Jurnal Masyarakat Informatika, 15(1), 2024
DOI:10.14710/jmasif.15.1.62835
ISSN: 2777-0648

Hybrid ERC20 Ethereum Blockchain Multisignature Wallet 3of3 with Withdrawal Pattern, External Effects, and Mutex as Single Key and Reentrancy Mitigation.

Jason Al Hilal Sabda Dewa*, Indra Waspada, Priyo Sidik Sasongko
Department of Informatics, Universitas Diponegoro, Semarang, Indonesia
** Corresponding author: jasonalhilal@gmail.com*

Abstract
In the rapidly evolving era of Decentralized Finance (DeFi), the convergence of Blockchain technology with intermediary-free financial services has forged a revolutionary landscape. However, this progress has been accompanied by critical challenges, notably the Single Key Risk and reentrancy attack threats against ERC20 smart contracts in private Ethereum Blockchain. This research formulated a proactive approach and implemented an innovative solution by embodying Reliable Decentralized Finance through the deployment of a 3-of-3 Hybrid Multisignature Wallet system with Withdrawal Pattern, External Effects, and Mutual Exclusion in the form of a Decentralized Application (DApps). The system not only applied withdrawal patterns but also integrated external effects and the principle of mutual exclusion to enhance the security of smart contracts. The system development methodology was executed comprehensively using Agile Software Engineering, encompassing the development of both smart contracts and external applications (decentralized applications). Testing was conducted using Ganache EVM (Ethereum Virtual Machine) connected to the Hot Wallet Metamask as an Externally Owned Account (EOA) for transaction signing. Valid results were obtained from comprehensive testing against the system's functional requirements, affirming the system's success in managing Single Key Risk and preventing reentrancy attacks, providing a reliable and concrete solution.

Keywords *: Decentralized Finance, Multi-Signature Wallet, Reentrancy Attack, Single Key Risk, Smart Contract Security*

1 Introduction

In the ever-evolving digital era, Blockchain technology has emerged as the backbone of decentralized financial (DeFi) innovation. DeFi, a system free from traditional intermediaries, paves the way for open and inclusive access to a myriad of financial services. Blockchain, as an immutable digital ledger, offers unparalleled transparency and security through its consensus mining process to verify each action within its domain [1]. In this landscape, Ethereum stands out as a trailblazer, being the first platform to support smart contracts, thus enabling the development of decentralized applications (dApps) within the DeFi ecosystem [2].

ERC20 smart contracts on Ethereum play a pivotal role in creating token interoperability and laying a consistent foundation for crypto token-related services development [3]. With Blockchain's capability to provide unmatched security and immutability, ERC20 smart contracts ensure the integrity

Received: 25 March 2024; Revised: 9 May 2024; Accepted: 13 May 2024; Published: 31 May 2024.

and safety of crypto assets. However, the management of tokens within crypto wallets introduces a single key risk, particularly when dealing with ERC20 smart contract withdrawals.

Addressing these vulnerabilities necessitates a more sophisticated security approach. The use of a 3-of-3 multisignature (multisig) wallet emerges as a potential solution, where at least three private keys are necessary to authorize a transaction [4]. This approach not only enhances security but also minimizes the single key risk that could grant full access to digital assets. This mechanism ensures that no single entity can unilaterally control the digital assets, creating a more secure environment for managing and executing transactions [4]. Moreover, implementing a secure and verified withdrawal pattern is crucial, with stringent verification requirements and comprehensive security checks to ensure transaction integrity and reliability in the DeFi ecosystem.

By integrating a multisignature wallet with a withdrawal pattern, along with mechanisms to handle external effects and mutual exclusions (mutex), DeFi can substantially improve the security of digital asset management. Withdrawal patterns enforce a controlled and secure method for transferring funds out of the wallet, ensuring that every transaction undergoes stringent verification and security checks [5]. The handling of external effects, crucial for maintaining the integrity of transactions affected by external state changes [5], and mutex for preventing reentrancy attacks, where an attacker could potentially drain funds by recursively calling the withdrawal function, are fundamental to safeguarding the assets within the DeFi ecosystem [5]. These measures offer additional protection against potential risks, such as reentrancy attacks that could compromise the security of ERC20 smart contract executions. Continuously developing and implementing innovative security solutions, the DeFi ecosystem can build a more robust foundation for broader growth and acceptance in the future.

2 Literature Review

2.1 Related Research

A study conducted by Jongbeen Han, Mansub Song, Hyeonsang Eom, and Yongseok Son in 2021 entitled 'An Efficient Multi-signature Wallet in Blockchain Using Bloom Filter' proposed an efficient multi-signature wallet for blockchain using the threshold elliptic curve digital signature algorithm (T-ECDSA) and Bloom Filter [4]. The system successfully guarantees transaction integrity as well as improves transaction validation efficiency and reduces transaction size without changing the blockchain protocol. The wallet uses T-ECDSA to process multiple signatures as a single signature for higher validation efficiency and uses Bloom filter to maintain privacy and reduce storage requirements by effectively managing transaction participant data.

Another study conducted by Shahriar Ebrahimi, Parisa Hasanizadeh, Seyed Mohammad Aghamirmohammadali, and Amirali Akbari in 2021, titled "Enhancing Cold Wallet Security with Native Multi-Signature Schemes in Centralized Exchanges," found that the use of multi-signature schemes in Cold Wallets could significantly improve asset data security by addressing the risk associated with storing private keys on a single device [6]. Similar to the first study, this research also aimed to enhance security in the management of digital assets on the blockchain, with a particular focus on the implementation of multi-signature wallets in Cold Wallets.

Research [4] and [6] shows that using multi-signature wallets can improve the security of digital wallets on the blockchain. Using multi-signature wallets can reduce the security risks associated with storing private keys on a single device and maintain transaction integrity without changing the existing blockchain protocol, making it more secure and efficient than conventional methods.

2.2 Blockchain

Blockchain is a decentralized database that employs independent nodes to store and retrieve data [7]. This technology links data blocks sequentially in a distributed ledger. Each block contains various content, including a "hash," which is a unique identifier of the block itself. The hash identifies and connects this block to all other blocks, both preceding and succeeding it. Therefore, Blockchain can be defined as a collection of blocks containing transaction data that are linked (chained) and arranged in sequence. Blockchain can be regarded as a digital data storage system where every new or last block is connected and contains the hash information (an alphanumeric code representing a word, message, or data) from the previous block [8]. Figure 1 and Figure 2 show that each block refers to the previous one and so on, forming a chain. Blockchain implements asymmetric cryptography to maintain the security of users and the consistency of the ledger on a distributed network.

