Blockchain Decisions: How Logic Constructs Secure Distributed Agreement in Ethereum Smart Contracts

Muhammad Nur Majiid - 13524028
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: muhammadnurmajiid22@gmail.com , 13524028@std.stei.itb.ac.id

***Abstract*— Blockchain technology offers a secure and transparent distributed system through consensus mechanisms that operate without a central authority. This paper explores how principles of propositional logic and Boolean algebra form the foundation of decision-making processes in blockchain networks, particularly in the implementation of smart contracts on Ethereum. We analyze how simple logical operators such as "AND," "OR," and "NOT" are employed to construct deterministic and secure consensus mechanisms. Furthermore, we demonstrate how logic-based design can mitigate vulnerabilities in smart contract logic, thereby enhancing the reliability and security of automated contract execution. The implementation and testing were conducted using the Remix IDE platform with Solidity-based smart contracts, illustrating trustless coordination and secure conditional execution.**

Keywords— blockchain; propositional logic; smart contracts; Ethereum; Boolean algebra; consensus.

I. Introduction

Blockchain technology has been widely discussed lately, especially about crypto and security which is quite sophisticated, such as how blockchain works in a decentralized manner (without a central control) as well as its level of security and transparency. In fact, the core of all the sophistication of blockchain technology lies in one main thing, where blockchain is run by thousands of computers (nodes) around the world to reach a common agreement. so that they all know which transactions are valid and which blocks should be added next without any interference from one party in power (the center).

Discussions about blockchain are often filled with quite complicated terms, especially for laypeople or people who are new to have entered the world of blockchain. However, behind all that complexity, the basic decision-making process is actually much simpler. In fact, the logical foundation in decision-making is very similar to how basic logic gates work.

In this paper, we will try to look at blockchain from a different perspective, which may be easier to digest. We will analyze how simple propositional logic with concepts like "AND", "OR", and "NOT" is the backbone of every "decision" made by blockchain nodes. For example, when a node receives a new transaction or block, it doesn't just believe it. It "thinks" logically: "Is this transaction valid AND is it signed correctly?". If all conditions are met, then it approves. By understanding simple logical operators, we intend to provide a clearer and more understandable view of how these simple logical principles are the foundation of the incredible security and reliability of decentralized systems on blockchain.

II. Theoretical Basis

A. Logic

Logic is the analysis and appraisal of arguments. Here we'll examine reasoning on philosophical areas (like God, free will, and morality) and on other areas (like backpacking, water pollution, and football). Logic is a useful tool to clarify.[1] Logic is the basis for systematic thinking. By using logic, it is expected to reduce errors in action in facing and solving a problem, so that the problem can be solved with a systematic answer. The way to think correctly using this basic logic can be made by a computer through a computer program, so that the computer can perform the ability to think logistically and systematically even though it is only simple.

1. Propositional logic

In logic, there is a sentence that has a true or false value but not both, this sentence is called a proposition. In the sentences "today it rains" and "100 is greater than 120" can have a true or false value so the sentence can be stated as a propositional sentence, while the sentence "what time is it?" is not a propositional sentence because it does not have a true or false value.

There are compound propositional or molecular proposition sentences, which are a combination of two or more propositional sentences connected by a logical operator. There are several basic logical operators such as "AND", "OR", "NOT" and implication ("IF…THEN…").

X	Y	NOT X	NOT Y	X AND Y	X OR Y
True	True	False	False	True	True
True	False	False	True	False	True
False	True	True	False	False	True
False	False	True	True	False	False

Table 2.1 implementation of basic logical operators

In the implementation of propositional logic, "NOT" returns a value that is the opposite of the proposition's truth value (True becomes False, and False becomes True). Conversely, "AND" yields a True value only if all propositions are true, while "OR" yields a False value only if all propositions are false.

2. Boolean Algebra

Algebra Boolean is a branch of mathematics developed by George Boole in the mid-19th century. Algebra Boolean discusses the logic value resulting from the operation using binary variables, where each variable can only have one of two values: true (1) or wrong (0). Boolean algebra is used as a way to formalize logical reasoning, and is the basis for digital logic and computer science.

