Implementation and evaluation of blockchain based e-voting system with Ethereum and Metamask

Deni Pramulia
Faculty of Computer Science
Universitas Indonesia
Depok, Indonesia
Email: deni.pramulia@ui.ac.id

Bayu Anggorojati
Faculty of Computer Science
Universitas Indonesia
Depok, Indonesia
Email: bayuanggorojati@cs.ui.ac.id

2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS) | 978-1-7281-9167-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICIMCIS51567.2020.9354310

***Abstract*—The implementation of an electronic voting (e-voting) system which has begun to be widely applied in place of traditional electoral systems, still has a major problem in the level of trust results. E-voting systems are very vulnerable to manipulation issues, such as changes in election results due to hacking or by the electoral system maker. Centralized systems in a network result in data sources coming from one party having the right to store and to manage data sources. The issue of trust caused by a centralized data distribution system can be overcome by spreading data in a system network. Blockchain is a distributed ledger, where every party in the network has the same data source, and it has a powerful characteristic known as immutability which is very suitable for e-voting system. This research proposed a blockchain based e-voting system with Ethereum and metamask. We show that the proposed e-voting system fulfills six basic principles of an election system, namely secret ballot, one-man one-vote, voter eligibility, transparency, votes accurately, recorded and counted, and reliability. Furthermore, the performance evaluation of the e-voting system shows that the slow gas price option gives the lowest gas price per second result, i.e. the best trade-off.**

***Index Terms*—Electronic voting, blockchain, smart contract, Ethereum**

I. Introduction

According to Oxford dictionary, *election* is defined as a process of choosing a person or a group of people for a position, especially a political position, by voting[1]. In general, a public election system is used as a parameter of successful democracy implementation in a nation scale, and even in a smaller scale such as an institution. A democratic election system has been widely implemented in democratic countries, including Indonesia, USA, Canada, etc. In a smaller scale, an election system has also been implemented in the election of head of village, or head of student council in schools or universities.

As per the election definition, voting is an important process. The voting process in the election system may adopt various methods, such as the traditional one as well as the *electronic voting* (e-voting). A traditional voting method is based on ballot paper and voting box to collect all those ballot papers. In case of Indonesian general election, the voters are authenticated by means of their national ID card and the list of eligible voters compiled and published by the commission of general election. Similar approach may also be used in other traditional election. There are many potential frauds in traditional method, such as fake national ID card, invalid list of eligible voters, ballot papers that are already used to vote one of the candidate, and manipulation of election results.

E-voting system is implemented to improve many drawbacks in traditional voting method. It refers to the adoption of technology that replaces the paper based system to record the election results and put it into counting system [1]. According to Kersting dan Baldersheim [2], e-voting system is divided into two main categories, namely internet and non-internet e-voting. The internet based e-voting clearly requires internet connection, while non-internet voting may use other electronic devices, such as short message service (SMS), vote by phone or digital TV. The main advantage of e-voting compared to traditional voting system is that it brings faster counting process, requires low budget, and could prevent pre-selected ballot papers. However, there are still some issues related to security of the e-voting system.

According to Corry [3], there are six basic principles of an election system, namely *secret ballot*, *one man one vote*, *voter eligibility*, *transparency*, *votes accurately recorded and counted*, and *reliability*. On the other hand, a so called *luber jurdil* principles in Indonesian's election, which means direct (no intermediary), general or public, free, confidential, honest and fair, can also be applied to e-voting principles. By following those principles, a correctly implemented e-voting system can improve ballot's security, speed up the processing time of election results (i.e. more efficient process), and gain more trust in both the election process and results [4].

Blockchain, as a technology that provides transparency in a transaction without losing security aspect, is a good candidate to be the solution for existing issues in e-voting. According to Cachin and Vukolic [5], Blockchain is a distributed ledger that records transactions, managed by distributed nodes without centralized authority through a distributed cryptographic protocol. This means that blockchain removes the need of centralized third party to validate a transaction, just like removing bank in money transfer [6]. By eliminating a third party to validate a transaction, blockchain provides four key characteristics to maintain trust in a transaction, namely immutable ledger, secure, shared and distributed [7]. With such key characteristics, blockchain is a good solution in e-voting

[1]https://www.oxfordlearnersdictionaries.com/definition/english/election

system due to several advantages, such as high availability, integrity, correctness, and verifiability [8].

