Applied Information Technology And Computer Science Vol. 2 No. 2 (2021) 001-020
© Universiti Tun Hussein Onn Malaysia Publisher's Office

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

Ethereum Blockchain in Preserving Integrity of Voting System

Arkayanik Law Yun Tong, Chuah Chai Wen*

Faculty of Computer Science & Information Technology,
Universiti Tun Hussein Onn Malaysia, Parit Raja, 86400, Malaysia

DOI: https://doi.org/10.30880/aitcs.2021.02.02.001
Received 14 June 2021; Accepted 09 September 2021; Available online 30 November 2021

Abstract: Voting is a method for a group of people makes a collective decision individually. It can be done by using paper-based method or voting system. Voting System has been around more than a decade. However, the most used voting method is still paper-based. The reason for this is because the concern of security risk that exist in voting system particularly the integrity of casted vote. With voting system being online, the risk of vote fraud happen increase. Thus, a mechanism to validate the integrity of casted votes is needed. In this project, the underlying technology that is proposed to solve this issue is Blockchain technology. Blockchain is known as immutable ledger that record every transaction that happened in a decentralized manner. The technology provides data integrity because of the nature of chaining each node hash value to each other in chronological order. The methodology for this project is Object-Oriented Analysis and Design (OOAD). The proposed system is developed using Java programming language. Overall, the developed system is able to save the casted vote into the Ethereum blockchain. Thus, the integrity of the casted vote is protected.

Keywords: E-voting, Blockchain, Data Integrity, Ethereum

1. Introduction

Voting is method used to reach a conclusion based on collected choices made by a group of people individually. In a formal voting process, it can be done either by paper or electronic machine. An electronic voting (e-voting) is voting done by using an internet connected device or machine. Although e-voting can facilitate voting process, it possessed security issues which can undermine the voting result [1]. In this study, the security issue that is focused is the integrity of the casted votes.

The integrity of casted vote is important to ensure the fairness of the poll result. If casted votes of a poll are modified without being notice, the result of the poll may be compromise. This is proven when in 2018, J. Alex Halderman, a computer scientist, ran a test on electronic voting systems in Massachusetts Institute of Technology (MIT). He demonstrated how to hack an e-voting system and compromised the voting result. The test is done by holding a mock contest between George Washington and Benedict Arnold. Three volunteers have casted their vote in Washington. However, Halderman, who is involved in the testing of security of the e-voting system, had meddled with the ballot

**Corresponding author: cwchuah@uthm.edu.my*

programming, changing the casted vote information. This caused Washington to lose to Arnold with one to two results, proving that the e-voting system is successfully compromised and the casted votes are modified [1]. Thus, the integrity of the casted vote must be protected. The proposed system saved the casted vote into Ethereum blockchain. By taking the advantages of the blockchain immutability property, the casted vote integrity can be preserved from any form of modification attack.

Besides the integrity of the casted votes, the anonymity of the voter also need to be protected. The reason for this is to protect voter's privacy and ensure the casted vote cannot be use to trace back the voter identity. In 2019, there is a case where a domestic abuse survivor is afraid to cast her ballot. She is afraid that her former abusive partner will be able to find her home address from the antique voting system from her area poll station [2]. Therefore, the anonymity of the voter must be protected to protect the voter identity and privacy. In the proposed system, the anonymity of the voters is protected by using two different mediums to stored casted vote information. The first medium is the system database which only stored the voter ID and poll ID. The second the medium is the Ethereum blockchain which only stored the casted information. The casted vote information includes the candidate ID and the poll ID. There is no information related to the voter stored inside the casted vote.

In the proposed system, the users can be classified as poll creator, candidate or voter. The user become poll creator when they create a new poll. When creating a new poll, the information need to input is the poll name, description and participant limit. After the poll is created, the poll creator can add candidates to the poll and published the poll afterward. After the poll is published, other users can send request to join the poll. The poll creator may approve or remove the request. If the user's request is approved, he or she became the voter of the poll. Each voter can only vote once for every poll. When the voter casted their vote, the casted vote information is stored into the blockchain. After the poll is closed, the poll creator, candidates and voters may view the poll result. The system generates the poll result by retrieving the casted vote of the poll from the blockchain.

2. Literature Review

This section explained about the literature reviews that have been conducted for this project. The goal of this literature review is to understand the background and technology used for this application.

2.1 Blockchain

Blockchain is a continuous block sequence that stores transaction records that are linked to one another. In blockchain, each block has its own hash value that acts as a unique identifier [3]. Figure 1 illustrates the blockchain structure.

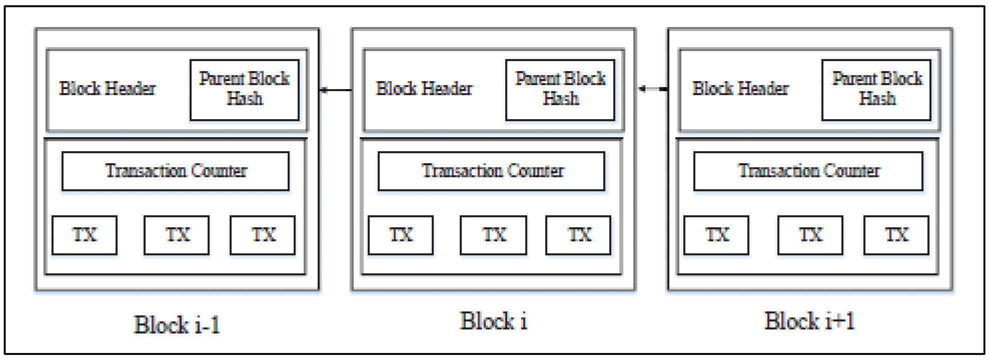


Figure 1: Example of Blockchain [3]

The block structure is divided into two different sections – block header and block body. The block header consists of six components. This includes block version, merkle time root hash, time stamp, nBits, nonce and parent block hash. On the other hand, a block body contains a transaction counter and the transaction itself [3]. Figure 2 illustrates the block structure.

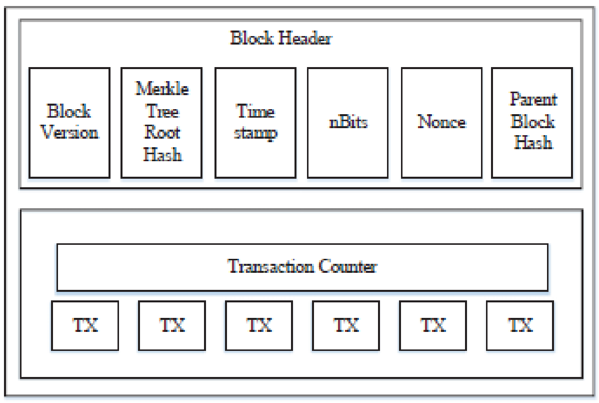


Figure 2: Block structure [3]

Table 1 describes six components of block header. These includes block version, merkle tree root hash, timestamp, nBits, nonce and parent block hash.

Table 1: Components of block header [3]

Components	Description
Block version	Contain block validation rules that need to be followed.
Merkle tree root hash	Hash value of all transactions contained in the block.
Timestamp	Current timestamp in seconds format.
nBits	Desire threshold of a valid block hash.
Nonce	Generated number used to get target hash value.
Parent block hash	Hash value of previous block.

Blockchain architecture design provides three security properties that protect its information. There are immutability, auditability and Traceability [4]. Table 2 shows each of the security properties of blockchain.

Table 2: Security properties of blockchain [4]

Security Properties	Description
Immutability	Each block has its own hash value that changes if the content is modified. If a block content is tampered, its hash value also changes. Thus, resulting the new hash value mismatches with the one that is already stored in the next block as the previous hash.
Auditability	All participants in the blockchain network have a copy of the blockchain transaction records. This allows them to search and validate transactions of specific blocks by themselves.
Traceability	Any errors or data m in the blockchain network can be traced and identified by using the consensus mechanism where every block holds parent hash value.

2.2 Ethereum Blockchain

Ethereum is a blockchain platform that uses Ether (ETH) as its cryptocurrency. Ether can be used to buy and trade goods and services. Ethereum platform allowed its users to create applications that operate on the blockchain in the same way that software does on a computer. Personal data can be stored and transferred, and complicated financial transactions can be handled using this software. Typically, this software is called decentralized application (dApp) [5].

A decentralized application on Ethereum operates automatically by using self-executing contracts which called smart contracts. A smart contract has similar function with conventional contract. It involves two parties to make an agreement about the delivery of goods or services. However, the Unlike conventional contracts, lawyers or intermediaries are not needed. The application creator codes the contract and deploy it on the Ethereum blockchain. Once the conditions of the contract are met, it self-executes and delivers Ether or services to the other party [5].

2.3 E-voting

E-voting provides an option that aims to be more reliable, efficient, and simple. To be reckoned as a secure voting system, six security properties must be present in the system. These properties are explained in Table 3.

Table 3: Security requirement of e-voting

Security Requirement	Description
Fairness [6]	No preliminary result should be disclosed before the voting process ends as it can influence other voters' votes.
Anonymity [7]	The casted votes should not contain any information that may expose the voter identity.
Voter Secrecy [7]	All casted votes are stored confidentially without disclosure to other people including the authority. The choices made by each voter only known by themselves.
Un-reusability [8]	Every voter can only vote once.
Individual Verifiability [7]	The voters' casted votes are correctly counted.
Un-forgiveness [8]	Voters cannot change their vote choice after casted it.

2.4 Follow-My-Vote (FMV)

Follow-My-Vote (FMV) is one of the existing system that is reviewed. FMV is an online voting system that was developed by a non-profit organization. FMV is one of the earliest voting systems implemented with Blockchain technology. It is developed with the intention of making an online voting system that is applicable to a bigger scale election [9].

