Implementation of Blockchain Technology in E-Voting Using Smart Contract and ZK-SNARK

Jimmy[1*], Kenny Rimba[2], Vincent[3], Ronsen Purba[4], Darwin[5]

[1,2,3,4,5]Department of Informatics Engineering, Faculty of Informatics, Universitas Mikroskil, Medan, North Sumatera, Indonesia
E-mail: [1*]211110476@students.mikroskil.ac.id, [2]211110404@students.mikroskil.ac.id, [3]211111031@students.mikroskil.ac.id, [4]ronsen@mikroskil.ac.id, [5]darwin.ng@mikroskil.ac.id

(Received: 17 Aug 2025, revised: 14 Sep 2025, accepted: 16 Sep 2025)

Abstract

E-voting systems are prone to challenges such as lack of transparency, risks of data manipulation, and dependence on centralized authorities, which can undermine trust in electoral processes. This research develops a blockchain-based e-voting system on the Polygon network, leveraging smart contracts and Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (ZK-SNARK) to enhance security, transparency, and voter anonymity. The study employs an application development approach, implementing a structured methodology with initialization, registration, voting, and tallying phases. Smart contracts automate voter verification, vote casting, and result tabulation, while ZK-SNARK ensures voters can cast ballots anonymously without revealing their identities. The system's transparency and immutability are tested using PolygonScan, demonstrating effective prevention of manipulations like double voting through cryptographic credentials (nullifier, commitment, and nullifier hash) and Merkle Tree structures. Results indicate that the system provides a secure, verifiable, and decentralized framework for elections. This implementation offers a robust foundation for future e-voting systems, promoting trust and integrity in digital voting processes.

Keywords: Blockchain, E-Voting, Smart Contract, ZK-SNARK, Polygon.

I. INTRODUCTION

Electronic voting (e-voting) is a method of collecting votes using electronic devices to record, process, and tally votes [1]. Blockchain-supported e-voting systems are increasingly viewed as the future of secure voting due to blockchain's immutable nature, which acts like a decentralized, tamper-proof digital ballot box. However, despite their ability to prevent data manipulation and ensure data integrity, blockchain-based voting systems still face significant challenges, particularly in protecting voter privacy and achieving fast transaction speeds [2].

Previous studies have shown that homomorphic encryption-based schemes on blockchain face several limitations. These include inefficiencies when handling elections with multiple candidates, difficulties in securely managing encryption keys, and complexities in organizing and tallying vote results, all of which limit their practical use in e-voting [3]. Other works emphasize that, despite improvements in transparency, blockchain-based systems remain vulnerable to threats such as traffic analysis and transaction inspection, which may compromise voter anonymity. To address these vulnerabilities, the integration of ZK-SNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) with Merkle tree structures has been proposed to ensure secure and anonymous voter verification while maintaining process efficiency. In this context, smart contracts play a critical role in facilitating ZK-SNARK verification by managing tasks such as voter registration, ticket validation, and vote counting, while also managing election protocols, securely storing votes, and verifying cryptographic proofs. These mechanisms ensure that all votes are authentic and that the entire process strictly follows predefined rules [4].

Given the potential of ZK-SNARK, blockchain technology emerges as an ideal foundation for implementing secure and reliable e-voting system. The transparent, immutable, and decentralized nature of blockchain enables users to access transaction data publicly without risks of manipulation [1], [5]. To further enhance efficiency and scalability, this study proposes implementing the e-voting system on Polygon, an Ethereum-based layer-2 network known for its faster transaction speeds and lower gas fees compared to the Ethereum mainnet [6]. By leveraging smart contracts, the system can automate key voting processes, reducing reliance

DOI: 10.34148/teknika.v14i3.1358
TEKNIKA, Volume 14(3), November 2025, pp. 433-443
ISSN 2549-8037, EISSN 2549-8045

on intermediaries and eliminating risks associated with centralized databases. This automation not only boosts efficiency but also enhances trust in the electoral process [7]. Moreover, Zero-Knowledge Proof, particularly ZK-SNARK, ensures that voter eligibility can be verified without revealing any personal information, thereby safeguarding voter privacy throughout the entire election process [4].

To overcome the ongoing challenges of e-voting systems, this study proposes the development of a secure, efficient, and user-friendly e-voting platform. By combining the automation capabilities of smart contracts with the privacy-preserving power of ZK-SNARK, the proposed system aims to deliver a transparent, tamper-proof, and anonymous voting process. This integrated approach is designed to strengthen the integrity of elections, protect voter identities, prevent unauthorized data changes, and ensure that the entire process is trustworthy and efficient in real-world applications.

II. LITERATURE REVIEW

A. E-Voting

E-voting refers to remote voting through internet-enabled devices such as computers and smartphones. It improves efficiency, accessibility, and result processing speed, but also raises concerns regarding fraud, hacking, and manipulation [8]. Major types of e-voting include punch-card systems, Direct Recording Electronic (DRE), optical scanning systems, ballot-marking devices, and internet-based voting (i-voting) [1]. Each method has strengths and weaknesses, particularly in terms of transparency, auditability, and voter privacy.

Despite its advantages, e-voting faces several challenges, including:

- Security risks: vulnerabilities to hacking and data breaches.
- Transparency issues: limited ability for voters to verify their votes.
- Trust concerns: skepticism from stakeholders due to centralized infrastructures.
- Anonymity risks: threats to vote secrecy, especially if authentication and voting occur in the same channel [9].

B. Blockchain

Blockchain is a decentralized ledger that records immutable data blocks linked cryptographically [10]. This technology ensures transparency, security, and immutability of data through a distributed network of nodes. Each block contains a list of transactions or data, secured by cryptographic hashes, and is linked to the previous block, forming a chain. Types of blockchain include:

- Public blockchain (permissionless): Open to anyone, allowing unrestricted participation in the network. Users can join as nodes, validate transactions, or mine blocks without requiring permission. Examples include Bitcoin, which facilitates peer-to-peer cryptocurrency transactions, and Ethereum, known for its smart contract capabilities.
- Private blockchain (permissioned): Access is restricted to specific participants, defined by the network owner. This type is typically used by enterprises for internal purposes, ensuring privacy and control over sensitive data. It is ideal for commercial applications where only authorized entities can interact with the network.
- Consortium blockchain: A hybrid model combining elements of both public and private blockchains. Access and data permissions are limited to a select group of participants, often organizations collaborating on a shared project. This setup allows for controlled access while leveraging the benefits of decentralization, such as enhanced security and transparency, making it suitable for inter-organizational collaborations.

