Blockchain-based Multisignature Wallet System for Decentralized Autonomous Organization

Vincentius Lienardo
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
vincentiuslienardo@gmail.com

Rinaldi Munir
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
rinaldi@informatika.org

***Abstract*—As a decentralized organization whose governance decisions are governed democratically by its members, a decentralized autonomous organization (DAO), just like a traditional organization, requires a system to manage its finances. In this case, the proposed system is in the form of a wallet to manage assets owned by DAO. Wallets need to be designed in such a way that before a transaction can be executed, it needs to be approved by a majority of the committee within the DAO. In addition, the system needs to ensure transparency or openness and integrity of each transaction so that the system can mitigate the potential for fraud or crime that is at risk of occurring. Therefore, a blockchain-based multisignature wallet system is designed and implemented that allows a transaction to be signed by DAO committees. By using blockchain, the system guarantees that no one party can change or manipulate transaction data.**

Keywords—DAO; multisignature wallet; blockchain; cryptocurrency.

I. INTRODUCTION

Blockchain is getting more popular. In 2015, Ethereum became the first public blockchain platform to support smart contracts with Turing-complete virtual machine called the Ethereum Virtual Machine (EVM). With smart contracts, users can build various decentralized applications (dApps) on the blockchain. Inspired by this, there is a thought that the management of an organization and operational rules can be coded on the blockchain in the form of smart contracts so that the organization will operate independently according to predetermined business logic without the intervention of third parties, which is then called a decentralized autonomous organization (DAO) [1]. Real implementations of DAOs benefit from the emergence of blockchain. With DAO, everyone can collaborate together around the world, there is no centralized authority or power thus eliminating hierarchies and bureaucracy that might exist. The decision-making process within the DAO is carried out on a bottom-up basis so that the direction of movement of the DAO is regulated by its members through a set of rules that define it, which resides in the blockchain.

In contrast to traditional organizations that need trust from certain parties in their development process, especially investors, DAOs can be trusted by outside parties through smart contract codes that represent the rules in DAO, which are transparent and verifiable. However, like organizations in general, DAO provides an opportunity for stakeholders to invest their capital, through tokens, which are cryptocurrencies. To run the economy and business in DAO, assets (treasury) need to be managed as effectively and safely as possible. This treasury management encourages the idea that it is necessary to implement a system that stores cryptocurrencies in the form of a wallet and a system that manages transactions carried out by the committees within the DAO.

As a place to store cryptocurrency assets, a wallet, or more specifically a crypto wallet, is a tool that allows users to interact with the blockchain network [2]. With this, cryptocurrency can be sent and received through a wallet. The majority of wallets that have been implemented, for example, MetaMask, Trust Wallet, etc., can generate one or more public and private key pairs. The public key is used to generate the address needed to receive the transaction, while the private key is used during the creation of digital signatures and the transaction verification process. Some wallets that have been in circulation can only sign a transaction individually, if these existing wallets are implemented in the DAO, the verification and approval process for transactions can only be carried out by one person who is on the DAO committees, a wallet is needed that can handle the majority of approvals from the committees. Therefore, a multisignature wallet may be an alternative solution.

Multisignature wallet aims to increase security by enabling a group of users to approve a transaction by providing a signature owned by each user and dividing the responsibility for the transmission of cryptocurrency assets among multiple users [3]. Various architectural wallet system solutions will be analyzed along with their advantages and disadvantages, as well as their relationship to DAOs.

The main problem that will be discussed and solved is the development of a multisignature wallet system for DAO using blockchain so as to ensure transaction security. The main problem can be divided into several subproblems: what type of blockchain is suitable for use in a multisignature wallet system for DAO, how to design an architecture of multisignature wallet system using blockchain, and how a multisignature wallet system using blockchain can solve problems that arise in

conventional systems. Therefore, the main goal to be achieved is to build a proposed multisignature wallet system using blockchain. To achieve this goal, it is necessary to achieve the following subgoals: analyzing the right type of blockchain to build a multisignature wallet system, designing an architecture of multisignature wallet system using blockchain, implementing a multisignature wallet system using blockchain, and testing the integrity of data in a multisignature wallet system using blockchain.

