



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

**IPV6 BLOCKCHAIN DATA COMMUNICATION FOR UAV
SWARM-INTELLIGENCE SYSTEMS BASED
ON PEER-TO-PEER, PEER-TO-MANY,
AND MANY-TO-PEER SCENARIOS**

by

Dymas Dymas

September 2022

Thesis Advisor:
Co-Advisor:

James B. Michael
Peter R. Ateshian

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC, 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 2022		3. REPORT TYPE AND DATES COVERED Master's thesis
4. TITLE AND SUBTITLE IPV6 BLOCKCHAIN DATA COMMUNICATION FOR UAV SWARM-INTELLIGENCE SYSTEMS BASED ON PEER-TO-PEER, PEER-TO-MANY, AND MANY-TO-PEER SCENARIOS				5. FUNDING NUMBERS
6. AUTHOR(S) Dymas Dymas				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000				8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A				10. SPONSORING / MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.				12b. DISTRIBUTION CODE A
13. ABSTRACT (maximum 200 words) This thesis explores the use of blockchains along with the Internet Protocol version 6 (IPv6) data packet messages to support secure, high-performance, and scalable communication with an intelligent swarm of unmanned aerial vehicles (UAVs). For this thesis, we investigate the exchange of encrypted data packets for three scenarios, those being peer-to-peer, peer-to-many, and many-to-peer. We simulate the swarm's behavior for each of these scenarios and vary the number of UAVs in a swarm over the simulation runs. The simulation-based results showed that for peer-to-peer scenarios and many-to-peer scenarios, there is no significant increase in latency even though in many-to-peer scenarios, the number of interacting nodes increases. In contrast, latency increases for the peer-to-many scenarios. Additional research needs to be performed to assess the security and scalability of the blockchain-IPv6 approach proposed in this thesis.				
14. SUBJECT TERMS blockchain, encryption, data communication, Internet Protocol version 6, IPv6, data packet, unmanned aerial vehicles, UAVs				15. NUMBER OF PAGES 89
				16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**IPV6 BLOCKCHAIN DATA COMMUNICATION FOR UAV
SWARM-INTELLIGENCE SYSTEMS BASED ON PEER-TO-PEER,
PEER-TO-MANY, AND MANY-TO-PEER SCENARIOS**

Dymas Dymas
Major, Indonesian Navy
BME, Sekolah Tinggi Teknologi Angkatan Laut, 2013

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

**NAVAL POSTGRADUATE SCHOOL
September 2022**

Approved by: James B. Michael
Advisor

Peter R. Ateshian
Co-Advisor

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis explores the use of blockchains along with the Internet Protocol version 6 (IPv6) data packet messages to support secure, high-performance, and scalable communication with an intelligent swarm of unmanned aerial vehicles (UAVs). For this thesis, we investigate the exchange of encrypted data packets for three scenarios, those being peer-to-peer, peer-to-many, and many-to-peer. We simulate the swarm's behavior for each of these scenarios and vary the number of UAVs in a swarm over the simulation runs. The simulation-based results showed that for peer-to-peer scenarios and many-to-peer scenarios, there is no significant increase in latency even though in many-to-peer scenarios, the number of interacting nodes increases. In contrast, latency increases for the peer-to-many scenarios. Additional research needs to be performed to assess the security and scalability of the blockchain-IPv6 approach proposed in this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	PROBLEM STATEMENT	1
B.	APPROACH.....	5
C.	SCOPE	6
D.	FINDING SUMMARY.....	6
E.	THESIS ORGANIZATION.....	6
II.	BACKGROUND	7
A.	BLOCKCHAIN.....	7
1.	Types of Blockchain.....	9
2.	Blockchain Working Principles	13
3.	Blockchain Components	15
4.	Smart Contract.....	26
5.	Blockchain Characteristics	26
B.	IPV6.....	27
1.	IPv6 Header Format.....	27
2.	IPv6 Encrypted Data Payload and Lightweight Blockchain	31
C.	ONE TIME PAD (OTP) BASED ON SHANNON’S OTP	31
III.	DESIGN OF EXPERIMENT.....	33
A.	IPV6 LIGHTWEIGHT BLOCKCHAIN DESIGN	33
1.	Framework	33
2.	Network Topologies	36
3.	Consensus.....	41
B.	DESIGN IMPLEMENTATION.....	42
1.	Software and Tool Installation	42
2.	Computing Platforms	43
3.	Time and Scalability Calculations.....	44
IV.	EXPERIMENT RESULTS.....	45
A.	SCENARIO PLANNING	45
1.	Simulation Preparation Stages	45
2.	The Implementation of the Simulation Based on Scenarios	50
B.	RESULTS AND ANALYSIS	55
1.	Simulation Results	55