Each type presents trade-offs in scalability, transparency, and control, depending on the use case and network requirements [11]. Central to blockchain's integrity are consensus algorithms, which coordinate agreement among decentralized nodes to validate transactions and maintain consistency. Common consensus algorithms include:

- Proof of Work (PoW): Used by Bitcoin, PoW requires miners to solve computationally intensive mathematical puzzles to validate transactions and add new blocks. Miners compete to find a nonce that produces a hash meeting specific criteria, a process called mining that ensures security but demands significant energy [5].
- Proof of Stake (PoS): An energy-efficient alternative to PoW, PoS selects validators (called forgers) based on the amount of cryptocurrency they hold and stake. The selection process, often based on coin age or randomization, reduces energy use while maintaining network consensus [11].
- Proof of Authority (PoA): Combining PoW and PoS, PoA starts with miners proposing empty blocks via PoW, followed by randomly selected validators (based on stake) verifying and signing the block. This hybrid approach enhances efficiency and decentralization, often used in private or consortium blockchains [5].

C. Smart Contracts

Smart contracts are self-executing programs stored on the blockchain that automatically enforce agreements between parties without requiring intermediaries [12]. These programs encode predefined conditions in code, enabling transparent and secure transaction execution once conditions are met. In e-voting systems, smart contracts automate critical processes like voter registration, vote validation, and result tallying, ensuring accuracy and reducing reliance on third parties.

D. Keccak

Keccak is a cryptographic hash function developed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche, which won the NIST SHA-3 competition. Based on the sponge construction, Keccak employs the Keccak-f permutation with varying state widths (25 to 1600 bits), organized in a 5x5 lane array, and undergoes transformation steps (θ, ρ, π, χ, ι) to ensure diffusion and non-linearity. The process involves initialization, padding, absorbing, and squeezing phases, with parameters like bit rate (r), capacity (c), and diversifier (d) defining its security and flexibility. This design is resistant to cryptographic attacks and serves as

the foundation for the SHA-3 standard, while also being utilized in the Ethereum blockchain to ensure transaction integrity and security [13].

E. Polygon

Polygon, an Ethereum-based layer-2 solution, enhances blockchain scalability by offering faster transaction speeds and lower gas fees compared to the Ethereum mainnet [6]. This makes it ideal for real-time e-voting systems, where high transaction volumes require cost-effective and rapid processing. Polygon supports smart contract execution, enabling seamless automation of e-voting processes like vote recording and verification. By periodically creating checkpoints on the Ethereum network, Polygon ensures transaction security while maintaining efficiency, making it a practical choice for decentralized applications requiring scalability and low-cost operations.

F. Zero-Knowledge Proofs (ZKP)

Zero-Knowledge Proofs (ZKPs) allow a prover to demonstrate knowledge of a secret to a verifier without revealing the secret itself, ensuring privacy and security in transactions [14]. ZK-SNARK (Succinct Non-Interactive Argument of Knowledge), a highly efficient ZKP variant, is widely adopted in blockchain for privacy-preserving applications due to its compact proof size and fast verification. In e-voting systems, ZK-SNARK enables voters to prove their eligibility such as possessing a valid voter ID without disclosing their identities, thus maintaining anonymity. This process also prevents issues like double voting by verifying votes securely on the blockchain, ensuring the integrity and confidentiality of the electoral process.

G. Wallet

A wallet, or digital wallet, is software or hardware that enables users to store, manage, and transact cryptocurrency assets like Bitcoin and Ethereum, acting as an interface to the blockchain network through cryptographic keys [15]. In e-voting, wallets securely store voter credentials or tokens, ensuring authenticated participation without compromising privacy. Various wallet types include desktop wallets, like Bitcoin Core or Exodus, which store private keys on a computer; mobile wallets, popular for their smartphone accessibility; hardware wallets, such as Ledger or Trezor, offering enhanced offline security; paper wallets, printed keys for secure offline backups; and web wallets, browser-based for convenience but requiring strong security measures [16].

III. RESEARCH METHODS

This study employs an experimental research approach through the design and implementation of a blockchain-based e-voting system integrated with Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (ZK-SNARK). The system leverages smart contracts to automate and secure critical processes such as voter registration, vote casting, and result verification on the blockchain.

The methodology is presented through several phases of the system workflow. Each phase—initialization, registration, voting, and tallying—is described in detail and accompanied by a dedicated flowchart to ensure clarity of the process, as well as to highlight how transparency, anonymity, and verifiability are maintained throughout the election cycle.

A. Initialization Phase

The initialization phase is the preparatory stage of the blockchain-based e-voting system, where the election organizer configures the infrastructure to ensure a secure and transparent voting process. During this phase, the election organizer deploys smart contracts to the blockchain, including a cryptographic MiMC hasher for generating secure voter commitments, a ZK-SNARK verifier for validating anonymous proofs, and the main voting contract to manage the election process.

These components enable the creation and verification of cryptographic structures, such as Merkle Trees, which store voter data anonymously, and Zero-Knowledge Proofs, which ensure voter privacy. The election organizer also sets the time periods for candidate registration, voter registration, and the voting phase to define the election timeline. Additionally, the front-end, developed using a framework like Vue.js, and the back-end, integrated with a blockchain network (e.g., Polygon), are hosted to provide accessible interfaces for voters and candidates. The workflow of this phase is illustrated in Figure 1, ensuring the system is fully prepared for subsequent stages like registration and voting.

