Journal of Information Technology Ampera
Vol. 4, No. 3, 2023 e-ISSN: 2774-2121
https://journal-computing.org/index.php/journal-ita/index

A Blockchain-Based Verifiable Decentralized Mechanism for Digital Voting System

Rini Deviani[1]

[1]Informatics Department, Syiah Kuala University, Banda Aceh, Indonesia
Email: [1]rini.deviani@usk.ac.id

Abstract

Digital voting systems have garnered significant attention in recent years due to their potential to increase accessibility and transparency in elections. However, concerns about security, transparency, and verifiability have limited their widespread adoption. This paper proposes a blockchain-based verifiable decentralized mechanism for digital voting systems, aiming to address these concerns. By leveraging blockchain technology, cryptographic techniques, and decentralized consensus, our proposed mechanism offers a secure and transparent platform for conducting digital elections. This paper outlines the architecture, components, and benefits of our system, highlighting its potential to revolutionize the way we conduct elections. The proposed mechanism incorporates advanced encryption algorithms to secure the privacy of individual votes. This privacy-centric approach safeguards voter anonymity while still allowing for transparent verification of the overall election results. Security analysis underscores the effectiveness of the blockchain-based verifiable decentralized mechanism in addressing key concerns surrounding digital voting systems. By leveraging innovative technologies and cryptographic principles, the proposed system exhibits a high degree of resilience, transparency, and protection against potential threats, making it a promising solution for secure and trustworthy digital elections.

Keywords: Digital voting, blockchain, ethereum, decentralized app, web 3.0.

1. INTRODUCTION

Elections represent a fundamental pillar of democratic societies, serving as the cornerstone of citizen participation in the decision-making process. The integrity and fairness of elections are paramount, and as technology advances, there is a growing interest in harnessing digital innovations to enhance the electoral process. Digital voting systems, which enable citizens to cast their votes electronically, have emerged as a potential solution to streamline elections, increase accessibility, and reduce administrative burdens. However, the adoption of digital voting systems comes with a set of complex challenges that must be addressed to ensure the security, transparency, and trustworthiness of the electoral process.

This paper explores an approach to digital voting systems by introducing a blockchain-based verifiable decentralized mechanism. We recognize that the existing concerns surrounding digital voting systems, including potential vulnerabilities to cyberattacks, issues with voter authentication, and doubts about the accuracy of results, have impeded their widespread adoption. To address these concerns, we propose a comprehensive solution that leverages cutting-edge technologies and cryptographic techniques to create a secure and transparent platform for conducting digital elections.

The central premise of our proposed mechanism is the utilization of blockchain technology, a decentralized ledger system that has demonstrated its resilience and security in various applications, most notably as the foundation of cryptocurrencies like Bitcoin [1]. By integrating blockchain technology into the digital voting process, we aim to provide a tamper-resistant and immutable record of votes, thereby ensuring the integrity of the electoral system.

Furthermore, cryptographic techniques play a pivotal role in preserving voter privacy and securing the authenticity of ballots. Techniques such as homomorphic encryption enable voters to cast their ballots without revealing their choices, while zero-knowledge proofs facilitate verifiable vote tallying without disclosing individual votes [2]. These cryptographic safeguards are pivotal in addressing the concerns associated with digital voting, such as vote manipulation and coercion.

The mechanism also incorporates decentralized consensus algorithms, such as Proof of Work (PoW) or Proof of Stake (PoS), to safeguard the blockchain against manipulation and control by a single entity. This decentralization ensures that no single authority can manipulate or compromise the voting process, enhancing the system's overall security and trustworthiness [3].

In this paper, we provide an in-depth exploration of the architecture and components of our proposed mechanism, highlighting its various benefits, including enhanced security, transparency, verifiability, accessibility, and efficiency. While we acknowledge that challenges remain, particularly related to voter authentication, scalability, and potential cyber threats, we believe that our approach represents a significant step toward redefining the future of digital elections.