The scope and limitations of this research are as follows: the system to be implemented includes but is not limited to transaction execution in the form of sending coins/tokens that are approved by the majority of parties; all parties who use this system are on the same blockchain network, the transactions carried out run in a certain network protocol; the parties referred to as transaction executors on the blockchain are elected members of the DAO committee; majority of miners/validators behave honestly, transactions validated by several miners/validators through the consensus used are valid transactions; various transaction data flowing in the blockchain is assumed to be true and trusted; and the system to be built may not be officially recognized by current regulations in Indonesia so it can be assumed that the implementation of the system has been approved by local regulations.

II. Related Works

Several studies on related systems will be described, consisting of research on an externally owned account (EOA) wallet system, namely MetaMask, research on a multisignature wallet system, namely Gnosis Safe, and research on a DAO system, namely Developer DAO.

A. MetaMask

MetaMask is a crypto wallet system in the form of a browser extension that is used to interact with the blockchain. MetaMask was first built by ConsenSys in 2016 [4]. With MetaMask, users can store and manage keys on their accounts, broadcast transactions, and send and receive cryptocurrencies [5]. MetaMask uses an account of type EOA, meaning that users have control over access to assets or contracts that can be done using a private key so that the private key needs to be strictly kept confidential.

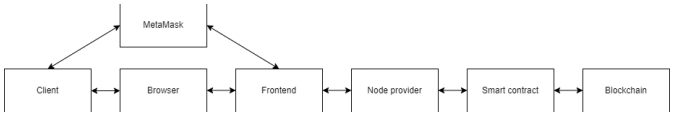

Fig. 1. MetaMask interaction in Web3

In Fig. 1., it can be seen that if the client wants to execute a transaction on the blockchain, the client can use MetaMask which connects the client and the frontend. Then, the data is passed to the node provider, such as Infura and Alchemy, which are the infrastructure for managing the nodes connected to the blockchain. After that, the data will be forwarded to a smart contract, which can be analogized as a backend, which functions to execute program commands that meet certain conditions that will change the state of the blockchain.

B. Gnosis Safe

Gnosis Safe is a system for managing multisignature wallets built by Gnosis. This system consists of several features: safe contracts, safe UI, safe transaction service, and safe apps. Unlike MetaMask which uses EOA, Gnosis Safe operates entirely with smart contracts that can define access control rights and can increase security in conducting transactions [6]. Gnosis Safe is perfect for a group of people who want to collectively manage their cryptocurrency assets.

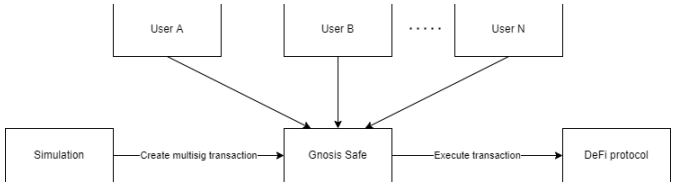

Fig. 2. Gnosis Safe interaction in Web3 [7]

In Fig. 2., Gnosis Safe as a multisignature wallet system will process a transaction if the proposed transaction is approved by the majority of users in the group. If the transaction has been approved by the majority, the transaction will be executed.

C. Developer DAO (DDAO)

Developer DAO (DDAO) is an organization/community that aims to educate and accelerate Web3 education. DDAO was founded by Nader Dabit in 2021. DDAO has four values, namely transparency, diversity and inclusion, responsibility, and kindness and empathy. With its uniqueness of being transparent and diverse, DDAO is made up of various people around the world who collaborate with each other on Web3. In accordance with general DAO rules, committees are selected from a group of people who are highly trusted and selected by the community to manage the wallet. Based on the results of the deliberation, DDAO uses Gnosis Safe as its multisignature wallet platform to manage treasuries with a 6/3 threshold [8]. 6/3 threshold means that of the six members responsible for the committee, three of whom need to sign a transaction in order for cryptocurrency assets to exit the treasury.