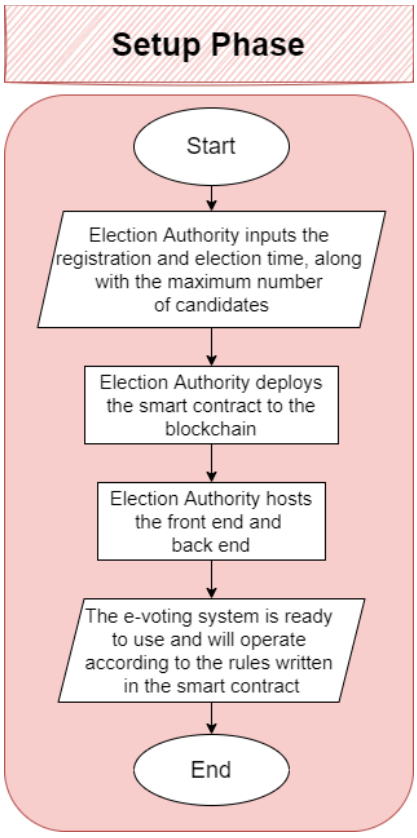


Figure 1. Flowchart of Initialization Phase

DOI: 10.34148/teknika.v14i3.1358
TEKNIKA, Volume 14(3), November 2025, pp. 433-443
ISSN 2549-8037, EISSN 2549-8045

B. Registration Phase

The registration phase enables candidates and voters to enroll in the blockchain-based e-voting system, ensuring only authorized participants can proceed to the voting phase. As illustrated in Figure 2, the process differs for candidates and voters based on their roles. Candidates submit personal details (e.g., ID number, name, gender, password, vision-mission, photo, and required documents) through the front-end interface. The backend server validates these details against an identity database. If valid, the election organizer reviews and approves the candidate, triggering a transaction to record their information on the blockchain, making them eligible for the election. If not approved, the candidate is rejected and cannot participate. For voters, voters submit identification data when register (e.g., ID number, name, gender, password), which the backend server verifies against the database. Upon successful validation, the system generates cryptographic credentials: a nullifier (a private, unique value to prevent double voting), a secret (a private value for creating commitments), a commitment (a public hash stored in the Merkle Tree to represent the voter anonymously), and a nullifier hash (a public hash to detect double voting attempts). Voters obtain their credentials by clicking "Download Voting Credential", which saves a .json file on their device. They are responsible for safeguarding this file, since if lost, the credential cannot be re-issued and the voter loses the right to vote. Each credential is cryptographically bound and single-use, preventing its reuse by another voter.

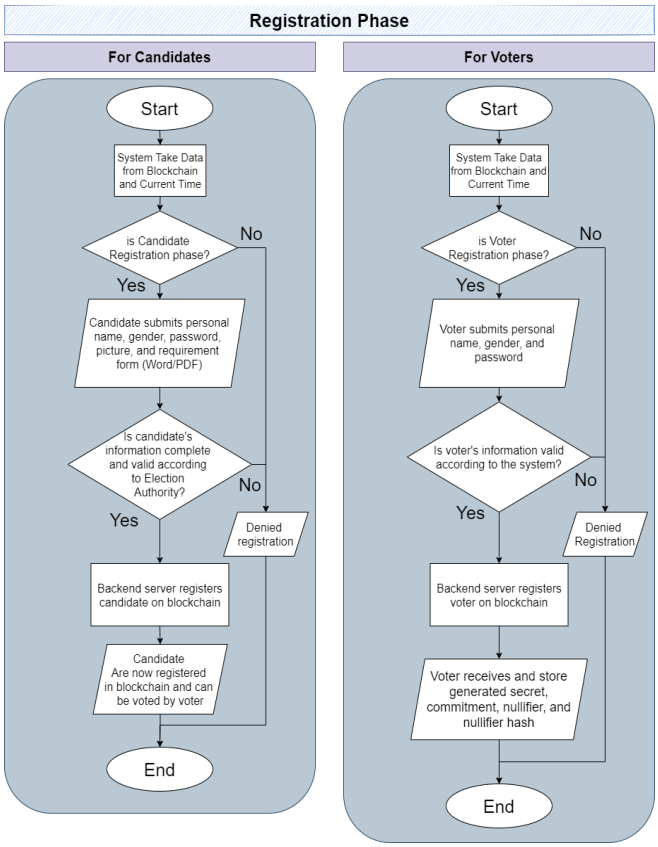


Figure 2. Flowchart of Registration Phase

C. Voting Phase

The voting phase, illustrated in Figure 3, is the stage where registered voters cast their ballots for their chosen candidates. Voters begin by authenticating themselves using cryptographic credentials (nullifier, secret, commitment, and nullifier hash) generated during registration. The system validates these credentials to confirm the voter's eligibility. Once validated, the voter proceeds to select a candidate through the front-end interface. The system then calculates the Merkle root and path using the deployed MiMC address from the initialization phase, together with the contract address and the commitment. Based on these parameters, a ZK-SNARK proof is generated to verify the vote's legitimacy without revealing the voter's identity. This proof, along with the voter's ballot, the current Merkle Tree root, and the nullifier hash, is submitted as a transaction to the blockchain. The ZK-SNARK verification ensures votes are valid, private, and protected against double voting or tampering. Voters can later verify their vote's inclusion in the blockchain using the transaction hash by comparing the nullifier in their ticket with the one stored on the blockchain, ensuring transparency and integrity throughout the process.

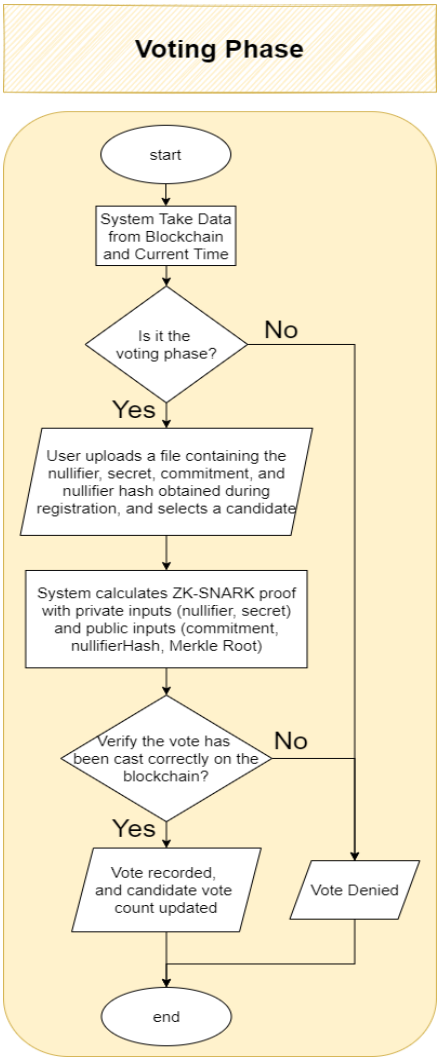


Figure 3. Flowchart of Voting Phase

D. Tallying Phase

The tallying phase, illustrated in Figure 4, is the final stage where the system calculates the total votes for each candidate. After the voting period ends, the election organizer, candidate or voters can access the vote tally through the front-end interface. The smart contract automatically aggregates the votes stored in the immutable blockchain ledger, ensuring no manipulation can occur. Once the voting period is complete, the system will declare the winner based on verified results, ensuring accuracy and fairness. This process is fully transparent, allowing the public to verify the outcome directly on the blockchain, guaranteeing integrity and trust in the election results.