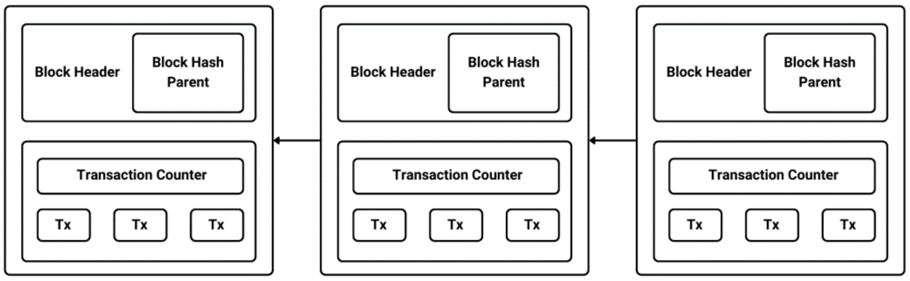

Figure 1 Illustration of Blockchain Consisting of Continuous Blocks

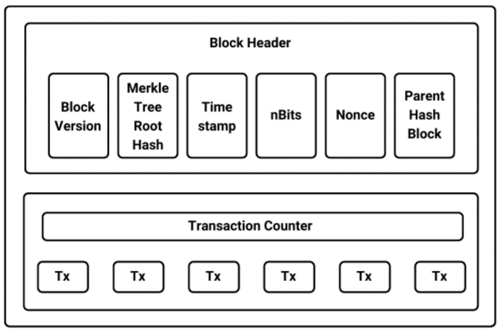

Figure 2 Block Structure in Blockchain [9]

An important aspect of Blockchain is its ability to maintain transparency while ensuring the privacy of its users [8]. Through the use of public key cryptography, each user can have a uniquely encrypted digital identity. Transactions conducted within the Blockchain are verified by the network through a process known as mining, which permanently and irreversibly adds the transactions to the public ledger as shown in Figure 3. This enables secure, anonymous transactions that are easily verifiable by all users. This technology not only changes the way digital transactions are conducted but also opens up potential for new applications in areas such as supply chain management, identity authentication, and more.

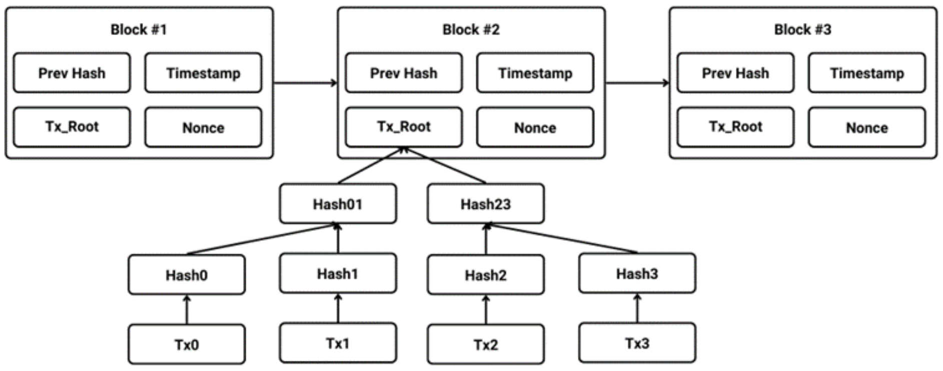

Figure 3 Blockchain Structure

2.3 Smart Contract

Smart contracts, a key implementation of Blockchain technology, aim to establish consensus among multiple parties based on the consensus mechanism used and are applied in the form of scripts or code serving as business logic in Blockchain-based systems or applications [9]. Smart contracts can be tailored to specific needs and actively utilized across any Blockchain platform like Ethereum, using a programming language called Solidity. Designing smart contracts and executing transactions without third-party involvement using Ether units on the Ethereum network should ideally enhance transparency and trust in Blockchain applications for users.

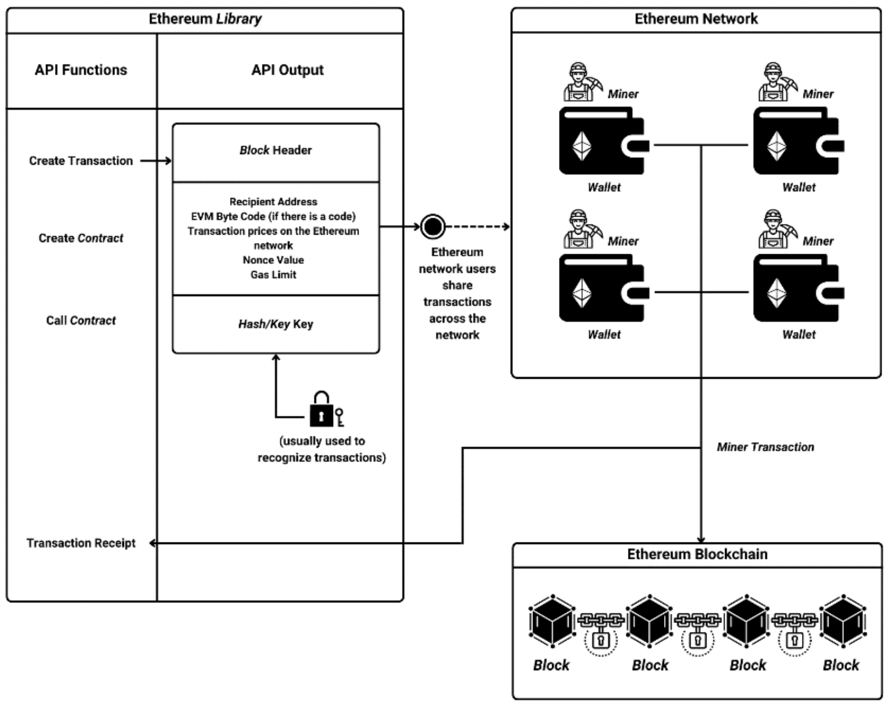

Figure 4 Scheme of Ethereum Network Operation

The application of Blockchain architecture to Ethereum involves various interacting components working together to fulfill their functions. Figure 4 shows ethereum's components include the Ethereum Virtual Machine (EVM), miners, blocks, mining, Ether, and gas [10]. The Blockchain network consists of nodes owned by miners and some nodes assisting in executing smart contracts and transactions. Smart contract execution and transaction processing occur on the Ethereum Virtual Machine (EVM), a Turing-complete device operating on the Ethereum network. EVM serves as a storage place for smart contracts, aiding Ethereum's expansion by encapsulating functionality or business logic in Blockchain-based application development. Smart contracts are executed as part of transactions and mining processes, with Ether serving as the cryptocurrency unit within the Ethereum network for transactions and smart contract functions.

2.4 ERC20

ERC20 stands for Ethereum Request for Comment 20. ERC20 was the first standard proposed by Ethereum in 2015 to regulate token creation on the Ethereum platform [11]. This standard defines functions that must exist within smart contracts used to create tokens, allowing various tokens to interoperate within the Ethereum ecosystem. Tokens following the ERC20 standard are the most commonly found digital assets within the Ethereum network and play a significant role in Blockchain projects, including Initial Coin Offerings (ICOs), Decentralized Finance (DeFi), and cryptocurrency exchanges.

