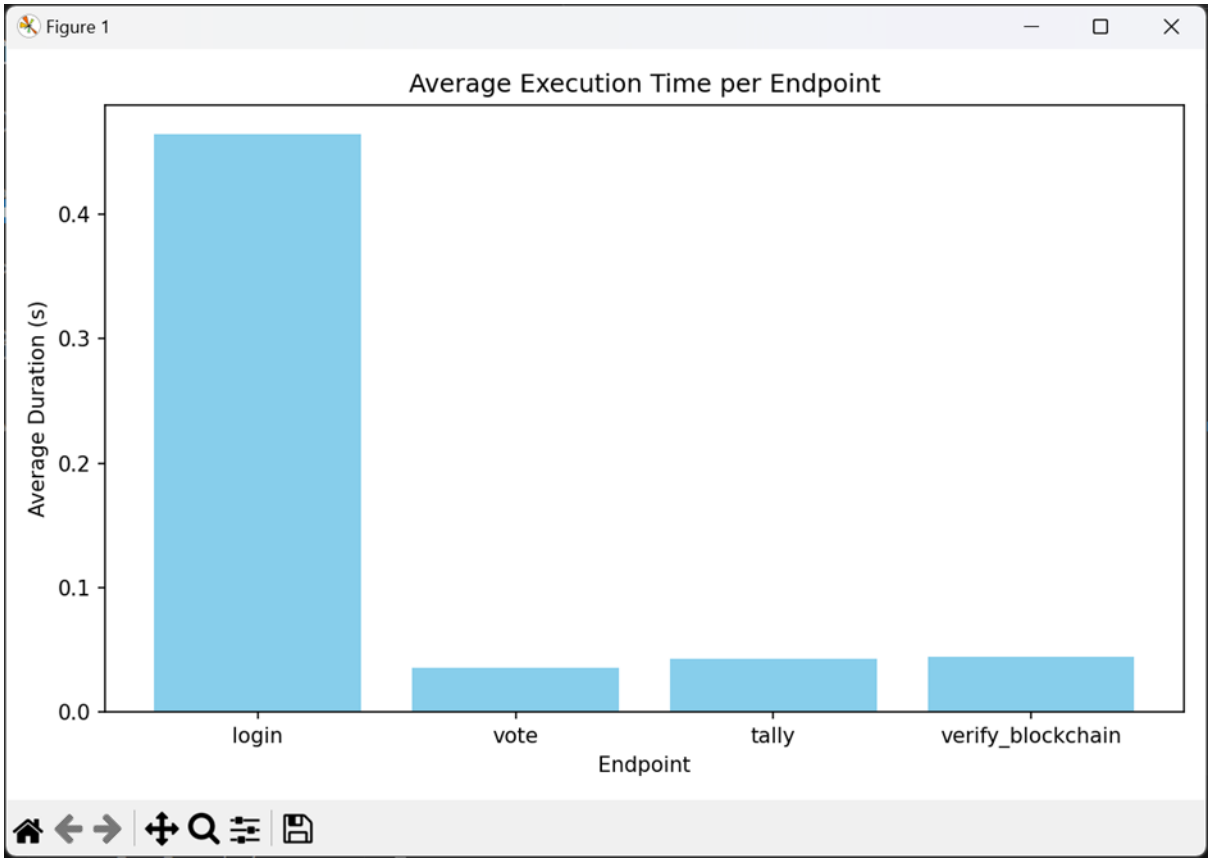
6.3.1 Graph Vote Submission Times



Vote Time Diagram

- Duration of each of the 100 submissions ranged from 0.017s to 0.026s.
- Confirms a consistently low latency voting experience.

6.3.2 Graph Average Execution Times



Average time diagram

- Distribution tightly centred at ~0.044s.

- Indicates stable hash-chain integrity checking, unaffected by increasing ledger length.

6.4 Comparative Evaluation

The table below compares this system's performance against a selection of real-world and academic electronic voting systems.