In this paper, an Ethereum blockchain based e-voting system is proposed. Ethereum is choosen because it is written in a turing-complete language, unlike Bitcoin, and supports smart contract [9] [10]. Furthermore, it is supported by big communities, including public Ethereum test network that are freely available for experimental purpose. However, building a Distributed Application (DApp) for Ethereum is still challenging with respect to the user experience because it requires the user to have an Ethereum account and able to manage her own keys, which can be a burden for a novice user who only wants to vote. To this end, a meta-transaction is needed. Meta-transaction is like a middle layer that relays a transaction between a user and the blockchain network. Metamask, one of the meta-transaction implementation for web browser, is also used in the proposed e-voting system.

II. Proposed solution

According to Cetinkaya [11], e-voting system involves three main actors, namely voter, registry authority (ies), and tallying authority (ies). Voter is someone who is eligible to vote in an election. Registry Authority is someone or an organization who is responsible to make sure that only the registered voters can give only one vote in an election. Tallying Authority is someone or an organization who is responsible to collect the ballots and count the election results. In our proposed e-voting system, both voter and registry authority are represented by Voter and Ballot Manager sub-systems respectively, while tallying authority is performed automatically by a function the smart contract.

The proposed e-voting system makes use of Ethereum Blockchain network because it supports smart contract and the availability of various development tools. Its main architecture consists of two main systems, namely Ballot Manager and Voter. Both systems communicate to the Ethereum Blockchain network through Metamask, i.e. an Ethereum meta-transaction implementation in a form of browser add-on. Fig. 1 illustrates the high level view of the system architecture as well as the main flow diagram.

A. Ballot manager

Ballot manager plays the most important role in the proposed e-voting system. It is responsible to deploy the smart contract into the blockchain network, create voting proposal, add voters, start voting and end voting process. To be able to perform all those tasks in the blockchain network, a ballot manager needs to have a valid blockchain account, e.g. Ethereum account in this case. Despite its name, it does not store information about ballot to satisfy *secret ballot* principle in voting [3]. Only name and voter's address are stored in blockchain network through smart contract, while vote information is implemented as private modifiers. More discussion about this issue is carried out in section IV-C. Details of all functionalities on ballot manager will be explained in the next sub-sections.

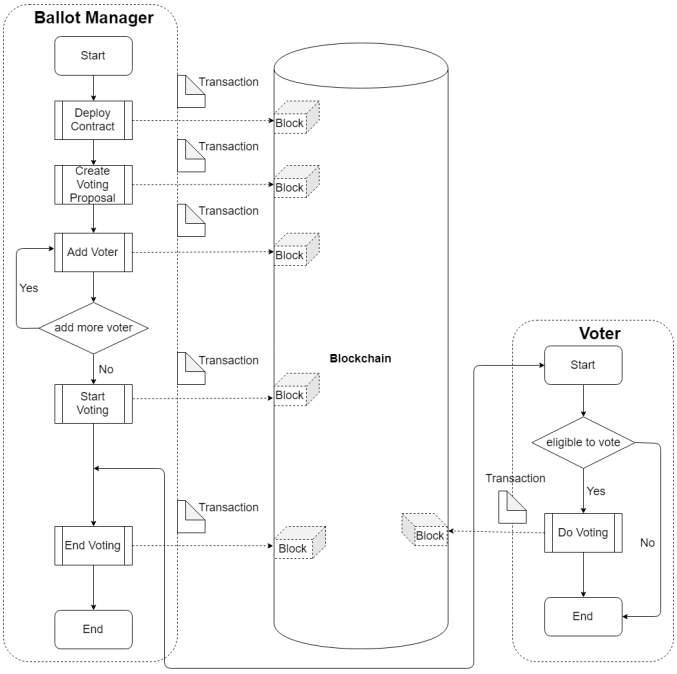

Fig. 1. High level architecture and flow diagram of the proposed e-voting system excluding the Metamask

1) Deploy smart contract: Deploying smart contract into the blockchain network is the first mandatory action before any others. The smart contract consists of all necessary variables, data structures and functions, including conditions, that defines how the e-voting mechanism is to be done. Brief implementation details of smart contract will be further explained in sub-section III-A.