ERC20 characteristics apply to payment transactions and are transferable, with all transaction histories traceable despite each token's identical and unique code, requiring an identified total supply of ERC20 tokens to ensure the ecosystem is aware of the circulating token amount [11]. Smart contract tokens operate by facilitating token transfers across accounts, checking the total number of created tokens, and verifying token balances at a specific address, similar to conventional digital currencies following the ERC20 standard. Decentralized Ethereum applications and Ethereum wallets can access tokens through this standard interface as ERC20 defines a constructor for token contracts to establish and initialize contract status. Standard functions define the fundamental transactional and token management operations of smart token contracts, and the interface for these standard functions is as follows [12].

2.5 Single Key Risk and Reentrancy Attack

Single key risk refers to the risk associated with using a single private key in the Blockchain system. The private key is fundamental to the cryptographic security of the Blockchain network, used to identify the owner of crypto assets and authorize transactions within the network. Single key risk occurs when a single private key controls access to all crypto assets in a wallet address or to control smart contracts as shown in Figure 5. If this key is compromised, lost, or misused, the owner may lose crypto assets or control over smart contracts. This vulnerability arises from an attacker's ability to sign permanent and immutable transactions [4]. Therefore, it is crucial to safeguard the security of private keys and mitigate potential misuse.

Reentrancy is a security attack that can occur in smart contracts on Blockchain platforms [13]. In Blockchain, reentrancy refers to a situation where a smart contract calls a function or external contract, and then the contract receives a callback from that external contract before the execution of its original function is complete, creating potential vulnerabilities to attacks. Attackers can exploit this opportunity to disrupt the execution of smart contracts illegitimately or steal crypto assets. In a reentrancy attack, the attacker leverages the execution sequence of smart contracts. A notable example is the 2016 DAO project attack, where hackers stole $60 million worth of Ether by exploiting the reentrancy vulnerability in TheDAO smart contract. This weakness was associated with the fallback mechanism in Solidity, allowing misuse of the call.value() function, which consumes contract gas for external calls. This incident underscores the importance of smart contract security in blockchain application development, leading to the first hard fork in Ethereum's history in response to financial losses and the need for strict security standards implementation and thorough testing to prevent potential vulnerabilities.

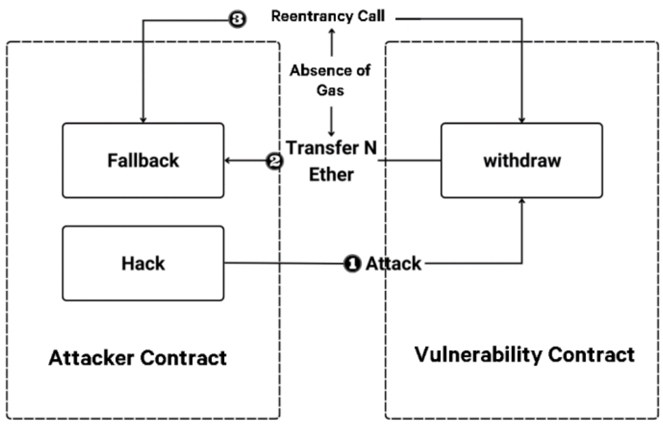

Figure 5 Principle of Reentrancy Vulnerability

2.6 Multisignature Wallet

Multisignature, often referred to as 'multi-sig', is a form of technology that provides additional security for cryptocurrency transactions. Essentially, this means that spending cryptocurrency requires more than one approval or signature for the transaction to be executed. A multi-signature wallet is a type of wallet that offers additional security by requiring unique dual signatures (hence multi-signature) to authorize and execute a transaction [4]. Traditional cryptocurrency wallets, or single-sig, contain token addresses, each with one associated private key that grants the key holder full control over the funds. With a multi-signature wallet, users can have token addresses with three or more associated private keys, requiring two of those keys for token usage as shown in Figure 6. Single-sig wallets (private keys) provide direct access to user funds, providing token ownership and requiring the private key to transact alongside the public key. If the private key is lost, all data will be lost, with no way to recover it. Distributing access to the wallet through multiple keys (multi-signature) is a better security measure.

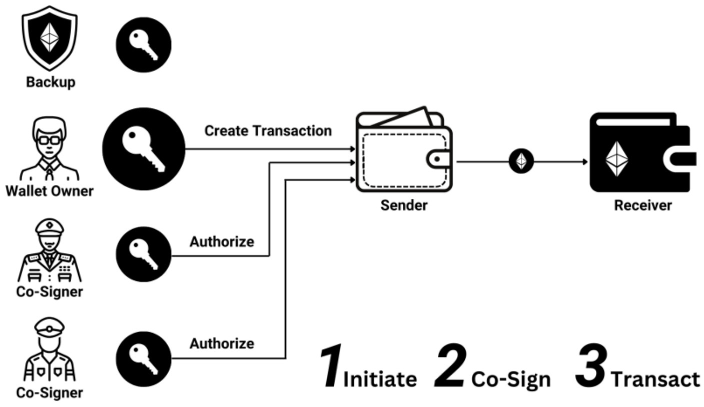

Figure 6 Operation of Multi-Signature Wallet

2.7 Withdrawal Pattern, External Effect, and Mutex

Withdrawal pattern in Blockchain refers to a method used to transfer or withdraw cryptocurrencies from one address or smart contract to another within the Blockchain network [5]. Figure 7 shows that the pattern is designed to mitigate the risk of attacks such as reentrancy attacks. The main objective of the withdrawal pattern is to allow cryptocurrency owners to send or withdraw assets as needed, such as transferring to a personal wallet, exchanges, or other smart contracts. All transactions involving the withdrawal pattern are recorded transparently on the Blockchain, allowing anyone to verify them.

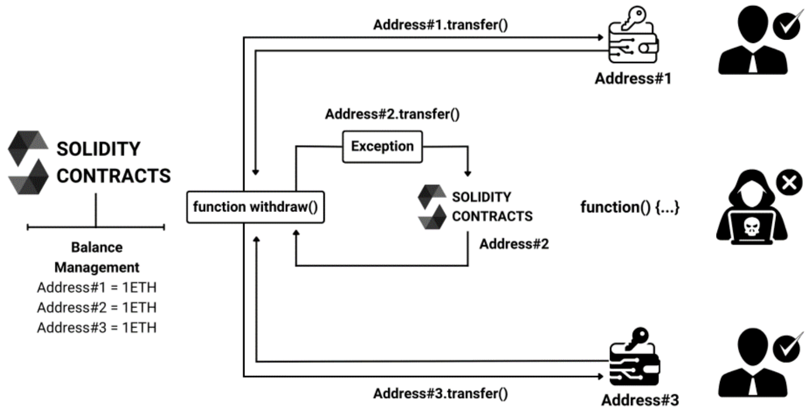

Figure 7 Operation of Withdrawal Pattern

External Effects Examination is a crucial principle in Ethereum smart contract development aimed at ensuring the security and integrity of smart contracts [14]. This principle refers to the practice of checking and ensuring that operations that can result in changes within the contract (such as fund transfers or status changes) are only executed after external operations are completed successfully. Thus, examining external effects reduces the risk of reentrancy attacks, where malicious external contracts attempt to exploit unfinished operations to harm running contracts.