2.	Simulation Analysis	61
3.	Comparison Results	62
V.	CONCLUSION AND FUTURE WORK	65
A.	SUMMARY OF FINDINGS	65
B.	FUTURE WORK POSSIBILITIES AND SUGGESTIONS.....	66
	LIST OF REFERENCES	67
	INITIAL DISTRIBUTION LIST	73

LIST OF FIGURES

Figure 1.	Basic process of drone communication. Source: [8].	3
Figure 2.	The concept of block generation in the UAVs swarm intelligence using blockchain technology	14
Figure 3.	Leveraging blockchain technology to reject an illegal drone	15
Figure 4.	Four main components of blockchain	16
Figure 5.	Avalanche effect. Source: [27].	21
Figure 6.	ZKP smart contract in blockchain guarantee the network security	34
Figure 7.	The class diagram shows only parent classes	35
Figure 8.	The send and receive data diagram	35
Figure 9.	Star topology design	37
Figure 10.	Point-to-point connection examples from the experiment	38
Figure 11.	Point-to-multipoint topology example from the experiment	39
Figure 12.	Mesh topology example for the experiment	40
Figure 13.	Hybrid topology example	41
Figure 14.	The ZKP process in the program	42
Figure 15.	Ubuntu 20.04.4 LTS specification	43
Figure 16.	Docker and Vagrant version information	44
Figure 17.	Compiler command in the VM terminal	45
Figure 18.	Successful connection between two terminals in the VM	46
Figure 19.	Connection failure in Ubuntu for Windows	46
Figure 20.	Back up file 01-network-manager-all.yaml	47
Figure 21.	Add IPv6 address and gateway 6	48
Figure 22.	Check updating new IP addresses	49

Figure 23.	Ping test from another VM	49
Figure 24.	Illustration of peer-to-peer communication between two UAVs.....	50
Figure 25.	Terminal displays that show the peer-to-peer connection is succeeded	51
Figure 26.	Illustration of the position of all nodes represented in peer-to-many scenario	52
Figure 27.	The command to initiate the communication on the GCS node to other nodes	53
Figure 28.	The illustration of the position of all nodes in many-to-peer scenarios.....	54
Figure 29.	The results of all the iterations performed in the peer-to-peer scenario	56
Figure 30.	The average transactions per milliseconds of the peer-to-peer scenario	57
Figure 31.	The results of all the iterations performed in the peer-to-many scenarios.....	58
Figure 32.	The average transactions per seconds of the peer-to-many scenarios	59
Figure 33.	The latency calculation results for three client nodes at 1 and 10 transaction iterations	60
Figure 34.	The latency calculation results for three client nodes at 100, 1000, and 10000 transaction iterations	60
Figure 35.	The average transactions per milliseconds of the many-to-peer scenario	61
Figure 36.	Comparison summary of latency test results	62

LIST OF TABLES

Table 1.	Blockchain block content.....	8
Table 2.	The advantages and disadvantages of each blockchain variant. Source: [13].....	12
Table 3.	Comparison of SHA functions. Source: [30].	23
Table 4.	IPv6 header format compared to IPV4 header. Source: [36].	28
Table 5.	Extension header options. Source: [37].	30
Table 6.	The latency calculation results of peer-to-peer scenario.....	57
Table 7.	The latency calculation results of peer-to-many scenarios	59
Table 8.	The latency calculation results of many-to-peer scenario.....	60