6.4.1 Comparative Performance

System	Vote Time (s)	Throughput (TPS)	Key Technologies
Helios	~2–5	N/A	Web, cryptographic proofs
Votereum	~15–30	~5–7	Ethereum, PoW
Hyperledger Fabric EVS	~0.1–0.5	~50–100	Hyperledger, PBFT
UniVote	~0.26*	N/A	Flask + SQLite (No blockchain)
This Project	0.0213	28.17–46.9	Python, JWT, Custom Blockchain

**UniVote measured login-to-submit time rather than pure vote submission latency (O'Laighin, 2024).* [2]

6.4.2 Key observations:

- The custom, single-node blockchain used here outperforms Ethereum- and Hyperledger-based systems in raw vote time due to reduced consensus overhead.
- Permissioned blockchains (Hyperledger) achieve sub-second latency but require multi-node infrastructure.
- This project's single-node design outperforms Ethereum in speed (0.0213s vs. 15–30s) and matches Hyperledger's throughput (46.95 TPS) while avoiding decentralization overhead.
- The throughput is competitive with other mid-scale blockchain EVS deployments and well above the rates required for small to medium-scale elections.
- Non-blockchain systems like UniVote provide ease of development and user experience benefits but lack a cryptographically verifiable audit trail.

6.5 Discussion

6.5.1 Summary of Findings

This study set out to evaluate whether a lightweight, single-node blockchain architecture can serve as a viable foundation for electronic voting systems (EVS) in trust-sensitive, controlled-scale environments. The results affirm this approach in key dimensions:

[2] O'Laighin, O. (2024). UniVote: Flask and SQLite voting system.

- Performance: achieves ~0.021 s per vote and throughput up to 46.9 TPS.
- Auditability: every vote is hashed, encrypted, and chained in an exportable JSON ledger.
- Tamper-resistance: integrity verification via SHA-256 linkage ensures detection of unauthorized edits.

However, the design also reveals critical limitations: lack of decentralization, limited voter privacy, and a single point of failure—making it unsuitable for national-scale, hostile, or coercion-risked elections.

6.5.2 Literature Context and Critical Comparison

System / Study	Strengths	Limitations / Relevance to This Study
Helios (Adida, 2008)	End-to-end verifiability, receipts	No blockchain; complex for small-scale use
Estonia i-voting (Springall et al., 2014)	Real-world deployment	Client-side vulnerabilities undermine security
Voatz (Specter et al., 2020)	Blockchain-backed mobile voting	MIT/USENIX analysis revealed endpoint weaknesses
DASH (Kiayias et al., 2015)	Strong anonymity via mixnets + ZK-proofs	High latency (~10–20s/vote)
This Project	Speed + simplicity + auditability	Centralization limits scalability/privacy

6.5.3 Key Takeaways:

- Helios and similar systems validate the importance of verifiable receipts but are architecturally complex.
- Estonian and Voatz case studies emphasize the fragility of systems when client or operational contexts are weak, strengthening the rationale for a minimal, transparent design.
- ZK-based systems (e.g., DASH) offer stronger privacy but are impractical for most institutions due to complexity

6.5.4 Justification of Thesis Approach

This project deliberately occupies the practical middle ground: combining verifiability and performance (thanks to blockchain linkage, receipts, and encryption) with simplicity and deploy ability.

It is justified in its context for three major reasons:

- Appropriate threat model: Designed for environments where decentralization is unnecessary and institutional trust (though not absolute trust) is realistic.

- Performance vs. complexity trade-off: Matches or exceeds performance of more complex systems without burdening institutions with large-scale infrastructure or cryptographic overhead.
- Auditability and transparency: Retain verifiability via exportable ledger and receipts—features missing in non-blockchain designs.

6.5.5 Limitations and Risks

- Centralization: A single-node architecture introduces a single point of failure and reliance on institutional integrity.
- Privacy limitations: Votes are encrypted at rest, but the absence of zero-knowledge proofs, mixnets, or blind signatures means voter anonymity and coercion-resistance are not guaranteed.
- Endpoint vulnerabilities: As seen in Estonia and Voatz deployments, client-side or administrative vulnerabilities can compromise the strongest backend systems.
- Scalability: Tested up to 100 automated votes. Performance under significantly larger loads (e.g., thousands of concurrent users) is unverified and may require redesign.

6.5.6 Ethical, Legal, and Societal Considerations

- Transparency vs. Comprehension: The system's auditability hinges on stakeholders' ability to interpret receipts and verify results. Usable audit tools and training are essential.
- Data protection concerns: Immutable ledgers conflict with "right to erasure" (e.g., GDPR). Deployment must include anonymizing mechanisms and clear retention policies.
- Governance clarity: Institutional control of nodes must be transparent, with documented roles and procedures to maintain public trust.