The concept of Mutual Exclusion (Mutex) is a key principle in the development of distributed systems and multithreading aimed at preventing access conflicts to shared resources by multiple processes or threads simultaneously [15]. This principle ensures that only one process or thread has exclusive access to the resource at any given time, avoiding situations that may lead to data inconsistency or deadlock.

3 Research Method

Research methodology serves as a crucial foundation to ensure the quality, security, and reliability of the developed systems. This method encompasses approaches and steps utilized throughout the research process. Below are the details of the research methodology adopted from the study "An Agile Software Engineering Method to Design Blockchain Application" [16], including steps such as Identity Resource, Identity Requirements, Design System (External and Smart Contract), Implementation System, and Integrate and Test.

In the process of developing Blockchain-based systems, the first step involves identifying the primary resources required through observing current industry business processes. The subsequent stage is to determine system specification details by dividing the system into smart contract subsystems and external systems. Following that, system design is carried out based on the previous specifications. The development stage is conducted to implement the system design, including selecting suitable platforms and programming languages. Integration of smart contracts and external systems is performed in the next stage, followed by testing to ensure the system operates according to the established specifications. By employing this method, this research aims to ensure the quality, security, and reliability of the developed systems, with a specific focus on the development of smart contracts and external systems within the context of blockchain technology as shown in Figure 8.

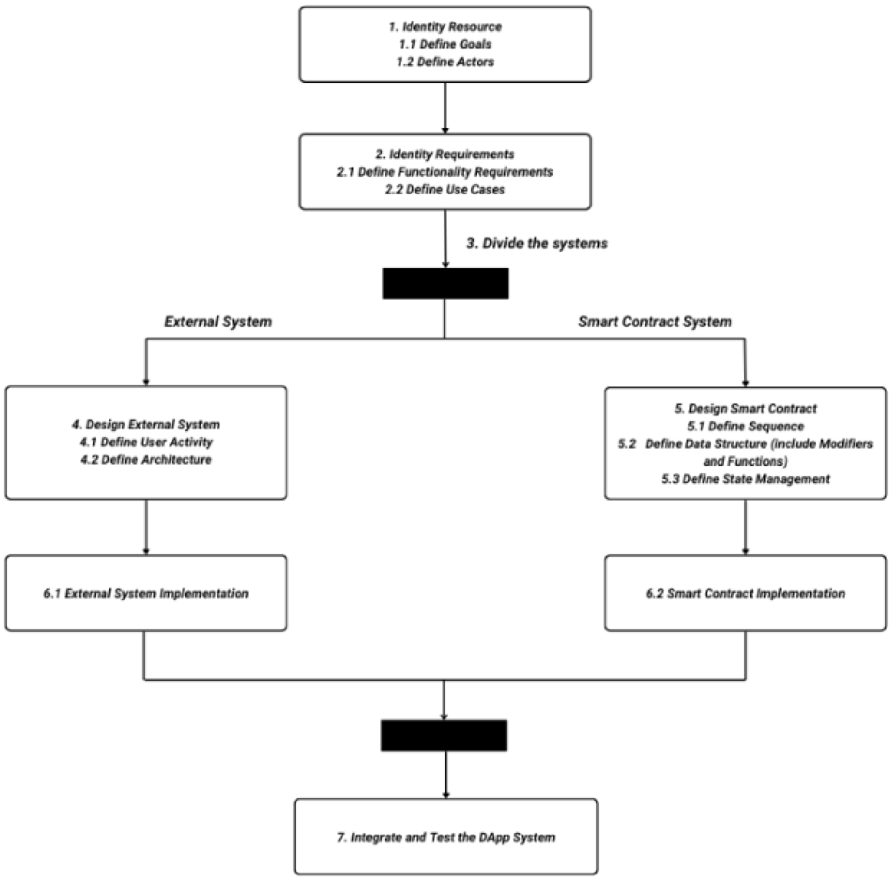
1. Identity Resource
1.1 Define Goals
1.2 Define Actors
2. Identity Requirements
2.1 Define Functionality Requirements
2.2 Define Use Cases
3. Divide the systems
External System
Smart Contract System
4. Design External System
4.1 Define User Activity
4.2 Define Architecture
5. Design Smart Contract
5.1 Define Sequence
5.2 Define Data Structure (include Modifiers and Functions)
5.3 Define State Management
6.1 External System Implementation
6.2 Smart Contract Implementation
7. Integrate and Test the DApp System

Figure 8 Flow of Research Methods

3.1 System Configuration and Functional Testing Plan

The Hybrid Multi-Signature Wallet System with Withdrawal Pattern, External Effects, and Mutual Exclusion is implemented in both hardware and software, as detailed in Table 1. The Hot Wallet is configured with the Ethereum Virtual Machine (EVM) on the system to provide user wallet access to the testnet Blockchain environment. The Hot wallet used is Metamask, with the EVM environment utilizing Ganache. After successfully adding the Ganache RPC network to Metamask, wallet accounts on Ganache can be imported for use with Metamask. This process can be done through the import account menu by entering the wallet's private key. In this case, 5 Ganache wallet accounts are imported into Metamask. These wallet accounts are used for testing purposes. All Ganache wallet accounts imported into Metamask are utilized. To determine whether the system can meet functional requirements, a functional system testing plan is created in Table 2.

Table 1 Specifications of Software and Hardware in System Implementation

No	Components	Specification
1	CPU	Intel® Core™ i7-9750H CPU @2.60GHz
2	Operation System	Windows 10 Home Single Language
3	Programming Language	Solidity (pragma ^0.8.15), Javascript
4	Framework	Next14, Moralis, Truffle, Openzeppelin
5	Blockchain	Ethereum
6	*Hot Wallet*	Metamask Wallet
7	Token Standard	ERC-20
8	EVM	Ganache
9	IDE	Remix
10	*Web Browser*	Google Chrome 113.0.5672.114

3.2 Scenario Solution for 3-of-3 Scheme Multisignature Wallet

Single key risk is a risk associated with using a single private key in a Blockchain system. In commonly used hot wallets, the majority employ a single key scheme where one wallet is held by a single account. In the event of a breach or unauthorized access by another account, there's a risk of risky transactions occurring, potentially depleting the currency balance in that wallet. To mitigate such risky transactions, a solution will be developed, namely the implementation of a multi-signature wallet with a 3-of-3 scheme.