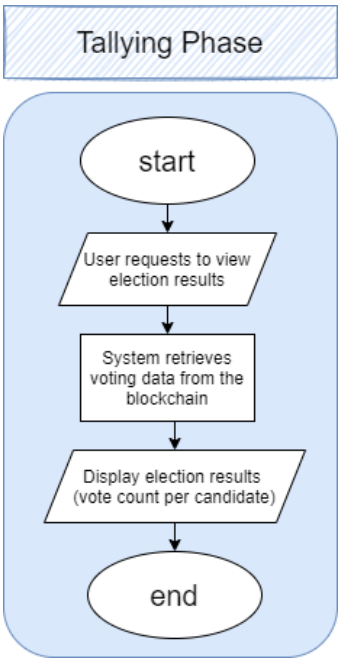


Figure 4. Flowchart of Tallying Phase

E. Database Design

In this blockchain-based e-voting system, the database architecture is strategically designed with a hybrid approach, minimizing centralization to uphold the decentralized ethos of blockchain technology. The design incorporates two primary centralized tables using an SQL database, which are exclusively utilized during the initial registration phase for identity verification. Once verification is complete, the system relies entirely on decentralized data structures recorded on the blockchain to manage the election process, ensuring security, transparency, and user privacy.

1. Centralized Database Structure

The centralized component consists of two main tables, *semua_anggota* and *kandidat_pending*, modeled using an Entity Relationship Diagram (ERD) as shown in Figure 5. The data is sourced from the university database to ensure that only eligible students can participate, but it can also be obtained from governmental, organizational, or custom datasets depending on the election context. This database is managed by the election organizer, who is responsible for deploying the smart contract and populating the initial identity data for verification purposes

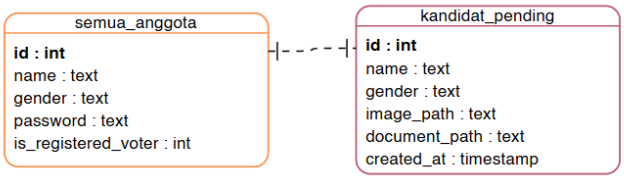


Figure 5. ERD of Database Design

a. *semua_anggota* Table: This table stores information about all potential users, including both candidates and voters, to be used as a reference for validation during registration. Key attributes include user ID (NIK or NIM), name, gender, registration status as a voter, and a password. During voter registration, the system compares the data submitted by the user against the records in this table. If the data matches, the system issues a ZK-SNARK-based ticket/hash, allowing the user to vote without their personal data being stored on the system, thus preserving privacy.

b. *kandidat_pending* Table: This table is used to manage information for candidates who are in the process of registering. It includes attributes such as the candidate's ID, name, gender, password, the server paths for their photo, and required documents (any supporting files specified by the election organizer as eligibility criteria). When a candidate registers, their data is held in this table with a "pending" status. The election organizer then verifies this information. Upon approval, the system registers the candidate on the blockchain, and they become a valid candidate in the election.

2. Decentralized Blockchain Data Storage

Once a candidate is approved, their data is recorded in a transaction on the blockchain. The smart contract design includes the following data structures to manage the election on the decentralized ledger (Figure 6):

a. Election: This structure holds the core parameters of the election, such as the organizer's wallet address (owner), the name of the election (electionName), registration and voting timelines, the maximum number of candidates, and the addresses for the ZK-SNARK verifier and hasher contracts.

b. *Kandidat* (Candidate): This structure stores information for each registered candidate approved by election organizer. The attributes include a unique ID, name, vision, mission, the total number of votes received (votecount), and a boolean status to indicate if the candidate is active (exists).

c. *Pemilih* (Voter): This structure represents the information for each registered voter on the blockchain. It includes a unique hash of the voter's identity for verification (uniqueHash), their registration status (isRegistered), and the ZK-SNARK commitment (commitment) that allows them to prove their right to vote without revealing their identity.

DOI: 10.34148/teknika.v14i3.1358
TEKNIKA, Volume 14(3), November 2025, pp. 433-443
ISSN 2549-8037, EISSN 2549-8045

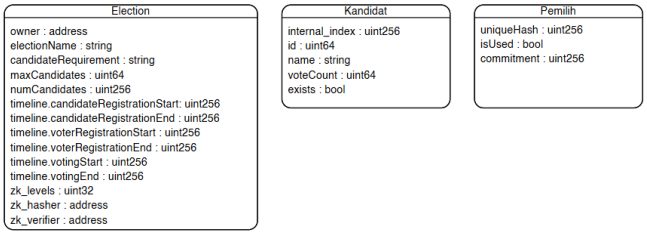


Figure 6. Storage Structure in Blockchain

IV. RESULT AND DISCUSSION

This section presents the results of the developed blockchain-based e-voting system and provides an in-depth discussion of its implementation. The evaluation demonstrates the system's functionality through user interface interactions and outputs generated by the smart contract on the blockchain. The analysis highlights how the system ensures transparency, anonymity, and integrity across the entire election process.