2) Create voting proposal: Create voting proposal essentially activates the e-voting system based on the smart contract that has been deployed earlier. It is important to note that the voting proposal can only be created by the ballot manager, i.e. the one who deployed the smart contract which can be identified by its address. Hence, a specific rule is included in the smart contract to accommodate such restriction. Furthermore, when the voting proposal is created, it will be bind with the smart contract's address.

3) Add voter: Once the voting proposal is created, ballot manager can start adding or registering voters to the system, i.e. blockchain. This mechanism is similar to the conventional voting method in which every eligible voters have to be registered in the system before they can give their voting right. Please note that this process can be done by adding voter one by one or add voters in batch. The first option requires extra work for the ballot manager but it is easier to estimate the required gas fee to execute this transaction, while the opposite applies for the later. Adding voters in batch is easier for the ballot manager, especially when there are huge numbers of voters, but it is difficult to estimate gas fee to execute transaction involving loop.

4) Start voting and end voting: The voting period is between the time when start voting and end voting transactions are sent to the blockchain network in sequential order. During this period, the voter can use their voting right by sending this vote to the blockchain network. The smart contract includes a voting status indicator to allows a voter in giving her vote.

B. Voter

In general, a voter can be anyone who has a valid Ethereum account. However, a voter needs to be registered by the ballot manager to be able to use her voting right in the e-voting system, as previously explained. Registering a voter essentially means that her address is linked to the voting proposal defined in the smart contract.

During the voting period, a voter can give her vote which technically means sending a transaction containing her address as well as her choice or vote to the blockchain network. A voter who already voted is not able to give another vote, i.e. to satisfy the *one man one vote* principle. Therefore a specific identifier is required to indicate whether a registered voter has already voted or not. Furthermore, votes from all registered voters during this period are accumulated and summed up to get the final voting result.

C. Metamask

The proposed solution also includes a meta-transaction component known as Metamask[2]. Metamask, as one of meta-transaction implementation, provides convenience both for the users and developers. For users, it hides the complexity for interacting with Ethereum network especially for those who are not familiar with Ethereum, e.g. create Ethereum account, manage account, keys and wallet, etc. Likewise, the developer only needs to interact with the globally available Ethereum Application Programming Interface (API). On top of that, a DApp can start the connection to Ethereum network without synchronizing a full node, as it channels the connection through Ethereum node provider, such as Infura[3].

III. Prototype Implementation

A. Smart contract

The smart contract is written in *Solidity* language [4] and and then the solidity code is compiled into bytecode and Application Binary Interface (ABI) by using *Remix* Integrated Development Environment (IDE) [5]. The bytecode version is then deployed into the Ethereum blockchain network, while ABI is then used by the DApp to interact with the smart contract. The Ropsten Ethereum test network[6] is used to deploy the smart contract as well as to record all the transactions done by our e-voting system prototype implementation.

For testing purpose, a simple yes or no voting proposal is implemented in the smart contract of the e-voting prototype. The smart contract consists of two data structures which are used to store the voter's name and a voter's chosen vote respectively. It also implements a static *enum* structure to indicate voting status, e.g. *created*, *voting* and *ended*. Furthermore, it has a constructor to initialize the smart contract, i.e. when voting proposal is created (see section II-A2). Finally, it implements the rest of functions described in II, including the rules that restricts several actions in the e-voting system, e.g. only allows registered voter to vote between the start and end of voting period. Due to space limitation, the detail of smart contract can be further reviewed from the Ropsten Ethereum address as shown in Fig. 2 or 3.

B. Ballot manager and Voter DApps

Both ballot manager and voter DApps are implemented as simple web applications written in Javascript. A Javascript library called web3js[7] is employed to simplify the interaction of both DApps with Ethereum network. The interaction between the web3js powered DApp and smart contract is possible through ABI which is in JSON format. The execution of smart contract itself is done by Ethereum Virtual Machine (EVM).