Table 2 Functional System Testing Plan

Test Classes	Test Cases
Cryptocurrency Token	Creating a cryptocurrency token and token supply by applying the ERC20 standard token smart contract and performing token approval.
Multisig Factory Contract	Initializing a multisig wallet smart contract dynamically based on the owner's wallet address using a factory contract.
Token *Wallet*	Adding the currency token created based on the ERC20 token standard to the multisig wallet and viewing both the currency token and coin that can be used in the multisig wallet.
Multisignature Wallet Owner	Adding, removing, and viewing wallet owners in the multisig wallet.
Multisignature Wallet Transaction	Performing deposit, checking balances, and currency withdrawals by implementing withdrawal patterns, external effects, and mutual exclusion on the multisig wallet.
Multisignature Wallet Transfer Transaction	Creating transfer requests and canceling transfer requests to a specific wallet address using currency tokens or coins in the multisig wallet.
Multisignature Wallet Approval Transaction	Approving transfer requests to specific wallet addresses and executing currency transfers by implementing withdrawal patterns, external effects, and mutual exclusion on a multisig wallet using a 3-of-3 scheme.

This solution will implement a scheme where private key usage is distributed among three authorized accounts. In other words, in the event of a wallet breach and risky transactions, as mentioned above, with multiple parties involved, the wallet will not execute those transactions until it receives authorization from all authorized parties or accounts. With the 3-Of-3 scheme, if any of the authorized accounts do not approve the transaction, it will fail. The transaction will be executed only if all authorized accounts approve it.

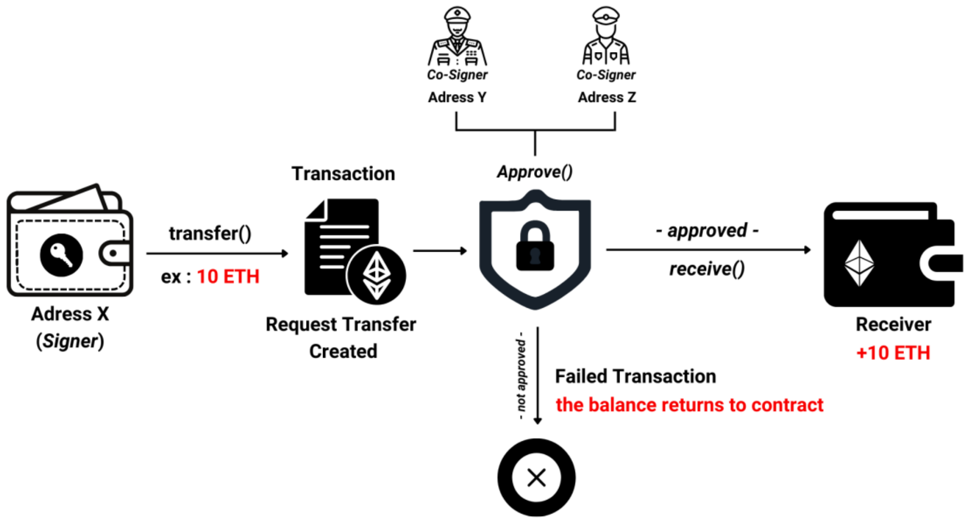

Figure 9 Multisignature Wallet 3-of-3 Scenario

Testing scenarios for the multi-signature wallet with a 3-of-3 scheme will also involve transferring 10 ETH from the sender's address (address X) to the recipient's address, where the owners of the multi-signature wallet are addresses X, Y, and Z as shown in Figure 9. When the transfer() function is executed, the 10 ETH balance will not be sent to the recipient's address until the two other owners besides address X approve it. This is because all balance transfer transactions require approval from the three owners as per the designed scheme (3-of-3). If the requirements are met, the 10 ETH balance will be sent; otherwise, it will not be sent, or it will return to the sender's contract address.

3.3 Scenario Solution for Withdrawal Pattern, External Effects, and Mutex Integration

Withdrawal Pattern, External Effects, and Mutual Exclusion (Mutex) are solutions to prevent reentrancy attacks (repeated transaction calls during the delay process) in smart contracts. This solution will be applied to the multi-signature wallet being created, resulting in a comprehensive hybrid system. Therefore, the solution to be implemented is the creation of middleware using withdrawal pattern, external effects, and mutual exclusion techniques in these transactions. This middleware is useful for limiting and preventing repeated calls in transactions within the smart contract, especially in the multi-signature wallet being created.

Testing scenarios for the implementation of withdrawal pattern, external effects, and mutual exclusion on the multisignature wallet with a 3-of-3 scheme will also involve conducting a withdrawal transaction worth 10 ETH from the sender's address to the contract address. The application of withdrawal pattern, external effects, and mutual exclusion will broadly serve as middleware to address the continuous invocation of transaction functions, in this case, withdrawal transactions, during the delay in transaction validation or mining processes. Figure 10 shows the middleware will continually check each transaction function call to validate whether the transaction is still delayed or not, in other words, whether the transaction has truly completed or not. Each called withdrawal transaction process will pass through this middleware for validation; if it meets the validation criteria, the 10 ETH balance will be sent, otherwise, if the transaction process is incomplete, the balance will not be sent.

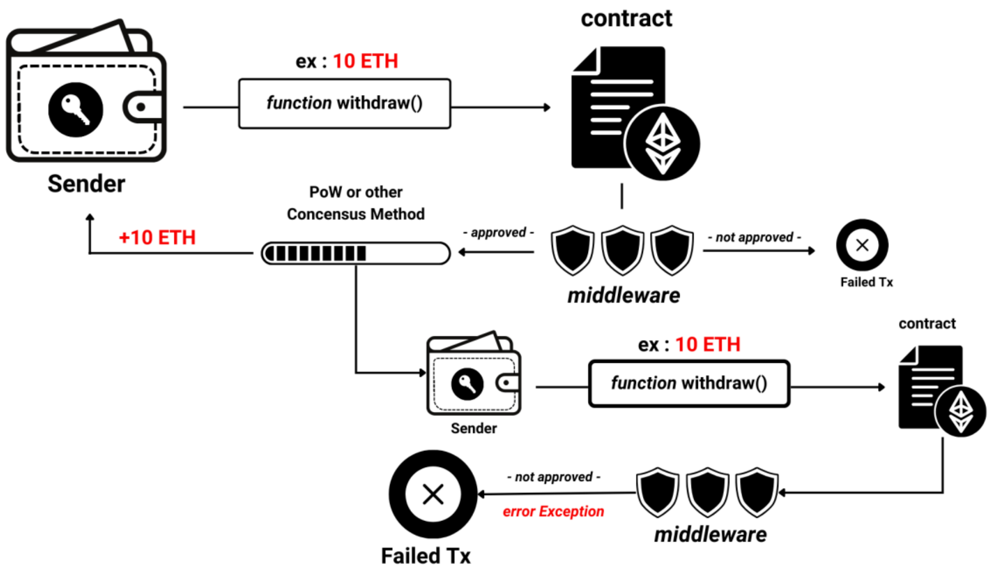

Figure 10 Scenario with Middleware (Withdrawal Pattern, External Effects, and Mutual Exclusion)

4 Result and Discussion

4.1 Smart Contract Implementation

The implementation of smart contracts in the system uses the Solidity programming language version 0.8.0 and above, and the OpenZeppelin framework is utilized for programming smart contracts on the Ethereum platform. The development of these smart contracts is carried out using an online Integrated Development Environment (IDE) that can be accessed directly through the browser at remix.ethereum.org. The use of the Remix IDE greatly facilitates the process of smart contract development. This is because the Remix IDE supports debugging, testing, and direct deployment of smart contracts on virtual Blockchain networks. The implemented smart contracts will be deployed on a private Blockchain network with the Ganache EVM Testnet, which can instantly validate transactions (Proof of Authority).

