TITLE OF PROJECT

CENG 3550, DECENTRALIZED SYSTEMS AND APPLICATIONS

Mandana Zooyousefin
180709711@posta.mu.edu.tr

Tuesday 28th January, 2025

Abstract

This study explores how decentralized voting systems provide a reliable and transparent election environment. Using blockchain technology, the system aims to eliminate the possibility of tampering with election results, thus increasing trust in the voting process. The study discusses the fundamental principles of the system, comparisons with existing works in the literature, and the prototype implementation of the proposed system.

1 Introduction

The decentralized voting system is implemented using the Ethereum blockchain to ensure transparency, security, and immutability. This project leverages the power of Solidity for smart contract development and provides a robust, tamper-proof voting mechanism suitable for educational purposes.

2 Fundamentals

Blockchain technology underpins the decentralized voting system, providing critical features such as:

- **Immutability:** Ensures that once votes are recorded, they cannot be altered.
- **Decentralization:** Eliminates the need for a central authority, ensuring unbiased operation.
- **Transparency:** Allows all participants to verify votes and results in real-time.

The use of Ethereum's smart contracts ensures that the voting process is automated and secure, leveraging the Solidity programming language for contract development.

3 Related Works

Decentralized voting systems have been explored in various academic and professional contexts. Previous implementations include:

- **Helios:** An online voting platform that emphasizes verifiability and security, though it does not leverage blockchain.

- **Follow My Vote:** A blockchain-based voting platform focused on transparency and voter anonymity.
- **Polys:** A blockchain voting system for organizations, providing secure and customizable voting experiences.

These projects highlight the potential of technology to improve voting systems but also demonstrate challenges such as scalability, voter authentication, and user adoption.

4 System Proposal

This project proposes a decentralized voting system that addresses common issues in traditional voting mechanisms, including:

- **Vote Integrity:** Ensures that votes cannot be altered or deleted once cast.
- **Accessibility:** Allows voters to participate from anywhere with an internet connection.
- **Security:** Prevents double voting and protects against malicious attacks using reentrancy guards.
- **Transparency:** Enables all participants to view the voting process and results without compromising voter privacy.

The proposed system is built on Ethereum and uses smart contracts to automate and secure the voting process. Future enhancements may include the integration of identity verification mechanisms to ensure voter eligibility.

5 IMPLEMENTATION

6 Core Components of the Voting Contract

6.1 Candidate Registration

The system allows for the registration of candidates through the contract constructor. Candidates are stored in a `mapping` structure, which links a unique identifier (ID) to the candidate's name. This ensures each candidate can be uniquely identified.

```
mapping(uint256 => string) public candidates;
uint256 public candidateCount;
```

Example: `candidates[0]` might return "Alice," and `candidates[1]` might return "Bob."

6.2 Voting Functionality

The `vote` function allows eligible participants to cast their votes. This function is secured against double voting and invalid candidate IDs:

```
function vote(uint256 candidateId) public {
    require(candidateId < candidateCount, "Invalid candidate ID");
    require(!_hasVoted[msg.sender], "You have already voted");
    require(reentrancyGuard == 0, "Reentrant call detected");

    // Reentrancy protection
    reentrancyGuard = 1;
    votes[candidateId]++;
    _hasVoted[msg.sender] = true;
    reentrancyGuard = 0;
}
```

6.3 Security Features

1. **Reentrancy Protection:**
   - A `reentrancyGuard` variable prevents recursive calls during vote processing.
2. **Double Voting Prevention:**
   - The `_hasVoted` mapping ensures that an address can only vote once.

6.4 Transparency and Data Retrieval

- **View Functions:**

  ```
  function candidates(uint256 id) public view returns (string memory);
  function votes(uint256 id) public view returns (uint256);
  ```

- **Voter Status:**

  ```
  function hasVoted(address voter) public view returns (bool) {
      return _hasVoted[voter];
  }
  ```

7 Deployment and Testing

7.1 Deployment

The smart contract is deployed using Remix IDE:

1. Constructor arguments (e.g., candidate names: `"Alice, Bob, Charlie"`) are passed during deployment.
2. The contract is tested on Remix VM (Cancun), a local test environment that simulates the Ethereum blockchain.

7.2 Testing Workflow

1. **Candidate Query:** Use the `candidates` function to retrieve registered candidates by ID.
2. **Vote Casting:** Call the `vote` function with a valid candidate ID.
3. **Vote Count Check:** Verify the updated vote count using the `votes` function.
4. **Duplicate Vote Prevention:** Attempt to vote again with the same address and observe the `"You have already voted"` error.

8 Blockchain-Specific Features

- **Decentralization:**
  - The system operates on a distributed blockchain network, eliminating the need for a central authority.
- **Immutability:**
  - Once deployed, the contract ensures data integrity and cannot be tampered with.
- **Security:**
  - Prevents vote manipulation through robust double voting and reentrancy protections.
- **Transparency:**
  - All votes and candidate data are publicly accessible on the blockchain.

9 Deployment and Testing

9.1 Deployment

The smart contract is deployed using Remix IDE:

1. Constructor arguments (e.g., candidate names: `"Alice, Bob, Charlie"`) are passed during deployment.
2. The contract is tested on Remix VM (Cancun), a local test environment that simulates the Ethereum blockchain.

9.2 Testing Workflow

1. **Candidate Query:** Use the `candidates` function to retrieve registered candidates by ID.
2. **Vote Casting:** Call the `vote` function with a valid candidate ID.
3. **Vote Count Check:** Verify the updated vote count using the `votes` function.
4. **Duplicate Vote Prevention:** Attempt to vote again with the same address and observe the `"You have already voted"` error.

10 Blockchain-Specific Features

- **Decentralization:**
  - The system operates on a distributed blockchain network, eliminating the need for a central authority.
- **Immutability:**
  - Once deployed, the contract ensures data integrity and cannot be tampered with.
- **Security:**
  - Prevents vote manipulation through robust double voting and reentrancy protections.
- **Transparency:**
  - All votes and candidate data are publicly accessible on the blockchain.

11 Potential Enhancements

- **Whitelist Implementation:**
  - Limit voting to specific authorized addresses.
- **Signature Verification:**
  - Ensure votes are signed cryptographically for added security.
- **Advanced Voting Mechanisms:**
  - Support for weighted voting or preferential voting systems.

12 Summary

This implementation demonstrates the core functionalities of a decentralized voting system built on Ethereum. While designed for educational purposes, it highlights essential blockchain features such as decentralization, transparency, and security. The system is simple yet effective, providing a foundational example for blockchain-based voting solutions.

13 Conclusion

In conclusion, this decentralized voting system offers a secure and transparent alternative to traditional voting mechanisms by leveraging blockchain technology. The implementation ensures fairness, prevents double voting, and enhances trust in the electoral process. While this project serves as an educational demonstration, future improvements such as identity verification and more advanced voting algorithms could further increase its usability in real-world applications. Blockchain-based voting has the potential to revolutionize elections by ensuring integrity, accessibility, and verifiability, making democratic processes more robust and reliable.

14 Acknowledgment

We would like to express our sincere gratitude to Dr.Enis Karaarslan for their guidance and support throughout this project. Finally, we acknowledge the open-source blockchain community and the resources provided by Ethereum developers, which were instrumental in this project.

For the full source code and implementation details, please visit our GitHub repository at:

`https://github.com/mandanazooyousefi/Decentralized_Voting_System/tree/main/Decentralized_18070971`