The subsequent sections of this paper will delve into the architecture of the proposed mechanism, discuss its benefits and challenges, and conclude by emphasizing the potential impact of our blockchain-based verifiable decentralized mechanism on the democratic process.

2. LITERATURE REVIEW

2.1 Blockchain Technology

Blockchain technology, originally designed for cryptocurrencies like Bitcoin, has found applications in various domains due to its properties of decentralization, immutability, and transparency. In the context of voting systems, blockchain can serve as a tamper-resistant ledger for recording votes securely and transparently. In essence, the blockchain functions as a decentralized database that stores transaction records shared among participants. The validation of each transaction relies on the consensus of a majority of members, preventing deceitful transactions from gaining collective approval. Once a record is established and endorsed by the blockchain, it becomes immutable and impervious to alterations or disappearance. In contemporary times, blockchain technology is hailed as a ground-breaking invention, rivalling the Internet in significance. While the Internet connects individuals for online business processes, blockchain addresses trust issues through peer-to-peer networking and public-key cryptography [4].

2.2 Cryptographic Techniques

Cryptography safeguards vital information by transforming it into encrypted data, accessible only to authorized recipients. These recipients use a specific key to decrypt the obscured data, restoring it to its original textual form. The conversion of original text into encrypted text (ciphertext) with a designated key is known as the encryption process, while the reverse is termed the decryption process [5]. Advanced cryptographic techniques, such as homomorphic encryption, zero-knowledge proofs, and multi-party computation, play a crucial role in securing the integrity and privacy of digital votes. These techniques enable voters to cast their ballots without revealing their choices and allow for the verifiable tallying of votes.

2.3 Decentralized Consensus

Decentralized consensus algorithms, like Proof of Work (PoW) and Proof of Stake (PoS), are used to ensure the immutability and security of blockchain-based systems. These algorithms prevent malicious actors from manipulating the voting records.

2.4 Web 3.0

Web 3.0, often referred to as the third generation of the internet, heralds a transformative shift in the way we interact with the digital world. Unlike its

predecessor, Web 2.0, which largely relies on centralized platforms and intermediaries, Web 3.0 is characterized by decentralization. It leverages technologies like blockchain and distributed ledger technology to create trustless networks, where data is secured cryptographically and control is distributed among users rather than held by centralized entities. This decentralization paves the way for a more transparent, equitable, and secure digital ecosystem.

At the heart of Web 3.0 is the concept of data ownership and privacy. Users regain control over their digital identities and personal information, deciding who can access their data and under what conditions. Decentralized Identity (DID) solutions play a pivotal role in this shift, allowing individuals to possess self-sovereign digital identities that can be used across the internet. Additionally, Web 3.0 fosters interoperability, enabling seamless communication and data sharing across various decentralized networks and platforms. While still in its early stages, Web 3.0 holds the promise of revolutionizing numerous industries, including finance, supply chain management, and social networking, by empowering users and reducing reliance on centralized authorities [6].

2.5 Ethereum

Ethereum, often abbreviated as ETH, is a ground-breaking blockchain platform that has reshaped the landscape of decentralized technology. Introduced by Vitalik Buterin in 2015 [7], Ethereum extends the capabilities of blockchain far beyond mere cryptocurrency transactions. At its core, Ethereum enables the creation and execution of smart contracts, self-executing code that automates agreements and tasks without the need for intermediaries. These smart contracts open the door to a vast array of applications, including decentralized finance (DeFi), non-fungible tokens (NFTs), decentralized applications (DApps), and more [8].

Ethereum's native cryptocurrency, Ether (ETH), serves as the fuel for these operations, facilitating transactions and incentivizing network participants [9]. Ethereum's commitment to decentralization and its transition to a more energy-efficient Proof of Stake (PoS) consensus mechanism with Ethereum 2.0 demonstrates its dedication to scalability and sustainability. As a result, Ethereum continues to be a driving force behind the evolution of the blockchain and decentralized technology space, sparking innovation and revolutionizing industries across the globe.