FMV primarily focuses on the properties of voter eligibility, individual verifiability. Voter eligibility is present during the authentication phase using government-issue ID and device camera. FMV allows voters to find their votes and verify if their votes are present and correct according to their voter unique ID. This is known as individual verifiability [9].

FMV has two requirements for user to be a voter. The first requirement is users need to have a government-issue ID. The second requirement is users required to have a device with a front camera or webcam. These two requirements are used for the register and authentication phase. Each voter also received a pass-phrase after successfully registered [10].

During the voting phase, FMV has a trusted central authority that verifies the voter identity. After voter identity has been verified, only eligible voters are authorized to cast their vote. FMV includes forgiveness in its system design. This means that FMV allows voters to change their casted vote. To change their vote, they are required to use a pass-phrase that was given to them after completion of registration phase. The casted votes are stored in the system blockchain. Voters are able to watch the progress of election in real-time as votes are casted. After voting process phase end, the system begins to tally the vote and publish it afterwards [10].

FMV has a central authority that manages the voters and votes information. Therefore, it is important that the central authority is a trustworthy entity. A trusted authority guaranteed the confidentiality of voters and concealed the connection between the voter real identity with their casted vote. If the central authority is tainted, the votes are at risk of becoming anonymous free. The central authority can be a potential threat that manipulates votes. Therefore, the integrity of the votes may be compromised [10].

Votes secrecy is not guaranteed in FMV. The reason is because votes are casted and stored without any encryption technique applied. This is a problem because the central authority is able to see the casted vote in plaintext. The fairness of the election was also undermined because of two reasons. The first reason is voters have the ability to change their casted vote. The second reason is the progress of the election can be watched by voters in real-time. The latter reason may influence the voter to change their candidate choice after casting their vote [10].

2.5 BitCongress

BitCongress is the second voting system that is reviewed. BitCongress is created by Bitcoin Kinetics. BitCongress is created as an open election voting platform. An open election means that it does not have a central authority that managed the election process. [11].

BitCongress is a voting system that requires three platforms to be functional. These three platforms are Bitcoin, smart contract from Ethereum and Counterparty. Smart contract is a programmatic protocol. It provides verification and compliance of contractual terms and clauses. Counterparty is a financial tools platform for transact business, exchange and participate in advanced financial contracts that are operated by the Bitcoin blockchain [11].

BitCongress provides the security properties of individual anonymity, individual verifiability and non-forgiveness. BitCongress applies the concept of open election. This means that anyone is eligible to become a voter by signing the election smart contract. Thus, it lacks authentication process which provide anonymity for voter. Each voter in the election has one token that represent as ballot. After casting their vote, the token is transacted from the voter block address to the candidate block address. This provide individual verifiability as voter can refer the candidate transaction record to find their address. Lastly, BitCongress only allow one vote per voter in the poll which provide non-forgiveness properties [11].

BitCongress used Bitcoin public blockchain as a voting platform for election. The elections in BitCongress is created using smart contracts. The election has a set of rules, duration, candidate and an URL accessible by public. Each election smart contract has a multi-signature contract held between voters and candidates. To register as a voter, the user need to sign the smart contract. Noted that authentication process is not required because of the open election concept. Upon entering the election, voter is given a token generated from Counterparty. The token represents a ballot. Each voter and candidate in the election has their own block address [11].

In voting phase, voter may cast their vote by transact the token to their desire candidate. Upon transacting the token, the transaction record is insert into both voter and candidate block address. This means that voter and candidate can view their transaction history to ensure the transaction is completed. The voter token is deducted after casting their vote which render them from voting again. Tallying processes are maintained by every voter in the network to conclude the election result [11].

Although BitCongress is more secure than FMV, it is not perfect. The most obvious flaw for BitCongress is, it has 3 dependencies that need to operate properly for the entire system to be functional.

If one of the dependencies failed, the whole system may crash and be unusable. Another issue is it does not provide any authentication since it used the concept of open election. [11].

2.6 Comparison between existing systems and proposed system

The comparison is based on different requirements of a voting system that needed to be effective and trusted. Table 4 shows the comparison between existing systems and proposed system.

Table 4: Comparison between existing systems and proposed system

Security Requirement	Follow-My-Vote	BitCongress	Proposed System
Fairness	✗	✗	✓
Anonymity	✓	✓	✓
Voter Secrecy	✗	✓	✓
Un-reusability	✗	✓	✓
Individual Verifiability	✓	✓	✓
Un-forgiveness	✗	✓	✓

Based on Table 4, the three systems have differences in four out of six requirements. These four requirements are fairness, un-reusability, voter secrecy and un-forgiveness.

For fairness, both FMV and BitCongress do not have these properties while the proposed system did. FMV design allows the voter to watch the election result in real-time which compromises the fairness of the election. For the case of BitCongress, it uses the concept of open election where they can see the vote casted at any time. In the proposed system, the result of the election is shown after the voting and tallying process ends. This provides fairness for the election.

For un-reusability, both BitCongress and the proposed system fulfilled the security properties. However, FMV did not. FMV allows its voters to changes their casted vote by using a passphrase. Therefore, the voters are able to cast their vote multiple times.

For voter secrecy, only FMV does not fulfill this security property as it has a central authority that manages the votes and casted vote. FMV does not apply encryption techniques when storing voters' votes. This allows the central authority to see the votes in a plaintext form.

Lastly, out of these three voting systems, only FMV does not has the property of Un-forgiveness which allows their voters to change their casted vote after they casted the vote. BitCongress and the proposed system have this property.

2.7 Penetration Testing

Penetration Testing (Pentest) is a method used for assessing the state of security and reducing security threats of an information system. It is a controlled attempt to break into a system or network in order to find security flaws. A Pentest result can deliver an insight on how secure is the information system and vulnerabilities that exist in it [12].

In this project, Pentest-Tools.com and SSL Trust are the two tools that are used to run pentest for the proposed system. Pentest-Tools.com provides website vulnerability scanning for web application header content. For example, Pentest-Tools.com scan the website to identify the implementation of secure response HTTP header of the website. This include Strict-Transport-Security, Content-Security-Policy, X-Frame-Options, X-XSS-Protection, X-Content-Type-Options, and Referrer-Policy. By entering the link of a website, Pentest-Tools.com scans the website and generates a Report that can be download as PDF file. The PDF file contained the website vulnerabilities information and recommendations on how to fix it [13]. Besides, SSL Trust provides two types of scans for a website. The first type of scan is for malware and virus detection. The second type of scan is for vulnerabilities examination. After the scanning process is completed, SSL Trust generates a PDF report that can be download [14].

3. Methodology

The methodology used in this project is Object-Oriented and Analysis Design (OOAD). OOAD is a software engineering approach that is used to develop a system as a group of interacting objects. Each object represents an entity that is characterized by its class, data elements and its behavior [15]. There are four phases in object-oriented methodology. These include object-oriented requirement analysis, object-oriented analysis, object-oriented design, object-oriented implementation and testing and object-oriented maintenance.

In object-oriented requirement analysis phase, the requirements for the proposed system are analyzed from academic papers. These academic papers are selected from IEEE explorer, ScienceDirect, ResearchGate and Google Scholar. The selected academic papers are related to e-voting, PKBDF2 and blockchain. Based on the finding, the security requirements for e-voting are are fairness, secrecy, anonymity, un-reusability, verifiability and non-forgiveness.

Follow-My-Vote (FMV) and BitCongress are two existing systems that are reviewed and taken as references for the development of the proposed system. BitCongress and FMV are online voting systems that utilize blockchain [25].

In object-oriented analysis phase, all collected information are analyzed. From the analysis of the collected information, the proposed system has eight modules. These modules are login, register, password reset, poll creation, poll list, voting, poll tabulation and password policy.

After determined the modules, functional requirements and non-functional requirements are examined. Functional requirements are related to the system modules. It provides the description of functionality for each modules. For non-functional requirements, the performance, security and operation of the system are determined.

Next, the roles exist in the system are identified. The roles are poll creator, candidate and voter. Poll creator is the one who create and configure a new poll in the system. In each poll, there are candidates who are nominated. Finally, voters are the one who responsible to cast their vote which determine the winner for a poll.

In object-oriented design, the test plan and the architecture of the proposed system is designed. The test plan is consisted of list of system functionalities which the proposed system should perform correctly. The architecture design includes the database design, class design and user interface design. These three designs are used as a guide during the implementation process

There are seven database tables designed for the proposed system. These tables are USERS table, VOTERS table, CANDIDATES table, TOKENS table, POLL_LIST table, CASTED_VOTE table and CMV_VOTE table. Noted that, CMV_VOTE represent the information saved in the Ethereum blockchain.

For class designs, there 19 classes which include HttpServlet, LoginServlet, UserServlet, PollServlet, User, CastedVote, Voter, Poll, Candidate, Token, Email, PBKDF2, UserDB, CastedVoteDB, VoterDB, PollDB, CandidateDB, TokenDB and DBDriver Each of these classes may have encapsulation methods or operational methods.

Lastly, there are a total of 17 modules for user interface (UI) design for the proposed system. These modules are Login, Register, Password Reset, Poll creation, Poll Result, Active Poll List, Active

Participated Poll, Active Created Poll, Poll Creation History, Poll Participation History, Voting Phase for voter, Pre-voting Phase for poll creator, Voting Phase for poll creator, Voting Phase for candidate, Poll Result for voter, Poll Result for poll creator, and Poll Result for candidate.