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ARP	Address Resolution Protocol
CPU	Central Processing Unit
DSCP	Differentiated Services Code Point
ECN	Explicit Congestion Notification
FANET	Flying Ad-Hoc Network
FPGA	Field-Programmable Gate Array
GCS	Ground Control Station
ICMPv6	Internet Control Message Protocol for IPv6
IoE	Internet of Everything
IoT	Internet of Things
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
LSB	Least Significant Bit
LTS	Long-Term Support
LWBC	Lightweight Blockchain
MTU	Maximum Transmission Unit
NAT	Network Address Translation
NIST	National Institute for Standards and Technology
NP	Non-deterministic Polynomial
NPS	Naval Postgraduate School
OS	Operating System
OTP	One Time Pad
P2P	Peer-to-Peer
PRNG	Pseudo Random Number Generator
RC	Remote Control
RNG	Random Number Generator
SCC	Smart Contract Consensus
SHA	Secure Hash Function
TCP	Transmission Control Protocol
TRNG	True Random Number Generator

UAV	Unmanned Aerial Vehicles
UDP	User Datagram Protocol
UML	Unified Modelling Language
VM	Virtual Machine
ZKP	Zero Knowledge Proof
ZK-SNARK	Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge

I. INTRODUCTION

The ever-increasing use of drones for military purposes, in addition to advances in automation such as equipping unmanned aerial vehicles (UAV) with various levels of autonomy and swarm intelligence, makes such vehicles tempting targets for adversarial forces. To obtain a competitive advantage, an adversary will attempt to find the exploitable physical and cyber vulnerabilities of a drone's flight controller, receiver, or transmitter, then apply kinetic, cyber, or some combination of kinetic- and cyber-attack mechanisms to manipulate the behavior of the drone, such as causing the drone to crash or leak sensitive data.

One avenue of attack against a military drone is to manipulate the communication mechanisms used by drones, either for drone-to-drone communication or communication between a drone and its human operators. For example, an adversary could modify, or block data exchanged among a swarm of drones to lower the effectiveness of the swarm's actions. It is important to provide military units with drones that have been assessed for kinetic and cyber vulnerabilities, that the risks associated with those vulnerabilities are mitigated before the operational use of the drone, and that risk assessment and mitigation occur as changes are made to the drone system over the drone's useful life.

Security risk management is also regulated in a framework, as was done by the National Institute for Standards and Technology (NIST), which published a risk management framework. Multiple technologies can be used to implement security-risk-reduction measures. For instance, Vikas Hassija and Vinay Chamola [1] assert that: "it is imperative to keep the transactions between the drones and other users secure, cost-effective, and privacy-preserving. Blockchain technology is a highly promising solution that can be used to deploy real-time drone applications."

A. PROBLEM STATEMENT

There is a symbiotic relationship between innovation and advancements in science and technology. Capabilities such as self-driving automobiles, autonomous Unmanned

Aerial Vehicles (UAVs), and smart home appliances, at one time considered to be in the realm of science fiction or too technically difficult to realize, are now commonplace.

The concept of a UAV first appeared in 1783 when Joseph-Michel and his partner, Jacques-Étienne Montgolfier publicly demonstrated a vehicle that can be said to be a drone or unmanned aircraft at that time [2], in the form of a hot air balloon in a place in France called Annonay in 1849. During that war, the balloon bombs, created by an Austrian lieutenant, Franz von Uchatius, were used to attack the city of Venice. Although the attack only caused minor damage, it can be claimed as a success because two days later Venice surrendered [3]. Nicolas Tesla patented a remote control (RC) in 1898, and about twenty years later a company called The Ruston Proctor Aerial Target invented the first pilotless winged aircraft based on RC technology that Tesla had previously patented [4].