2.6 Solidity

Solidity is a high-level, statically typed programming language primarily used for developing smart contracts on the Ethereum blockchain and other Ethereum-compatible networks [10]. Created by Gavin Wood in 2014, Solidity has become the go-to language for Ethereum developers due to its simplicity and efficiency in writing self-executing contracts that facilitate decentralized applications (DApps) [11].

Solidity's syntax is influenced by JavaScript and Python, making it accessible to a wide range of developers. One of its key features is the ability to define and enforce rules and conditions within smart contracts, enabling trustless and transparent interactions on the blockchain. However, Solidity code must be meticulously crafted, as any vulnerabilities can lead to significant financial losses, highlighting the importance of rigorous auditing and security practices in the world of blockchain development. Solidity continues to evolve, with regular updates and improvements, to support the growing ecosystem of decentralized applications and blockchain-based solutions [12].

3. METHODS

3.1 Architecture Overview

The proposed blockchain-based verifiable decentralized mechanism for digital voting systems is designed to provide a secure, transparent, and tamper-resistant platform for conducting elections. The architecture consists of the following key components, also detailed in Figure 1. This architecture provides fairness by keeping the casted vote encrypted till the ending time of the election. After ending time, the voter can verify their casted vote, ensuring verifiability [13] . Detailed steps are as below:

1) Voter Registration: The first step in our digital voting mechanism is the registration of eligible voters. Each voter is assigned a unique cryptographic key pair, consisting of a public key and a private key. The public key is used to verify the authenticity of the voter, while the private key is used to cast a vote securely.
2) Vote Casting: When a voter decides to cast their vote, they create a digital ballot containing their choices. The ballot is encrypted using homomorphic encryption techniques to ensure the voter's privacy. The encrypted ballot, along with the voter's digital signature, is then recorded on the blockchain.