A. User Interface

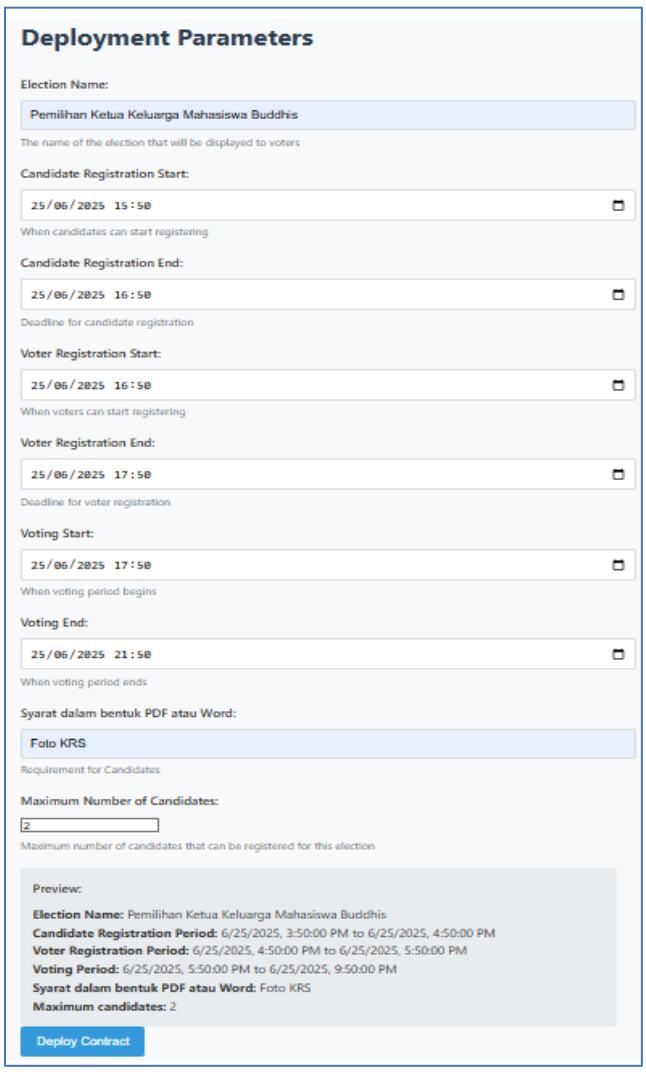


Figure 7. Initialization Interface.

The system provides several role-based interfaces that allow organizers, candidates, and voters to interact with the application. Each interface was designed to align with the registration records stored on the blockchain, thereby ensuring both usability and security.

In the initialization phase, as shown in Figure 7, the election organizer can define the registration and voting periods, as well as the maximum number of candidates.

After initialization, the system does not immediately redirect to another page; instead, the user can directly access the dashboard page as shown in Figure 8. The only difference between the election organizer dashboard and other user dashboards is the presence of menus for candidate verification and administrative settings.

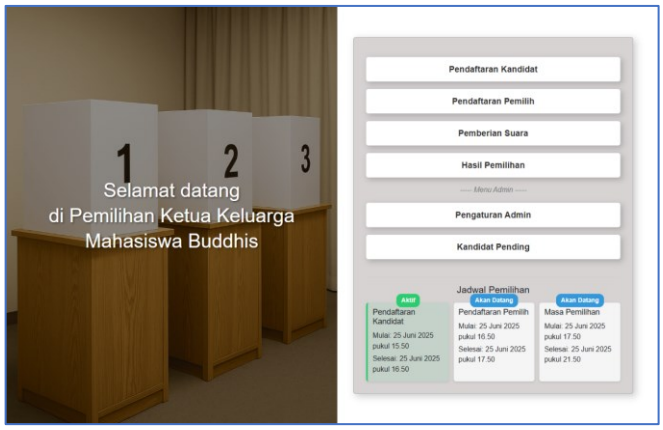


Figure 8. Dashboard Page

From the dashboard, the election organizer can navigate to the candidate verification page to verify the registered candidate, as presented in Figure 9.

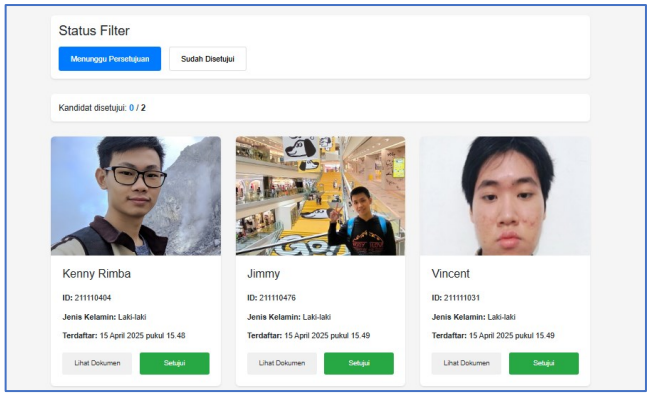


Figure 9. Candidate Verification Page.

In case, there is a wrong period set, the organizer can change it on the admin page as shown in Figure 10.

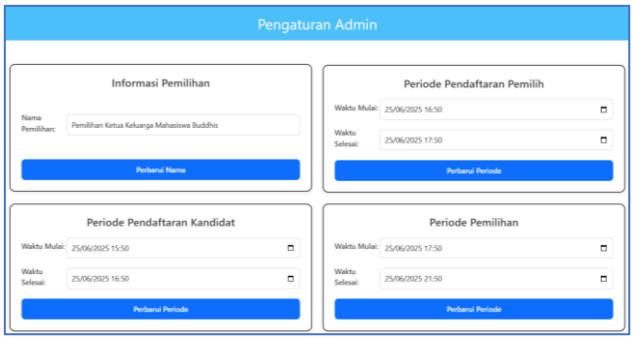


Figure 10. Admin Page

The candidate register interface is shown in Figure 11, while the voter register interface is displayed in Figure 12. By separating the login processes, the system enforces role-based access and ensures that user identities correspond to the blockchain registration records

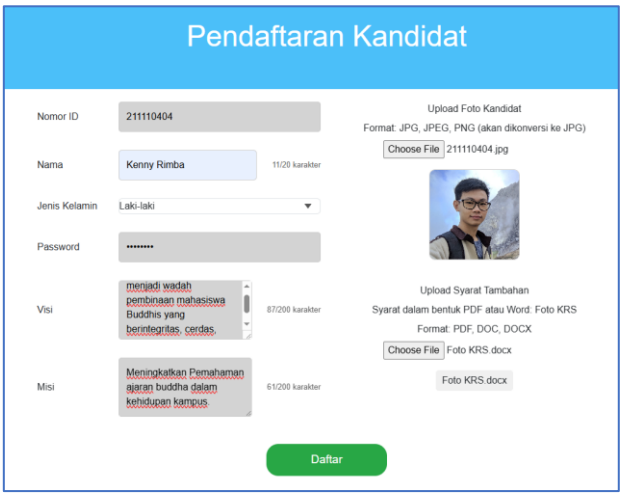


Figure 11. Candidate Registration Page

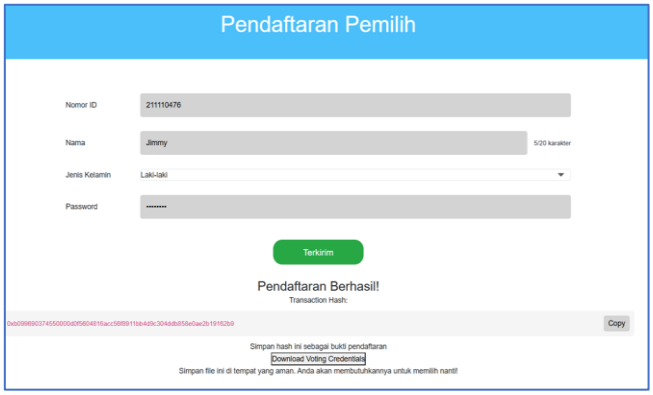


Figure 12. Voter Registration Page

Once registered, voters will get a credential ticket that they can use to verify themselves which shown in Figure 13.

