DLSU RESEARCH CONGRESS 2023
MANILA, PHILIPPINES
JULY 5-7, 2023
Fostering a Humane and Green Future:
Pathways to Inclusive Societies and Sustainable Development

Implementation of a Double Voting Prevention Algorithm for an E-voting Blockchain Application

John Henry F. Cagaoan[1], Jacob Israel R. Salazar[1] and Katrina Ysabel C. Solomon[1,*]
[1]*Advanced Research Institute for Informatics, Computing, and Networking (AdRIC), De La Salle University*
**Corresponding Author: katrina.solomon@dlsu.edu.ph*

Abstract: A blockchain is a data structure that tracks information and transactions through a digital ledger. Its main difference from databases is that blockchain is decentralized, and the data stored in blockchains are immutable. This characteristic promotes data integrity, which makes it a potentially excellent platform for industries and applications heavily relying on non-repudiation, like e-voting. Smart contracts, the programs running on the blockchain, implement election rules, but are not inherently designed to prevent instances of multiple voting attempts from a given user. While a legitimate vote may be stored and remain immutable, an additional layer of security is necessary to reject succeeding voting attempts from users who have already voted. This study aims to improve smart contracts to prevent double voting on blockchain-based e-voting systems. An e-voting architecture with user-facing web servers and an offchain module coordinates with the blockchain and smart contract to filter out multiple attempts of double voting. The developed prototype prevented instances of double voting for both frontend and backend access through the offchain module, which checks voter data, address used to cast the vote, and generated digital signatures from the database and blockchain. The system successfully blocks situations such as re-casting a vote for the same user, and using a different address to cast a vote, namely the administrator's address, another registered voter's address, and an unrecognized address.

Key Words: blockchain; e-voting; smart contracts; double voting prevention; digital signatures

1. INTRODUCTION

A blockchain is a data structure that allows users to keep track of information and transactions using a digital ledger. Each block in a blockchain refers to a data point that holds various information. This flexible nature of blockchain technology allows it to become applicable to different fields; medical, business, etc. Furthermore, blockchain technology possesses a unique property that makes it efficient and secure compared to other technologies - its decentralized nature. Other technologies such as online banking systems, electoral management

systems, and the like, require third parties to facilitate transactions and information storage; this makes it susceptible to data tampering, erroneous data gathering, and malicious attacks. In comparison, blockchain is secure because data becomes immutable once it is stored in the ledger (Khandelwal, 2019).

E-voting is one such field that can benefit from the decentralized and secure nature of blockchain. Nowadays, institutions rely on electoral management systems to perform vote counting on their behalf. However, since most of these systems require third parties, erroneous vote collection and double voting might occur which lowers the integrity of the election. This predicament can be addressed by implementing blockchain frameworks on e-voting systems. Blockchain-implemented e-voting systems can perform vote counting operations through the utilization of smart contracts. These smart contracts are pieces of instruction programmed to tally votes independently once the election time has been closed. Various studies have been done on this area and among those, a comprehensive theoretical framework and system design on how the blockchain technology can be applied by using smart contracts on e-voting systems was defined by Hjálmarsson et al. (2018).

Thuy et al. (2019) proposed a decentralized voting platform framework that utilized both web application and mobile application approaches in the Ethereum blockchain. Event administrators would use the web application to fill in data regarding the details of the voting event. The web application then submits an HTTP request to the event management system to use that data to deploy smart contracts to the network. The smart contracts that will be used in the blockchain contain the following components: an Ethereum wallet which serves as the address, a full node to communicate with the Ethereum network, and a database containing a list of all the contract addresses.

Khoury et al. (2018) created a system architecture of e-voting blockchain with the utilization of a mobile application for registration and voting. Ethereum Virtual Machine (EVM) was used as the blockchain runtime environment, on which transparent, consistent, and deterministic smart contracts were deployed by organizers for each voting event to run the voting rules.