Since then, drone technology and its adoption have grown steadily. They have been used in scientific studies, such as to gather data about the activity of volcanoes where it would be too dangerous or expensive to use piloted aircraft. In the 1990s Abraham Karem introduced the Predator, a drone equipped with cameras and other sensors for surveillance. The defense community has equipped the Predator with weaponry, including missiles [5]. The predators themselves have been used in several conflicts, such as those in Afghanistan, Pakistan, Bosnia, former Yugoslavia, Iraq, Yemen, Libya, Syria, and Somalia [6]. In 2022, they have also been used extensively for combat by both the Ukrainian and Russian armed forces.

A significant advancement of UAV technology is the application of swarm intelligence, in which a group of drones imitate the intelligent behavior of large numbers of homogeneous animals such as ant colonies, flocks of birds, and colonies of bees. The swarm exhibits collective behavior through coordination amongst the members of the swarm. The behavior of swarms can be encoded as algorithms which in turn can be implemented in software for execution on computers, such as the embedded computers used in drones [7]. The swarm behavior has even been used to put on drone-based light shows, such as at the opening ceremony of the Tokyo 2020 Olympic Games.

In the bee colony, the queen bee is the controller and similarly, in the swarm intelligence UAV, there is a control center in the system, typically the controller named Ground Control Station (GCS). UAVs work in a straightforward way, which involves data exchange between UAVs and GCS, then GCS can be connected to satellites or satellites can be directly connected to UAVs and everything happens in real-time. Figure 1 illustrates one way in which drones and their infrastructure can communicate. At a minimum, the communication needs to be low latency and secure. [8].

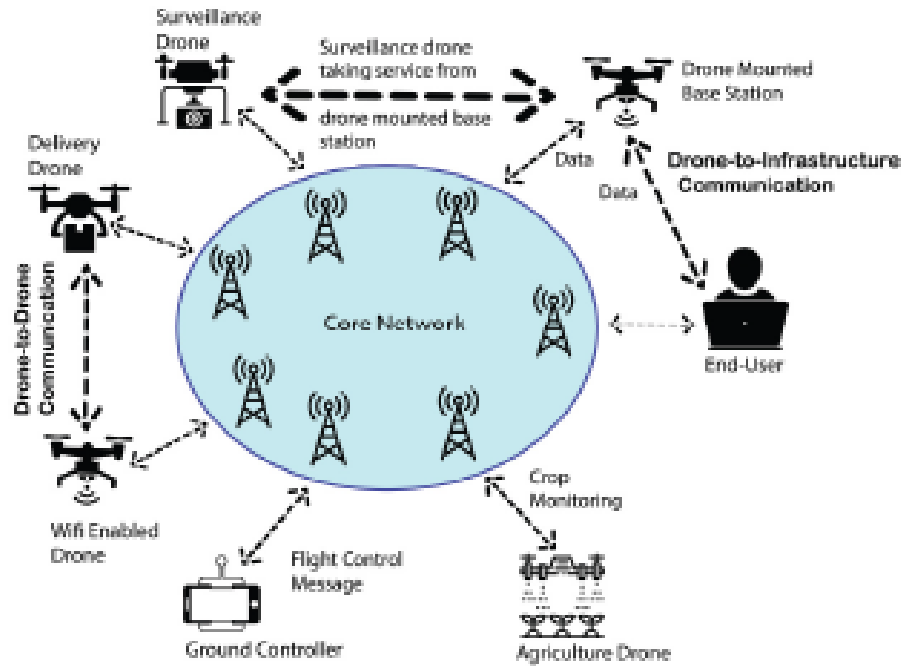


Figure 1. Basic process of drone communication. Source: [8].

There are two techniques for communicating between GCSs and UAVs. The first technique is the swarm infrastructure-based GCS and the second is the Flying Ad-Hoc Network (FANET). The swarm infrastructure-based GCS itself has a GCS that is used for centralized communication. All UAV swarms will communicate with the GCS so that the swarm can function. However, a drawback of this technique is that it is dependent on the availability and correct functioning of the GCS. If the GCS is disturbed, the entire UAV swarm will also experience interference. In contrast, FANET uses one transmitter to send

commands to a certain UAV, then that UAV relays those commands to a second UAV. The commands will then be distributed to other drones in either a serial or concurrent manner. All UAVs will communicate and have a list of commands given by the transmitter so that if this transmitter fails, all UAVs can still execute commands because each UAV has a list of valid commands. In the end, by using this FANET technique, each UAV will have redundancy and not depend entirely on the communication infrastructure. However, this technique also has drawbacks. For example, an intruder or an unknown UAV can enter and disrupt the UAV swarm. As another example, the authorized members of the UAV swarm cannot detect, so intruder (i.e., unauthorized participant) UAVs and thus can get a list of commands that will be carried out by the authorized UAVs [9].