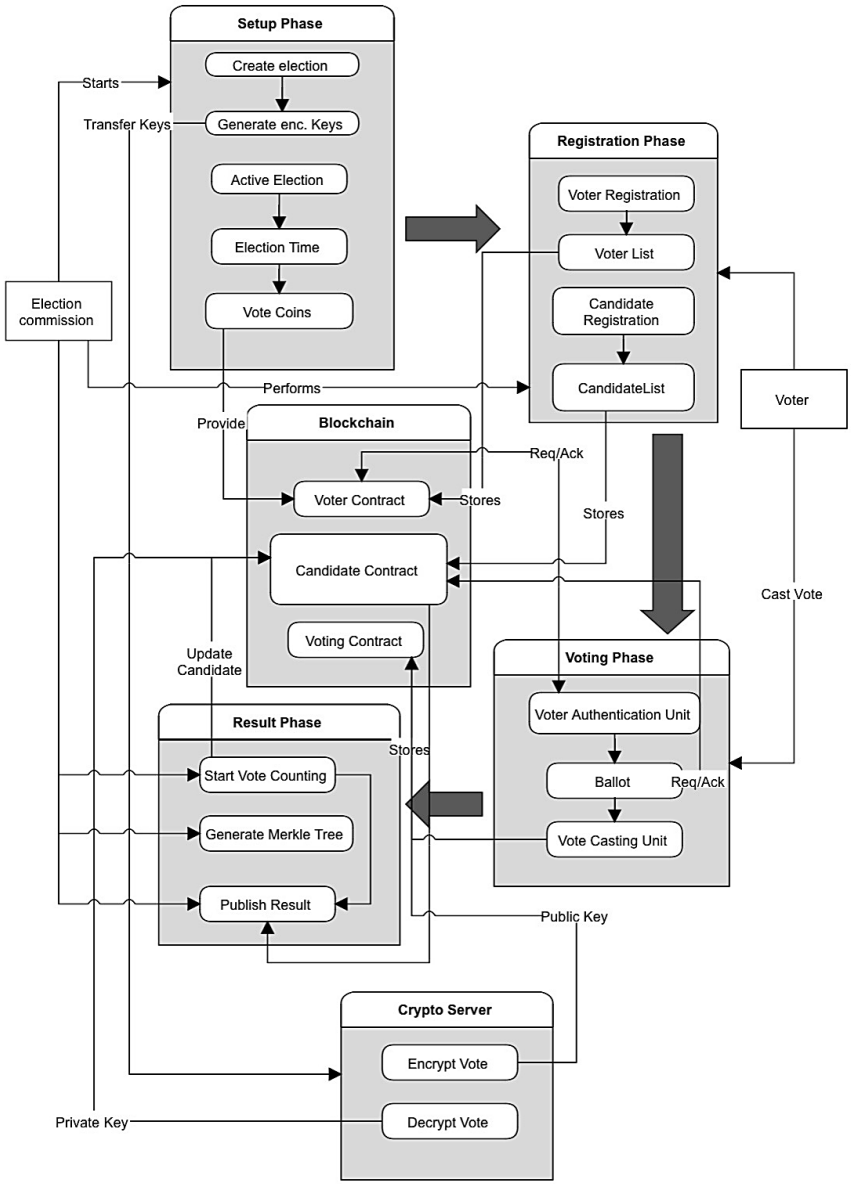

Figure 1. Blockchain based Voting Flowchart

3) Verifiable Tallying: Tallying the votes is a transparent and verifiable process. Each node in the blockchain network can independently verify the validity of the recorded votes using cryptographic proofs. Zero-

knowledge proofs are employed to confirm that the votes were cast correctly without revealing their content. Once the tallying is complete, the election results are stored on the blockchain.

4) Decentralized Consensus: To maintain the security and integrity of the blockchain, a decentralized consensus algorithm (e.g., PoW or PoS) is utilized. This prevents any single entity from controlling the network and ensures that the blockchain remains tamper-resistant.
5) Voter Verification: After the election, voters can verify that their votes were correctly recorded on the blockchain without revealing their choices. This verification process enhances transparency and trust in the system.

3.2 Proposed Algorithm

The blockchain network used in this system can be based on a permissioned blockchain, ensuring that only authorized participants (e.g., election authorities and validators) can participate in the consensus process. The use of a permissioned blockchain enhances security and control while still maintaining decentralization.

Algorithm 1 defines a voting system with participants represented by the Voter struct. Each participant is assigned a weight, indicating their voting influence, and can delegate their vote to another participant. Proposals, represented by the Proposal struct, are initialized in the constructor, and participants can vote for their preferred proposal. The chairperson, initially set to the contract deployer, has a default weight of 1. The contract provides a basic framework for decentralized voting, allowing for delegation of votes and accumulation of vote weight.

Algorithm 1 : Add candidate (proposal)

```
// Define the Voter structure
struct Voter {
    uint weight;         // weight is accumulated by delegation
    bool voted;          // if true, that person already voted
    address delegate;    // person delegated to
    uint vote;           // index of the voted proposal
}

// Define the Proposal structure
struct Proposal {
    bytes32 name;        // short name (up to 32 bytes)
    uint voteCount;      // number of accumulated votes
   }

// Declare public variables
address chairperson;
mapping(address => Voter) voters;
Proposal[] proposals;

// Constructor to create a new ballot with given proposal names
function Ballot(bytes32[] proposalNames) {
    chairperson = msg.sender;
```

```
    voters[chairperson].weight = 1;

    // Iterate through proposalNames to create Proposal objects and add them to proposals
    for each name in proposalNames {
        proposals.push(Proposal({
            name: name,
            voteCount: 0
        }));
    }
}
```

Give right to vote in Algorithm 2 grants the right to vote to a specified address (voter). The function checks two conditions using 'require' statements: first, that the sender of the transaction invoking the function is the chairperson, ensuring that only the chairperson can assign voting rights; second, that the specified voter has not already voted, preventing double voting. Additionally, the function ensures that the voter's weight is initially set to 0, avoiding the possibility of assigning voting rights multiple times. If all conditions are met, the function sets the voter's weight to 1, effectively giving them the right to participate in the voting process.