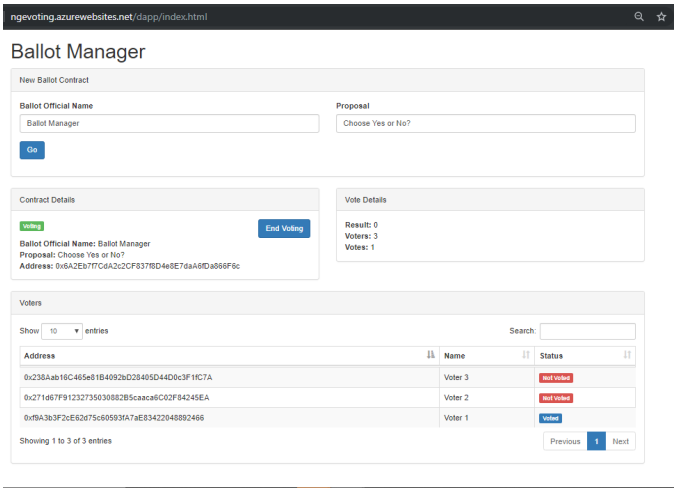

Fig. 2. Ballot manager user interface

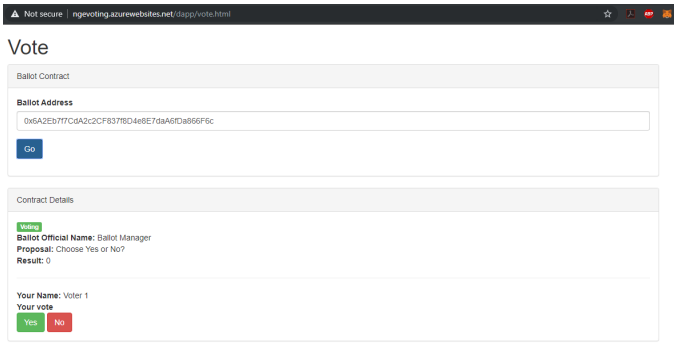

Fig. 3. Voter user interface

[2]https://metamask.io/
[3]https://infura.io
[4]https://github.com/ethereum/solidity
[5]https://remix.ethereum.org/
[6]https://ropsten.etherscan.io/
[7]https://github.com/ethereum/web3.js

The graphical user interface (GUI) of ballot manager and voter DApps prototype are shown in Fig. 2 and 3 respectively. Please note that the GUIs presented in the figures show the complete User Interface (UI) components, e.g. panels, buttons, list, etc, after the voting process is started and one voter gave her vote. Initially, both ballot manager and voter DApps only show the upper most panel, while the other panels are hidden. As a prototype implementation, the DApps are implemented as simple HTML pages without HTTPS and particular user account to access the web interface, as the account is directly connected to Ethereum account through Metamask. Finally, both DApp can be accessed through the URL shown in Fig 2 and 3 (i.e. http://ngevoting.azurewebsites.net/dapp/index.html and http://ngevoting.azurewebsites.net/dapp/vote.html).

IV. Evaluation and analysis

A. Functional test

The functional test is conducted to verify the implemented blockchain based e-voting system works as intended. It comprises two portions of testing which corresponds to two main actors who are involved in the system, namely the ballot manager and voter.

TABLE I
List of functional test cases in ballot manager DApp

Nr.	Test case	Expected result	Result
1	Deploy smart contract	Remix IDE's error checkers confirms there is no error in smart contract	Pass
		Metamask ask confirmation to send the transaction	Pass
2	DApp connects to Ethereum network through Metamask	Metamask asks permission to allows DApp accessing Metamask account to connect to Ethereum network	Pass
3	Create voting proposal	DApp is connected to Metamask and the address of voting creator corresponds to the address of the deployed smart contract	Pass
4	Add a voter to the eligible voters list	The voter will be added to the eligible voting list along with its address and voting status	Pass
5	Start voting	The status voting proposal is changed from created to voting	Pass
		The button Start Voting is changed to End Voting	Pass
6	Voter gives her vote	Votes number in Vote Details panel is incremented	Pass
		Voter's status is changed from Not Voted to Voted	Pass
7	End Voting	End Voting button disappears	Pass
		Vote details does not change anymore	Pass

First part of the functional tests include all functionalities related to the ballot manager, and it is not strictly bounded to the ballot manager DApp only. For instance, smart contract is deployed through Remix IDE. Complete list of functional test cases related to ballot manager along with the results are displayed in Table I. The test results show that all functions related to ballot manager pass the tests. Moreover, the *voter eligibility* and *votes accurately, recorded and counted* principles in voting are satisfied which are indicated by the 4^{th} and 6^{th} test cases respectively. [3]. Please note that in the sixth test case, the expected results are those that can be observed in the ballot manager DApp.

