BlockVOTE : An Architecture of a Blockchain-based Electronic Voting System

1st Chinnapong Angsuchotmetee
Department of Computer Science
Faculty of Science
Prince of Songkla University
Songkhla, Thailand
chinnapong.a@psu.ac.th

2nd Pisal Setthawong
Department of Management Information Systems
Assumption University
Samut Prakarn, Thailand
pisalstt@msme.au.edu

3rd Sapjarern Udomviriyalanon
Department of Computer Science
Faculty of Science
Prince of Songkla University
Songkhla, Thailand
5910210111@psu.ac.th

***Abstract*—Electronic voting systems provide many advantages over traditional ballot based voting systems mainly over the accuracy and speed of the tallying process of the voting. However, electronic voting systems suffer from many technical and security issues which have limited its deployment in voting scenarios such as company voting and political elections. Centralized electronic voting systems are, by nature not secure, and there are many avenues of cyber-attacks that could tamper the voting result. Electronic voting system should be highly secured, tampered-proof guaranteed, and the voting should be trusted worthy. In this study, we propose BlockVOTE, a Blockchain-based electronic voting system. Our proposal uses Blockchain to ensure that the voting process can be kept secure and trustable through the consensus handling mechanism of the Blockchain. The architecture design and implementation suggestion are provided in this study. The implementation of the proposal was developed and tested via experimentation. The experiment result and the discussion on the possibility of adopting our proposal in an actual election is provided at the end of this study.**

***Index Terms*—Blockchain, Voting system, Electronic Vote**

I. Introduction

An *election* is a core mechanism of the democratic system. A traditional ballot-based voting system is the most frequently adopted approach in an election due to its implementation simplicity. However, a paper ballot-based voting system can be prone to errors and can also be time-consuming. An ideal voting system for an election should be robust such that human errors and fraud votes are kept minimized while ensuring that the result tallying process can be completed as fast as possible.

To overcome the challenges as mentioned, an *electronic voting system* has been proposed to solve many of the issues discussed earlier. There are several existing electronic voting systems which are ready to be implemented [1]. These systems, however, have not yet been widely adopted in an actual election, especially a high-profile election (e.g., a political election). The primary reason comes from the fact that existing electronic voting systems rely mainly on centralized-based architecture. This leads to the possibility of having the voting result to be manipulated or tampered through cyber-attacks or internal manipulation. Hence, a paper ballot-based voting system is still preferable nowadays because not every voter may trust an electronic voting system. An electronic system, which can be trusted by every voter that the result are kept secured, trustable, and non-violable, is needed to be proposed

978-1-7281-2544-2/9/$31.00 ©2019 IEEE

In 2009, an emerging financial technology so-called *Bitcoin*, has been proposed by Satoshi Nakamoto [2]. Bitcoin allows financial transactions, typically processed in a centralized manner, to be processed in a decentralized approach through the underlying mechanism called the *Blockchain*. The Blockchain mechanism enables users to make and record transactions in a distributed, efficient, permanent, non-violable, and verifiable manner. By adopting Blockchain, users in Bitcoin can exchange cash directly in a peer-to-peer manner while every transaction is guaranteed to be trustable without relying on a centralized control from a central banking system. The capability of recording global transactions in a distributed, yet secured and verifiable manner, makes researchers and industrialists have applied Blockchain approaches in many different applications domain, not limited to only financial application. For example, a study in [3] applies Blockchain in a medical data sharing application. A study in [4] applies Blockchain in a logistic monitoring application. To date, however, the suitable method, design, and the implementation of the application of Blockchain in an electronic voting system are still being proposed.

In this study, the requirements for a suitable decentralized-based electronic voting system are identified. The proposal of the architecture design and all the related data models of such a system are described next. The proposed architecture is named the *BlockVOTE: A Blockchain-based Electronic Voting System*. The implementation detail of *BlockVOTE* is also given and the validation experiment has also been conducted.

The organization of this paper is as follows. Section II describes the motivation scenario. The selected scenario is a *political election*. Challenges on proposing a high-security voting system are given and discussed against the given scenario in this section. This section follows by Section III which describes related studies and state of the art of Blockchain. The proposed BlockVOTE is described in Section IV. The implementation and validation experiments of BlockVOTE are described in Section V. The experiment results and challenges are discussed in Section VI. SectionVII concludes the study.