In object-oriented implementation phase, the database tables, classes and user interface (UI) are implemented. All the databases tables, user interface and classes are linked together to ensure the system function correctly. For example, the Poll class, PollDB class, POLL_LIST table and Poll List UI module are linked together. This allows poll information that stored inside POLL_LIST table to be retrieved using operational method of PollDB class. The retrieved information then passes on from PollDB class to Poll class. Finally, the poll information is display to Poll List UI module using encapsulation method of Poll class.

In the object-oriented testing phase, the developed system is tested using the test plan and pentest tools. The test plan is used to test if the proposed system is functioning as it intended to. The test plan consists of two type of tests. The first test is system functionality and the second is security test. If any errors happen, the debugging process is done to solve the errors. Finally, the pentest process is done on the system using two tools. These tools are Pentest-Tools.com and SSL Trust. The pentest result provides an insight of system security level and vulnerabilities.

In the object-oriented maintenance phase, the maintenance of the system is done by running a test after the system implementation is completed. Documentations are written to identify the improvements needed to be done for future work. Error and bugs for the system are also recorded after running the tests for the system. Solutions for any error and bugs is provided afterward. Due to time limitation, this phase will not be implemented for this project.

4. Analysis and Design

This section explained about the system analysis and design that have been conducted for this project.

4.1 General System Architecture

General system architecture defines the structure and behavior of a system. It includes the process and flow of the information of the system. The general system architecture for the proposed system is illustrated in Figure 3.

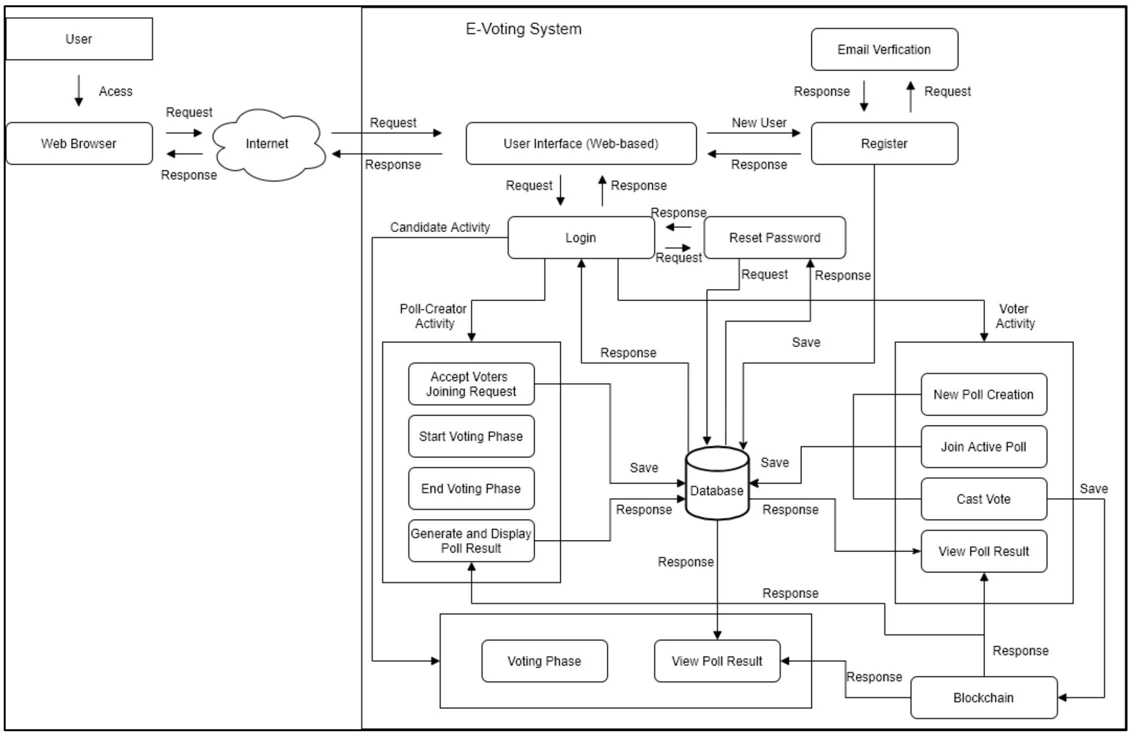


Figure 3: General system architecture for proposed system

As shown in Figure 3, the general system architecture for the proposed system is a web-based system. The user need to use a web browser to access the system. When accessing the system, user may login into their account. User can also reset their password in case they forgot it. For a new user, they may register a new account which required email verification. After successfully registered, the new user information is stored inside the database.

After login, there are three type of activities can be done by user. These activities are poll creator activity, voter activity and candidate activity. For voter activity, the voter may create a new poll which allow them become the poll creator the new poll. Next, voter may also request to join an active poll. When voter requests to join a poll, their request is saved to the database to indicate that they had request to join the poll. In voting phase of a poll, voter can cast their vote to a desired candidate. When voter casted their vote, the casted vote is saved into the database and blockchain. Lastly, voter may view the result of a poll.

For poll creator activity, poll creator may accept voter request for joining the poll they created. When poll creator approved the voter request, the voter is saved into the database to indicate that they had joined the poll. Next, poll creator may start and end the voting phase. When the voting phase end, the poll creator may view the result of a poll.

For candidates' activity, they can view the other list of candidate in the poll during the voting phase. After the voting phase ended, candidate can view the poll result.

4.2 Functional Requirements

The proposed system has seven functional requirements as shown in Table 5. There is register module, login module, password reset module, poll creation module, poll list module, voting module and poll tabulation module.

Table 5: Functional requirement of proposed system

Requirement Modules	Functionality
Register	• New users need to register their information to the system. • Information collected are account ID, name, date of birth, email and password. • Account ID and email must be unique. • Email address needs to be verify. • Passwords created must follow the password policy.
Login	• Users can log in into the system by using their account ID and password.
Password Reset	• Users can reset their password by entering their registered email address. • Upon entering their email address, an email containing a link for the password reset page is sent to their registered email address.
Poll Creation	• Every user can create their own poll with its settings. • The poll name, candidates, poll description and voters' capacity are determined during the configuration process. • The creator of the poll cannot vote. • Poll creator may approve voter request to join their poll.
Poll List	• Users can request to join an active poll; they can only enter if they are approved by poll-creator. • Users are not able to join if the voter's capacity is full.
Voting	• The poll creator has the ability to start and end the voting phase of the created poll. • Every voter can vote as long as the voting phase is still ongoing. • Each voter can only vote once.
Poll Tabulation	• Result of the poll is published in table form to the poll creator, candidates and voters.
Password Policy	• Password complexity include at least one number, one alphabet (uppercase and lowercase), one special character and minimum 10 characters long. • Password is hashed with PBKDF2.

4.3 Non-functional Requirements

Non-functional requirements are divided into three categories which are operational, performance and security. Table 6 shows the non-functional requirements for the proposed system.

Table 6: Non-functional requirement of proposed system

Requirement	Description
Operational	• The system only available when there is internet connection.
Performance	• System must locate to the correct session depend on who authorized.
Security	• Users can access the system with correct account ID and password only. • The password is hashed using PBKDF2 algorithm. • Password must consist of number, alphabet (lowercase and uppercase), special character and at least 10 characters long. • Email that contain link for password reset is valid for 5 minutes. • User is not able to reset password using expired link.

4.4 Entity Relationship Diagram

Entity Relationship Diagram (ERD) describes the relationship of tables exist in the database. Figure 4 shows the ERD for proposed system

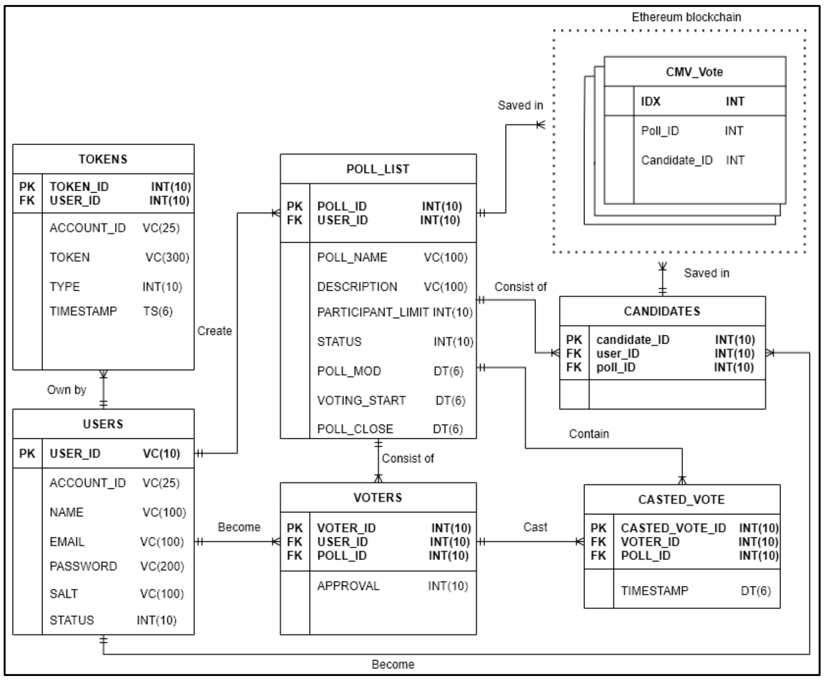


Figure 4: ERD for the proposed system

As shown in Figure 4, the ERD of the proposed system has seven tables. These include USERS, VOTERS, POLL_LIST, CANDIDATES, TOKENS, CASTED_VOTE and CMV_VOTE. Each of the entity may contain primary or foreign key. The CMV _ VOTE entity is stored in the Ethereum blockchain.