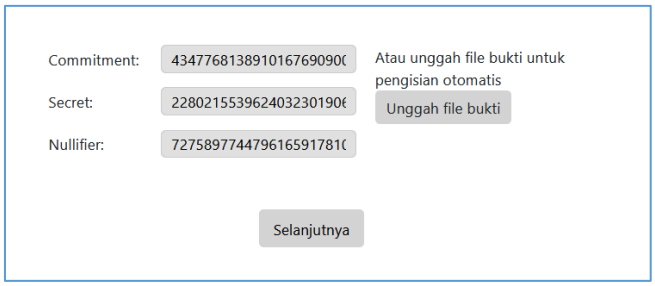


Figure 13. Voter Ticket Verification

Once the ticket has been verified, the user will be taken to a page where they can select the candidate they want that displayed in Figure 14.

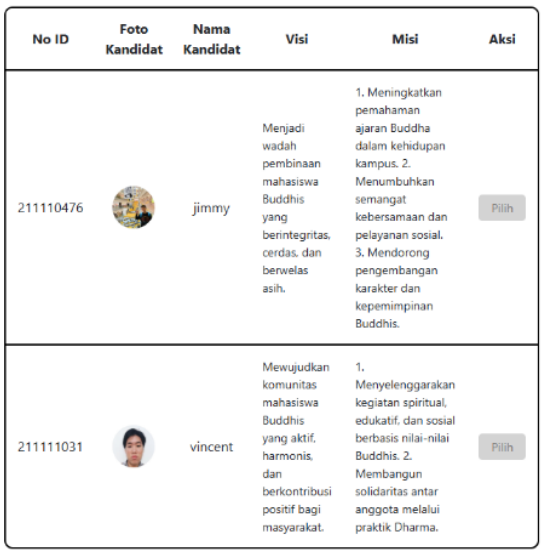


Figure 14. Voting Page

Before submission, the system presents a confirmation screen, allowing the voter to review their choice. And if the voting process is successful, a notification will appear as displayed in Figure 15 containing a link or blockchain address that redirect the voter to: "https://polygonscan.com/tx/{Transaction_Hash}"

The voter may either follow this link directly or copy the transaction hash to manually verify it on Polygonscan. But this verification is optional, as voters may also choose not to perform this step.



Figure 15. Voting Confirmation

DOI: 10.34148/teknika.v14i3.1358

Finally, after the voting period ends, the result page is presented, as illustrated in Figure 16.

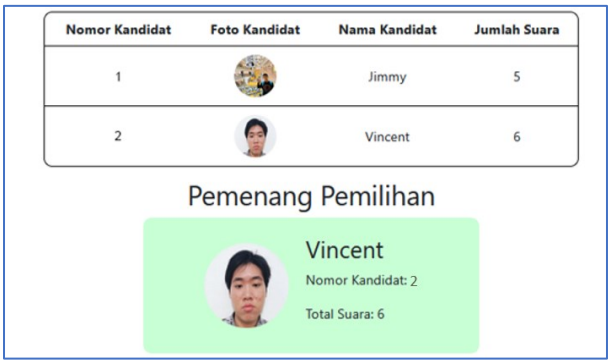


Figure 16. Result Page

B. System Results

The System Results section describes the entire election process from initialization, through candidate and voter registration, to voting, ensuring that every step is recorded transparently and immutably on the Polygon blockchain. This immutable record prevents manipulation and significantly increases trust in the system.



Figure 17. Initialization Transaction on PolygonScan

The first stage is initialization, which can be seen in Figure 17. During this step, the election organizer configures the core parameters of the election. Once executed, these parameters are permanently stored on the blockchain and include:

- **Arg [0] _levels (uint32)** = Represents the number of levels in the Merkle Tree.
- **Arg [1] _hasher (address)** = Address of the previously deployed MiMC Sponge.
- **Arg [2] _verifier (address)** = Address of the previously deployed Verifier contract.
- **Arg [3] _timeline (tuple)** = Contains the list of time periods such as candidate registration, voter registration, and voting periods, in tuple format.
- **Arg [4] _electionName (string)** = Official election name.
- **Arg [5] _candidateRequirement (string)** = Additional requirements that must be fulfilled by candidates, provided in Word or PDF format.
- **Arg [6] _maxCandidates (uint64)** = Maximum number of candidates that can be selected.

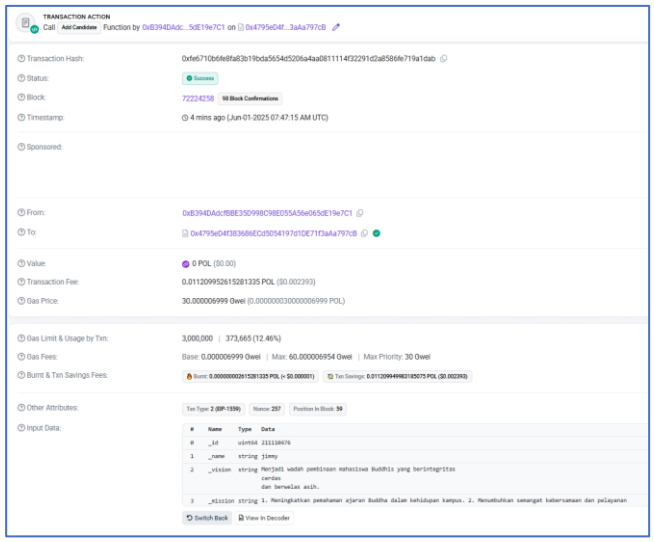


Figure 18. Candidate Registration Transaction

Transaction 1 in Figure 18 represents the candidate registration process. From the decoded input data, it can be seen that the registered candidate is Jimmy with ID 211110476, along with his stated vision and mission. The 'from' field shows that the transaction was executed by 0xB394DAdcfBBE35D998C98E055A56e065dE19e7C1, which is the address of the election organizer. This indicates that the registration fee was covered by the organizer, ensuring that candidates are not required to pay for blockchain transaction costs. The validity of this transaction can be verified on the Polygon blockchain explorer (Polygonscan). By adding the address after the URL **https://polygonscan.com/address/,** the full link becomes: "https://polygonscan.com/address/0xB394DAdcfBBE35D998C98E055A56e065dE19e7C1"

Through this link, anyone can directly check the balance, transactions, and related details of the election organizer's wallet, ensuring transparency and verifiability.