II. Motivating Scenario: A Political Election

An election is a process in which the population chooses from a pool of individuals to hold public office, which in turn represents the needs of the public. A central voting authority is responsible for overseeing, organizing, validating, and announcing the result of the election. The choice of how the voting system is organized can be varied depending on the choice of the voting authority of each organization or country. In general, there are two main types of the voting system, which are (i) a ballot-based voting system and, (ii) an electronic-based voting system. They are described as follows.

A. Ballot-based Voting Systems

The most common centralized voting system is the ballot system. The voting authority would determine the list of potential voters based on predetermined criteria. Each of the voters is given a paper ballot, and then the voter would select one of the possible choices before casting their ballot into a box at a polling station.

Traditional ballot-based voting systems suffer from many disadvantages. The preparation of paper-based voting ballot is an expensive and time-consuming endeavor in which the paper ballots must be prepared beforehand to account for all potential voters for every polling station and are not reusable. Another significant issue is the long voting tally process. The vote tallying process is time-consuming as all the ballots have to be examined, classified, and tallied. Also, the tally process can be tampered with if the voting authorities are biased and/or no stakeholders were monitoring the tallying process to double check the tallying.

B. Electronic-based Voting Systems

To improve paper-based ballot voting system, electronic-based voting systems utilizing *Electronic Voting Machine (EVM)* [5] have been proposed, in which may differ in implementation details. Examples include EVM systems provide on-demand ballot printing and marking, systems that can tally the voting, systems that allow transmission of tabulated results, and online voting systems.

Online voting systems offer the best solution when considering efficiency in voting systems, but there are numerous issues due to its centralized design which makes the system vulnerable to external cyber-attacks, and manipulation by the voting authorities. The infrastructure of the voting must be designed such that, and all the voting result must be kept secured from external online threats and internal tampering.

C. Challenges: A High-Security Voting System

In an ideal case, every election prefers a voting system that is guaranteed to be highly secured and tampered-proof, while the tallying process and result announcing are kept to be as efficient and as soon as possible. To do so, we summarize challenges that have to be addressed as follows.

- *Securing Network Infrastructure:* A highly secured electronic voting system must ensure that all network infrastructure related devices must be able to resilient against cyber-attacking;
- *Securing Voting Data:* A highly secured electronic voting system must keep all the voting data in such a way that the result can always be available for tallying while ensuring that all the data cannot be tampered either through cyber-attacking nor by internal tampering;
- *Trust Management:* A highly secured electronic voting system where the results are 100% reliable and trustworthy for all related parties.

III. Related Studies: Blockchain

Blockchain is an algorithm that handles transaction between peer-to-peer ledgers. Initially, it is designed to support transaction processing in Bitcoin [2]. Bitcoin is a decentralized banking system where each transaction between ledgers can be made directly without relying on a centralized server or central banking authority for validating transactions. Ledgers in the Bitcoin ecosystem attempts to validate the validity of the transaction on their own, in which each transaction within the ecosystem are modeled as a *Block*. A newly created block can only be connected to the shared global chain of blocks only if the block is validated by all ledgers within the Bitcoin ecosystem to be valid. This process helps Bitcoin to keep all transactions to be highly-secured, non-violable, and tampered-proof without relying on a central bank to validate a transaction [6].

Breakthrough in Blockchain came in the form that the block can be embedded with a built-in programmable function to support a custom business logic. Such an extension is regarded as a *Smart Contract* feature of *Blockchain*. A *smart contract* is an autonomous process which is capable of regulating the flow of transactions within a Blockchain network such that a specific set of programmable instructions that every ledger is agreed on would be executed each time a predefined action or events within Blockchain network were made [7].

Blockchain and Smart Contract are deemed suitable for storing global knowledge and business logic for any application domain in a distributed manner while maintaining the security and the privacy of the stored knowledge. This allows for a wider range of application domains. Examples of application domains which utilizes Blockchain include medical record storage [8], supply chain management [9] and authentication systems [10].