III. Design and Implementation

There are several important things that need to be considered in the development of a system to manage electronic currency assets, namely that each transaction cannot be falsified (transaction data needs to be immutable) and the system must also guarantee the integrity and transparency of transaction data.

The plan to develop a system for managing electronic currency assets in the DAO has several alternative solutions based on the wallet architecture used and the type of blockchain as the system building platform.

There are two alternative blockchain-type solutions as a platform for the development of this multisignature wallet system, namely Ethereum and Hyperledger Fabric. In DAO, it is important to have aspects of freedom and transparency, every transaction made needs to be visible to the public. The main difference between Ethereum and Hyperledger Fabric is the permission type. On Ethereum, the blockchain is permissionless,

meaning that any individual can access the blockchain. Whereas in Hyperledger Fabric, only certain groups/individuals who are given access can access the blockchain.

Ethereum is a platform for creating B2C businesses and decentralized applications. Ethereum was created to be able to run smart contracts on EVM and build decentralized applications that are used by the general public. Meanwhile, Hyperledger Fabric is a platform for creating B2B business and cross-industry applications. Hyperledger Fabric helps this industry or business to collaborate with the developers who build Distributed Ledger Technology (DLT). Ethereum is also a public blockchain network where every transaction is completely transparent and anyone on the Internet can view and access the details of the transaction. While Hyperledger Fabric is a blockchain network with limited access, only authorized organizations/individuals (that have an authorization certificate) can see all transactions on the network.

According to the information above, to support the transparency of data so that every transaction flowing through the DAO can be publicly verified by everyone, Ethereum as a permissionless blockchain is a more appropriate blockchain choice compared to the permissioned Hyperledger Fabric.

The smart contract based system for managing finances in the DAO includes a wallet. There are several alternative wallet solutions based on their respective uses, namely externally owned account (EOA) software wallet and smart contract account, which consists of single-signature (singlesig) smart contract wallet and multisignature (multisig) smart contract wallet.

Fig. 3. Architecture of externally owned account (EOA) and contract account

Externally owned account (EOA) wallet allows users to be given a public key (0x…) which can be used to send and receive electronic currency. To be able to enter the EOA wallet, users are given a private key in the form of a seed phrase which consists of twelve random words and needs to be backed up because knowing the seed phrase means having full control over the wallet account. In contrast to an externally owned account (EOA), a contract account consists of a single-signature (singlesig) smart contract wallet and a multisignature (multisig) smart contract wallet. Single-signature (singlesig) smart contract wallets are similar to EOA wallets in that there is a public key and a private key. However, the difference lies in the account represented by the address tied to the wallet. In singlesig, there is a contract account consisting of codes that can be set to add some additional features, such as recovery, 2FA, and so on. Multisignature (multisig) smart contract wallets are similar to single-signature wallets but require a minimum number of signatures to approve a transaction before it can be executed. Multisig ensures that no one can commit a crime.

In DAO, the multisignature (multisig) wallet system is very suitable to be used to ensure verification from several financial managers to sign every transaction you want to make and increase security against criminals. By using a contract account that is executed through the code it contains (in the form of a smart contract), a multisignature wallet can be implemented.

Non-functional requirements that must be met, in this case by a multisignature wallet system, are availability and integrity, as represented in Table I.

TABLE I. NON-FUNCTIONAL REQUIREMENTS

ID	Parameter	Requirement
NF-01	Availability	The system as a whole can still run even though there are some nodes that are not functioning properly.
NF-02	Integrity	The system can prevent modification of transaction data in any form.

The architecture of the multisignature wallet system is divided into three parts: the user interface (frontend), the part that interacts with the smart contract (backend), and the system part that executes orders based on input from the client (smart contract) which is directly related to the blockchain. The general architecture of the multisignature wallet system is represented in Fig. 4., which describes the relationship between these three sections.