To overcome the problem of intruder UAVs, it may be possible to apply Blockchain to prevent unauthorized UAVs from accessing lists using the UAV swarm command. Blockchain itself has been widely used in the financial sector with the aim of eliminating third parties from the verification process of each transaction.

In the blockchain, when the data is distributed, it will be difficult to hack and get the complete data because it is validated by a network that uses cryptographic means. Each block consists of the hash value of the previous block, a random number to verify the hash, or nonce, and a timestamp. The integrity assurance is provided by the blockchain for the formation of the first block which is the result formed of a validated transaction, called the genesis block. Since the value of the hash is unpredictable or unique, an act of fraud or duplication will be detected. Each validated block has its hash value and any changes to that block will have an impact on other blocks. The block is added to the chain if all or most nodes give permission or agree as the consensus mechanism which arranges the validity of the transaction in a certain block of the block's validity.

This consensus mechanism on the blockchain can be performed in three ways, those being Proof of Work, Proof of Stake, and voting, Practical Byzantine Fault Tolerance. In the world of cryptocurrency, Proof of Work is used for mining. It works by doing computation of mathematical equations on each node, then each node that completes the computation first will have the right to enter the latest block into the blockchain. Using Proof of Stake, only legal nodes can perform computations to reach consensus. On the

other hand, Practical Byzantine Fault Tolerance is voting based and requires at least one third. of the authorized nodes be Byzantine.

The authentication process is performed by generating One-Time Pad (OTP) with a pseudo random function. The UAVs were registered in the Blockchain, and each drone determines the nearest drone that it is able to authenticate based on the relationship stored in the Blockchain nodes. The request for authentication is sent from a drone to the related drone which observes and checks in the Blockchain whether this drone is related and would be able to authenticate it. This scheme is able to thwart the attack of external malicious drones or third-party attacks, even if the adversary knows the first token.

B. APPROACH

In this thesis, we investigate the way communication takes place between UAVs using IPv6 (Internet Protocol Version 6). IPv6 has many advantages over IPv4 (Internet Protocol Version 4), namely, it is faster, and more effective because it has fewer routing tables than IPv4, so the routing process will be more organized and effective, and more secure because it is equipped with encryption capabilities for exchanging data. Bandwidth is more efficient because IPv6 supports multicast. Configuration is easier because it runs automatically. Overall, IPv6 is more suitable for mobile devices such as drones because there is no need to go through a network address table (NAT) and thus is low latency. IPv6 will be combined using blockchain with Proof of Stake consensus.

Like with cryptocurrency, every node in the blockchain must do the payment. In this research, the payment is replaced with an OTP. Each node generates the same or synchronous OTP. The use of blockchain and OTP here is to detect unauthorized UAVs and prevent them from reading or updating command lists used by the UAV swarm. Furthermore, we explore the capabilities of blockchain, Smart Contract Consensus (SCC), and Distributed Ledger technology for swarm communication. In addition, a simulation was performed based on the concept of proposed the UAV swarm intelligence communication architecture.

C. SCOPE

The scope of this thesis is limited to exploration of the combined use of blockchain technology and OTP, both of which are populated in the IPv6 data packets.

D. FINDING SUMMARY

After conducting an experiment that simulates the operation of physical UAVs in peer-to-peer, peer-to-many, and many-to-peer scenarios and using 1–10000 iterations or transactions, the results of the comparison of the latency of each scenario are obtained. From these results, it is concluded that for peer-to-peer scenarios and many-to-peer scenarios, there is no significant increase in latency even though in many-to-peer scenarios, the number of interacting nodes increases. Whereas in the peer-to-many scenarios where a node conducts a transaction in the form of broadcast messages to several nodes at once, this results in increased latency. The simulation results and a summary of these conclusions are explained in Chapter IV and Chapter V. In addition, in Chapter V, the possibilities and suggestions for future work related to the matters in this thesis are discussed.