Boolean algebra also has an important basic operator such as in the logic of propositions, such as Conjunction "AND" (∧) or (.), Disjunction "OR" (∨) or (+) and negation "NOT" (¬) or (`).

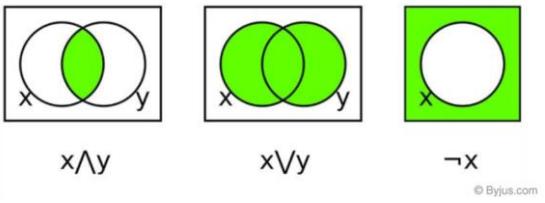


Fig 2.1 Boolean Algebra Operators
Source: https://byjus.com/maths/boolean-algebra/

Implementation of operators in the Boolean algebra is also same as the implementation in the proposition logic.

A	B	A ∧ B	A ∨ B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

A	¬A
True	False
False	True

Fig 2.2 Table of Operators implementation in Boolean Algebra
Source: https://byjus.com/maths/boolean-algebra/

In Boolean algebra there are some basic axioms such as commutative, distributive, identity, and complement. For any given $x, y, z \in B$, where B represents a Boolean Algebra, these elements can exclusively hold a value of either 0 or 1, with no possibility of simultaneously holding both. the following equations are applicable:

a. commutative :
   i. $x + y = y + x$
   ii. $x \cdot y = y \cdot x$
b. distributive :
   i. $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
   ii. $(x + y) \cdot z = (x \cdot z) + (y \cdot z)$
c. identity :
   i. $x \cdot 1 = x$
   ii. $x + 0 = x$
d. complement :
   i. $x + x` = 1$
   ii. $x \cdot x` = 0$

B. Distributed Systems and Consensus

Centralized system is still often used by most people, where the control of a system can be held or regulated by people/organizations that are the center of the system. However, the centralized system has several deficiencies that are quite fatal, which are vulnerable to manipulation and interference of people who can regulate the system, making the centralized system begin to be less trusted by some people. Distributed systems are created as solutions and substitutes for centralized systems that are still often used.

1. Distributed Systems

There is a big difference between centralized system and distributed system, where in centralized systems there is a center that can regulate the overall system. while the distributed system is a collection of independent computers that appears to its users as a single coherent system.[2] Distributed system also includes a collection of nodes that contain computer programs that utilize computing resources that run separately to achieve common goals. Also known as distributed computing or distributed database, distributed systems are very dependent on separate nodes synchronized through general networks to communicate with each other between nodes. Distributed systems aim to eliminate bottlenecks or central failure points of a system.

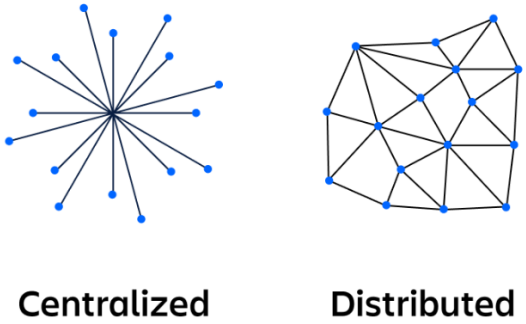


Fig 2.3 Difference between centralized system and distributed system
Source:https://www.atlassian.com/microservices/microservices-architecture/distributed-architecture

Characteristics in distributed systems include: Sharing resources, where hardware, software, or data can be used together (sharing) on distributed systems (can be used by other users/nodes). Simultaneous processing where several machines can process the same function simultaneously. Scalability, computing and processing capacity can increase as needed when expanded to additional machines. Detection of errors in which failures in the system can be more easily detected because they are run by many computers simultaneously. Transparency, in

a distributed system, nodes can access and communicate with each other.

2. Consensus

In a distributed system, there is a theory namely Byzantine Generals' Problem (BGP). BGP is a classic problem in computer science, especially in distributed systems that explain fundamental challenges in reaching agreements (consensus) between several separate parties, especially when some of them may be dishonest, damaged, or unexpected behavior. This is a crucial concept that underlies why designing a safe distribution system is very difficult, and why blockchain appears as a revolutionary solution.