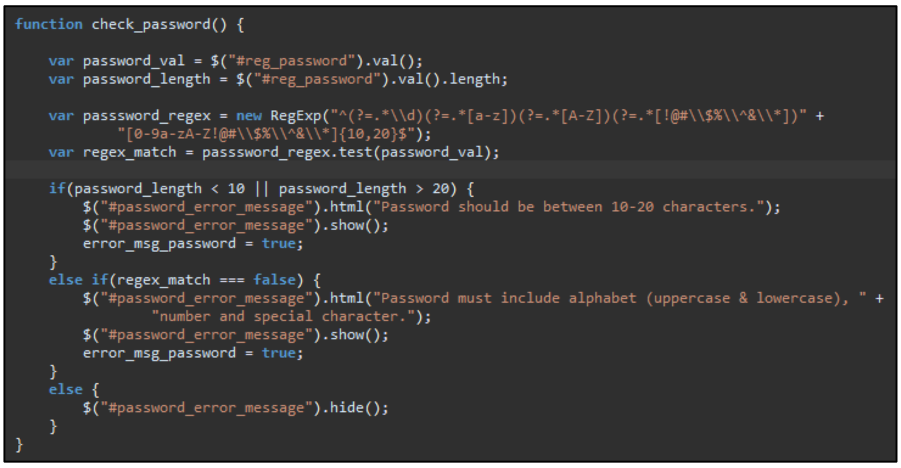
5. Implementation

This section examines the implementation and testing result of the proposed system. These include implementation of system security properties and implementation of e-voting security properties.

5.1 Implementation of System Security Properties

This section discusses the implementation of system security properties. These include strong password requirement, session timeout, PBKDF2 and Content-Security-Policy.

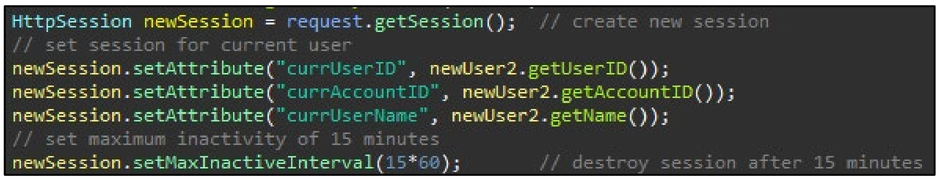
Figure 5 shows the source code for strong password policy. The code is written in JavaScript programming language. The source code explained that the password should be at least 20 characters long. The password pattern should also include at least one number, alphabet (lowercase and uppercase) and a special character. This password requirement is applied to user registration process and password reset process.



```
function check_password() {

    var password_val = $("#reg_password").val();
    var password_length = $("#reg_password").val().length;

    var passsword_regex = new RegExp("^(?=.*\\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#\\$%\\^&\\*])" +
            "[0-9a-zA-Z!@#\\$%\\^&\\*]{10,20}$");
    var regex_match = passsword_regex.test(password_val);

    if(password_length < 10 || password_length > 20) {
        $("#password_error_message").html("Password should be between 10-20 characters.");
        $("#password_error_message").show();
        error_msg_password = true;
    }
    else if(regex_match === false) {
        $("#password_error_message").html("Password must include alphabet (uppercase & lowercase), " +
                "number and special character.");
        $("#password_error_message").show();
        error_msg_password = true;
    }
    else {
        $("#password_error_message").hide();
    }
}
```

Figure 5: JavaScript source code for Strong Password checking

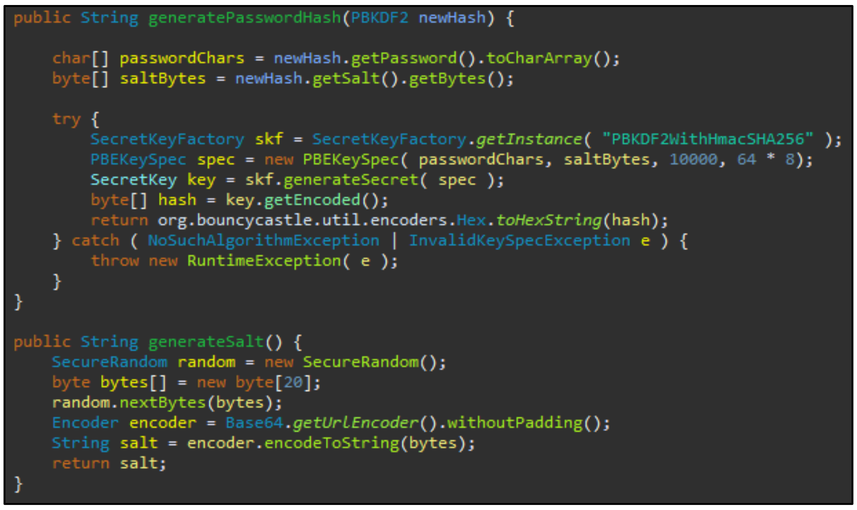
Figure 6 shows the source code for session timeout. The code is written in Java programming language. The source code indicates if user is inactive for 15 minutes, their session will be destroyed and the user is required to re-login in order to continue using the system.



```
HttpSession newSession = request.getSession();  // create new session
// set session for current user
newSession.setAttribute("currUserID", newUser2.getUserID());
newSession.setAttribute("currAccountID", newUser2.getAccountID());
newSession.setAttribute("currUserName", newUser2.getName());
// set maximum inactivity of 15 minutes
newSession.setMaxInactiveInterval(15*60);        // destroy session after 15 minutes
```

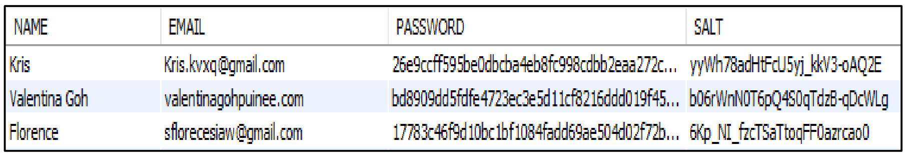
Figure 6: Java source code for setting session idle duration

Figure 7 shows the source code for PBKDF2 hashing on user account's password. The code is written in Java programming language. Besides that, salt is also added during the hashing process. Next, Figure 8 shows an example of users' hashed password and salt stored in the database.



```
public String generatePasswordHash(PBKDF2 newHash) {

    char[] passwordChars = newHash.getPassword().toCharArray();
    byte[] saltBytes = newHash.getSalt().getBytes();

    try {
        SecretKeyFactory skf = SecretKeyFactory.getInstance( "PBKDF2WithHmacSHA256" );
        PBEKeySpec spec = new PBEKeySpec( passwordChars, saltBytes, 10000, 64 * 8);
        SecretKey key = skf.generateSecret( spec );
        byte[] hash = key.getEncoded();
        return org.bouncycastle.util.encoders.Hex.toHexString(hash);
    } catch ( NoSuchAlgorithmException | InvalidKeySpecException e ) {
        throw new RuntimeException( e );
    }
}

public String generateSalt() {
    SecureRandom random = new SecureRandom();
    byte bytes[] = new byte[20];
    random.nextBytes(bytes);
    Encoder encoder = Base64.getUrlEncoder().withoutPadding();
    String salt = encoder.encodeToString(bytes);
    return salt;
}
```

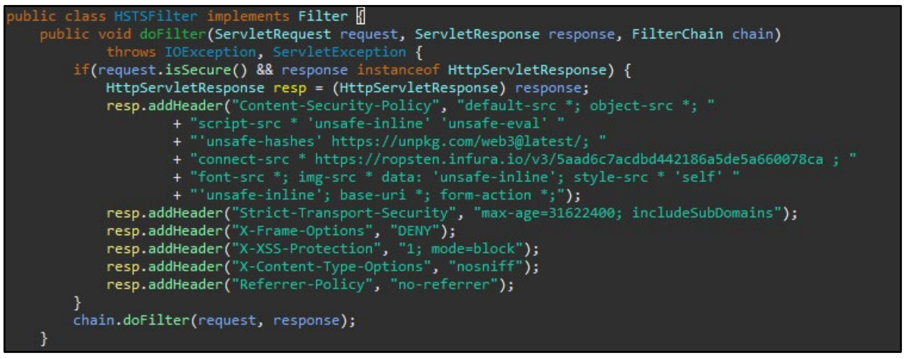
Figure 7: Java source code for password hashing using PBKDF2



NAME	EMAIL	PASSWORD	SALT
Kris	Kris.kvxq@gmail.com	26e9ccff595be0dbcba4eb8fc998cdbb2eaa272c...	yyWh78adHtFcU5yj_kkV3-oAQ2E
Valentina Goh	valentinagohpuinee.com	bd8909dd5fdfe4723ec3e5d11cf8216ddd019f45...	b06rWnN0T6pQ4S0qTdzB-qDcWLg
Florence	sflorecesiaw@gmail.com	17783c46f9d10bc1bf1084fadd69ae504d02f72b...	6Kp_NI_fzcTSaTtoqFF0azrcao0

Figure 8: User accounts' hashed password and salt stored in database

Figure 9 shows the source code for content-security-policy. The code is written in Java programming language. Content-Security-Policy is also known as secure HTTP response header. The purpose of Content-Security-Policy is to improve the security of the web page by restricting the browser to load data from unknown source.