Double voting is the act of casting more than one vote of a legitimate voter; this act usually occurs on remote E-voting systems that do not have security implementations (Augoye & Tomlinson, 2018). In recent studies, E-voting systems are now being adopted over blockchain platforms such as Ethereum and Hyperledger Fabric. These early e-voting blockchain applications are using smart contracts to emulate the elections: create a set of candidates, set the start and expiration of the election, allow the voters to vote during the election, and display the results. Registered voters are given a wallet with a value of one to allow them to cast one vote, the security problem occurs when a voter attempts to increment the value of their wallet, such that it would be possible for them to cast another vote. This act would result to double voting and ruin the integrity of the e-voting blockchain. Currently, smart contracts on e-voting blockchain applications do not have any double voting prevention algorithm to address this issue.

This study aims to improve existing smart contracts to prevent the possibility of double voting on blockchain-based e-voting systems based on the framework by Thuy et al. (2019).

2. METHODOLOGY

The implemented system consists of two (2) main components: the web servers and the blockchain as seen in Fig. 1 and will be further discussed in the succeeding subsections.

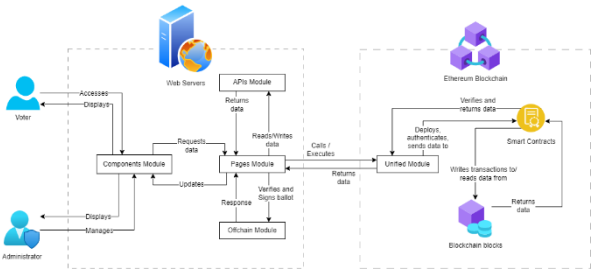
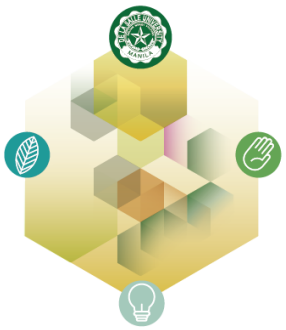


Fig. 1. Overall System Overview

2.1 Web Servers

The web servers in Fig. 2 consist of two (2) user-facing servers which host the ballot casting and login for voters, and an administrator dashboard for election administrators to manage elections, candidates, and voters. Next.js and Tailwind CSS are used in place of Handlebars to easily create routes for the website as well as for the REST API, and web pages for both the dashboard and voter ballot.



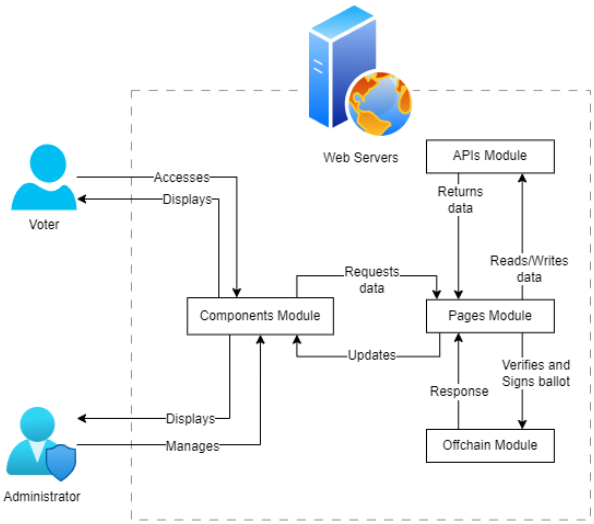


Fig. 2. Overview of the Web Servers Component

Both web servers use Node.js and MongoDB for the backend, with the database housing the administrator credentials, voter list, and candidate list prior to deploying all election details to the blockchain. Web3.js was used to allow the web servers to communicate with the Ethereum test network, particularly for the administrator deploying the election to the blockchain and voters casting their ballots.

The administrator dashboard allows election administrators to create new voters which will automatically be added along with existing voters once an election has been created, to create candidates to include in the upcoming election, as well as to create and monitor an election and deploy it to the blockchain.

The voters' server retrieves the ballot data from the database, then generates the ballot for the voters using the candidate list included in the election data, with the candidates and positions being patterned after De La Salle University's University Student Government election code.