The system flow process generally starts with the user interacting with the frontend. The user will send a request to the frontend through a browser. Then, the frontend will interact with the provider and backend nodes, depending on each function to be operated. In addition, there is MetaMask, an EOA wallet in the form of a browser extension that functions as a link between the browser and the frontend to handle accounts and connect users to certain blockchains. The provider node will then interact with the Ethereum blockchain, which consists of smart contract code that is executed depending on the particular function being called. Every command execution in the smart contract, the Ethereum Virtual Machine (EVM) needs to broadcast any state changed in the blockchain to other nodes so that every miner/node has the same state. This state change is in the form of adding a new block to the blockchain.

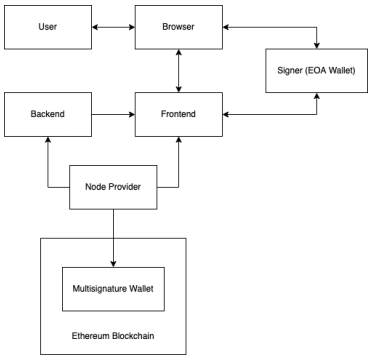

Fig. 4. Architecture diagram of multisignature wallet system

A. Frontend

The frontend functions as a system interface that interacts directly with the user. In this section, several processes will be carried out, namely receiving and validating input from the user, sending requests to the backend or provider nodes, interacting with MetaMask, and displaying the data results obtained from the blockchain. The frontend is made in the form of a web application to facilitate access by DAO members.

According to Fig. 4., the frontend interacts with four entities: browser, wallet, provider node, and backend, making it a component/part that connects the entire multisignature wallet system. Judging from the communication method, the communication between the frontend and the backend uses a REST API; communication between frontend and provider nodes using JSON RPC; and communication between the frontend and MetaMask using a browser extension.

With a system design as shown in Fig. 4., the frontend requires a strong and reliable client-side programming language. Therefore, we consider TypeScript which has several advantages over JavaScript, namely static/strong typing and pre-compilation. Meanwhile, React will be used as a library to support the development of an interactive frontend interface.

B. Backend

The backend section functions as an intermediary between the frontend and the provider node which will then be forwarded to the blockchain. In this section, several processes will be carried out according to system requirements, namely receiving requests from the frontend, performing actions according to requests from the frontend, and translating these actions to then be able to execute smart contract commands that are communicated through the provider node.

According to Fig. 4., the backend interacts with two entities: the frontend and the provider node. In terms of the process/way of communication, communication between the frontend, backend, and node provider is about the need to deploy smart contracts when a user wants to create a new multisignature wallet.

With a system design as shown in Fig. 4., the backend, just like the frontend, uses the TypeScript programming language in its development. Additionally, the backend uses a dedicated development environment for Ethereum, namely Hardhat. By using Hardhat, the system development process becomes easier because Solidity files can be run locally.

C. Smart Contract

Smart contract in this multisignature wallet system are divided into several functions, depending on the operations you want to perform. There are owners, minimum number of confirmations, and several other variables that function as state variables (variables stored in the contract). In addition, there are functions to submit transactions, add confirmations, execute transactions, request transaction cancellations, approve transaction cancellations, execute transaction cancellations, and several functions to retrieve state variables in this contract. These functions can only be operated under certain conditions, for example only the owner of this contract can call the function, function calls can only be made if the Transaction struct has been created, function calls can only be made if the transaction has been submitted, and various other conditions. In addition, there are functions for hashing, verifying, and recovering to support the digital signature validation process.