6.5.7 Future Directions

1. Privacy enhancements: Add blind signatures or zero-knowledge mechanisms to strengthen ballot secrecy.
2. Distributed permissioned replication: Employ multi-node consensus (e.g., PBFT or Raft) to increase resilience without full decentralization.
3. Scalability testing: Simulate high-concurrency scenarios (10k+ users) and stress-test client and server performance.
4. Independent auditing: Publish code, test harnesses, and perform third-party security reviews to build confidence and transparency.

6.5.8 Conclusion

This research confirms that blockchain EVS, when scoped sensibly, can deliver high-speed, auditable, and secure voting for institutional applications. While not a replacement for national election systems, it offers a pragmatic, deployable solution: stronger than non-

blockchain systems yet much less complex than full decentralized or ZK-based architectures. With the enhancements and operational guardrails outlined above, such a system is well-positioned to raise the integrity and trustworthiness of small- to medium-scale elections significantly.

7 Conclusion and Future Work

This research set out to determine whether a lightweight blockchain-based electronic voting system could deliver the core EVS objectives of vote integrity, tamper resistance, and high performance using standard Python libraries and a RESTful architecture. The implemented system achieved all three objectives, validating the feasibility of a minimalist blockchain design for controlled-scale elections.

7.1 Key contributions and outcomes:

- Encrypted, hash-linked vote storage ensuring immutability and traceability.
- Tamper detection through SHA-256 hash-chain verification.
- Real-time tallying enabling immediate and transparent results.
- Performance efficiency with an average vote submission time of ~0.0213 s and throughput up to 46.95 TPS.
- Simplified architecture that omits mining, complex consensus, and multi-node coordination while retaining essential blockchain auditability.

7.2 Applicability and significance:

The system is most appropriate for medium-trust environments—such as university elections, NGO referenda, and institutional polls—where stakeholders value transparency but can tolerate a single trusted node. The findings align with recent literature indicating that properly scoped blockchain EVS designs can outperform large-scale decentralized systems in speed and usability, while still providing verifiable results.

7.3 Limitations:

The system's single-node architecture introduces a central point of failure and limits resilience against insider threats. Privacy protections, while sufficient for the tested use case, fall short of advanced anonymity protocols such as mixnets or zero-knowledge proofs. The scalability evaluation was limited to 100 automated votes, leaving large-scale concurrency untested.

7.4 Future work:

1. Pilot deployment in a live institutional election to evaluate user experience and operational readiness.
2. Scalability testing under high concurrency (≥10,000 voters) and varied network conditions.
3. Enhanced privacy mechanisms, e.g., blind signatures or zero-knowledge proofs, to strengthen anonymity and coercion resistance.

4. Multi-node permissioned replication to eliminate single points of failure while maintaining simplicity.
5. Independent security audits and open-source release to improve transparency and stakeholder trust.