Based on Byzantine Generals' Problem (BGP), scientists develop an algorithm called Byzantine Fault Tolerant (BFT). This algorithm is designed to enable the distributed system to reach consensus even though there are a number of nodes that do not behave honestly or damaged. In the case of this blockchain algorithm is used to reach consensus when finding and adding a new block to the blockchain network.

There are two consensus mechanisms in the blockchain that are often used:

i. Proof of Work (POW)

POW is used by Bitcoin to overcome problems related to consensus by making the block validation process very expensive computing (required large computing resources). The attacker needs to control most of the network computing power to carry out attacks and make fake consensus (51%attack), which economically it makes no sense. This is a form of probabilistic BFT.

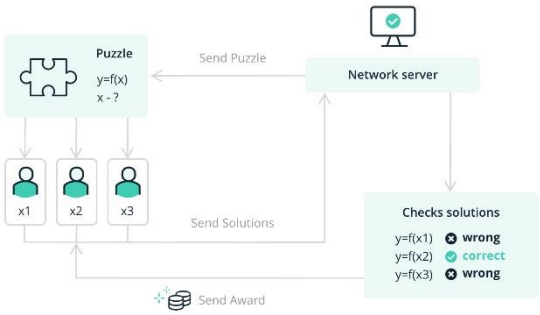


Fig 2.4 Proof of Work mechanism
Source:https://www.ledger.com/academy/blockchain/what-is-proof-of-work

ii. Proof of Stake (POS)

Post is used by Ethereum to overcome problems related to consensus with incentive and penalty mechanisms (called slashing). Validators that are dishonest or intend to manipulate consensus (making fake consensus) will lose most or all of the stakes (crypto assets they lock into the network), making the behavior very expensive and not economically profitable. This is designed to achieve economic finality and is a form of economic BFT.

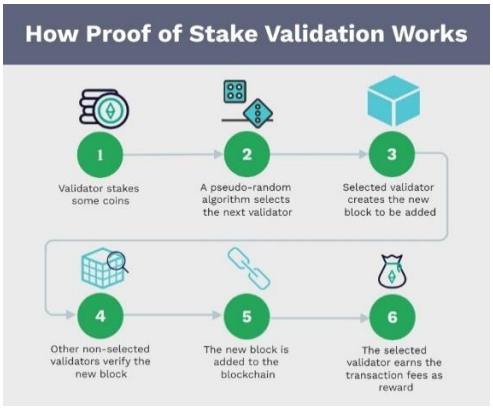


Fig 2.5 Proof of Stake mechanism
Source : https://www.myetherwallet.com/blog/proof-of-work-vs-proof-of-stake/

C. blockchain System

A blockchain system consists of a network of computing nodes, sharing a common data structure (the blockchain) with consensus about the state of this structure.[3] blockchain technology allows us to share information transparently in a Web3 network (blockchain network). Data on blockchain is stored in blocks that are linked together in a chain. The data is consistent because you cannot delete or change the chain without consensus from the network. blockchain technology is often used to create immutable or permanent ledgers to track transactions, payments, orders, account information, and more. The system has built-in mechanisms to prevent unauthorized transaction entries (BFT Algorithm) and create consistency in the shared view of these transactions.

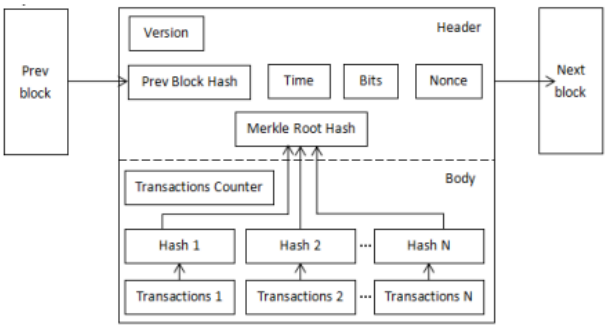


Fig 2.6 Block Structure in blockchain system
source:https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8732934

Regarding transaction recording on the blockchain network, the blockchain system sets rules where approval from participants (nodes) is needed so that the transaction carried out is considered valid and the transaction recording process can be carried out. Recording new transactions requires the addition of a new block to the blockchain network, so the approval needed is more than 51% of participants (nodes) or the majority of participants (nodes) agree to the transaction (nonce).