Algorithm 2 : Give right to vote

```
// Function to give the right to vote to a specific address
function giveRightToVote(voter)
    // Check if the caller is the chairperson
    if msg.sender is not equal to chairperson
        throw error "Only chairperson can give the right to vote."

    // Check if the voter has not voted already
    if voter has already voted
        throw error "The voter already voted."

    // Check if the voter does not already have the right to vote
    if voter's weight is not equal to 0
        throw error "The voter already has the right to vote."

    // Assign the right to vote by setting the voter's weight to 1
    voter.weight = 1
```

Function 'delegate' allows a voter to delegate their vote to another address. It checks that the sender has not already voted and that self-delegation is not allowed. The function then iterates through the delegation chain to ensure there are no loops, indicating a valid delegation path. If the delegate has already voted, the sender's weight is added to the corresponding proposal's vote count; otherwise, the delegate's weight is increased. The sender's voted status is updated, and the delegation information is recorded, effectively implementing a delegation mechanism for votes in the contract.

Algorithm 3 : Delegate a Vote

```
// Function to delegate your vote to the voter 'to'
function delegate(to)
    // Get the reference to the sender's Voter structure
    sender = voters[msg.sender]
```

```
    // Check if the sender has not already voted
    if sender has already voted
        throw error "You already voted."

    // Check if self-delegation is disallowed
    if to is equal to msg.sender
        throw error "Self-delegation is disallowed."

    // Loop to find the final delegate in case of chained delegation
    while voters[to].delegate is not equal to address(0)
        // Update 'to' to the next delegate
        to = voters[to].delegate

        // Check for a loop in delegation
        if to is equal to msg.sender
            throw error "Found loop in delegation."

    // Mark the sender as voted and set the delegate
    sender.voted = true
    sender.delegate = to

    // Get the reference to the delegate's Voter structure
    delegate_ = voters[to]

    // Check if the delegate has already voted
    if delegate_.voted
        // If the delegate already voted, add to the number of votes for the chosen
proposal
        proposals[delegate_.vote].voteCount += sender.weight
    else
        // If the delegate did not vote yet, add to her weight
        delegate .weight += sender.weight
```

Function ‘vote’ allows a participant to cast their vote for a specific proposal. It verifies that the sender has the right to vote by checking their weight, ensuring they haven't voted before. If these conditions are met, the function marks the sender as having voted for the specified proposal and increments the vote count for that proposal by the sender's weight. The code includes a safety check to prevent out-of-bounds access to the proposals array, automatically reverting any changes if the provided proposal index is beyond the array's range.

Algorithm 4 : Cast a Vote

```
// Function to give your vote to a specific proposal
function vote(proposal)
    // Get the reference to the sender's Voter structure
    sender = voters[msg.sender]

    // Check if the sender has the right to vote
    if sender.weight is equal to 0
        throw error "Has no right to vote."

    // Check if the sender has not already voted
    if sender has already voted
        throw error "Already voted."

    // Mark the sender as voted and set the voted proposal
    sender.voted = true
    sender.vote = proposal

    // Add the sender's weight to the vote count of the chosen proposal
    proposals[proposal].voteCount += sender.weight
```

Algorithm 5 : Count Winning candidate (proposal)

```
function winningProposal() returns winningProposal
    winningVoteCount = 0
    winningProposal = 0
    for each proposal in proposals
        if proposal.voteCount > winningVoteCount
            winningVoteCount = proposal.voteCount
            winningProposal = index of proposal in proposals
    return winningProposal
```

Function winningProposal determines the index of the winning proposal by iterating through the array of proposals. It initializes a variable winningVoteCount to 0 and checks each proposal's vote count. If a proposal's vote count surpasses the current winning vote count, it updates the winning vote count and sets the winningProposal variable to the index of that proposal. After the loop completes, the function returns the index of the proposal with the highest vote count, signifying the winning proposal in the voting system.

3.3 Deployment Details

Thirdweb is a development framework that allows you to build web3 functionality into your applications [15].