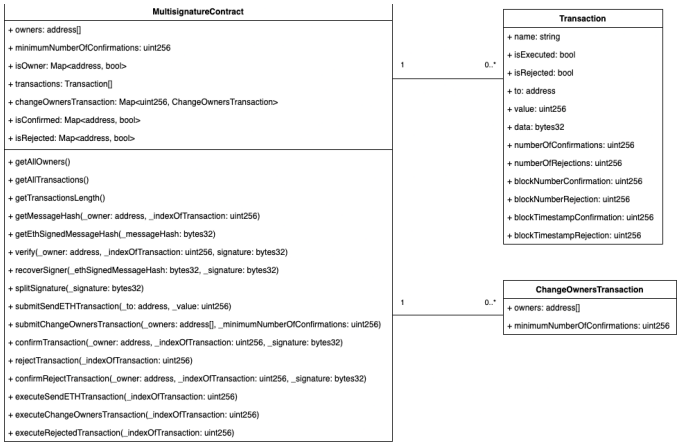

Fig. 5. Class diagram of multisignature wallet

Based on Fig. 5., there are three entities: MultisignatureContract, Transaction, and ChangeOwnersTransaction. MultisignatureContract contains some public attributes and functions. Transaction and ChangeOwnersTransaction contain public attributes. The relation of MultisignatureContract with Transaction and MultisignatureContract with ChangeOwners can be seen in the figure. MultisignatureContract can have 0 or more Transactions or ChangeOwnersTransactions. In addition, there are several functions to perform signing and verifying in MultisignatureContract, namely getMessageHash, getEthSignedMessageHash, verify, recoverSigner, and splitSignature. To submit a transaction, there are three functions depending on the transaction type (sending cryptocurrency, changing owner addresses and minimum confirmations, or submitting a transaction cancellation), namely submitSendETHTransaction, submitChangeOwnersTransaction, and rejectTransaction. To approve the transaction, i.e. counter the confirmations, confirmTransaction and confirmRejectTransaction are used which will confirm the transaction to be executed and confirm the transaction to be canceled, respectively. Finally, to execute the transaction, there are three functions depending on the transaction type as well (sending cryptocurrency, changing owner addresses and minimum confirmations, or requesting transaction cancellation), namely executeSendETHTransaction, executeChangeOwnersTransaction, and executeRejectedTransaction.

IV. Experiment and Analysis

Several testing methods are discussed on the multisignature wallet system, starting from testing certain transaction cases and

non-functional testing which includes testing data integrity and system availability.

Testing of certain transaction cases includes all tests of the types of cases that may occur in the multisignature wallet system. This aims to ensure that every type of case that may occur in the multisignature wallet system has been handled properly. This type of testing is declared successful if the testing of the implementation of the multisignature wallet system is in accordance with the expectations of the transaction handling flow of the multisignature wallet system. There are three types of cases tested on the multisignature wallet system, namely testing certain transaction execution cases, testing on-chain rejection transaction cases, and testing non-owner address cases.

In the first transaction case test, it is tested whether the transactions should be executed sequentially.

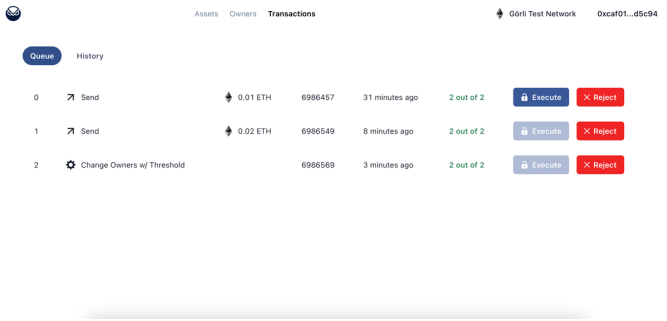

Fig. 6. First transaction case test: sequential transaction execution

As in Fig. 6., it can be seen that each authorized user to manage this multisignature wallet system can only execute transactions with nonce 0, which is the earliest transaction in the queue. It can be seen that the user cannot execute transactions with nonce 1 and 2.

In the second transaction case test, it is tested regarding transaction rejection.