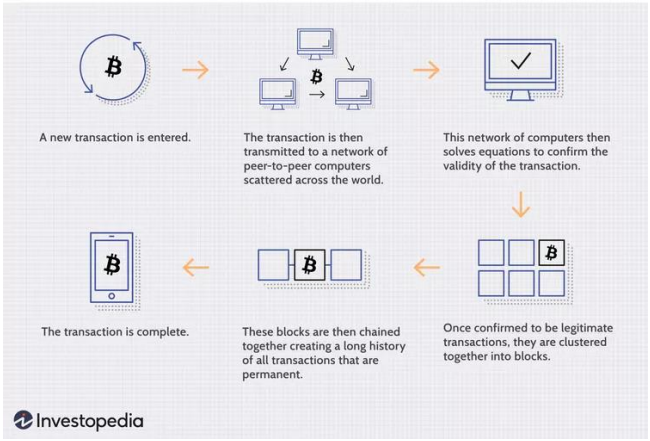


Fig 2.7 recording transactions process on the blockchain
source : *https://www.investopedia.com/terms/b/blockchain.asp*

blockchain technology allows us to use various available features on the blockchain, as follows:

1. Decentralization

   in blockchain the decision-making process is based on consensus generated by nodes in a distributed network, this removes control and decision-making from a centralized entity (individual, organization, or group). Decentralized blockchain networks processes are carried out transparently, reducing the need for trust among participants (nodes). This network also prevents participants (nodes) from controlling each other in a way that reduces the functionality of the network to control the network.

2. Immutability

   In the blockchain network, something that has been entered into the network, cannot be changed or deleted. No one can change a transaction after someone records it in the shared ledger, even the person who recorded/made the transaction. If there is an error in the transaction record, the user must create a new transaction to correct the previous error and enter it into the blockchain network, both transactions can be seen by anyone in the network.

3. Consensus

   The blockchain system sets rules about recording transactions into the network. You can record a new transaction only if the majority of participants in the network give their approval.

blockchain also has several main components in its network, including:

1. Distributed Ledger

   A distributed ledger is a shared database in a blockchain network that stores transactions. The number of transactions that can be recorded in a block varies, depending on the block size and the complexity of the transactions. For example, complex smart contracts will require more space, so the number of transactions recorded will be smaller. Distributed ledger technology has strict rules about who can edit and how to edit. You cannot delete an entry once it is recorded.

2. Smart Contract

Smart contracts are used to manage contracts independently (automatically) without the need for third-party assistance. Smart contracts will automatically activate after being deployed and stored in the blockchain system. and when the conditions in the smart contract are met, the smart contract will automatically execute the program. Smart contracts run if-then checks so that transactions can be completed with certainty.

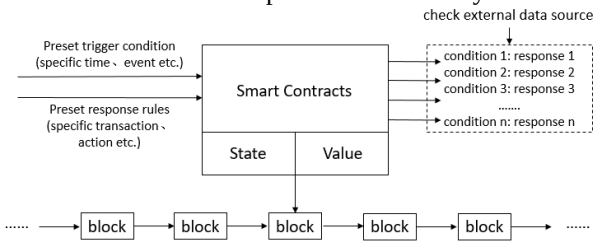
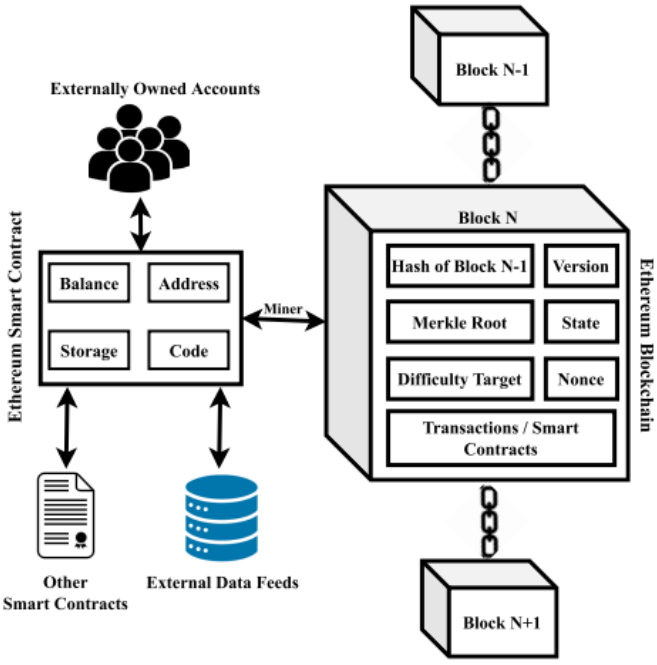