1) Smart Contract Development: Smart contract is written in Solidity and has been compiled to bytecode with an associated ABI (Application Binary Interface).
2) Access ThirdWeb's Platform: The official website or platform is thirdweb.com. It is required to create a account
3) Connect to a Wallet: Some platforms required to connect a cryptocurrency wallet (such as Metamask) to interact with the platform.
4) Deploy Smart Contract:
   - Look for a section or feature related to smart contract deployment.
   - Uploaded the compiled bytecode and provide any necessary configuration parameters (e.g., constructor arguments).
5) Confirm and Deploy:
   - Review the details of the deployment.
   - Confirm the transaction and wait for the deployment process to complete.
6) Retrieve Deployment Information: After successful deployment, there should be information such as the contract address, transaction hash, etc.

4 RESULTS AND DISCUSSION

In a blockchain-based voting system using a public network, the goal is to leverage the transparency, security, and decentralization features of blockchain technology to ensure the integrity of the voting process. The key components of such a system, including gas fees and the use of the Metamask extension is depicted in Figure 2.

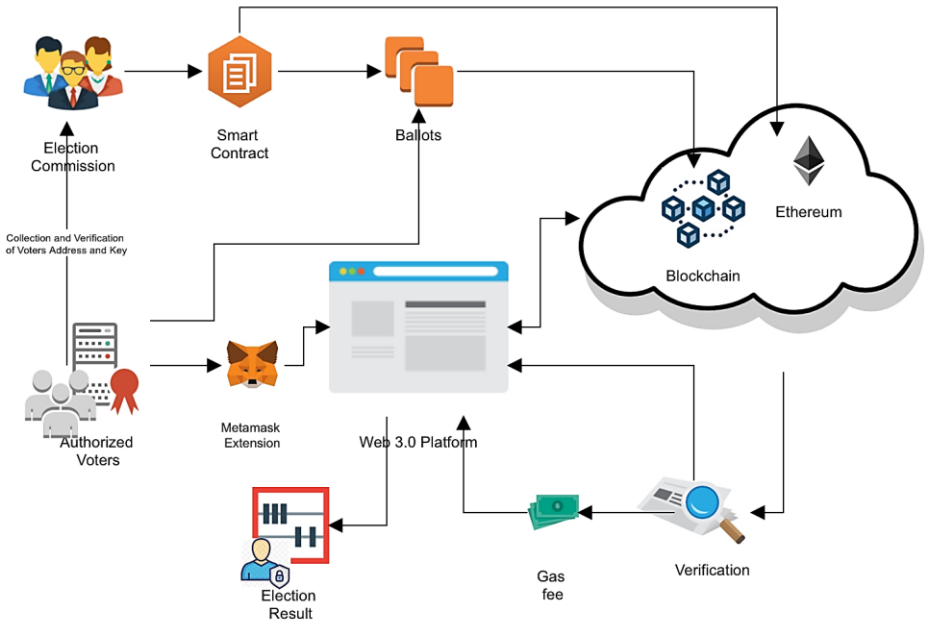

Figure 2. Voting Process Diagram

A blockchain-based voting system on a public network harnesses the power of decentralized technology to revolutionize the electoral process. Utilizing smart contracts on platforms like Ethereum, the system assigns unique voting tokens to eligible participants, ensuring a secure and transparent method of casting votes.

The public ledger records each vote, guaranteeing transparency and immutability while protecting voter identities through cryptographic techniques. As seen in Figure 3, gas fees, incurred for transaction processing, are a vital aspect, and users, equipped with Metamask extensions, can seamlessly manage these fees and interact with the voting smart contract.

Metamask serves as a user-friendly interface, allowing voters to securely store their tokens, submit votes, and cover associated transaction costs. The decentralized nature of blockchain ensures multiple nodes validate each transaction, reducing the risk of fraud, and the immutable record safeguards against tampering. This system enhances trust in the electoral process by combining the security of blockchain technology, the transparency of public networks, and the user-friendly interface of Metamask.