An offchain module was also implemented to help process signatures and verification of voter ballot from the blockchain to reduce gas costs. Helper functions are still present in the smart contract so the web servers can call on these functions to countercheck the ballot's validity and voter authenticity from the blockchain against the web server's data received from the voters.

2.2 Blockchain

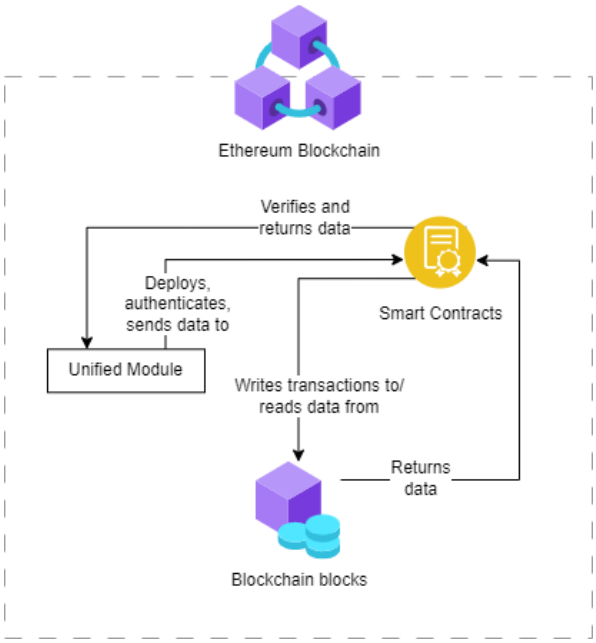


Fig. 3. Overview of the Blockchain Component

The blockchain component illustrated in Fig. 3 consists of a unified module which serves as the bridge between the web servers to the blockchain, and a smart contract, written using the object-oriented language Solidity, that houses the elections details, candidates list and candidate vote count, voters list, and functions to help validate voter data on the blockchain side of the system. Ganache (Truffle Suite, 2023) was used to set up a test network, launched with default settings and is set to automine blocks to have transactions automatically reflect on the network. Accounts used for the voters were also generated by the software, which in turn was imported to MetaMask (MetaMask, 2023), a cryptocurrency wallet, to allow voters to cast their ballots. Web3.js and Ethers.js were used as libraries to allow the web servers to communicate with the blockchain.

3. RESULTS AND DISCUSSION

Multiple test cases were created to verify the security of the voting system. These are different attempts to cast malicious instances of double voting. Postman (Postman, 2023) was used to send requests with the appropriate payload.

2.1 Casting Two Consecutive Votes

The test case assumes that the voter is attempting to re-cast their ballot to do double voting and expects the first vote to be successful and the second to indicate an error. Fig. 4 shows the voter UI with an error after clicking the submit ballot for the second time.

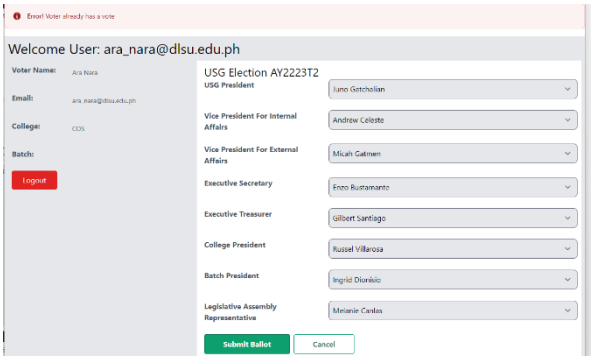


Fig. 4. Voter UI Indicating Voter Has Already Casted a Vote

2.2 Casting a Vote Using a Different Address

This test handles cases where a voter is a valid voter for the election but used a different address to cast their vote. All vote casts were assumed to be the first-time casting of the vote, as casting votes again from the same voter account will still be blocked. Several addresses were tested for this scenario, and which are further explained below:

- **Voter used the administrator's address** - This case assumes that the voter used the same address of the election administrator, which should not be allowed as election administrators should only use their address to deploy the election contract to blockchain.
- **Voter used a different valid voter's address** - This case handles situations wherein the malicious agent used a different voter's address to cast their own vote, of which the said voter is also a valid voter for the current election.
- **Voter used an unrecognized address** - This case tackles events where the malicious agent used an address that is not in use by any voter - an address that is not registered with the system.