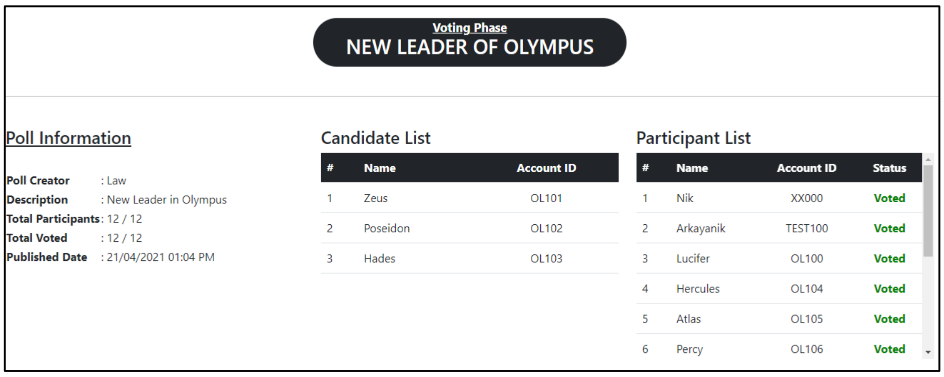
```java
public class HSTSFilter implements Filter {
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
            throws IOException, ServletException {
        if(request.isSecure() && response instanceof HttpServletResponse) {
            HttpServletResponse resp = (HttpServletResponse) response;
            resp.addHeader("Content-Security-Policy", "default-src *; object-src *; "
                    + "script-src * 'unsafe-inline' 'unsafe-eval' "
                    + "'unsafe-hashes' https://unpkg.com/web3@latest/; "
                    + "connect-src * https://ropsten.infura.io/v3/5aad6c7acdbd442186a5de5a660078ca ; "
                    + "font-src *; img-src * data: 'unsafe-inline'; style-src * 'self' "
                    + "'unsafe-inline'; base-uri *; form-action *;");
            resp.addHeader("Strict-Transport-Security", "max-age=31622400; includeSubDomains");
            resp.addHeader("X-Frame-Options", "DENY");
            resp.addHeader("X-XSS-Protection", "1; mode=block");
            resp.addHeader("X-Content-Type-Options", "nosniff");
            resp.addHeader("Referrer-Policy", "no-referrer");
        }
        chain.doFilter(request, response);
    }
```

Figure 9: Java source code for implementation for content-security-policy

5.2 Implementation of E-voting Security Properties

This section discusses the implementation of E-voting security properties. These include fairness, anonymity, voter secrecy, un-reusability, individual verifiability and un-forgiveness.

The fairness property indicates that no preliminary result should be disclosed before the voting process ends. In the propose system, the poll creator, voters and candidates are not able to see the current vote count during the voting phase. Figure 10, Figure 11 and Figure 12 shows the user interface of the proposed system during voting phase from the perspective of poll creator, voter and candidate respectively.



Voting Phase
NEW LEADER OF OLYMPUS

Poll Information

Poll Creator	: Law
Description	: New Leader in Olympus
Total Participants	: 12 / 12
Total Voted	: 12 / 12
Published Date	: 21/04/2021 01:04 PM

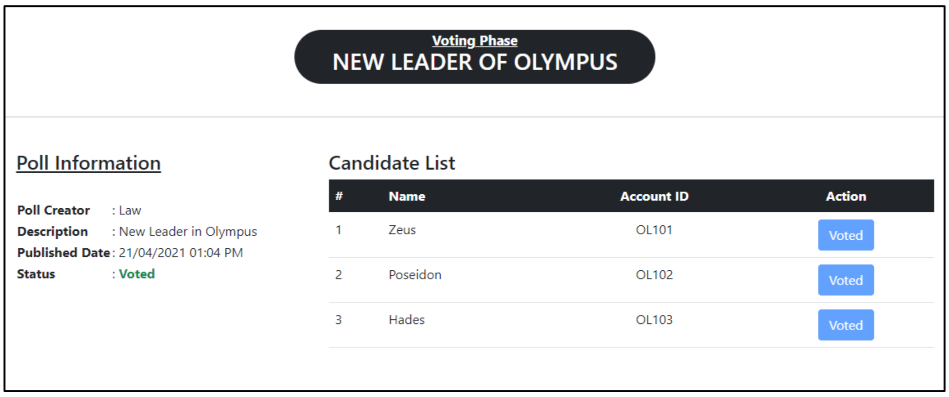
Candidate List

#	Name	Account ID
1	Zeus	OL101
2	Poseidon	OL102
3	Hades	OL103

Participant List

#	Name	Account ID	Status
1	Nik	XX000	Voted
2	Arkayanik	TEST100	Voted
3	Lucifer	OL100	Voted
4	Hercules	OL104	Voted
5	Atlas	OL105	Voted
6	Percy	OL106	Voted

Figure 10: User interface for perspective of poll creator during voting phase



Voting Phase
NEW LEADER OF OLYMPUS

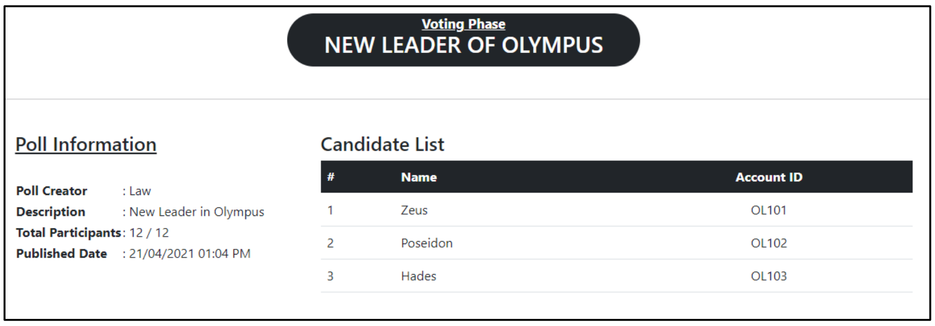
Poll Information

Poll Creator	: Law
Description	: New Leader in Olympus
Published Date	: 21/04/2021 01:04 PM
Status	: Voted

Candidate List

#	Name	Account ID	Action
1	Zeus	OL101	Voted
2	Poseidon	OL102	Voted
3	Hades	OL103	Voted

Figure 11: User interface for perspective of voter during voting phase



Poll Information

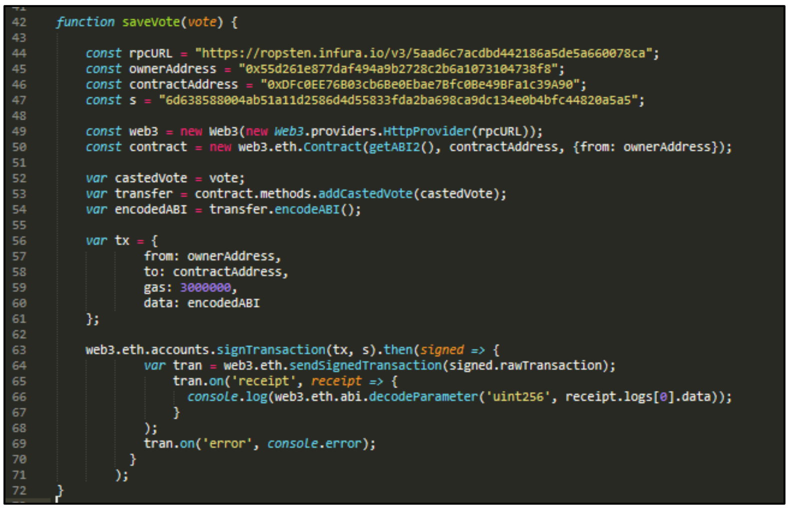
Poll Creator	: Law
Description	: New Leader in Olympus
Total Participants	: 12 / 12
Published Date	: 21/04/2021 01:04 PM

Candidate List

#	Name	Account ID
1	Zeus	OL101
2	Poseidon	OL102
3	Hades	OL103

Figure 12: User interface for perspective of candidates during voting phase

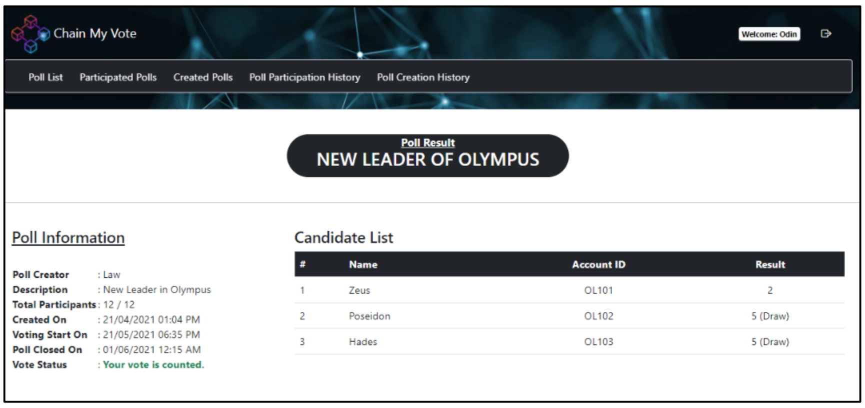
The anonymity property describes that the casted votes should not contain any information that may expose the voter identity. In the propose system, the casted votes are stored inside the Ethereum blockchain. The information send to the Ethereum blockchain is candidate ID and poll ID. There is no information related to the voter stored inside the blockchain. Figure 13 shows the JavaScript source code used to submit the casted vote information to the Ethereum blockchain.



```javascript
function saveVote(vote) {

    const rpcURL = "https://ropsten.infura.io/v3/5aad6c7acdbd442186a5de5a660078ca";
    const ownerAddress = "0x55d261e877daf494a9b2728c2b6a1073104738f8";
    const contractAddress = "0xDFc0EE76B03cb6Be0Ebae7Bfc0Be49BFa1c39A90";
    const s = "6d638588004ab51a11d2586d4d55833fda2ba698ca9dc134e0b4bfc44820a5a5";

    const web3 = new Web3(new Web3.providers.HttpProvider(rpcURL));
    const contract = new web3.eth.Contract(getABI2(), contractAddress, {from: ownerAddress});

    var castedVote = vote;
    var transfer = contract.methods.addCastedVote(castedVote);
    var encodedABI = transfer.encodeABI();

    var tx = {
            from: ownerAddress,
            to: contractAddress,
            gas: 3000000,
            data: encodedABI
    };

    web3.eth.accounts.signTransaction(tx, s).then(signed => {
            var tran = web3.eth.sendSignedTransaction(signed.rawTransaction);
                tran.on('receipt', receipt => {
                  console.log(web3.eth.abi.decodeParameter('uint256', receipt.logs[0].data));
                }
            );
            tran.on('error', console.error);
          }
        );
}
```

Figure 13: JavaScript source code to save voters' casted vote to Ethereum blockchain

The voter secrecy property explained that all casted votes are stored confidentially without disclosure to other people. In the proposed system, the vote casted by each voter is only known by themselves. Figure 14, Figure 15 and Figure 16 illustrates the poll result user interface of voter, candidate and poll creator respectively. These figure shows that the system does not disclose who the other voter voted.