4.2 External System Implementation

Development of the external system referred to in the implementation of the multisignature wallet system involves the entire system outside the blockchain network. The external system developed in this final project is a client application serving as the interface system used to interact with the smart contract. The implementation of the client application in this final project is developed using the JavaScript programming language with the Next.js framework. Additionally, the client application uses the web3.js library and Moralis to interact with the smart contract on the blockchain via HTTP/Websocket.

4.3 Smart Contract and External System Integration

The integration process of the smart contract and external system is carried out using Truffle. Truffle is a development environment for compiling, testing, and asset pipeline for Ethereum blockchain-based systems. The steps involved in integrating the smart contract and external system include installing Truffle, initializing a Truffle project, configuring the blockchain address used, and the location of the client application requiring smart contract metadata in the Truffle configuration file. Additionally, configuring the deployment mechanism for each smart contract and compiling and deploying the smart contract.

4.4 Functional Testing Results

4.4.1 Cryptocurrency Token

This test was conducted to check how users, as wallet owners, can perform transactions related to Currency Tokens. Firstly, the wallet owner deploys the Token contract in Token.sol to create currency tokens and token supply by applying the ERC20 standard smart contract token. The created currency token is named "DIYUT" with the currency code "DYT" and a total supply of 10000000000000000000000000 Wei or 10000000 Ether.

This currency token will be used as a means of transaction payment besides using the Ether (ETH) currency coin as shown in Figure 11. Once the transaction is successfully executed, the transaction data will enter the Blockchain through the consensus process. After the currency token is successfully created, the wallet owner can perform approval transactions for the multisig wallet that will use the currency token. Approval of the multisig wallet for the token can be done by executing a transaction on the Token contract by entering parameters such as the multisig wallet address and the maximum token supply that can be used. The test results for transactions related to Currency Tokens are declared valid and the successful transaction results have been added to the Blockchain.

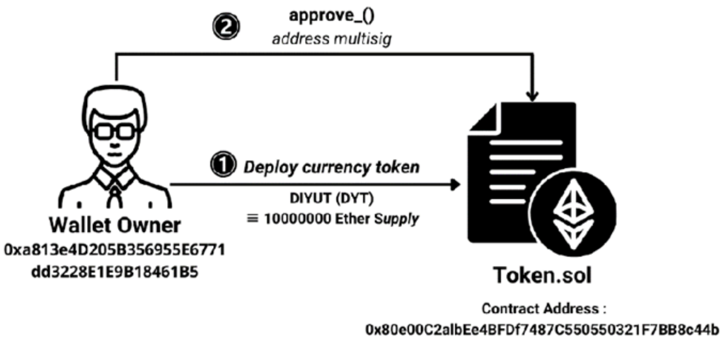

Figure 11 Currency Token Testing Mechanism

4.4.2 Multisig Factory Contract and Multisignature Wallet Owner

This test was conducted to check how users, acting as wallet owners, can perform transactions related to the Multi-Signature Factory Contract and Multi-Signature Wallet Owner. Figure 12 shows that the wallet owner can deploy the MultisigFactory contract in MultisigFactory.sol and MultisigTA in Multisig.sol and then perform transactions to initialize the multisig wallet smart contract dynamically based on the wallet address owner with the factory contract. After the multisig wallet instance is created, the wallet owner can also perform transactions to add, remove, and display new wallet owners on the multisig wallet smart contract instance. A check was conducted on the MultisigTA contract to ensure and display that the wallet owner's address has been added as an owner of the multisig wallet.

The test results show that the wallet owner successfully added another additional address to use the 3-of-3 approval scheme on the multi-signature wallet. In addition to adding wallet owners, transactions can also be performed to remove wallet owners who hold authority from the multisig wallet by the main wallet owner. Based on the test results of transactions related to the Multi-Signature Factory Contract and Multi-Signature Wallet Owner, the test is declared valid and the successful transaction results have been added to the Blockchain.

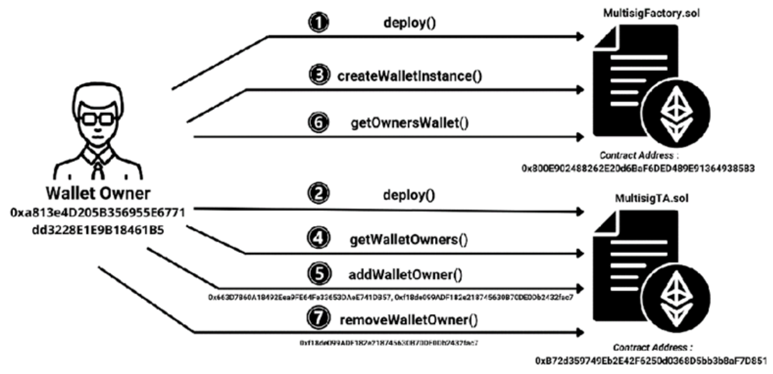

Figure 12 Multisignature Factory Contract and Multisignature Wallet Owner Testing Mechanism

4.4.3 Token Wallet

This test was conducted to determine how users, acting as wallet owners, can perform transactions related to the Token Wallet. The wallet owner can add the currency token that has been created in the Token contract, namely "DYT," to the multisig wallet. This process is carried out so that the wallet owner can have the option to use currency tokens in transactions without always relying on the Ether currency coin as shown in Figure 13. Subsequently, the wallet owner can display the currency tokens and currency coins that can be used in the wallet.

Based on the test results of transactions related to the Token Wallet, the test is deemed valid and the successful transaction results have been added to the Blockchain.