IV. BlockVOTE: An Architecture of a Blockchain-based Electronic Voting System

In this study, we propose an architecture of a blockchain-based electronic voting system. Our architecture is named *BlockVOTE*. This section describes the architecture overview, data models, algorithms, and all related formal definitions. The details regarding the implementation of our proposal are later described in the next section. The architecture of *BlockVOTE* is depicted in Figure 1. Our architecture is designed based on the *smart contract* capability of Blockchain. For the clarity of the explanation, we explain our architecture in a step-by-step manner according to the three main steps of any voting

process, which are (i) Poll Creation, (ii) Voting, and (iii) Result Tallying. The details are as follows.

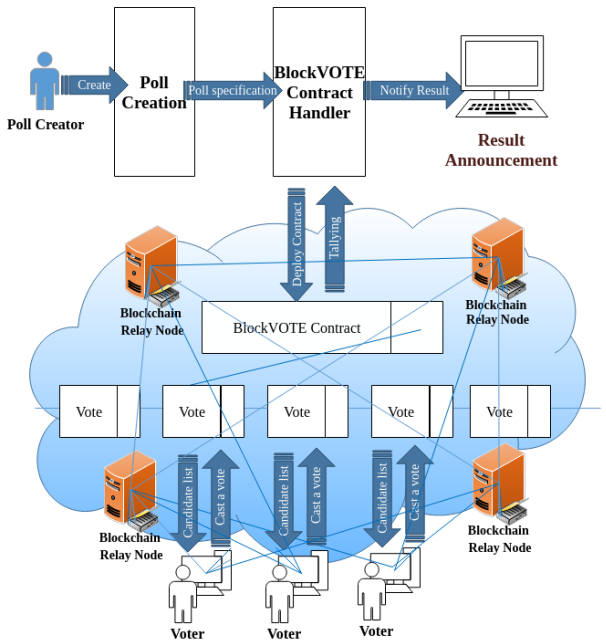

Fig. 1. BlockVOTE: Architecture Overview

A. Poll Creation

Before creating a poll, a list of candidates would be needed to be prepared first. The internal process of choosing candidates (e.g., verifying the profile of each candidate in a political election) can be varied between each organization. This process is not covered within our architecture as it is considered to be outside of the scope of the system. When the list of candidates are ready, the poll creator must send the list of candidates to the *Poll Creation* (see Fig. 1) module of the architecture. This module is used for modeling the list of candidates into a machine-readable format. The data model used for modeling a candidate is described in Def 1.

Definition 1: Candidate: A *Candidate*, c, is a choice within a poll that a voter can cast a vote to. It is modeled as a 3-tuple $c =< id, desc, score >$, where:

- id: is a unique number for identifying c;
- $desc$: is a user-friendly short textual description for describing c;
- $score$: is a number representing number of votes that c got from voters.

After all the candidates are modeled, the *Poll Creation* would automatically pass the list of modeled candidates to the *BlockVOTE Contract Handler* module. *BlockVOTE Contract Handler* is responsible for creating a *BlockVOTE Contract*, which is a smart contract designed specifically for the proposed architecture. The formal definition of the *BlockVOTE Contract* is given in Def. 2.

Algorithm 1 Vote: A BlockVOTE built-in function

Require:
BV: a BlockVOTE contract that a voter invokes
id: an ID number of a candidate that a voter who invokes the contract wants to vote
v: a blockchain-based cryptographic hash of a voter who invokes BV

1: **if** v is in $BV.V$ **then**
2: return $False$ ▷ v has already casted a vote
3: **else if** none of $c.id$ in $BV.C$ is equal to id **then**
4: return $False$ ▷ invalid candidate id
5: **else**
6: $c.score = c.score + 1$ where $c \in BV.C \wedge c.id = id$
7: add v to $BV.V$
8: return $True$ ▷ Successfully cast a vote
9: **end if**

Definition 2: BlockVOTE Contract: A *BlockVOTE Contract*, BV, is a smart contract which is responsible for managing a poll, handling vote activities, and tallying the result automatically. It is defined as a 3-tuple, $BV =< C, V, exp >$ where as:

- C: is a set of candidates defined as $C = \{c_1, c_2, c_3, ..., c_n\} \wedge \forall c_i \in C, c_i =< i, desc, score >$ (see Def. 1);
- V: is a set of voters who have casted a vote to one of candidates in C, $V = \{v_1, v_2, v_3, ..., v_n\} \wedge \forall v_i \in V, v_i$ is a blockchain-based cryptographic hash representing an identity of a given voter;
- exp: is a date-time value indicating that when voters can still cast a vote to C;

When *BlockVOTE Contract Handler* receives a poll creation request, it creates a new BlockVOTE contract according to Def 2, while keeping V empty, and setting exp according to the request from a poll creator. When a new contracted is created successfully, it is automatically deployed onto a pre-deployed Blockchain-based infrastructure on the Internet cloud in such a way that, users can cast a vote to the contract. The detail of the *vote* function of BV is described in the next sub-section.