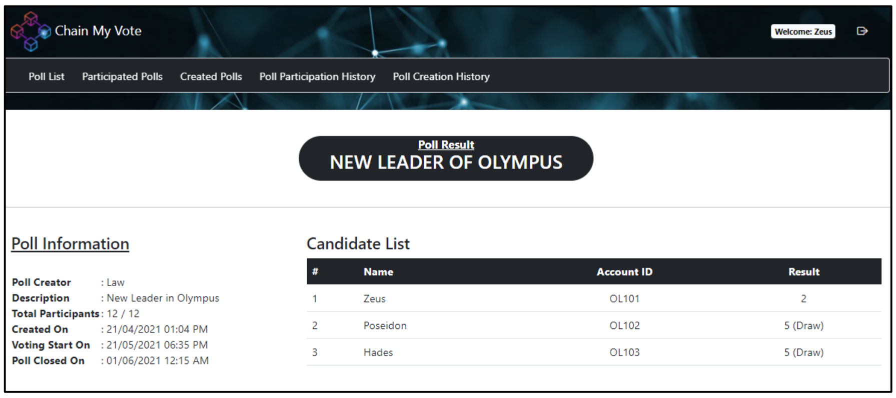
Poll Information

Poll Creator	: Law
Description	: New Leader in Olympus
Total Participants	: 12 / 12
Created On	: 21/04/2021 01:04 PM
Voting Start On	: 21/05/2021 06:35 PM
Poll Closed On	: 01/06/2021 12:15 AM
Vote Status	: Your vote is counted.

Candidate List

#	Name	Account ID	Result
1	Zeus	OL101	2
2	Poseidon	OL102	5 (Draw)
3	Hades	OL103	5 (Draw)

Figure 14: Poll result webpage for voter



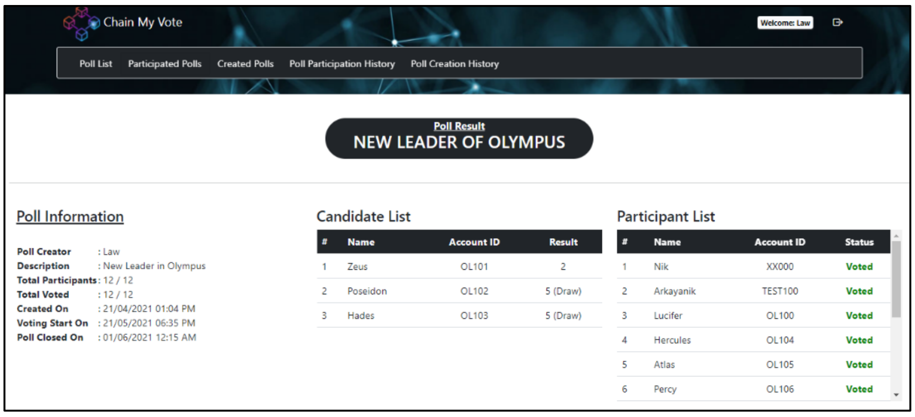
Poll Information

Poll Creator	: Law
Description	: New Leader in Olympus
Total Participants	: 12 / 12
Created On	: 21/04/2021 01:04 PM
Voting Start On	: 21/05/2021 06:35 PM
Poll Closed On	: 01/06/2021 12:15 AM

Candidate List

#	Name	Account ID	Result
1	Zeus	OL101	2
2	Poseidon	OL102	5 (Draw)
3	Hades	OL103	5 (Draw)

Figure 15: Poll result webpage for candidate



Poll Information

Poll Creator	: Law
Description	: New Leader in Olympus
Total Participants	: 12 / 12
Total Voted	: 12 / 12
Created On	: 21/04/2021 01:04 PM
Voting Start On	: 21/05/2021 06:35 PM
Poll Closed On	: 01/06/2021 12:15 AM

Candidate List

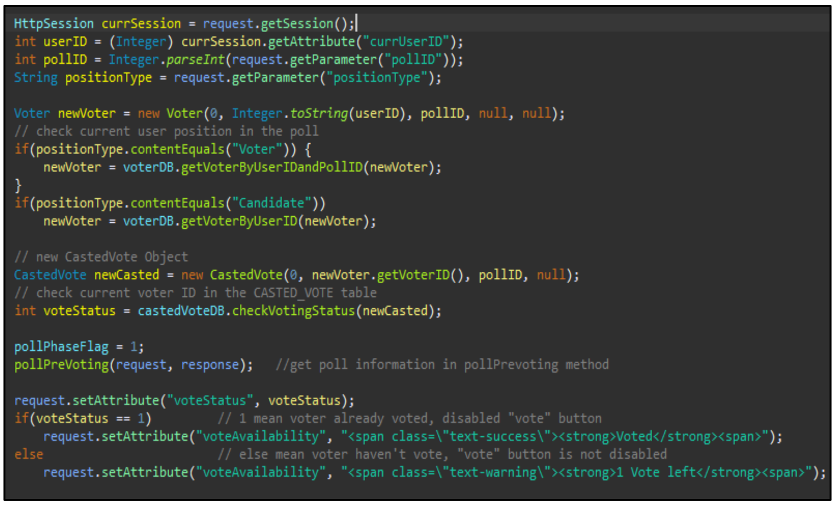
#	Name	Account ID	Result
1	Zeus	OL101	2
2	Poseidon	OL102	5 (Draw)
3	Hades	OL103	5 (Draw)

Participant List

#	Name	Account ID	Status
1	Nik	XX000	Voted
2	Arkayanik	TEST100	Voted
3	Lucifer	OL100	Voted
4	Hercules	OL104	Voted
5	Atlas	OL105	Voted
6	Percy	OL106	Voted

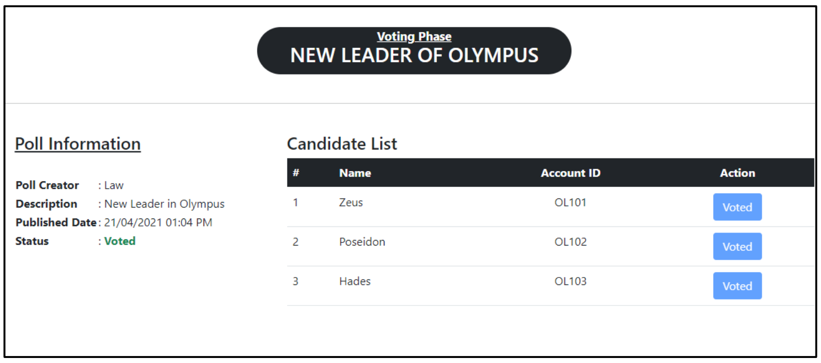
Figure 16: Poll result webpage for poll creator

The un-reusability property indicates that every voter can only vote once. In the proposed system, once a voter had casted their vote. Their voter ID are recorded in the database's casted_vote table. The next time the same voter revisits the voting phase webpage, the vote button is disabled. This prevent the voter to cast their more than once. Figure 17 shows the source code on how the system check whether the user has voted or not. Besides that, Figure 18 illustrates the voting webpage for voter that already casted their vote and their "vote" button is disabled.



```java
HttpSession currSession = request.getSession();
int userID = (Integer) currSession.getAttribute("currUserID");
int pollID = Integer.parseInt(request.getParameter("pollID"));
String positionType = request.getParameter("positionType");

Voter newVoter = new Voter(0, Integer.toString(userID), pollID, null, null);
// check current user position in the poll
if(positionType.contentEquals("Voter")) {
    newVoter = voterDB.getVoterByUserIDandPollID(newVoter);
}
if(positionType.contentEquals("Candidate"))
    newVoter = voterDB.getVoterByUserID(newVoter);

// new CastedVote Object
CastedVote newCasted = new CastedVote(0, newVoter.getVoterID(), pollID, null);
// check current voter ID in the CASTED_VOTE table
int voteStatus = castedVoteDB.checkVotingStatus(newCasted);

pollPhaseFlag = 1;
pollPreVoting(request, response);   //get poll information in pollPrevoting method

request.setAttribute("voteStatus", voteStatus);
if(voteStatus == 1)          // 1 mean voter already voted, disabled "vote" button
    request.setAttribute("voteAvailability", "<span class=\"text-success\"><strong>Voted</strong><span>");
else                         // else mean voter haven't vote, "vote" button is not disabled
    request.setAttribute("voteAvailability", "<span class=\"text-warning\"><strong>1 Vote left</strong><span>");
```