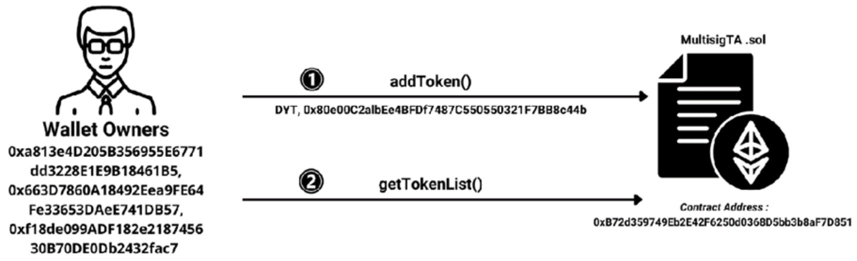

Figure 13 Token Wallet Transaction Testing Mechanism

4.4.4 Multisignature Wallet Transaction

In this case, the wallet owner deposited 15 Ether (ETH) or 15000000000000000000 wei into the smart contract. This process aims to make payments related to transactions conducted within the smart contract, especially for gas fee payments. After the deposit process succeeds, the wallet owner can check the balance of Ether currency coins deposited in the MultisigTA contract by entering the parameter "ETH" according to the currency whose balance they want to view, and similarly, the wallet owner can view the currency balance from the smart contract.

In addition to making deposits, the wallet owner can also withdraw funds from the smart contract through the MultisigTA contract. This withdrawal process must be initiated by executing a transaction through the initiateWithdrawal() function before executing the withdraw() function as a security implementation aspect of one form of withdrawal pattern. Failure to do so will result in a failed transaction. In this case, the smart contract balance will be reduced to 9 Ether or 9000000000000000000 wei, and the withdrawal process will be for 0.5 Ether or 500000000000000000 wei to facilitate observing the difference before and after the withdrawal transaction process.

Based on the test results of transactions related to the Multi-Signature Wallet Transaction, the test is deemed valid and the successful transaction results have been added to the Blockchain as shown in Figure 14.

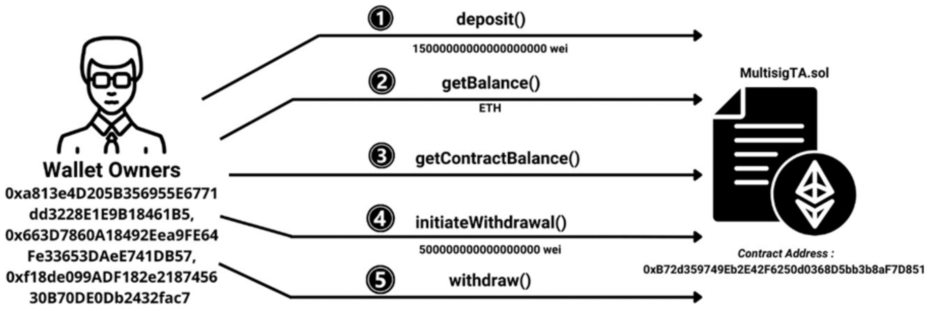

Figure 14 Multisignature Wallet Transaction Testing Mechanism

4.4.5 Multisignature Wallet Transfer Transaction

This test was conducted to determine how users, acting as wallet owners, can perform transactions related to the Multi-Signature Wallet Transfer Transaction. The wallet owner can initiate a transaction to send currency coins or currency tokens to another wallet address. This process can be carried out in the MultisigTA contract. This function is used to create a transfer request before being approved by other wallet owners according to the multi-signature scheme. Thus, the transfer process must go through approval first, which involves calling the function to create a transfer request. In this case, the transfer process will be initiated from the main wallet owner with the address ***0xa813e4D205B356955E6771dd3228E1E9B18461B5*** to the recipient wallet address ***0x13d6166f563Fa4752E72F7F633031e14613f53b8***, with a transfer amount of 0.5 Ether or 500000000000000000 wei.

After initiating the transfer request, in this case, the wallet owner can also cancel the transfer request made. Based on the test results of transactions related to the Multi-Signature Wallet Transfer Transaction, the test is deemed valid and the successful transaction results have been added to the Blockchain as shown in Figure 15.

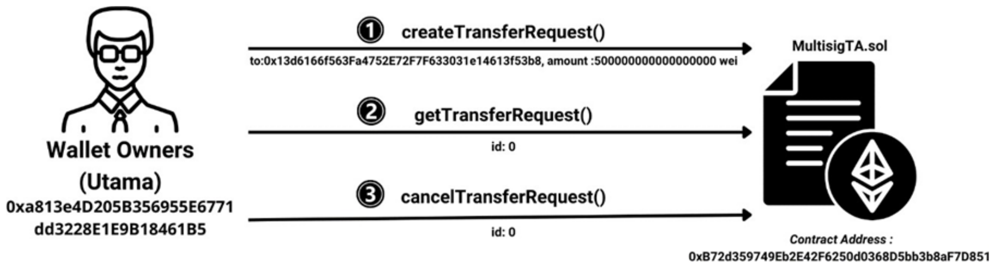
Wallet Owners
(Utama)
0xa813e4D205B356955E6771
dd3228E1E9B18461B5
1 createTransferRequest()
to:0x13d6166f563Fa4752E72F7F633031e14613f53b8, amount :500000000000000000 wei
2 getTransferRequest()
id: 0
3 cancelTransferRequest()
id: 0
MultisigTA.sol
Contract Address :
0xB72d359749Eb2E42F6250d0368D5bb3b8aF7D851

Figure 15 Multisignature Wallet Transfer Transaction Testing Mechanism

4.4.6 Multisignature Wallet Approval Transaction

This test was conducted to assess how users, acting as wallet owners, can perform transactions related to the Multi-Signature Wallet Approval Transaction. After the main wallet owner creates a transfer request to the wallet address ***0x13d6166f563Fa4752E72F7F633031e14613f53b8***, other wallet owners besides the main one can approve the transfer request through a function call in the MultisigTA contract by entering "ETH" as the currency coin being transferred and "1" as the ID of the transfer request.

Wallet owners who can approve include ***0x663D7860A18492Eea9FE64Fe33653DAeE741DB57*** as Wallet Owner A and ***0xf18de099ADF182e218745630B70DE0Db2432fac7*** as Wallet Owner B. Before giving approval, a wallet owner can check the approval status of the transfer request to see if they have already approved it or not. Wallet owners can also view the approval limit for the transfer request.

Based on the test results of transactions related to the Multi-Signature Wallet Approval Transaction, the test is deemed valid and the successful transaction results have been added to the Blockchain as shwon in Figure 16.

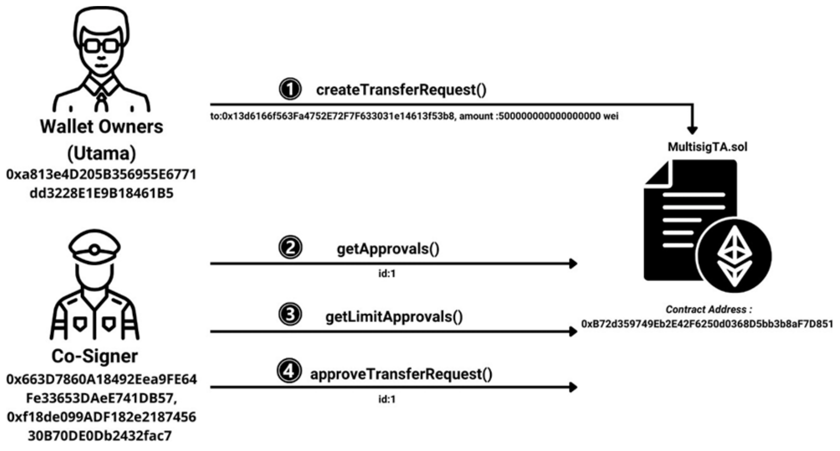

Figure 16 Multisignature Wallet Approval Transaction Testing Mechanism

4.4.7 Testing Summary

Based on the functional test cases that have been described, the results of this test are proven to be valid by the specifications and security standards set. This indicates that the implementation and functionality of the multi-signature wallet system have been successful. A summary of the test results can be seen in more detail in Table 3.