B. Voting

After the poll creator uses *Poll Creation* module and *BlockVOTE Contract Handler* module to create a new BlockVOTE contract, voters can invoke the *vote* functionality of the contract to cast a vote. The functionality is defined as a built-in function which uses BV as an input. The algorithm of the *vote* function is described in Alg. 1. In short, a voter casts a vote by passing, (i) a contract to be voted, (ii) an ID of a candidate that a voter wants, and (iii) their own cryptographic hash identity, to the *vote* function. The vote is a success only if when that voter has not yet cast a vote, and provided candidate ID is valid. The vote function rejects a vote when a voter has already cast a vote, or when provided candidate ID is invalid.

Each time a vote has been cast by using a function as described in Alg. 1, a new *Block* is created. One block within BlockVOTE architecture stores one vote transaction from one voter, and the most recent state of the contract (i.e., a list of all candidates and their corresponding score received).

C. Result Tallying

The result tallying is a process that is done when the poll deadline has been reached. The tallying process is modeled as a smart contract-based function that is executed automatically by the *BlockVOTE Contract Handler* module when a poll deadline is reached. The formal definition of the result tallying function is given in Def. 3.

Definition 3: Result Tallying Function: A result tallying function, tf, is a function for tallying the result of a given poll. It is defined as

$$tf(BV, cdate) = \begin{cases} false & \text{if } cdate < BV.exp \\ R & \text{otherwise} \end{cases}$$

where as:

- BV: is a BlockVOTE contract to be tallying tf, $BV =< C, V, exp >$ (see Def. 2);
- $cdate$: is a date-time when a tallying process is requested.
- R: is an ordered set, $R = \{c_1, c_2, c_3, ..., c_n\} \wedge \forall c_i \in R, c_i \in BV.C \wedge c_i.score < c_{i+1}.score$

In short, the result tallying function returns a list of candidates within a contract ordered by the score that each candidate is received in descending order. The function returns *false* to indicates that the tallying cannot be done if the poll has not yet ended.

So far, the architecture has been described by using all related formal definitions and algorithms. The following section describes the implementation of the proposed architecture.

V. BlockVOTE: Implementation & Validation

To ensure the viability of the proposed architecture design, the proposed architecture should be generic enough to be developed with different Blockchain application frameworks. Two separate prototypes were developed for the proposed system. The first prototype was developed using *Ethereum* framework[1]. The other prototype was developed using *HyperLedger* framework[2]. This section describes the implementation detail and the validation experiment of the prototypes.

A. BlockVOTE: Ethereum-based Implementation

Ethereum is one of the most famous cryptocurrency ecosystems which supports a smart-contract capability, in a way that any Ethereum wallet owner can deploy their own smart contract onto the Ethereum network globally to support their own business logic. To deploy BlockVOTE in Ethereum, complying to the smart contract development framework of the Ethereum foundation called the *Truffle Framework*[3] was required. The architecture of the Etherum-based BlockVOTE implementation is given in Figure 2.