E. THESIS ORGANIZATION

Chapter II presents the background of UAV swarm intelligence communication blockchain capabilities and leveraging it as a means of communication in UAV swarm intelligence. It also gives an overview of the IPv6 structure format. Chapter III discusses the analysis of data transmission for communication based on the IPv6 blockchain. Specifically, the analysis of IPv6 blockchain data packet scenarios, confidentiality, integrity, and availability. The research results of the possibilities and challenges of IPv6 blockchain implemented in the UAV swarm intelligence are explained in Chapter IV. Chapter V provides conclusions and recommendations for future research.

II. BACKGROUND

The concept of UAV Swarm intelligence is based on the assumption that UAVs is exceptionally reliable in terms of deployment, high mobility, inexpensive maintenance, and can be programmed efficiently to loiter while waiting for the next command. Hence, UAV swarm intelligence is predicted to increase in various fields, such as traffic or target surveillance, perimeter surveillance, remote sensing, or even package delivery. To anticipate the rapid developments in its use in the future, this technology must be robust and safe for its users.

An advantage of using a UAV is that it can be set up to transmit packets with minimal or no human intervention and performs object detection or tracking in real-time. Attempts to carry out malicious activities like transferring malicious packages can damage the reliability of the swarm's communication. These attacks can be categorized according to their impact on data secrecy (confidentiality), data authentication (integrity), accessibility of services, and data (availability) [10].

To overcome protect against attacks on swarm communication it may be possible to leverage blockchain solutions. In order to be suitable for use on mobile Swarm Intelligence UAVs that require high throughput and low latency, this blockchain needs to have a smart contract that functions to regulate agreements between drones. Use of One Time Pad (OTP) and zero-knowledge proof (ZKP) will strengthen the guarantee of transaction security between drones. UAV Swarm intelligence also requires a lightweight blockchain, so a combination with IPv6 is an alternative that needs to be considered.

To find out how blockchain combined with IPv6 can provide security guarantees, an overview of the blockchain and IPv6 structure format is given in this chapter.

A. BLOCKCHAIN

Blockchain users recognized that blockchain is a series of sequential pieces of information. Every blockchain starts with the first block, commonly called a genesis block. Every entry in the blockchain is stored as a record. The record contains information about every financial transaction and every new transaction request will be entered into the queue

mechanism to become a new block in the chain. Each new block that is created based on a basic structure consists of 4 main parts. The first part contains a hash value. The second part is the block header which contains the number to track the serial number of the updated blockchain software, the Merkle root hash, information about the hash of the previous block, timestamp which has information about when the transaction has been made, nonce and the difficulty information algorithm of proof of work for this block. The third part is the transaction counter to count how many transactions are in this block. The last part of the block contains the transactions that have been processed and have been included in this block. The blockchain design determines how many and when a transaction is recorded in each block which will then be secured with digital signatures and cryptographic hashes. Each block is connected to the previous block and holds the address of the previous block. The blockchain design structure is made by the combination of blocks which hold each other's data. For this reason, the intervention against a block can be prevented because each block also stores the hash value of the previous block together with its own hash value, this makes the blocks in the blockchain have bonds that cannot be intervened. Table 1 shows the block content of the blockchain.