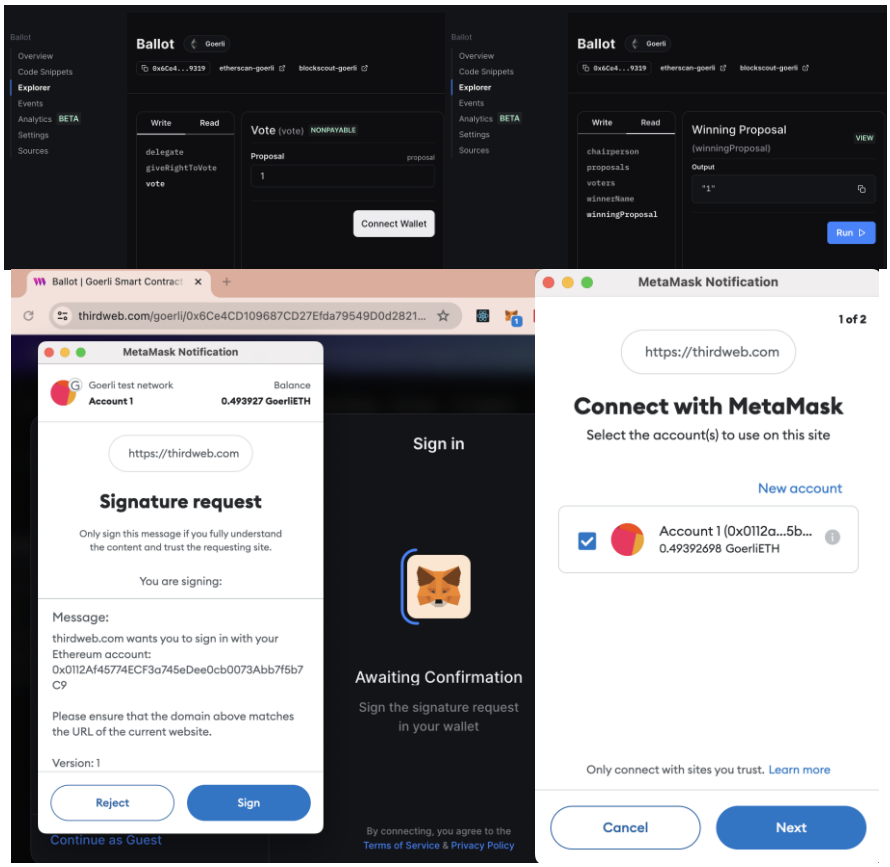

Figure 3. Smart Contracts Deployed and Run on Thirdweb

Decentralized digital voting system with verifiable mechanism utilizing blockchain, Web 3.0, and Metamask introduces both promising security features and potential challenges. Below is a security analysis highlighting key aspects:

1) Immutability and Integrity
   - Strengths: The use of blockchain ensures data immutability, enhancing the integrity of the voting system. Once votes are recorded, they cannot be altered, providing a reliable and tamper-resistant record.
   - Considerations: While immutability is a strength, it also means that any vulnerabilities in the smart contracts or issues with the recorded data are permanent. Rigorous testing, code audits, and constant monitoring are essential to prevent and quickly respond to potential security threats.
2) Smart Contract Security
   - Strengths: Smart contracts execute the logic of the voting system. Implementing secure coding practices, conducting thorough audits, and regular code updates can mitigate potential vulnerabilities.
   - Considerations: Smart contracts are susceptible to various attacks such as reentrancy and overflow exploits. Continuous monitoring and prompt patching of vulnerabilities are critical. Establishing a bug bounty program may enhance security by incentivizing the community to identify and report vulnerabilities.
3) User Security with MetaMask
   - Strengths: MetaMask provides a user-friendly interface for interacting with decentralized applications and ensures secure key management.
   - Considerations: Users must be educated about phishing risks, ensuring they only interact with authentic contracts and applications. Developers should implement secure MetaMask integration practices, and users should verify the legitimacy of the application through trusted channels.
4) Privacy and Confidentiality
   - Strengths: The decentralized nature of blockchain can enhance privacy by eliminating a central authority. However, the transparency of the blockchain can compromise voter privacy.
   - Considerations: Employing privacy-preserving technologies, such as zero-knowledge proofs, can address concerns related to voter privacy while maintaining the benefits of decentralization.