Fig 2.8 Smart Contract Mechanism
source : *https://www.researchgate.net/figure/The-mechanism-of-smart-contracts_fig2_325899686*

D. Ethereum Smart Contracts

Smart contract is a set of protocols created by contract participants in a blockchain network to execute automatically when conditions set out in the contract are met. Smart contracts allow two or more parties to create digital agreements that can be executed without intermediaries, with transparent and immutable outcomes once the contract is executed.

Smart contract work is quite simple. when the smart contract is approved (signed) by the parties involved, then the smart contract will be stored in the blockchain network and can be accessed by anyone. when the conditions in contract are met the contract will automatically execute the specified



action. all smart contract execution results are immutable and

transparent, thus preventing data manipulation and other fraud.

Fig 2.9 Ethereum blockchain based smart contract source:https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9762279

Smart contracts can be used in various cases. Here are some cases that can be done using smart contracts:

1. Financial Contracts

   DeFi (Decentralized Finance) services or applications can handle large financial transactions without banks. DeFi applications can be more profitable than traditional applications in terms of trust and security of transactions.

2. Digital Identity

   In traditional systems, identity management and trust management are facilitated by government administration. Ethereum smart contracts can manage digital identities and build trust and security rather than management in legacy systems.

3. Supply Chain Management

   Traditional supply chain systems lack transparency and traceability, while in Ethereum smart contracts, in addition to transparency and traceability, there is also an automatic verification process, where the system will automatically pay/make transactions after the item arrives at its destination.

To run Ethereum smart contracts, supporting components are required, including:

1. Ethereum Virtual Machine (EVM)

   EVM is a decentralized runtime environment that executes smart contract bytecode. The EVM is Turing complete, meaning it can execute complex computational instructions, and all nodes on the Ethereum network execute contracts in a consistent manner.

2. Solidity

   Solidity is the primary programming language for writing smart contracts on Ethereum. Solidity code is compiled into bytecode that can be executed by the EVM.

3. Gas

   Each instruction in a smart contract requires an execution fee called gas. Users must pay gas fees using Ether to run contracts on the network.

III. PROBLEM DEFINITION AND SOLUTION

A. Problem Definition

In a blockchain network like Ethereum, achieving secure and trustworthy agreements among participants (nodes) who do not know each other and trust each other is a major challenge. The absence of a central authority necessitates reliance on smart contracts, which are prone to logic-based vulnerabilities. Furthermore, the lack of formal guarantees in most smart contract implementations leaves them open to exploitation.

Common problems in distributed systems, especially Ethereum smart contracts, include:

- Trustless Coordination in Open Networks: How can parties who do not trust each other reach a valid and unmanipulated agreement?
- Vulnerability in Smart Contract Logic: How to structure contract logic so that it is not only functional but also secure?

B. Solution

Ethereum smart contracts can overcome the challenge of explicit trustless coordination by implementing a Boolean logic-based consensus mechanism. By encoding conditions under which transactions, such as fund transfers or contract activations, are executed only if a series of logical propositions are true, deterministic consensus can be achieved across network nodes. This approach inherently eliminates the reliance on external arbitration, while ensuring consistent and predictable contract behavior.

Logic-based design can mitigate common smart contract vulnerabilities by making conditions explicit, testable, and verifiable. By representing each decision point in Boolean expressions and analyzing them through logical tables or formal verification tools, the possibility of undefined states or logical bugs can be drastically reduced.

IV. IMPLEMENTATION

The implementation of the given solution cannot be shown in full in this paper, due to issues related to code length and source code complexity, readers can still access the full implementation of the solution and relevant documentation in the GitHub repository at.
https://github.com/MAJIIDMN/Ethereum-Smart-Contract-with-Remix-IDE

1. Trustless Coordination in Open Networks

   To build trust in a smart contract, the program will be set to only execute actions when all parties involved in the contract agree (approve). The program will also verify the approval first, to ensure that only people involved in the contract can call/use the APPROVE() function.