TABLE II
List of functional test cases in Voter DApp

Nr.	Test case	Expected result	Result
1	Start voting process	Ongoing "voting" status appears in the Contract details panel	Pass
		The buttons that give available voting options are provided	Pass
2	Unregistered voter	No option to give vote, thus not able to vote	Pass
3	Registered voter give vote	Voted status appears	Pass
		No option to give vote anymore	Pass
4	Voting process is ended	No more option to give vote	Pass
		"Ended" voting status appears in the Contract details panel	Pass

At the Voter DApp side, the actual functions can be executed by voters and thus tested, only after the voting has been started by the ballot manager. Therefore, the test cases shown in Table II are based on this assumption. Before the voting is started by ballot manager, a voter can only go to the voting proposal address in Ethereum network. Table II lists all functional test cases along with the results for the functionalities related to voter DApp. It can be concluded that all the tests pass successfully, including the second test case when an unregistered makes an attempt to vote even though she can obtain the voting proposal's address. Moreover, *one man one vote* principle in voting is satisfied by the proposed system which is indicated by the "no option to give vote anymore" result in the 3^{rd} test case in Table II.

As a final note, all transaction history that involves the smart contract used in this research can be traced through etherscan by searching through the contract's address. Likewise, the transaction history that involves ballot manager and any voter can be traced in the same way. This can also provide transparency to the voting process, i.e. satisfying *transparency* voting principle [3], and serve as monitoring system and audit trail, e.g. in case of any malicious voter attempting to abuse the voting.

B. Performance evaluation

Performance evaluation of the e-voting system prototype is focused on the required time for the transaction until it is confirmed or verified in the blockchain network. On top of that, three gas price options provided in Metamask, i.e. fast, medium and slow, are tested. The fast gas price option demands the most expensive gas price than the other options but it gives the fastest "estimated" transaction time, although the estimated time is not always true. It is important to note that the proposed e-voting system smart contract is deployed in a public Ethereum Ropsten test network, thus the performance

TABLE III
TRANSACTIONS COST AND TIME IN E-VOTING SYSTEM FOR DIFFERENT GAS PRICE OPTIONS IN METAMASK

User	Transaction	Fast			Medium			Slow		
		Gas price	Est. time	Actual time	Gas price	Est. time	Actual time	Gas price	Est. time	Actual time
Ballot manager	Deploy contract	0.008832 ETH	30 sec	35 sec	0.004416 ETH	168 sec	18 sec	0.000883 ETH	312 sec	35 sec
	Create ballot	0.013087 ETH	24 sec	40 sec	0.007852 ETH	126 sec	13 sec	0.001309 ETH	318 sec	50 sec
	Add voter 1	0.001042 ETH	24 sec	47 sec	0.000625 ETH	126 sec	43 sec	0.000104 ETH	558 sec	59 sec
	Add voter 2	0.000817 ETH	24 sec	28 sec	0.000409 ETH	222 sec	47 sec	0.000082 ETH	648 sec	49 sec
	Add voter 3	0.000817 ETH	24 sec	39 sec	0.000409 ETH	222 sec	36 sec	0.000082 ETH	600 sec	15 sec
	Start voting	0.00067 ETH	24 sec	41 sec	0.000402 ETH	120 sec	86 sec	0.000067 ETH	600 sec	53 sec
Voter 1	Vote	0.001695 ETH	30 sec	16 sec	0.000923 ETH	216 sec	23 sec	0.000168 ETH	630 sec	34 sec
Voter 2	Vote	0.00114 ETH	24 sec	10 sec	0.00057 ETH	234 sec	30 sec	0.000114 ETH	450 sec	30 sec
Voter 3	Vote	0.00114 ETH	24 sec	21 sec	0.000614 ETH	234 sec	20 sec	0.000123 ETH	516 sec	47 sec
Ballot manager	End voting	0.000775 ETH	24 sec	37 sec	0.000426 ETH	198 sec	42 sec	0.000078 ETH	642 sec	23 sec
	Total	0.030015 ETH	172 sec	314 sec	0.016646 ETH	1866 sec	358 sec	0.00301 ETH	5274 sec	395 sec

evaluation results do not involve any computational resource from our side.

The overall performance evaluation results of the e-voting system prototype is listed in Table III. According to the results, the actual time always go beyond the estimated time, especially in the medium and slow gas price options in which the actual time is way faster than the estimated time. Interestingly, as opposed to the medium and slow gas price, the actual time in fast option mostly take slower than the estimated time. Furthermore, the total sum of actual time for each gas price scenario does not differ significantly while the difference among the total sum of gas price for each gas price option is quite significant. All in all, the evaluation results show that all transactions are successfully executed no matter how long they take. It shows that the *reliability* principle in voting is satisfied [3].