Table 3 Testing Summary

Test Classes	Description	Results
Cryptocurrency Token	Creating a cryptocurrency token and token supply by applying the ERC20 standard token smart contract and performing token approval.	Cryptocurrency token (DYT) and token supply are successfully created by applying ERC20 standard token smart contract.
Multisig Factory Contract	Initializing a multisig wallet smart contract dynamically based on the owner's wallet address using a factory contract.	Multisig wallet smart contract is dynamically initialized based on the wallet address owner with a factory contract.
Token *Wallet*	Adding the currency token created based on the ERC20 token standard to the multisig wallet and viewing both the currency token and coin that can be used in the multisig wallet.	A currency token (DYT) created based on the ERC20 standard token is registered and can be used in the multi-sig wallet.
Multisignature Wallet Owner	Adding, removing, and viewing wallet owners in the multisig wallet.	Users can successfully add, delete, and view wallet owners with a multi-sig wallet system.
Multisignature Wallet Transaction	Performing deposit, checking balances, and currency withdrawals by implementing withdrawal patterns, external effects, and mutual exclusion on the multisig wallet.	Users can successfully deposit, display the balance, and withdraw the currency coin (ETH) or currency token (DYT) that has been created by implementing withdrawal patterns, external effects, and mutual exclusion in the multi-sig wallet.
Multisignature Wallet Transfer Transaction	Creating transfer requests and canceling transfer requests to a specific wallet address using currency tokens or coins in the multisig wallet.	User successfully made a transfer request and canceled a transfer request to a specific wallet address in the multi-sig wallet.
Multisignature Wallet Approval Transaction	Approving transfer requests to specific wallet addresses and executing currency transfers by implementing withdrawal patterns, external effects, and mutual exclusion on a multisig wallet using a 3-of-3 scheme.	User successfully approves a transfer request to a specific wallet address and transfers currency coin (ETH) or currency token (DYT) by implementing withdrawal pattern, external effects, and mutual exclusion in a multi-sig wallet using 3-of-3 schema.

5 Conclusion

Based on the research, the multi-signature wallet system with a 3-of-3 scheme on ERC20 smart contracts in the Ethereum Blockchain has successfully enhanced transaction security. By requiring approval from three different key holders, the risk of a single key is minimized, making transactions safer. The architecture of this system also integrates withdrawal patterns, external effects, and mutual exclusion, reducing the chances of reentrancy attacks and ensuring that transaction operations are executed correctly.

Testing the system through the external system created, Ganache EVM Testnet, and Remix IDE showed good performance, in line with the research goal to enhance security in transaction execution in the Blockchain environment. The findings of this research contribute positively to the development of security in the context of smart contracts and DeFi on the Ethereum platform. It is hoped that these findings will serve as a foundation for further development, enhancing the security and reliability of DeFi transaction systems in the future.

References

[1] A. Beije, N. Vyas, and B. Krishnamachari, *Blockchain and the Supply Chain: Concepts, Strategies and Practical Applications*, Second Edition. Kogan Page, 2022.

[2] A. M. Antonopoulos and G. Wood, *Mastering Ethereum: Building Smart Contracts and DApps*, First Edition. Sebastopol, California: O'Reilly Media, 2018.

[3] W.-M. Lee, *Beginning Ethereum Smart Contracts Programming: With Examples in Python, Solidity, and JavaScript*, 2nd Edition. New York: Apress, 2023.

[4] J. Han, M. Song, H. Eom, and Y. Son, "An Efficient Multi-signature Wallet in Blockchain Using Bloom Filter," in *Proceedings of the ACM Symposium on Applied Computing*, Association for Computing Machinery, Mar. 2021, pp. 273–281. doi: 10.1145/3412841.3441910.

[5] R. Ma, J. Gorzny, and E. Zulkoski, *Fundamentals of Smart Contract Security*. New York: Momentum Press, 2023.

[6] S. Ebrahimi, P. Hasanizadeh, S. M. Aghamirmohammadali, and A. Akbari, "Enhancing Cold Wallet Security with Native Multi-Signature schemes in Centralized Exchanges," Oct. 2021, doi: 10.48550/arXiv.2110.00274.

[7] J. W. Lim, *Handbook of Digital Currency: Bitcoin, Innovation, Financial Instruments, and Big Data*, First Edition. Massachusetts : Academic Press, 2015.

[8] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," *Proceedings - 2017 IEEE 6th International Congress on Big Data, BigData Congress 2017*, pp. 557–564, Sep. 2017, doi: 10.1109/BigDataCongress.2017.85.

[9] T. Laurence, *Blockchain For Dummies*, 3rd Edition. Hoboken: John Wiley & Sons, Inc, 2023.

[10] V. Dhillon, D. Metcalf, and M. Hooper, *Blockchain Enabled Applications Understand the Blockchain Ecosystem and How to Make it Work for You*. California: Apress Berkeley, 2017. doi: https://doi.org/10.1007/978-1-4842-3081-7.

[11] J. Sun, S. Huang, C. Zheng, M. Wang, Z. Hui, and Y. Ding, "A Novel Method to Prevent Multiple Withdraw Attack on ERC20 Tokens," in *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*, IEEE, Dec. 2021, pp. 1–7. doi: 10.1109/QRS54544.2021.00011.

[12] V. Buterin, "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform," 2014.

[13] ChainLink, "Reentrancy Attacks and The DAO Hack Explained." Accessed: Nov. 21, 2023. [Online]. Available: https://blog.chain.link/reentrancy-attacks-and-the-dao-hack/

[14] C. Diligence, "Ethereum Smart Contract Best Practices." Accessed: Nov. 12, 2023. [Online]. Available: https://github.com/ConsenSys/smart-contract-best-practices

[15] Maurice. Herlihy and N. Shavit, *The Art of Multiprocessor Programming*, 1st Edition. Massachusetts: Morgan Kaufmann, 2012.

[16] M. Marchesi, L. Marchesi, and R. Tonelli, "An Agile Software Engineering Method to Design Blockchain Applications," vol. 8, pp. 1–8, Oct. 2018, doi: 10.1145/3290621.3290627.