A (approverA)	B (approverB)	C (approverC)	Exec = A ∧ B ∧ C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	**1**

Fig 4.1 Consensus Execution Truth Table

Algorithm 1 Boolean Logic-Based Consensus Execution

Require: Approvals from parties A, B, and C
Ensure: Contract action is only executed upon unanimous approval

Initialize approval flags $A, B, C \leftarrow false$
function APPROVE(party)
  if party $\in \{A, B, C\}$ **then**
    Set corresponding approval flag to true
  end if
end function
function ISCONSENSUSREACHED
  return $A \wedge B \wedge C$
end function
function EXECUTEACTION
  if executed = false **and** isConsensusReached() = true **then**
    Execute contract logic
    Set $executed \leftarrow true$
  else
    Reject execution
  end if
end function

Fig 4.2 Logic-Based Algorithm

This algorithm ensures smart contracts execute correctly, only taking action when all parties involved agree. This builds greater trust among users, even if they don't know each other and there's no third party involved.

2. Vulnerability in Smart Contract Logic

Logic-based design principles can be applied to strengthen the security of smart contracts. By using basic principles of logic and Boolean algebra, contracts ensure that execution only occurs when all required states are met. In this example, the contract uses three logical flags—verification, timeliness, and user consent—that must all be true to trigger payment execution.

V (Verified)	T (OnTime)	A (Approved)	Exec
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	**1**

Fig 4.3 Payment Execution Truth Table

Algorithm 2 Logic-Based Payment Execution

Require: Status flags: Verified (V), OnTime (T), Approved (A)
Ensure: Payment is only released when all conditions are satisfied

function SETFLAGS
  Buyer sets V, T, and A via separate approval functions
end function
function CHECKCONDITIONS
  return $V \wedge T \wedge A$
end function
function EXECUTEPAYMENT
  if CheckConditions() = true **and** paid = false **then**
    Proceed with payment to seller
    Set paid $\leftarrow$ true
  else
    Reject payment execution
  end if
end function

Fig 4.4 Logic-Based Algorithm for payment Execution

These Boolean conjunctions form the basis for reliable conditional execution, reducing the risk of undefined behavior or missed validation steps. This algorithm uses Boolean conjunctions and basic forms of logic for reliable conditional execution, reducing the risk of undefined behavior or missed validation steps. so that the level of security and accuracy of program execution will be higher.

V. TESTS AND RESULTS

The test was conducted on the Remix platform, which is a web-based Integrated Development Environment (IDE) designed specifically for creating, compiling, deploying, and debugging Solidity smart contracts on the Ethereum blockchain. By using remix, we can use virtual environments or Ethereum testnet for developing and testing smart contracts before deploying them to the Ethereum mainnet.

After creating a smart contract in Solidity (.sol), we can compile the contract in the compile menu, the compiled contract can be directly deployed on Remix deploy menu.

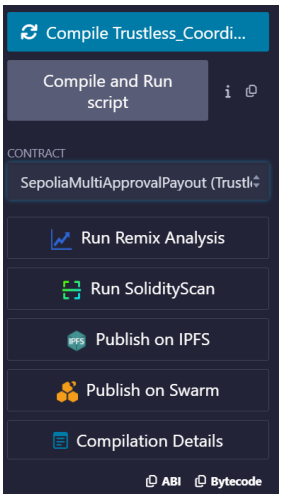


Fig 5.1 Compile menu on Remix

We can deploy on a virtual machine (VM) provided by Remix or use a testnet on the blockchain network that we use (ETH) by connecting Remix to the wallet (Web3 wallet) that we use.