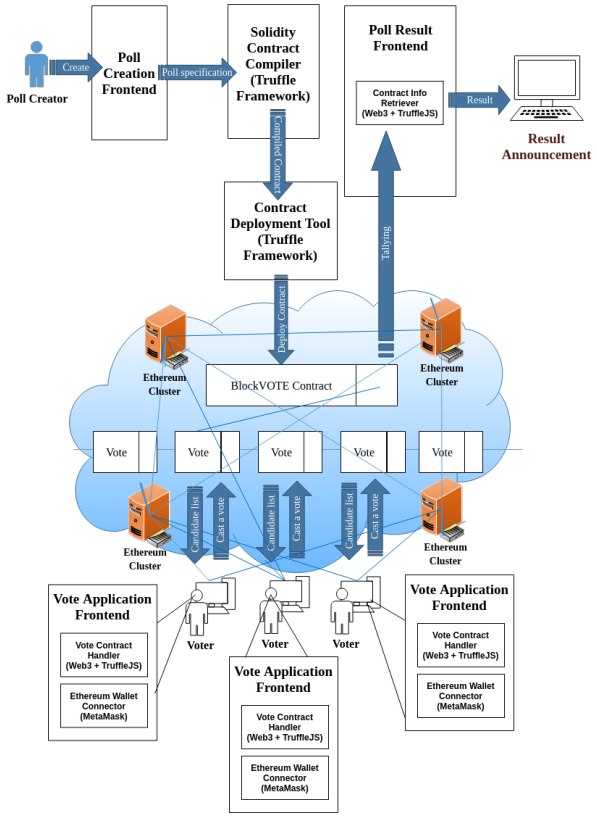

Fig. 2. BlockVOTE: Ethereum-based Implementation

For the proposed prototype, as depicted in Figure 2, the poll creator will create a poll using a web-based application frontend. This frontend application takes the list of candidates given by users to generate a smart contract source code that would later be used by Ethereum. Ethereum specifies that a smart contract must be written in *Solidity* language. The template of the *BlockVOTE* contract in Solidity language is defined as the following:

```
pragma solidity ^0.5.0;
contract BlockVote {
    struct Candidate {
        uint id;
        string desc;
        uint score;
    }
    mapping(address => bool) public voters;
    mapping(uint => Candidate)  public candidates;
    uint public candidatesCount;
    uint256 public exp_date;

    function addCandidate (string memory _name) private {
        candidatesCount ++;
        candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
    }
    function vote (uint _candidateId) public  {
        require(!voters[msg.sender]);
        require(_candidateId > 0 && _candidateId <= candidatesCount);
        voters[msg.sender] = true;
        candidates[_candidateId].score++;
    }
}
```

After the contract code was generated, it would be compiled using the Solidity compiler tool provided by the Truffle

[1]https://www.ethereum.org/
[2]https://www.hyperledger.org/
[3]https://trufflframework.com

Framework. The compiled contract would later be deployed onto the Ethereum network by using the contract migration tool.

For each voter would need to have access to a voting machine which could be a PC, a smartphone, or a tablet that is installed with the application and is Internet-capable. The application utilizes Web3[4] and TruffleJS library for interacting with the deployed BlockVOTE contract.

Even though voting should be free of charge, the Ethereum platform specification requires all users to spend some amount of *Ether* (the name of the currency in Ethereum ecosystem) before they are allowed to create a new block. The amount of Ether that a user needs to spend depends on the size of data within the created block. Therefore, in the proposed Ethereum-based voting application, each user needs to connect their own Ethereum wallet to the voting application before they can cast a vote. The cost per vote in BlockVOTE application, according to the automatic cost calculation in Ethereum network, is 0.000652 ETH per vote.

For the result tallying process, a separate frontend was created using the Web3 and TruffleJS libraries. The tallying process was done by listing all the candidates and their scores as noted within the contract to the frontend. The frontend will reject the tallying request if the poll has not ended yet according to the expiry date of the poll given in the contract.

B. BlockVOTE: HyperLedger-based Implementation

HyperLedger is an opensource blockchain platform hosted by Linux Foundation. Unlike Ethereum, the designed of HyperLedger is not designated for creating a public cryptocurrency-based ecosystem. Instead, it was designed for developing a private blockchain ecosystem where private organizations can create their own blockchain network internally, and deploy their business logic onto their network using a smart contract-based mechanism. The architecture overview of the HyperLedger-based BlockVOTE implementation of the proposed system is given in Figure 3.