TABLE IV
GAS PRICE PER SECOND RATIO FOR ALL TRANSACTIONS IN DIFFERENT GAS PRICE OPTIONS

Transaction	**Gas price per second**		
	Fast	**Medium**	**Slow**
Deploy contract	0.000252343	0.000245333	2.52286E-05
Create ballot	0.000327175	0.000604	0.00002618
Add voter 1	2.21702E-05	1.45349E-05	1.76271E-06
Add voter 2	2.91786E-05	8.70213E-06	1.67347E-06
Add voter 3	2.09487E-05	1.13611E-05	5.46667E-06
Start voting	1.63415E-05	4.67442E-06	1.26415E-06
Vote	0.000105938	4.01304E-05	4.94118E-06
Vote	0.000114	0.000019	0.0000038
Vote	5.42857E-05	0.0000307	2.61702E-06
End voting	2.09459E-05	1.01429E-05	3.3913E-06
Average	9.63326E-05	9.88579E-05	7.63251E-06

Taking the gas price per second ratio for all the transaction for each gas price scenario, it can be seen in Table IV that the gas price per second average in the slow gas price option is the lowest one among the three. From this finding, it can be concluded that slow gas price option gives the best trade-off between the gas price and transaction time. Nevertheless, the experiment results highly dependent on the Ethereum blockchain network being used, in which the price may go high when the transaction volume in the network is high, while it may go lower when the transaction volume in the network is low.

C. Security and data transparency

Out of three important security aspects known as CIA triad (confidentiality, integrity and availability), blockchain based e-voting system ensures the integrity and availability. The integrity is ensured by the immutable characteristic of blockchain, i.e. recorded transaction in the blockchain cannot be changed. Meanwhile, the availability can be ensured thanks to the distributed and large public Ethereum network utilized by the system. As for the confidentiality of the information recorded in the blockchain, it highly depends on the smart contract implementation.

One way to prevent the user's vote from being accessed by other party is by using private access type for the variable that carries vote information. However, having private access type does not fully prevent other party from figuring out the user's vote information. The following steps are taken to decode the transaction content of the e-voting system:

1) Trace the transaction in Ethereum network: Every user can trace any transaction in any available public Ethereum networks through Etherscan. As earlier mentioned, our e-voting system is deployed in Ropsten test network, thus one need to check the corresponding network in Etherscan. After that, one can trace the transaction performed by any voter given the voter's address.

2) Analyze the vote function in the smart contract: When a voter votes during the voting period, there is an interaction between voter DApp through web3js with the smart contract which has been compiled in an ABI form and deployed in the blockchain network. One thing is to capture the transaction sent by a voter and see its content, another thing is to know what function in the smart contract is executed in that transaction. Those two information is important in figuring out the voter's choice in a transaction because every transaction is sent in a hashed version. In principle, since every transaction data is visible to all the nodes, one could read private variables if the transaction is known.

To get the exact function name used by a voter to vote, one needs to get the ABI by decompiling the smart contract's bytecode from Etherscan, asuming that one has no access to the source code. Performing a bytecode decompilation means extracting the information in the bytecode's runtime and transform it into a human readable form by the EVM. The ABI as a result of bytecode decompilation has high similarity with the ABI obtained from IDE. The most important thing is that one knows the function name to give a vote (i.e. `doVote(bool)`). On a side note, bytecode decompilation is not required if a deployed smart contract has been verified and published by Etherscan. In such a case, a smart contract's source code is publicly available for any Etherscan user.

3) Read the transaction data sent by a voter: One can have a look at a transaction data by reading its *transaction hash*. There are several information provided in a transaction hash, such as transaction status, sender's and receiver's addresses, transaction fee, and transaction's input data. In this case, one is interested in the transaction's input data. An example of input data from our experiment is `0x87caea780000000000000000000000000000000000000000000000000000000000000001` in hexadecimal. We will come back to this input data later.