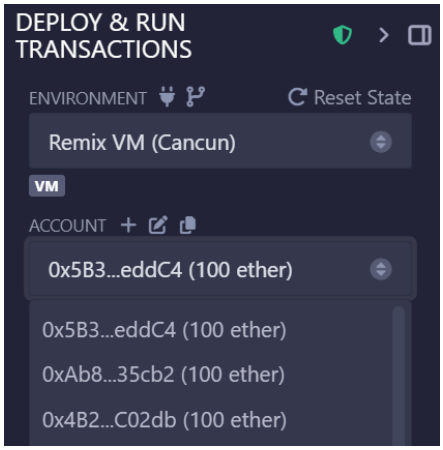


Fig 5.2 Deploy contract on VM

1. Trustless Coordination in Open Networks

In the first smart contract, we need to enter the addresses of the approvers and recipients, where the Approver will act as a party that can approve the contract to reach consensus, when consensus has been reached, actions can be taken, for example sending a certain amount of ether to the recipient's address.

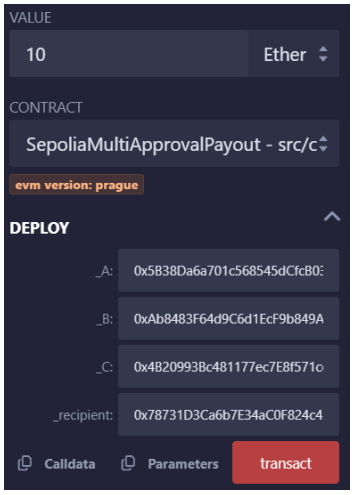


Fig 5.3 Deploy the first smart contract

After successful deployment, we can perform several actions that we have programmed on the smart contract using the Application Binary Interface (ABI) provided by Remix.

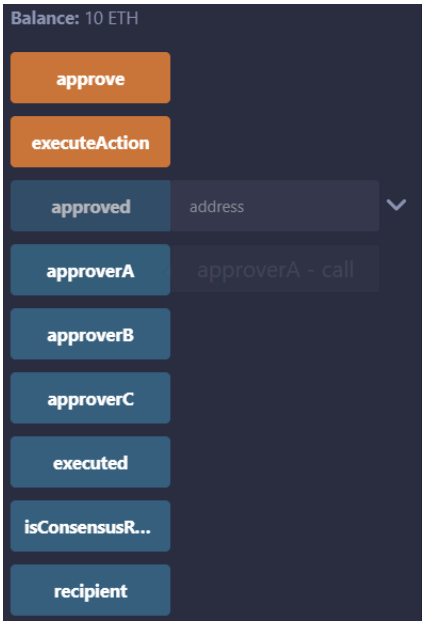


Fig 5.4 First smart contract (ABI) display

This function can be used to check whether a consensus has been reached or an action has been taken.

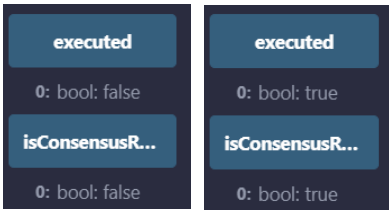


Fig 5.5 Have not reached a consensus and action has not been taken

Fig 5.6 Consensus has been reached and action has been taken

2. Vulnerability in Smart Contract Logic

In the second smart contract we need to enter the address of the seller and buyer, the base price, the packaging deadline (before shipping), and the delivery deadline, the buyer will lock a certain amount of ether in the contract as payment. and after the transaction is completed, the smart contract will automatically send ether at the agreed price to the seller's address, and the buyer can also take back the remaining ether that was not used during the transaction.

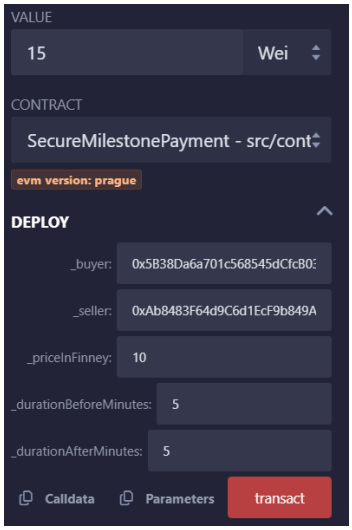


Fig 5.7 Deploy the second smart contract

After successfully deployed, we can perform several actions that we have programmed on the smart contract using the Application Binary Interface (ABI) provided by remix. as in the first smart contract.