The architecture of BlockVOTE Hyperledger implementations is similar to the Ethereum implementation. The differences between both implementations are the languages for modeling a contract and the set of tools required. In HyperLedger, data models and built-in functions are programmed separately. Data models are written using a language-specific to HyperLedger named *CTO* language, while built-in functions are written using Javascript. The template that the proposed architecture uses for generating a data model and built-in function for creating a new BlockVOTE contract are as follows:

```
//Data Model Template//
namespace org.acme.blockvote

participant voter identified by voterID {
  o String voterID
  o String fullName
}

asset ifVoted identified by voterID {
  o String voterID
  o Boolean isvote
}

asset candidateVote identified by candidateID {
  o String candidateID
  o String short_desc
  o Integer totalVote
}

transaction vote {
  --> candidateVote candidateVoteAsset
  --> ifVoted ifVotedAsset
}

//Script Template//
'use strict';

function vote(tx) {
    if (!tx.ifVotedAsset.isvote) {
        tx.candidateVoteAsset.totalVote = tx.candidateVoteAsset.totalVote + 1;
        return getAssetRegistry('org.acme.blockvote.candidateVote')
            .then(function (assetRegistry) {
                return assetRegistry.update(tx.candidateVoteAsset)
                  .then(function(){
                      return getAssetRegistry('org.acme.blockvote.candidateVote')
                  })
            })
            .then(function () {
                return getAssetRegistry('org.acme.blockvote.ifVoted')
                    .then(function (assetRegistry) {
                        tx.ifVotedAsset.isvote = true;
                        return assetRegistry.update(tx.ifVotedAsset);
                    })
            });
    } else {
        throw new Error('Vote already submitted!');
    }
}
```

[4]https://web3js.readthedocs.io/en/1.0/

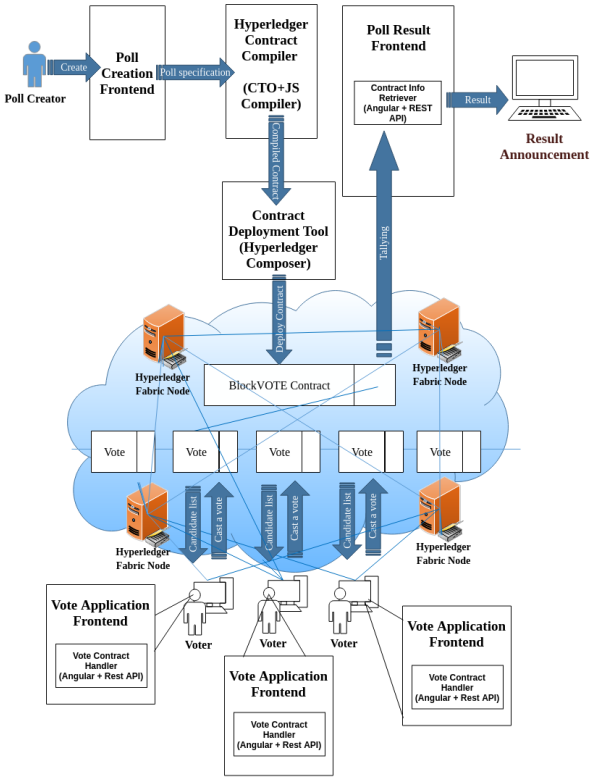

Fig. 3. BlockVOTE: HyperLedger-based Implementation

The contract has been generated would be compiled and deployed using *HyperLedger Composer* toolsets. The compiled BlockVOTE contract would then be deployed in a network of a pre-deployed *HyperLedger Fabric*[5] nodes. This step is obligatory when using HyperLedger as HyperLedger is a private blockchain-based platform. Hence, a set of nodes

[5]HyperLedger Fabric is a software that is used for creating a blockchain server node for HyperLedger

are required for creating the infrastructure for HyperLedger. This step was not required in Ethereum-based prototype as the Ethereum network already has an extensive network of Ethereum node clusters available.

To cast a vote, voters need to have a frontend application to cast a vote. The frontend application was developed using Angular[6] and a set of web-based REST API provided by HyperLedger Fabric for connecting to the nearest HyperLedger Fabric node. The HyperLedger prototype does not require a cryptocurrency wallet because HyperLedger assumes that the application owner themselves owns all the related infrastructure. Hence, voters do not need to pay any fee to cast a vote in HyperLedger-based implementation of the architecture, which is a considerable advantage over the Ethereum based implementation.

C. Validation Experiment

To validate the proposed system, a mock-up poll with 10 candidate choices were created and deployed on both the Ethereum-based and HyperLedger-based prototypes.

The performance of the tallying process would be crucial for a voting application. Three sets of experiments with 10, 50, and 100 votes were conducted. For each set, five rounds of experiments are conducted and the average time delay between the time that the result tallying function was requested and when the result is available on the result notification from the frontend. The result is displayed in Figure 4.