The rejection transaction is used when one of the users in the DAO wants to change an action that has already been submitted. According to Fig. 7., the first transaction in the queue has been submitted for rejection by someone in the DAO and suppose this rejection is to be executed. Therefore, this transfer of 0.01 ether will not take place. However, the transaction will still be executed and it will be in the History tab.

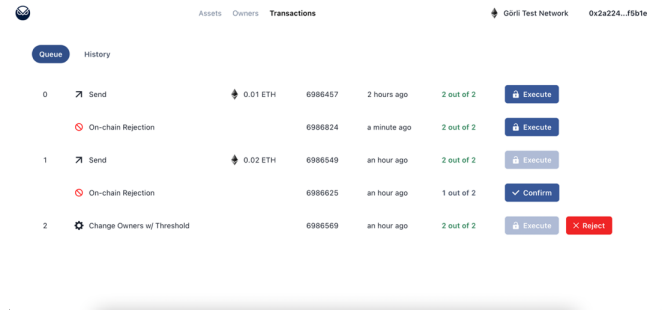

Fig. 7. Second transaction case test: on-chain rejection

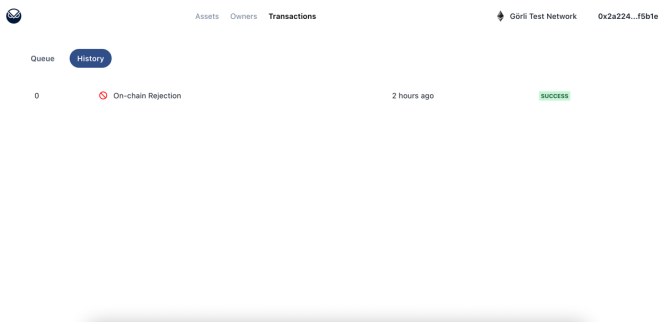

Fig. 8. Second transaction case test: on-chain rejection

After the rejection transaction is executed, according to Fig. 8., this transaction is said to have been completed and no action can be taken against it. With this, it can be seen that the system passed the on-chain rejection transaction case test.

In the third transaction case test, it is tested regarding transaction rejection.

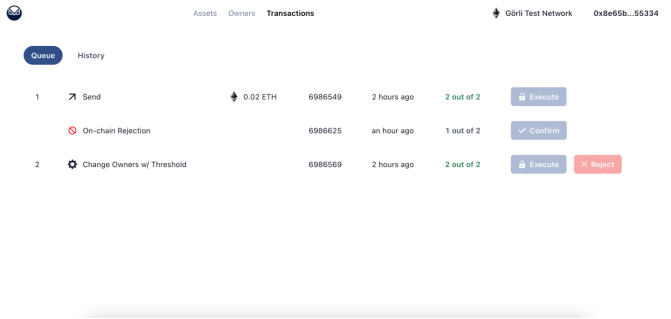

Fig. 9. Third transaction case test: non-owner address

As in Fig. 9., it can be seen that the non-owner address is not allowed to perform any actions. This user can only see transactions, which include transactions that are in the queue and transactions that have been executed in the history. With this, it can be seen that the system passes the non-owner address case test.

Non-functional testing includes aspects of integrity and availability of the multisignature wallet system. Integrity testing is carried out to ensure that the data contained in the blockchain cannot be changed, or in other words immutable. Meanwhile, availability testing is carried out to ensure that the blockchain can still run properly and properly even though one or more nodes are not running.

Integrity and availability testing can be declared successful if each test result meets the standard criteria in accordance with what has been described in the points in the non-functional requirements.

In integrity testing, an experiment will be carried out to change data that has been persistent on the blockchain. Integrity testing includes changing transaction data. Attempts to change transaction data cannot be carried out on the smart contract side because the user can only perform actions defined as functions in the smart contract. Changing transaction data in a block will

change the signature of the block so that the change request will be easily considered invalid and rejected by miners.

In availability testing, according to the current multisignature wallet system, the Ethereum goerli test network is used, where there are real miners around the world in such a way that they build the original Ethereum network, only it does not use the original cryptocurrency.