References

1. Banerjee, A. (2021). A Fully Anonymous e-Voting Protocol Employing Universal zk-SNARKs and Smart Contracts. [online] Available at: https://eprint.iacr.org/2021/877.pdf
2. Adida, B. (2008). Helios: web-based open-audit voting. [online] Proceedings of the 17th USENIX Security Symposium, pp.335–348. Available at: https://www.usenix.org/legacy/event/sec08/tech/full_papers/adida/adida.pdf
3. Emami, A., Yajam, H., Akhaee, M.A. and Asghari, R. (2023). A scalable decentralized privacy-preserving e-voting system based on zero-knowledge off-chain computations. [online] Journal of Information Security and Applications, 79, p.103645. Available at: https://doi.org/10.1016/j.jisa.2023.103645
4. ElSheikh, M. and Youssef, A.M. (2022). Dispute-free scalable Open Vote Network using zk-SNARKs. [online] arXiv preprint arXiv:2203.03363. Available at: https://arxiv.org/abs/2203.03363
5. Hajian Berenjestanaki, M., Barzegar, H.R., El Ioini, N. and Pahl, C. (2023). Blockchain-based e-voting systems: a technology review. [online] Electronics, 13(1), p.17. Available at: https://doi.org/10.3390/electronics13010017
6. Hardwick, F.S., Gioulis, A., Akram, R.N. and Markantonakis, K. (2018). E-Voting with blockchain: An e-voting protocol with decentralisation and voter privacy. [online] IEEE International Conference on Internet of Things (iThings). Available at: https://arxiv.org/abs/1805.10258
7. Iordache, C.A. and Marian, C.V. (2022). Enhanced Accessibility and Anti-Fraudulent System for Polling Stations and Mobile Voting in Hospitals. [online] *2022 E-Health and Bioengineering Conference (EHB)*, Iasi, Romania, pp.1–4. Available at: https://doi.org/10.1109/EHB55594.2022.9991362
8. Kshetri, N. and Voas, J. (2018). Blockchain-Enabled E-Voting. [online] IEEE Software, 35(4), pp.95–99. Available at: https://doi.org/10.1109/MS.2018.2801546
9. Naidu, P.R., Bolla, D.R., P.G., Harshini, S.S., Hegde, S.A. and Harsha, V.V.S. (2022). E-Voting System using Blockchain and Homomorphic Encryption. [online] 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), Mysuru, India, pp.1–5. Available at: https://doi.org/10.1109/MysuruCon55714.2022.9972661
10. Russo, A., Fernández Anta, A., González Vasco, M.I. and Romano, S.P. (2021). Chirotonia: a scalable and secure e-voting framework based on blockchains and linkable ring signatures. [online] arXiv preprint arXiv:2111.02257. Available at: https://arxiv.org/abs/2111.02257

11. Specter, M., Koppel, J. and 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. [online] Proceedings of the 29th USENIX Security Symposium (USENIX Security 2020), pp.1535–1553. Available at: https://www.usenix.org/conference/usenixsecurity20/presentation/specter
12. Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., MacAlpine, M. and Halderman, J.A. (2014). Security analysis of the Estonian internet voting system. [online] Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS '14), pp.703–715. Available at: https://doi.org/10.1145/2660267.2660315
13. Tang, B., Tan, M., Liu, M., Liu, Z. and Tian, W. (2023). A privacy protection method of blockchain-based e-voting using homomorphic encryption and order-preserving encryption. [online] 2023 5th International Conference on Artificial Intelligence and Computer Applications (ICAICA), Dalian, China, pp.86–90. Available at: https://doi.org/10.1109/ICAICA58456.2023.10405563
14. Tsai, C. and Song, K. (2023). Research on e-voting based on blockchain. [online] 2023 IEEE 6th International Conference on Computer and Communication Engineering Technology (CCET), Beijing, China, pp.63–67. Available at: https://doi.org/10.1109/CCET59170.2023.10335141
15. Wang, X., Feng, T., Liu, C. and Fang, J. (2024). Multi-party confidential verifiable electronic voting scheme based on blockchain. [online] Journal of Cloud Computing, Article 160. Available at: https://doi.org/10.1186/s13677-024-00723-8
16. Wang, B., Guo, F., Liu, Y., Li, B. and Yuan, Y. (2024). An efficient and versatile e-voting scheme on blockchain. [online] Cybersecurity, 7, Article 62. Available at: https://doi.org/10.1186/s42400-024-00226-8
17. Zheng, Z., Xie, S., Dai, H., Chen, X. and Wang, H. (2017). An overview of blockchain technology: architecture, consensus, and future trends. [online] 2017 IEEE International Congress on Big Data (BigData Congress), pp.557–564. Available at: https://doi.org/10.1109/BigDataCongress.2017.85
18. Zyskind, G., Nathan, O. and Pentland, A. (2015). Decentralizing privacy: Using blockchain to protect personal data. [online] 2015 IEEE Security & Privacy Workshops, pp.180–184. Available at: https://doi.org/10.1109/SPW.2015.27
19. Santoso, I. and Yuli Christyono (2023). Zk-SNARKs As A Cryptographic Solution For Data Privacy And Security In The Digital Era. *International Journal of Mechanical Computational and Manufacturing Research*, 12(2), pp.53–58. doi: https://doi.org/10.35335/computational.v12i2.122.
20. Oulu University (2023). Oulu Repository. [online] Available at: https://oulurepo.oulu.fi/handle/10024/46455