Figure 17: Java source code on how system decide to enabled or disabled "Vote" button



Voting Phase

NEW LEADER OF OLYMPUS

Poll Information

Poll Creator : Law
Description : New Leader in Olympus
Published Date : 21/04/2021 01:04 PM
Status : Voted

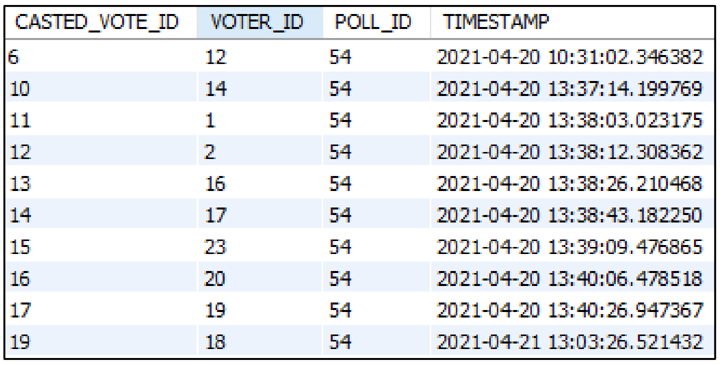
Candidate List

#	Name	Account ID	Action
1	Zeus	OL101	Voted
2	Poseidon	OL102	Voted
3	Hades	OL103	Voted

Figure 18: Voting webpage for voter that already casted their vote; the button is disabled

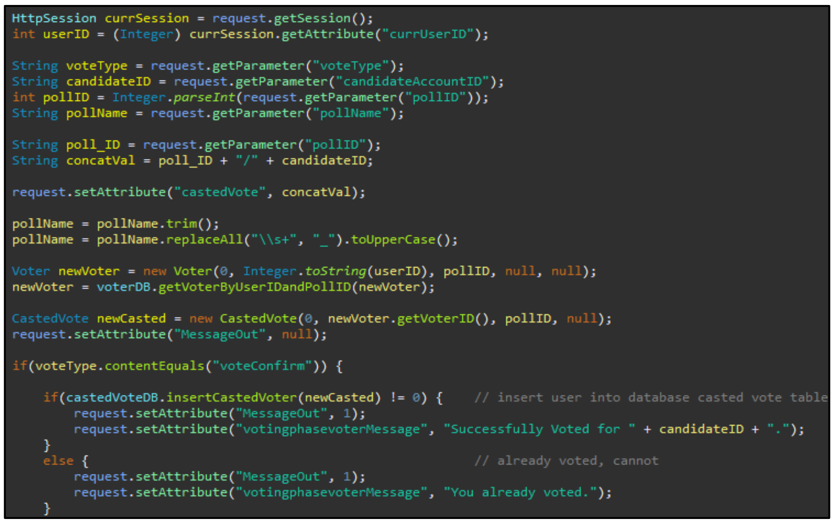
The un-forgiveness property indicates that voters cannot change their vote choice after they casted it. As shown in Figure 18 above, after voters casted their votes, the “vote” button is disabled immediately. There is no feature that allows the voter to change their vote choices afterward.

The individual verifiability property indicates that the voters can ensure their votes are counted. In the proposed system, after the voter casted their votes. Their information is stored in two different mediums. These medium are the Ethereum blockchain and the system database. In order to successfully count the voter’s casted vote, the system need to successfully insert the voter ID into the database table CASTED_VOTE first. Only then, the casted vote is saved to the Ethereum blockchain which indicates the casted vote is counted. Figure 19 shows the CASTED_VOTE table of the system database. Next, Figure 20 shows the Java source code for vote casting process.



CASTED_VOTE_ID	VOTER_ID	POLL_ID	TIMESTAMP
6	12	54	2021-04-20 10:31:02.346382
10	14	54	2021-04-20 13:37:14.199769
11	1	54	2021-04-20 13:38:03.023175
12	2	54	2021-04-20 13:38:12.308362
13	16	54	2021-04-20 13:38:26.210468
14	17	54	2021-04-20 13:38:43.182250
15	23	54	2021-04-20 13:39:09.476865
16	20	54	2021-04-20 13:40:06.478518
17	19	54	2021-04-20 13:40:26.947367
19	18	54	2021-04-21 13:03:26.521432

Figure 19: Database table CASTED_VOTE



```java
HttpSession currSession = request.getSession();
int userID = (Integer) currSession.getAttribute("currUserID");

String voteType = request.getParameter("voteType");
String candidateID = request.getParameter("candidateAccountID");
int pollID = Integer.parseInt(request.getParameter("pollID"));
String pollName = request.getParameter("pollName");

String poll_ID = request.getParameter("pollID");
String concatVal = poll_ID + "/" + candidateID;

request.setAttribute("castedVote", concatVal);

pollName = pollName.trim();
pollName = pollName.replaceAll("\\s+", "_").toUpperCase();

Voter newVoter = new Voter(0, Integer.toString(userID), pollID, null, null);
newVoter = voterDB.getVoterByUserIDandPollID(newVoter);

CastedVote newCasted = new CastedVote(0, newVoter.getVoterID(), pollID, null);
request.setAttribute("MessageOut", null);

if(voteType.contentEquals("voteConfirm")) {

    if(castedVoteDB.insertCastedVoter(newCasted) != 0) {    // insert user into database casted vote table
        request.setAttribute("MessageOut", 1);
        request.setAttribute("votingphasevoterMessage", "Successfully Voted for " + candidateID + ".");
    }
    else {                                                  // already voted, cannot
        request.setAttribute("MessageOut", 1);
        request.setAttribute("votingphasevoterMessage", "You already voted.");
    }
```

Figure 20: Java source code for voter’s vote casting process

6. Result and Discussion

This section presents the result and discussion of the proposed system. These include security test plan result, pentest result and user acceptance result.

6.1 Test Plan Result

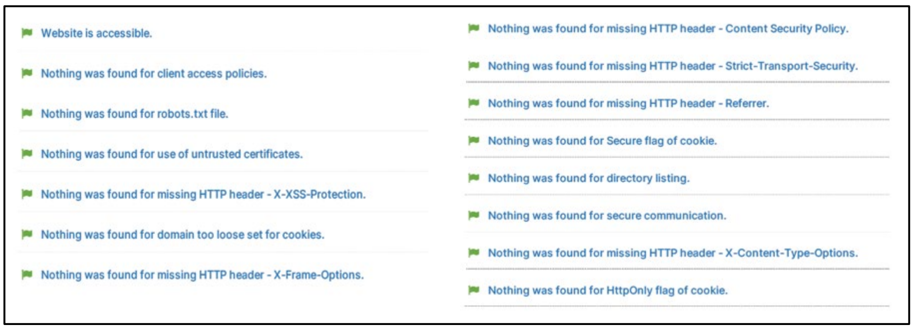
Table 7 shows the result of security test plan for the proposed system.

Table 7: Security Test plan result

No	Requirement	Result
1	When user fails to login, the error message does not indicate which part of the credential data is incorrect	Pass
2	Enforce strong password policy	Pass
3	Password is obscured in the textbox	Pass
4	Ensure user is not allowed to reset password using expired button or button that already been used.	Pass
5	Session is destroyed after 15 minutes of user inactivity.	Pass
6	Minimum value/length and maximum value/length in input field is specified.	Pass
7	To reset password in user profile page, user must input the correct old password; the new password must satisfied the strong password requirement.	Pass
8	Password in the database is hashed with PBKDF2 algorithm	Pass
9	Salt is added to the password before hashed with PBKDF2 algorithm.	Pass

6.2 Pentest Result

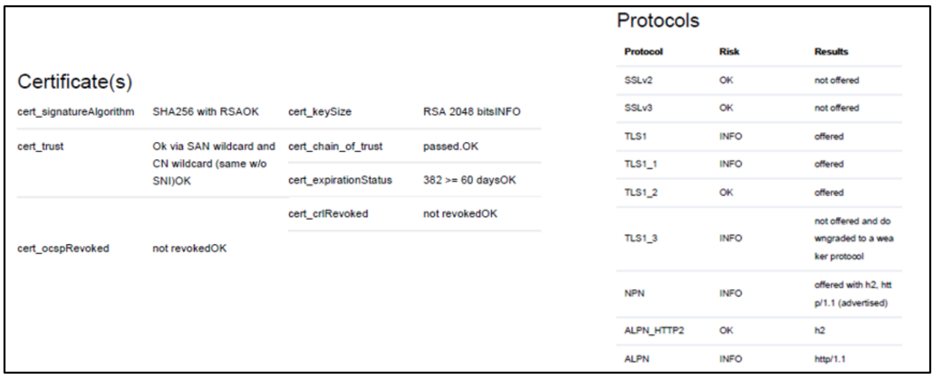
There are two pentest tools namely Pentest-Tools.com and SSL Trust are used to scan the developed system. Based on the Pentest-Tools, the developed system has secure response HTTP header include Strict-Transport-Security, Content-Security-Policy, Referrer-Policy, X-Frame-Options, X-XSS-Protection and X-Content-Type-Options. The Strict-Transport-Security header instructs the browser to only open secure HTTPS connections version of the developed system to the web server and deny unencrypted HTTP connection attempts. The Content-Security-Policy header triggers a protection mechanism implemented in web browsers. This is done by instructing the browser to only connects and loads the specific webpage's resources based on the policy to prevent exploitation of Cross-Site Scripting (XSS) vulnerabilities. The Referrer-Policy header prevents user tracking and inadvertent information leakage. The X-Frame-Options header prevent attackers from embedding this website into an iframe of a third party website. The X-XSS- Protection header instructs the browser to immediately stop loading the web page when it detects the web page is embedded with Cross-Site Scripting (XSS) attack. Lastly, The X-Content-Type-Options header prevents the browser from reinterpreting the content of web page which may override the value of the Content-Type header. Figure 21 shows the Pentest-tools.com result.