Fig. 10. Non-functional testing: integrity testing

In Fig. 10., it can be seen that there are 17 active nodes out of a total of 18 nodes, or in other words, there is one node that is down or not running. However, Ethereum as a whole is still doing well and can accept transactions that are submitted for validation.

In accordance with the implementation and testing of the multisignature wallet system, Ethereum as a public blockchain is suitable to be used as a platform of this system because every transaction made by the DAO can be verified publicly, increasing the transparency of the system. Any flow of funds through the DAO will be easily tracked and known by the public.

The proposed multisignature wallet system architecture is also possible to implement and fulfill every certain transaction case and non-functional requirement of the system. Regarding testing, there are two types of tests performed: specific transaction case tests and non-functional tests.

In certain transaction case testing, which includes sequential transaction execution cases and on-chain rejection transaction testing cases, it can also be seen that every possible case went well. In non-functional testing, which includes integrity and availability, integrity testing cannot be carried out from the smart contract side so that testing cannot be carried out directly, while availability testing can be carried out and meets the availability standard metrics, namely the system can still run well even though there is one or some dead nodes, as evidenced by 17 of the 18 running nodes.

From the tests in various aspects that have been carried out, the multisignature wallet system has several constraints, namely the system runs on a secure network and the system is deployed on a network built from original miners (Goerli test network) which results in the system not being simulated locally.

With this, it can be concluded that the overall test in general went well. All possible cases have been tested, along with non-functional requirements.

V. CONCLUSION

The blockchain-based multisignature wallet system has successfully guarantee the integrity, availability, and transparency. The system that has been designed and implemented has also met the standards of certain transaction cases and non-functional testing.

Ethereum as a public blockchain is very suitable to be used as a platform for the development of a multisignature wallet system. With the nature of the public blockchain, everyone can participate in the blockchain network. Everyone does not need to ask for special access permissions to enter the network.

The proposed multisignature wallet system already has the main functionalities needed to perform certain tasks, such as sending cryptocurrency that requires majority of signers, managing the logic of queued transactions, and handling change of owners and minimum number of confirmations. Nevertheless, the system needs to be tested for scalability to ensure the system is ready for use in a production environment that involves many users.

ACKNOWLEDGMENT

The author would like to express gratitude to Dr. Ir. Rinaldi Munir, M.T., as his advisor on author's thesis at Bandung Institute of Technology (ITB) for guiding and providing advice, that helped the author to successfully finish this paper. The author would also like to thank the parents of the author, who have provided support and allowed the author to finish his study. Lastly, the author would like to thank the friends of the author, who have helped the author on the writing of this paper.

REFERENCES

[1] Wang, S., "Decentralized Autonomous Organizations: Concept, Model, and Applications", *IEEE Transactions on Computational Social Systems*, 2019.

[2] Volhov, D., "*Wallet*", 2020. [Online]. Available: https://academy.binance.com/en/glossary/wallet

[3] Han, J., "An Efficient Multi-Signature Wallet in Blockchain Using Bloom Filter", *36th Annual ACM Symposium on Applied Computing*, 2021.

[4] Piore, A., "How Blockchain Technology Could Help Us Take Back Our Data from Facebook, Google, and Amazon", 2018.

[5] Leising, M., "Metamask's Blockchain Mobile App Opens Doors for Next-Level Web", 2020.

[6] Sawinyh, N., "Gnosis Safe: Smart Contract-based Multisig Wallet", 2019. [Online]. Available: https://defiprime.com/gnosis-safe

[7] Ulfo, F., "Multisig Transactions with Gnosis Safe", 2021. [Online]. Available: https://medium.com/gauntlet-networks/multisig-transactions-with-gnosis-safe-f5dbe67c1c2d

[8] Kempster, W., "Initial Developer DAO Treasury Setup", 2021. [Online]. Available: https://forum.developerdao.com/t/initial-developer-dao-treasury-setup/549