4) Get the hash value of `doVote(bool)` function: The first step in obtaining a voter's vote, one need to encode or get the hash value of the `doVote(bool)` function. The hash algorithm standard used by Ethereum is Keccak-256 which is part of Secure Hash Algorithm (SHA)-3 standard released by national institute of standards and technology (NIST) in 2015. The hash value of `doVote(bool)` by using Keccak-256 is `0x87caea78a8df593f9fa3be5e3ecbde321879259f66dc52b0cbdef3745aa4757c` in hexadecimal.

5) Obtain the voter's vote: Based on the results obtained in point 3) and 4), one can see that the first four bytes of the voter transaction's input data and the hash value of `doVote(bool)` is identical, i.e. `0x87caea78`. The first four bytes of a transaction data always refers to the method signature (i.e. `doVote(bool)`). The following characters in the transaction data refers to the method's parameter, represented by hexadecimal value padded to 32 bytes. Since the `doVote` method takes `boolean` data type as a parameter, only two values are expected from the transaction data, i.e. `True` or `False`, which correspond to `0x01` and `0x00` in hex respectively (padded to 32 bytes). Looking back at the transaction data obtained in point 3), one can conclude that the particular voter votes *Yes*. This finding shows that the *secret ballot* voting principle is not fully satisfied. However, this can be solved by applying additional cryptographic technique, such as zero-knowledge proof which is available in public Ethereum blockchain network.

V. Conclusion

The design and implementation of the proposed blockchain based e-voting system with Ethereum and Metamask can serve as a solution to security and trust issue in the e-voting system, thanks to the blockchain characteristics: immutable ledger, shared, secure and distributed. Furthermore, the proposed blockchain based e-voting system also satisfies basic principles of an election system defined by Equal Justice Foundation, namely *secret ballot*, *one-man one-vote*, *voter eligibility*, *transparency*, *votes accurately, recorded and counted*, and *reliability*. Based on the performance evaluation results, the time required to validate a transaction, especially *Vote*, ranges from 10 to 47 seconds, which is tolerable considering that this is a non-realtime transaction. Additionally, slow gas price option gives the best trade-off between the required gas price and transaction time. Last but not least, the proposed blockchain based e-voting system provides transparency, but the input data in *Vote* transaction can be revealed through reverse engineering technique.

This e-voting can still be improved in many ways. First, add voter function can be modified so it can add multiple voters in batch, which is really convenient if there are millions of eligible voters to be registered. Second, encryption or other technique, such as zero-knowledge proof, need to be incorporated to keep the confidentiality of user's vote. Third, to prevent voter's private key to be stolen, an additional method to access voter's Ethereum or Metamask account, such as biometric based authentication, can also be incorporated.

Acknowledgment

This research is supported by Universitas Indonesia under "Publikasi Terindeks Internasional (PUTI) Prosiding" research grant NKB-884/UN2.RST/HKP.05.00/2020.

References

[1] M. Kumar and E. Walia, "Analysis of electronic voting system in various countries," *International Journal on Computer Science and Engineering*, vol. 3, pp. 1825–1830, May 2011.

[2] N. Kersting and H. Baldersheim, "Electronic voting and democratic issues: An introduction," in *Electronic Voting and Democracy: A Comparative Analysis*, N. Kersting and H. Baldersheim, Eds. London: Palgrave Macmillan UK, 2004, pp. 3–19.

[3] C. E. Corry, "Basic voting principles," in *Vote Fraud and Election Issues*. http://www.ejfi.org/Voting/Voting.htm, 2009.

[4] P. Wolf, R. Nackerdien, and D. Tuccinardi, "Introducing electronic voting: Essential considerations," Policy Paper, International Institute for Democracy and Electoral Assistance (IDEA), December 2011.

[5] C. Cachin and M. Vukolic, "Blockchain consensus protocols in the wild," *CoRR*, vol. abs/1707.01873, 2017. [Online]. Available: http://arxiv.org/abs/1707.01873

[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf," 2008.

[7] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," National Institute of Standards and Technology (NIST), Internal Report 8202, October 2018. [Online]. Available: https://doi.org/10.6028/NIST.IR.8202

[8] C. Meter, "Design of Distributed Voting Systems," Master's thesis, Department of Computer Science, Heinrich-Heine-University Düsseldorf, September 2015.

[9] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," White Paper, 2014. [Online]. Available: https://ethereum.org/en/whitepaper/

[10] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger petersburg version," 2020. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[11] O. Cetinkaya and D. Cetinkaya, "Verification and validation issues in electronic voting," vol. 5, no. 2, p. 117-126, 2007.