In Fig. 5, it is shown that all three (3) voting attempts using the administrator's address, another valid voter's address, and an unrecognized address were logged as invalid.

Fig. 5. All Three Voting Attempts Using Different Addresses Caught by the System

2.3. Concurrency Test

In this test, two separate machines within the same local network are trying to access the voter application, to simulate casting a vote concurrently. In Fig. 6, the administrator's election logs successfully caught the double voting attempt and indicated the timestamp of both machines casting a vote.

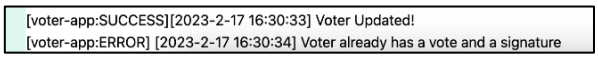


Fig. 6. Election Logs Showing a Concurrent Double Voting Attempt

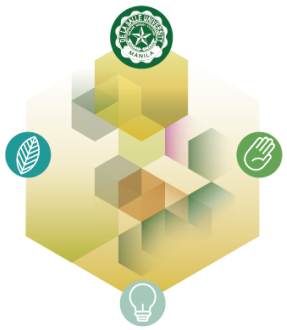
2.3. Smart Contract Static Analysis

Security tests for the smart contract were performed to ensure that the contract provides sufficient security to ensure the integrity of the election.

Remix (Remix Software, 2023), Ethereum IDE's Solidity Static Analysis plugin provides a way to examine the smart contract without execution, and helps developers identify security vulnerabilities. Modules used for testing should receive a passing mark to ensure the robustness of the contract. Table 1 indicates the relevant security modules used for testing based on Remix IDE's documentation.

Table 1. Summary of Static Analysis Test Results

Module	Description	Status
'tx.origin' is used	If used for authentication, must be modified to ensure a contract can only execute processes when invoked	Passed



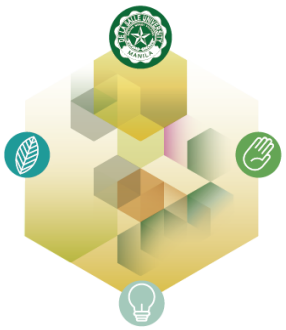
Module	Description	Status
Check effects: Potential reentrancy bugs	Check-Effects-Interaction pattern should be followed to avoid re-entrancy vulnerabilities	Passed
Inline assembly used	Inline assembly should only be used in edge cases and must be avoided as often as possible	Passed
Block timestamp: Semantics maybe unclear	Usage of keyword now must be clear as Solidity uses now as an alias for `block.timestamp`, which can influence miners.	Passed
Low level calls: Semantics maybe unclear	Using `call`, `callcode`, `delegatecall` in the contract should be avoided. Use `transfer` in transactions, if possible, for cases where ether transfer transaction should rollback in case of failure.	Passed
Blockhash usage: Semantics maybe unclear	A miner may influence a transaction outcome in the current hash if left uninitialized.	Passed
Selfdestruct: Beware of caller contracts	Callee contract can leave calling contracts unusable if implemented wrongly.	Passed

4. CONCLUSIONS

E-voting applications have been susceptible to double-voting due to the lack of security controls especially in the smart contract level. This research aimed to provide a system architecture for blockchain-based e-voting applications with security controls not only within the smart contracts, but also in offchain processing to mitigate double voting. The framework by Thuy et al. (2019) has been used as the basis for the system implementation. The system implementation has two main parts, a web application in which the administrator and the voters interact with, developed using Node.js, Next.js, and Tailwind CSS, and blockchain with Ganache serving as the test network, and linked to the web application with Web3.js and Ethers.js. An offchain module was developed to process signatures and execute verifications outside the smart contracts to help mitigate gas costs and to maintain that blockchain serves mainly to store data. The system was tested against multiple double-voting scenarios, particularly in using different addresses to ensure that the offchain module successfully identifies valid ballots and deters instances of double voting. All tests against the web application frontend, as well as accessing the backend API through a third-party application, were successful in blocking instances of double voting. Since the tests were able to catch malicious instances of double voting, non-malicious attempts such as accidentally clicking the submit ballot button twice or re-clicking the button due to slow Internet connection should also be prevented.