Table 1. Blockchain block content

BLOCK CHAIN BLOCK CONTENT			
Magic number		4 byte	Fixed value in the form of "0xD9C4AEF9" in 4 bytes
Height		4 byte	Block number
TITLE (HEADER)	Version	4 byte	Version used
	Previous title	32 byte	Title (hash) of previous block
	<u>Merkle root</u>	32 byte	Hash value of transactions (hash)
	Timestamp	4 byte	Time since January 1, 1970
	Difficulty	4 byte	Difficulty information of the network
	Nonce	4 byte	Random number adjusted for network difficulty
Number of operations		1-9 byte	Integer value of certain length (<u>integer</u>)
Transactions		-	List of transfers in the block

Blockchain is commonly identified as a distributed ledger. The blockchain can be set up so that every transaction is encrypted. A regular implementation of blockchain technology is cryptocurrency transactions. Every time a financial transaction occurs, a new block is created which is encrypted, then the new block will be passed to each individual user in the blockchain network so that they are aware of the transaction. Blockchain technology in cryptocurrency can replace the role of banks by eliminating intermediaries between sellers and buyers because they process transactions themselves and are validated by other users.

While predominately used in the financial sector, blockchain technologies can be leveraged for other uses. Such as surveillance, supply chain management, tracking, communications logging, and notary transactions. Due to its safe, transparent, and reliable nature, many systems are starting to use blockchain technology [11].

1. Types of Blockchain

It is important to cover the characteristics of blockchain in order to adapt the technology to other technical concepts and designs. In general, most people know about blockchain from its use as the technology based on one of the most widely recognized cryptocurrencies, Bitcoin. When Bitcoin was first introduced it quickly became a phenomenon. This is because bitcoin transactions appear to have supplanted the function of banks as “intermediaries” or liaisons between the recipient and the sender of a transaction. Blockchain technology provides the capability for a system to work without relying on a centralized function [12].

There are three main types of blockchain technologies [13]:

a. Public Blockchain

The Public blockchain is the earliest variant of the type of blockchain that exists today. The public blockchain was created with the aim of reducing dependence on the use of intermediaries in a transaction system. Because of its nature which can be accessed by the public (permission-less distributed ledger), anyone can join and conduct decentralized transactions in it [12]. Each joined node will have a copy of the ledger because each node

has an equal position. This means that no single entity has more rights or privileges to manage this system, and all nodes will be able to populate and validate the blockchain.

Because public blockchain is open and transparent to the public, it allows fraud or manipulation of data to occur. Thus, public blockchain requires a security mitigation measure. A mutual agreement must be created to ensure its security. For example, the use of consensus methods is the Proof of Work and Proof of Stake method whose algorithm ensures that in the distributed ledger there are verified and valid transaction records.

Due to the excessive time and great energy required to complete mathematical computations when completing a transaction, it can be said that this type of blockchain has problems with transaction speed. For comparison, bitcoin that uses the public blockchain is only able to complete seven transactions per second [14] compared to VISA, which can complete more than 24000 transactions in one second [15]. When the speed decreases, as well as energy consumption, will increase as the number of nodes increases, then another fact arises, which in the end it is concluded that public blockchains also have scalability problems [12].

b. Private Blockchain

Different from the public blockchain, which is a permission-less distributed ledger, the private blockchain is known as the permissioned blockchain which operate on a closed network and restricted environment. This means that in this type of blockchain network there is an entity that regulates who may verify, read, write, access and grant authorization levels, or in other words, it is not fully decentralized. This is one of the biggest drawbacks of private blockchains, but on the other hand, it also provides advantages over public blockchain [12].

Because verification and validation have been regulated and determined by certain entities, the network can meet consensus and transactions more quickly and with less energy than public blockchains [13] Private blockchains are also believed to be more scalable because by only giving mandates to certain nodes in managing the network or authorizing, this will have an impact on the network so that it is free to grow on the

condition that new nodes must have received permission to join and carry out transactions, and even public blockchain can join it [13].

The level of trust and security is inferior when compared to public blockchains due to the possibility of manipulation and attacks on mandated nodes or centralized nodes. If the attack is successful, it will endanger the identity of other nodes, transactions, and everything else on the network [13].

c. Consortium Blockchain

To reduce the weaknesses that occur in the private blockchain, there is a consortium blockchain, where the network model is like a private blockchain because this type is still a centralized network, but the consortium blockchain nodes that regulate or have the right to give permission are carried out by groups consisting of more than one organization instead by a single node or by several nodes in one organization only. The nodes that validate the transactions also can act as the participant node, which can send and receive transactions [12].