Figure 19. Voter Registration Transaction

The voter registration transaction (Figure 19) is intended to securely enroll eligible voters into the system by issuing a cryptographic voting ticket that preserves their anonymity. The public data stored on the blockchain includes:

- **Unique Hash** = Serves as the unique ID of the registered voter.
- **Commitment** = Hash generated from two secret values:
  - The nullifier prevents double voting by ensuring that each voter can only vote once.
  - The secret is used to form the commitment and guarantees that even if the nullifier becomes known, the commitment cannot be reconstructed without the secret itself.

This design ensures that voter identities remain private while still maintaining the verifiability of each vote.

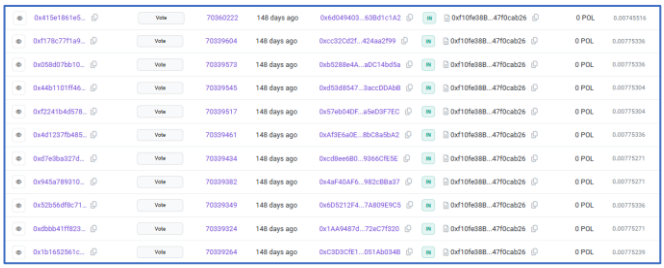


Figure 20 List of Voting Transactions on PolygonScan

In this election, a total of 11 voters successfully cast their votes, resulting in 11 distinct transaction hashes recorded on the blockchain which displayed in Figure 20. Each vote submission is stored as a unique transaction, ensuring full transparency.

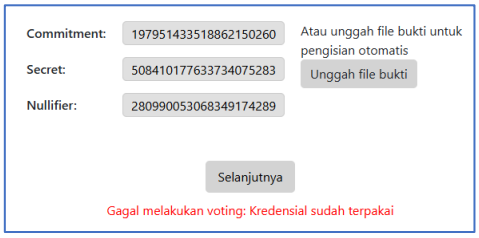


Figure 21. Double Voting Attempt

Any attempt to vote more than once is automatically detected and rejected by the smart contract (Figure 21). If they try to put the invalid ticket they will also get rejected by the smart contract as shown in Figure 22.

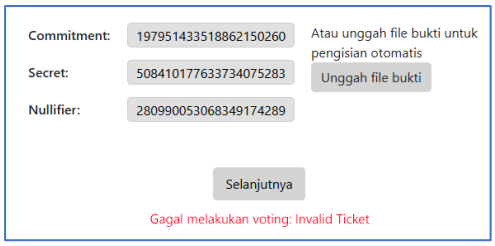
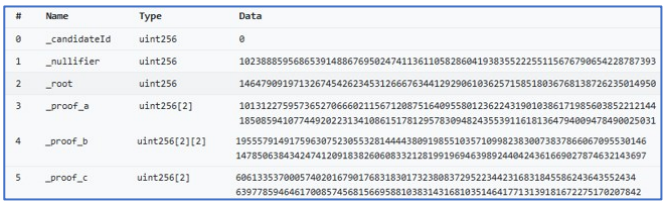


Figure 22. Attempt using Invalid Ticket



#	Name	Type	Data
0	_candidateId	uint256	0
1	_nullifier	uint256	10238885956865391488676950247411361105828604193835522255115676790654228787393
2	_root	uint256	14647909197132674542623453126667634412929061036257158518036768138726235014950
3	_proof_a	uint256[2]	10131227595736527066602115671208751640955801236224319010386171985603852212144 185085941077449202231341086151781295783094824355391161813647940094784900025031
4	_proof_b	uint256[2][2]	19555791491759630752305532814444380919855103571099823830073837866067095530146 14785063843424741209183826060833212819919694639892440424361669027874632143697
5	_proof_c	uint256[2]	60613353700057402016790176831830173238083729522344231683184558624364355243 4 63977859464617008574568156695881038314316810351464177131391816722751702078 42

Figure 23. ZK-SNARK Proof of Voting Data

The transaction depicted in Figure 23 represents a vote casting action. The decoded input data shows that the voter selected the first candidate (indexed from 0). The transaction includes a ZK-SNARK proof, ensuring the vote's validity without revealing the voter's identity, while the nullifier hash prevents double voting.

During vote casting, the transaction structure mirrors voter registration but includes distinct data submitted to the blockchain. This public data comprises:

- **Candidate ID**: The index of the chosen candidate.
- **Nullifier Hash**: A value to check for prior voting attempts, preventing double voting.
- **Root**: The Merkle Tree root for commitment verification.
- **Proof A, Proof B, Proof C**: ZK-SNARK proof components to verify vote validity without disclosing voter identity. These three proofs are generated by the voter's system (the prover) using their private information (Nullifier, Secret, Path Elements and Path Indices) and public inputs (Nullifier Hash and Merkle Root). They are then submitted to the verifier smart contract on the blockchain

The verifier uses these components to execute a bilinear pairing operation based on the following Equation 1:

$$e(A,B)=e(\alpha,\beta)\times e((public_inputs\times vk),\gamma)\times e(C,\delta) \quad (1)$$

In this formula:

- **A, B, and C** are the cryptographic elements from the proof, which belong to specific elliptic curve groups (G1 and G2).
- **e** is the bilinear pairing function.
- Other variables like
  α, β, γ, δ, and the verification key (vk) are parameters established during a one-time trusted setup phase.

Essentially, the verifier smart contract takes Proof A, Proof B, and Proof C, plugs them into this equation, and checks if the two sides are equal. If they match, the proof is considered valid, and the vote is accepted and recorded on the blockchain. If they do not match, the proof is invalid, and the vote is rejected. Through ZK-SNARK proofs, the system ensures:

- The vote is valid and cast by an eligible registered voter.
- The voter's identity remains anonymous.
- No double voting occurs.