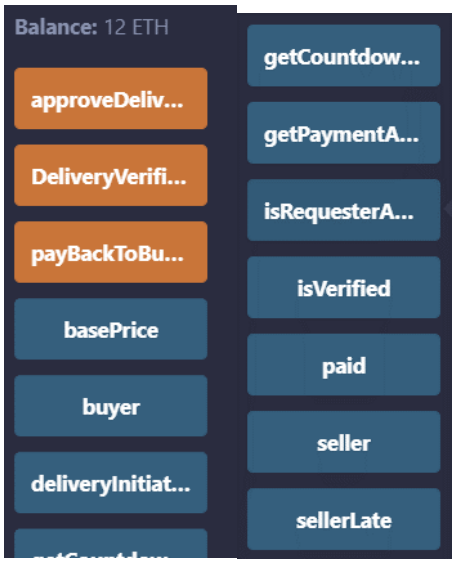


Fig 5.8 Second smart contract (ABI) display

This function can be used to check whether it has been shipping to the buyer and whether payment has been made.

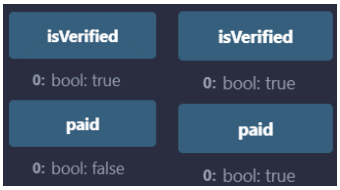


Fig 5.9 Item in shipping process

Fig 5.10 Item has arrived at the buyer

VI. APPENDIX

video explanation of the paper is at: https://youtu.be/J8FruTpH5MI?si=si-o_cVXPUH99tIu

The program is available in the GitHub repository at: https://github.com/MAJIIDMN/Ethereum-Smart-Contract-with-Remix-IDE

VII. ACKNOWLEDGMENT

All praise is due to Allah, the author can complete this paper on time. The author would like to express his deepest gratitude to all parties who have supported the learning process of the Discrete Mathematics IF1220 course, especially to Mr. Arrival Dwi Sentosa, M.T as the lecturer of the course. Lastly, the author apologizes for any shortcomings in this paper. Hopefully this paper can be used as a reference for future research.

REFERENCES

[1] H. J. Gensler, *Introduction to logic*. New York Routledge, 2017.

[2] A. S. Tanenbaum and V. Steen, *Distributed Systems - Principles and Paradigms, 2nd Edition*. Pearson Higher Education, 2007.

[3] F. Idelberger, G. Governatori, R. Riveret, and G. Sartor, "Evaluation of Logic-Based Smart Contracts for Blockchain Systems." [Online]. Available: https://research.csiro.au/data61/wp-content/uploads/sites/85/2016/08/ruleml16.pdf (accessed Jun. 18, 2025).

[4] J. Liu and Z. Liu, "A Survey on Security Verification of Blockchain Smart Contracts," *IEEE Access*, vol. 7, pp. 77894–77904, 2019, doi: 10.1109/access.2019.2921624.

[5] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur, and H.-N. Lee, "Ethereum Smart Contract Analysis Tools: A Systematic Review," *IEEE Access*, vol. 10, pp. 57037–57062, 2022, doi: 10.1109/access.2022.3169902.

[6] A. Hayes, "Blockchain Facts: What Is It, How It Works, and How It Can Be Used," Jun. 2025. [Online]. Available: https://www.investopedia.com/terms/b/blockchain.asp (accessed Jun. 18, 2025).

[7] Ledger, "What is Proof-of-Work," Jun. 2023. [Online]. Available: https://www.ledger.com/academy/blockchain/what-is-proof-of-work (accessed Jun. 18, 2025).

[8] byjus.com, "Boolean Algebra (Boolean Expression, Rules and Examples)." [Online]. Available: https://byjus.com/maths/boolean-algebra/

[9] M. E, "Proof of Work vs Proof of Stake," Jun. 2024. [Online]. Available: https://www.myetherwallet.com/blog/proof-of-work-vs-proof-of-stake/

(accessed Jun. 18, 2025).
[10] K. Zettler, “What Is a Distributed system?” [Online]. Available: https://www.atlassian.com/microservices/microservices-architecture/distributed-architecture (accessed Jun. 18, 2025).
[11] Amazon Web Services, “What is Blockchain Technology? - Blockchain Technology Explained - AWS,” *Amazon Web Services, Inc.*, 2024. https://aws.amazon.com/what-is/blockchain (accessed Jun. 18, 2025).

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Juni 2025

Muhammad Nur Majiid (13524028)