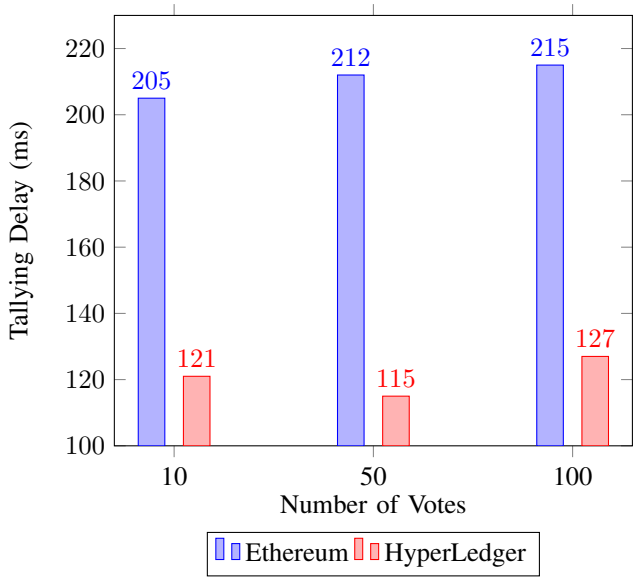

Fig. 4. Tallying Delay of Ethereum-based and HyperLedger-based BlockVOTE implementation

Figure 4 demonstrates that the tallying delaying for all cases is almost constant for each respective implementation. The reason comes from the fact that every block in the network always contains the latest status of the contract. Hence, the tallying process does not require accessing the whole data starting from the beginning to the end of the chain. Only the latest block is required for the result tallying.

It can also be observed from the result that the tallying process of HyperLedger is slightly faster than Ethereum. This comes from the difference in the internal consensus handling algorithm between Ethereum and HyperLedger. Ethereum adopts a Proof-of-Stake based consensus which requires the users to pay Ether to warrant a transaction [11]. HyperLedger uses a certificate-based consensus handling algorithm through the Proof-of-Authority consensus [12] using Apache Kafka library[7]. Proof-of-Authority consensus works faster than Proof-of-Stake consensus leading to the faster tallying process delay when implementing BlockVOTE on HyperLedger. Nevertheless, Proof-of-Stake has more secure consensus handling mechanism [13] than the Proof-of-Authority according to many Blockchain researchers and developers.

VI. Discussion

Significant technical challenges, as mentioned in Section II, are related to securing network infrastructure and voting data. To overcome these challenges, this study proposes a decentralized-based electronic voting system using Blockchain. The proposed system stores voting data among the decentralized blockchain nodes. Tampering with voting results in one of the node is no longer possible because other nodes would not accept the tampered result through the consensus handling mechanism of Blockchain. Hence, the challenges in securing both infrastructure and voting data could be overcome.

Prototypes using both the Ethereum and HyperLedger platform were created for conducting a poll, and the results were tallied in the validation experiment. Based on the experiment, one of the major issues to consider is that the Ethereum-based prototype requires voters to own Ethereum wallet before they can cast a vote, whereas there is no such a requirement for the HyperLedger-based prototype. This comes from the fact that deploying BlockVOTE on Ethereum means that all the voting data has to be stored in the public Ethereum network and all the voters need to follow the *Proof-of-Stake* consensus protocol of Ethereum by paying some amount of Ether before a vote can be cast. This is considered a considerable disadvantage on adopting Ethereum over HyperLedger because it is not viable to force every voter to own an Ethereum wallet and to own Ethereum credits.

On the other hand, deploying BlockVOTE using HyperLedger does not require voters to own a cryptocurrency wallet. However, the main disadvantage of adopting HyperLedger is that it requires the deployment of several HyperLedger Fabric nodes before it is possible to develop and deploy any application. The deployment is required as there is no free public extensive network of HyperLedger.

Developers who seek to implement an electronic voting application on Blockchain would need to weight these mentioned limitations of Ethereum and HyperLedger before starting any implementation.

[6]https://angular.io/

[7]https://kafka.apache.org/

VII. Conclusion and Future Work

Though traditional centralized voting systems have been used since the invention of democracy, there are many possibilities in improving the voting systems. In the area of centralized voting systems, there had been the exploration of usage of online voting systems which offer a compelling improvement from ballot systems and electronic voting systems. However, online voting systems are complex, and the system has security issues that could potentially be abused and would have severe ramifications to the results of the elections.