Using PolygonScan, the smart contract's read functions can be called to verify the integrity of these transactions. This combination of transparency, immutability, and privacy strengthens the trustworthiness of the election process.

After the voting period ends, Figure 24 displays the vote counts for each candidate, retrieved from the blockchain's immutable ledger. For example, the second candidate, Vincent (ID: 211111031), received six votes. This transparent and unalterable record allows the public to verify the results, reinforcing trust. The election organizer can then declare the winner based on these verified vote counts, ensuring a fair and transparent outcome.

DOI: 10.34148/teknika.v14i3.1358
TEKNIKA, Volume 14(3), November 2025, pp. 433-443
ISSN 2549-8037, EISSN 2549-8045



Figure 24. Candidate Vote Counts

V. CONCLUSION

Based on the experiments and evaluations conducted, several conclusions can be drawn from this study:

1. The developed e-voting application has successfully implemented smart contracts on the Polygon network to automate the election process. All critical data, such as voter registration and ballot casting, are directly recorded on the blockchain, ensuring immutability and transparency. The testing results confirmed that the smart contract accurately logged transactions in accordance with predefined rules, and the data could be verified publicly through PolygonScan.
2. The integration of ZK-SNARK proved effective in preserving voter anonymity. The experimental results demonstrated that voters were able to prove their eligibility without disclosing their real identities, while the system reliably verified these proofs. Furthermore, the system was capable of detecting and preventing double voting attempts, thereby enhancing the security and trustworthiness of the overall election process.

These findings confirm that the combination of blockchain technology and ZK-SNARK provides a secure, transparent, and privacy-preserving solution for digital elections. The proposed system not only ensures the integrity and verifiability of electoral data but also offers a practical foundation for scalable, decentralized e-voting implementations in the future.

REFERENCES

[1] M. Hajian Berenjestanaki, H. R. Barzegar, N. El Ioini, and C. Pahl, "Blockchain-Based E-Voting Systems: A Technology Review," *Electronics*, vol. 13, no. 1, p. 17, Dec. 2023, doi: 10.3390/electronics13010017.

[2] R. Taş and Ö. Ö. Tanrıöver, "A Systematic Review of Challenges and Opportunities of Blockchain for E-Voting," *Symmetry*, vol. 12, no. 8, p. 1328, Aug. 2020, doi: 10.3390/sym12081328.

[3] K. Yuan, P. Sang, S. Zhang, X. Chen, W. Yang, and C. Jia, "An electronic voting scheme based on homomorphic encryption and decentralization," *PeerJ Comput. Sci.*, vol. 9, p. e1649, Oct. 2023, doi: 10.7717/peerj-cs.1649.

[4] F. Rabia, A. Sara, and G. Taoufiq, "ZkSNARKs and Ticket-Based E-Voting: A Blockchain System Proof of Concept," *Data Metadata*, vol. 3, p. .341-.341, Oct. 2024, doi: 10.56294/dm2024.341.

[5] M.-V. Vladucu, Z. Dong, J. Medina, and R. Rojas-Cessa, "E-Voting Meets Blockchain: A Survey," *IEEE Access*, vol. 11, pp. 23293–23308, 2023, doi: 10.1109/ACCESS.2023.3253682.

[6] H. Yuhao and S. Peng, "A Decentralized Voting System on the Polygon Blockchain," *Procedia Comput. Sci.*, vol. 247, pp. 1304–1313, 2024, doi: 10.1016/j.procs.2024.10.156.

[7] H. Taherdoost, "Smart Contracts in Blockchain Technology: A Critical Review," *Information*, vol. 14, no. 2, p. 117, Feb. 2023, doi: 10.3390/info14020117.

[8] P. Baudier, G. Kondrateva, C. Ammi, and E. Seulliet, "Peace engineering: The contribution of blockchain systems to the e-voting process," *Technol. Forecast. Soc. Change*, vol. 162, p. 120397, Jan. 2021, doi: 10.1016/j.techfore.2020.120397.

[9] O. Oyelude and I. Olojede, "Evaluating the Effectiveness of Electronic Voting Systems in Nigeria: Challenges and Opportunities," *Afr. J. Polit. Adm. Stud.*, vol. 16, no. 2, pp. 84–104, Dec. 2023, doi: 10.4314/ajpas.v16i2.5.

[10] S. D. K. Hu, H. N. Palit, and A. Handojo, "Implementasi Blockchain: Studi Kasus e-Voting".

[11] B. Shrimali and H. B. Patel, "Blockchain state-of-the-art: architecture, use cases, consensus, challenges and opportunities," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 9, pp. 6793–6807, Oct. 2022, doi: 10.1016/j.jksuci.2021.08.005.

[12] K. Zīle and R. Strazdiņa, "Blockchain Use Cases and Their Feasibility," *Appl. Comput. Syst.*, vol. 23, no. 1, pp. 12–20, May 2018, doi: 10.2478/acss-2018-0002.

[13] E. B. Kavun and T. Yalcin, "A Lightweight Implementation of Keccak Hash Function for Radio-Frequency Identification Applications," in *Radio Frequency Identification: Security and Privacy Issues*, vol. 6370, S. B. Ors Yalcin, Ed., in Lecture Notes in Computer Science, vol. 6370. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 258–269. doi: 10.1007/978-3-642-16822-2_20.

[14] M. Sheikhi, R. Giustolisi, and C. Schuermann, "Receipt-Free Electronic Voting from zk-SNARK:," in *Proceedings of the 20th International Conference on Security and Cryptography*, Rome, Italy: SCITEPRESS - Science and Technology Publications, 2023, pp. 254–266. doi: 10.5220/0012140000003555.

[15] K. Karantias, "SoK: A Taxonomy of Cryptocurrency Wallets," 2020, 2020/868. Accessed: Apr. 15, 2025. [Online]. Available: https://eprint.iacr.org/2020/868

[16] M. Moniruzzaman, F. Chowdhury, and M. S. Ferdous, "Examining Usability Issues in Blockchain-Based Cryptocurrency Wallets," in *Cyber Security and Computer Science*, vol. 325, T. Bhuiyan, Md. M. Rahman, and Md. A. Ali, Eds., in Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 325. , Cham: Springer International Publishing, 2020, pp. 631–643. doi: 10.1007/978-3-030-52856-0_50.

DOI: 10.34148/teknika.v14i3.1358