This research will benefit election administrators to have a tamper-proof system with blockchain as the technology provides emphasis on data integrity and non-repudiation which are important to maintain an election's integrity and security. This will also allow decentralized applications to further take off and allow election applications to become much more robust from malicious agents. The blockchain security integration via smart contracts provides the web application another layer of security as it is possible for data to be easily manipulated on the web application side by tampering the votes via MongoDB Atlas or by sending a malicious payload on a backend API of the server. The blockchain double voting prevention implementation proves to be a strong layer of security as it can be used to cross-reference data changes from a web application's basic database to the data stored on a secured blockchain via smart contracts.

One of the main disadvantages of deploying the system over blockchain is additional costs, as wallets to be used to cast votes need to have a balance, unlike centralized e-voting apps which



solely rely on databases, and as such, server hosting costs. Furthermore, blockchain processing tend to be slower, which was not evident in the implementation tests as the environment was local and Ganache was set to automine blocks. This may heavily impact user experience, and lead to non-malicious double voting attempts.

Despite the success of implementing double voting checks via digital signature on blockchain through offchain and on-chain module implementation, there are multiple recommendations the researchers indicate to improve further the system security and gas cost efficiency of the smart contract.

Due to hardware and cost limitations, the researchers advise deploying the smart contract to an Ethereum live network and the web application to web deployment services such as Heroku or Vercel to further test the resiliency of the smart contract and the offchain module against double voting attacks. With the study carried out using Ganache's basic settings, which only provides ten blockchain accounts and is set to automine, the researchers also advise using loaded Ethereum wallets to test the actual performance of the e-voting application in a production setting.

The researchers also recommend future work to address the high gas cost fee on smart contract implementation by making the digital signature verification more efficient and less data-intensive in storing voter and election information. A possible route for this is to change the structs used and transform election data into strings that can be parsed by an application.

5. REFERENCES

Augoye, V., & Tomlinson, A. (2018). Mutual Authentication in Electronic Voting Schemes. *2018 16th Annual Conference on Privacy, Security and Trust (PST)* (pp. 1-2). Belfast: IEEE. doi:10.1109/PST.2018.851421

Hjálmarsson, F. Þ., Hreiðarsson, G. K., Hamdaqa, M., & Hjálmtýsson, G. (2018). Blockchain-Based E-Voting System. *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)* (pp. 983-986). San Francisco: IEEE. doi:10.1109/CLOUD.2018.00151

Khandelwal, R. (2019). Taxation of Cryptocurrency Hard Forks. *The Contemporary Tax Journal, 8*(1). doi:https://doi.org/10.31979/2381-3679.2019.080105

Khoury, D., Kfoury, E. F., Kassem, A., & Harb, H. (2018). Decentralized Voting Platform Based on Ethereum Blockchain. *2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)* (pp. 1-6). Beirut: IEEE. doi:10.1109/IMCET.2018.8603050

MetaMask. (2023). *MetaMask developer documentation*. Retrieved from MetaMask: https://docs.metamask.io/

Postman, I. (2023). *Postman API Platform*. Retrieved from Postman: https://www.postman.com/

Remix Software, I. (2023). *Remix Docs Home*. Retrieved from Remix: https://remix.run/docs/en/main

Thuy, L.-C., Cao-Minh, K., Dang-Le-Bao, C., & Nguyen, T. A. (2019). Votereum: An Ethereum-Based E-Voting System. *2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF)* (pp. 1-6). Danang: IEEE. doi:10.1109/RIVF.2019.8713661

Truffle Suite. (2023). *Ganache*. Retrieved from Truffle Suite: https://trufflesuite.com/ganache/index.html