An alternative approach in the voting system is to explore other approaches. Blockchain technology, which is an open, distributed and self-auditing ledger that can record transactions between parties in an efficient, permanent, and verifiable manner is one technology that could be adapted for use in voting systems. Hence, in this study, the *BlockVOTE: An Architecture of a Blockchain-based Electronic Voting System* was proposed. The architecture, data models, and all related formal definitions were proposed in this study. The architecture was validated employing prototyping and experimenting using two different implementation frameworks, which are Ethereum and HyperLedger. The result shows that the proposed system, in both prototype implementations, could be used for conducting a poll, keeping the result secured while minimizing the result tallying time.

Though our proposed voting system may provide many advantages, numerous issues need to be addressed before the broader adoption of the technology. Hence, for future work, there are several topics that the team is tackling. The first area is to finalize the development and deploy the prototype for a large scale voting as opposed to the limited test experiments. A more massive experiment with live voters could propagate the usefulness of Blockchain technology in elections to a broader audience so that the voting authorities and governmental organizations would examine the technology and may adopt it for future voting. Another direction is to explore the proposal of a Blockchain-based application framework that is more suitable to be used in an electronic voting system than Ethereum or HyperLedger.

References

[1] M. K. Alomari, "E-voting adoption in a developing country," *Transforming Government: People, Process and Policy*, vol. 10, no. 4, pp. 526–547, 2016. [Online]. Available: https://doi.org/10.1108/TG-11-2015-0046

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf," 2009.

[3] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, "Medshare: Trust-less medical data sharing among cloud service providers via blockchain," *IEEE Access*, vol. 5, pp. 14 757–14 767, 2017.

[4] R. Casado-Vara, A. González-Briones, J. Prieto, and J. M. Corchado, "Smart contract for monitoring and control of logistics activities: Pharmaceutical utilities case study," in *International Joint Conference SOCO'18-CISIS'18-ICEUTE'18*, M. Graña, J. M. López-Guede, O. Etxaniz, Á. Herrero, J. A. Sáez, H. Quintián, and E. Corchado, Eds. Cham: Springer International Publishing, 2019, pp. 509–517.

[5] T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach, "Analysis of an electronic voting system," in *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, May 2004, pp. 27–40.

[6] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 3–16. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978341

[7] H. Watanabe, S. Fujimura, A. Nakadaira, Y. Miyazaki, A. Akutsu, and J. Kishigami, "Blockchain contract: Securing a blockchain applied to smart contracts," in *2016 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2016, pp. 467–468.

[8] A. Roehrs, C. A. da Costa, and R. da Rosa Righi, "Omniphr: A distributed architecture model to integrate personal health records," *Journal of Biomedical Informatics*, vol. 71, pp. 70 – 81, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1532046417301089

[9] D. Tse, B. Zhang, Y. Yang, C. Cheng, and H. Mu, "Blockchain application in food supply information security," in *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Dec 2017, pp. 1357–1361.

[10] J.-H. Huh and K. Seo, "Blockchain-based mobile fingerprint verification and automatic log-in platform for future computing," *The Journal of Supercomputing*, vol. 75, no. 6, pp. 3123–3139, Jun 2019. [Online]. Available: https://doi.org/10.1007/s11227-018-2496-1

[11] W. Li, S. Andreina, J.-M. Bohli, and G. Karame, "Securing proof-of-stake blockchain protocols," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, J. Garcia-Alfaro, G. Navarro-Arribas, H. Hartenstein, and J. Herrera-Joancomartí, Eds. Cham: Springer International Publishing, 2017, pp. 297–315.

[12] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18. New York, NY, USA: ACM, 2018, pp. 30:1–30:15. [Online]. Available: http://doi.acm.org/10.1145/3190508.3190538

[13] L. Ismail, H. Hameed, M. AlShamsi, M. AlHammadi, and N. AlDhanhani, "Towards a blockchain deployment at uae university: Performance evaluation and blockchain taxonomy," in *Proceedings of the 2019 International Conference on Blockchain Technology*, ser. ICBCT 2019. New York, NY, USA: ACM, 2019, pp. 30–38. [Online]. Available: http://doi.acm.org/10.1145/3320154.3320156