- Website is accessible.
- Nothing was found for client access policies.
- Nothing was found for robots.txt file.
- Nothing was found for use of untrusted certificates.
- Nothing was found for missing HTTP header - X-XSS-Protection.
- Nothing was found for domain too loose set for cookies.
- Nothing was found for missing HTTP header - X-Frame-Options.
- Nothing was found for missing HTTP header - Content Security Policy.
- Nothing was found for missing HTTP header - Strict-Transport-Security.
- Nothing was found for missing HTTP header - Referrer.
- Nothing was found for Secure flag of cookie.
- Nothing was found for directory listing.
- Nothing was found for secure communication.
- Nothing was found for missing HTTP header - X-Content-Type-Options.
- Nothing was found for HttpOnly flag of cookie.

Figure 21: Pentest-Tools.com Result

For SSL Trust, there are two types of scan completed. The first scan is for malware and virus detection. The second scan is for vulnerability examination. For the malware and virus section, the result shows that the developed system did not infected by any malware and virus. SSL Trust report indicates that the developed system has a secure connection through TLS protocol with a valid certificate. For the vulnerability report, the developed system is secure from attacks such as Heartbleed, CCS, ROBOT, DROWN, and LOGJAM. However, the developed system has a low protection and against attacks such as BREACH, BEAST_CBC_TLS1, BEAST, and LUCKY13. Figure 22 shows the SSL Trust Certificate Check and Protocol Check Result.



Certificate(s)

cert_signatureAlgorithm	SHA256 with RSAOK	cert_keySize	RSA 2048 bitsINFO
cert_trust	Ok via SAN wildcard and CN wildcard (same w/o SNI)OK	cert_chain_of_trust	passed.OK
		cert_expirationStatus	382 >= 60 daysOK
		cert_crlRevoked	not revokedOK
cert_ocspRevoked	not revokedOK		

Protocols

Protocol	Risk	Results
SSLv2	OK	not offered
SSLv3	OK	not offered
TLS1	INFO	offered
TLS1_1	INFO	offered
TLS1_2	OK	offered
TLS1_3	INFO	not offered and do wngraded to a wea ker protocol
NPN	INFO	offered with h2, htt p/1.1 (advertised)
ALPN_HTTP2	OK	h2
ALPN	INFO	http/1.1

Figure 22: SSL Trust Certificate Check and Protocol Check Result

6.3 User Acceptance Result

The user acceptance form collects data from users by using Google Form. The Google Form consist of Section A, Section B and Section C. Section A collects the respondents' information such as name, email and job scope. Section B collects the testing result for the respondents. Lastly, Section C collects comments from the respondents regarding the system. Table 8 shows the Section B questions in Google Form.

Table 8: Google Form Questions in Section B

No	Section B Questions	Answer
1	Each participated poll can only be voted for one time.	Yes / No
2	Casted vote for each poll cannot be changed.	Yes / No
3	During the voting phase, the system display the current vote counts for each candidate.	Yes / No
4	The system shows the other voters voted candidate(s) for each poll.	Yes / No

The respondents for the user acceptance test is the Science Association Club of Sekolah Menengah Saint Paul Beaufort. The voting system is used to create a poll for committee member selection. Figure 23 shows the result of Section B questions in Google Form. There are a total of 17 respondents.

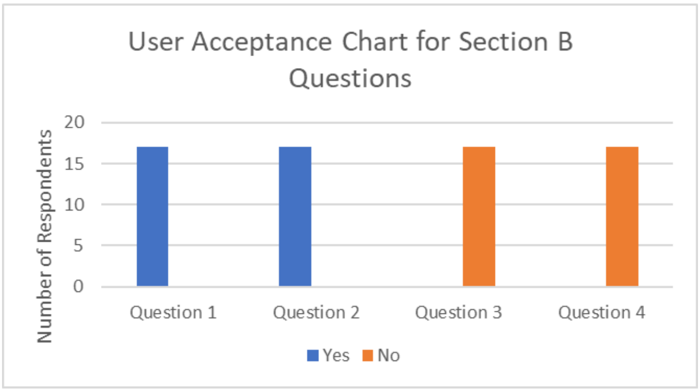


Figure 23: User Acceptance Result based on Google Form in Section B

Based on Figure 23, all the respondents answered “Yes” for question 1 and question 2. This indicates that they are able to vote one time for each poll and the casted vote cannot be change afterward. Contrarily, all respondent answered “No” for both question 3 and question 4. This means that they are not able to see the current vote count for each candidate during the voting phase. Besides that, they are also not able to which candidate the other voters voted.

For the respondents’ comment in Section C, there are two notable comments regarding the proposed system. The first comment suggests that the “Vote” buttons should be taken out after voter voted. In the current system, after the voter voted, the “vote” button is only disabled. The reason for this is because the respondent thought they can vote again after they casted their vote. The second comment suggests to add “Remember Me” features to allow the system remember the user account ID and password.

7. Conclusion

The project “Ethereum Blockchain in Preserving Integrity of Voting System” is developed to protect the integrity of the casted vote. This is done by integrating the proposed system with Ethereum blockchain. The system been developed successfully and achieved all of the objectives. Besides, the system also fulfilled the security requirements of e-voting.

The advantages of the system include the implementation of PBKDF2 to protect the user password. Next, the system enforces strong password policy for every user account. Besides that, session destroy is also implemented when user is inactive for 15 minutes. Lastly, the proposed system fulfilled the security requirements of e-voting which are fairness, anonymity, voter secrecy, un-reusability, individual verifiability and un-forgiveness. Contrarily, the disadvantages of the system are during the candidate addition process for the system, the poll creators need to scroll down the dropdown list to find the particular candidate. This may inconvenience when there are too many users. Besides, after the user session is destroyed due to 15-minutes inactivity, the system webpage still stays on same page and does not redirect to login page.

Therefore, for the future implementation, during the candidate addition process for the system, add a search function that allow the poll creator to find a particular user easily. Next. the system only destroyed the session of the users when they are inactive for 15 minutes. For future implementation, automatically redirect user to login page when their session is destroyed.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support and encouragement throughout the process of conducting this study.

References

[1] D. Dill, “Our elections are not secure,” Sci. Am., vol. 316, no. 3, p. 12, 2017.

[2] M. Oppenheim, “Domestic abuse survivors ‘ blocked from voting due to an antiquated system that puts lives at risk,” Independent, no. November 2019, pp. 1–11, 17-Nov-2019.

[3] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, “An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends,” Proc. - 2017 IEEE 6th Int. Congr. Big Data, BigData Congr. 2017, no. October, pp. 557–564, 2017.

[4] E. Bellini, Y. Iraqi, and E. Damiani, “Blockchain-Based Distributed Trust and Reputation Management Systems: A Survey,” IEEE Access, vol. 8, pp. 21127–21151, 2020.

[5] G. A. Oliva, A. E. Hassan, and Z. M. (Jack) Jiang, “An exploratory study of smart contracts in the Ethereum blockchain platform,” Empir. Softw. Eng., vol. 25, no. 3, pp. 1864–1904, 2020.

[6] F. S. Hardwick, A. Gioulis, R. N. Akram, and K. Markantonakis, “E-Voting with Blockchain: An E-Voting Protocol with Decentralisation and Voter Privacy,” Proc. - IEEE 2018 Int. Congr. Cybermatics 2018 IEEE Conf. Internet Things, Green Comput. Commun. Cyber, Phys. Soc. Comput. Smart Data, Blockchain, Comput. Inf. Technol. iThings/Gree, pp. 1561–1567, 2018.

[7] D. H. Vu, T. D. Luong, T. B. Ho, and C. T. Nguyen, “An Efficient Approach for Electronic Voting Scheme without An Authenticated Channel,” Proc. 2018 10th Int. Conf. Knowl. Syst. Eng. KSE 2018, pp. 376–381, 2018.

[8] K.-H. Wang, S. K. Mondal, K. Chan, and X. Xie, “A Review of Contemporary E-voting: Requirements, Technology, Systems and Usability,” Ubiquitous Int., vol. 1, no. 1, pp. 31–47, 2017.

[9] M. Soud, S. Helgason, G. Hjalmtysson, and M. Hamdaqa, “TrustVote: On Elections We Trust with Distributed Ledgers and Smart Contracts,” pp. 176–183, 2020.

[10] M. Chaieb, S. Yousfi, P. Lafourcade, and R. Robbana, “Verify-Your-Vote : A Verifiable Blockchain-based Online Voting Protocol To cite this version : HAL Id : hal-01874855,” 2018.

[11] X. Yang, X. Yi, S. Nepal, A. Kelarev, and F. Han, “Blockchain voting: Publicly verifiable online voting protocol without trusted tallying authorities,” Futur. Gener. Comput. Syst., vol. 112, pp. 859–874, 2020.

[12] D. Dalalana Bertoglio and A. F. Zorzo, “Overview and open issues on penetration test,” J. Brazilian Comput. Soc., vol. 23, no. 1, pp. 1–16, 2017.

[13] “Pentest-Tools,” 2012. [Online]. Available: https://pentest-tools.com/website-vulnerability-scanning/website-scanner#.

[14] “SSL Trust,” 2019. [Online]. Available: https://www.ssltrust.com.au/ssl-tools/website-security-check.

[15] J. Hunt, “Object Oriented Analysis and Design,” no. June, pp. 343–351, 1998.