5 CONCLUSION

In conclusion, the blockchain-based verifiable decentralized mechanism for digital voting systems presented in this paper represents a significant step forward in addressing the security, transparency, and verifiability concerns associated with digital elections. By leveraging blockchain technology, cryptographic techniques, and decentralized consensus, this mechanism offers a secure, transparent, and efficient platform for conducting elections in the digital age. Further research and development in this field will be essential to overcome the remaining challenges

and pave the way for the widespread adoption of digital voting systems, ultimately enhancing the democratic process.

REFERENCES

[1] Nakamoto, Satoshi, and A. Bitcoin. "A peer-to-peer electronic cash system." *Bitcoin.–URL: https://bitcoin. org/bitcoin.* vol. 4, no. 2, 2008.

[2] M. Tejedor-Romero, D. Orden, I. Marsa-Maestre, J. Junquera-Sanchez, and J. M. Gimenez-Guzman, "Distributed remote e-voting system based on shamir's secret sharing scheme," *Electronics (Switzerland)*, vol. 10, no. 24, Dec. 2021, doi: 10.3390/electronics10243075.

[3] Delgado-Segura S, Pérez-Sola C, Navarro-Arribas G, Herrera-Joancomartí J. Analysis of the bitcoin utxo set. InFinancial Cryptography and Data Security: FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers 22 2019 (pp. 78-91). Springer Berlin Heidelberg.

[4] D. Efanov and P. Roschin, "The All-Pervasiveness of the Blockchain Technology," *Procedia Comput Sci*, vol. 123, pp. 116–121, 2018, doi: https://doi.org/10.1016/j.procs.2018.01.019.

[5] Y. Alemami, M. A. Mohamed, and S. Atiewi, "Research on various cryptography techniques," *International Journal of Recent Technology and Engineering*, vol. 8, no. 2 Special Issue 3, pp. 395–405, Jul. 2019, doi: 10.35940/ijrte.B1069.0782S319.

[6] C. Chen *et al.*, "When Digital Economy Meets Web3.0: Applications and Challenges," *IEEE Open Journal of the Computer Society*, vol. 3, pp. 233–245, 2022, doi: 10.1109/OJCS.2022.3217565.

[7] Buterin V. A next-generation smart contract and decentralized application platform. white paper. 2014 Jan 14;3(37):2-1.

[8] Buterin, Vitalik. "Ethereum: platform review." *Opportunities and Challenges for Private and Consortium Blockchains* 45 (2016).

[9] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur, and H. N. Lee, "Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract," *IEEE Access*, vol. 10. Institute of Electrical and Electronics Engineers Inc., pp. 6605–6621, 2022. doi: 10.1109/ACCESS.2021.3140091.

[10] C. Dannen, "Solidity Programming," in *Introducing Ethereum and Solidity*, Apress, 2017, pp. 69–88. doi: 10.1007/978-1-4842-2535-6_4.

[11] Bauer, D.P. (2022). Solidity. In: Getting Started with Ethereum. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-8045-4_2

[12] S. Hwang and S. Ryu, "Gap between theory and practice: An empirical study of security patches in solidity," in *Proceedings - International*

Conference on Software Engineering, IEEE Computer Society, Jun. 2020, pp. 542–553. doi: 10.1145/3377811.3380424.

[13] S. T. Alvi, M. N. Uddin, L. Islam, and S. Ahamed, "DVTChain: A blockchain-based decentralized mechanism to ensure the security of digital voting system voting system," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 9, pp. 6855–6871, Oct. 2022, doi: 10.1016/j.jksuci.2022.06.014.

[14] J. Sadowski and K. Beegle, "Expansive and extractive networks of Web3," *Big Data Soc*, vol. 10, no. 1, p. 20539517231159628, Jan. 2023, doi: 10.1177/20539517231159629.