A consortium network uses a multi-party consensus algorithm that regulates approval. Approval is obtained from voting results by the largest number of validating nodes, for example, if in a consortium network there are ten nodes that are joined to represent several organizations, then to validate a transaction that occurs only requires approval from a minimum of six nodes only [13], so that there is no node that would absolutely dominate in this network.

Although this type of blockchain is more robust than private blockchains in terms of security, the entire network can still leave the possibility for manipulation and attack, but it has better management over structure and customizability [12].

To provide a depiction of the differences between the three variants of blockchain technology. Table 2 displays the advantages and disadvantages of each blockchain variant [13].

Table 2. The advantages and disadvantages of each blockchain variant.
Source: [13].

Entries	Public Blockchain	Private Blockchain	Consortium Blockchain
Applied in	Ethereum, Bitcoin, Alternative coins The government can apply it for general election matters, voting due the trust and transparency reason	Hyperledger Fabric, Hyperledger Sawtooth, Corda Private companies can provide budget transparency for their direction boards	Marco polo, Energy Web Foundation, IBM Food Trust. Multinational food companies can apply it for tracking their food material so that they can maintain their material quality
Database	Public peer-to-peer distributed system	Private peer-to-peer distributed system	Private peer-to-peer distributed system
Function	Decentralized	More centralized	More centralized, but not fully centralized like Private Blockchain
User and Authorization	All users or nodes are responsible to validate the transactions and has a right to access everything in it	The authorized nodes decide levels of access to other user nodes	Consortium nodes can make and validate transactions, they also vote to grant permission for other nodes Having both functions of Public and Private Blockchain
Anonymity	Pseudonymous	Known Identities	Known Identities
Consensus Algorithm	Proof of Work, Proof of Stake	Multi-party consensus algorithm, Practical Byzantine Fault Tolerance, Proof of Elapsed Time	Multi-party consensus algorithm, Practical Byzantine Fault Tolerance, Proof of Elapsed Time, <u>similar to Private Blockchain</u> but rules depend on the agreement of the authorized groups of nodes
Consumption of Energy	High, because all nodes need huge energy to solve the mathematical computation to add in the block in the blockchain	Less power and energy	Less power and energy
Speed of Transactions	Slow	Fast	Fast
Cost of Operation	High	Low	Low
Scalability	Has scalability issues	Yes	Yes
Security	High, the more user will make the harder network to be attacked	More vulnerable to attacks and manipulation	More vulnerable to attacks and manipulation

2. Blockchain Working Principles

To explain the working principle of blockchain, it will be easier if the implementation is described directly with an example of communication between drones. The premise for the scenario is that after launching there is communication between two drones, the first communication data that occurs is framed in the first block (genesis block) and the data sent is expressed in a certain value. Next, drone A will transfer data to drone B, where drone A will store records of the transfers that occur into transfer records. These records will be called ledgers. The ledgers contain everything related to the communication data transfer that just happened. The details of the compartments in the ledger have been explained in the blockchain block content section. This compartment can contain when the data transfer occurred (date and time or timestamp), the contents of the communication data that has been transferred, and all information related to the record.

Now imagine again, drone A once again transfers other data to drone B, using a blockchain network. Requests from drone A will be broadcast to all drones in the network or colony, because blockchain has a peer-to-peer network method that usually called nodes, then the information on the data transfer will automatically spread to all drones on the same network. Each drone or node in the network will have a certain algorithm in common, so when data transfer occurs and is known by all nodes, each node will use this algorithm to carry out the verification process. The arrival of information about data transfers that occur simultaneously with the verification and validation process which takes time, then all data transfer information will be stored.

Like in a book, when information is verified, it will be sequentially recorded on a sheet of paper in the notebook, as well as the data transferred earlier. After being validated, it will be entered into a new block. Then it is coupled with the previous genesis block or block, that is in the ledger and so on to form a blockchain. When a new block has been added to the blockchain strand it can be said that the data transfer process is complete [16]. Figure 2 illustrates the concept of block generation in the blockchain on the drone's group in UAV swarm intelligence.

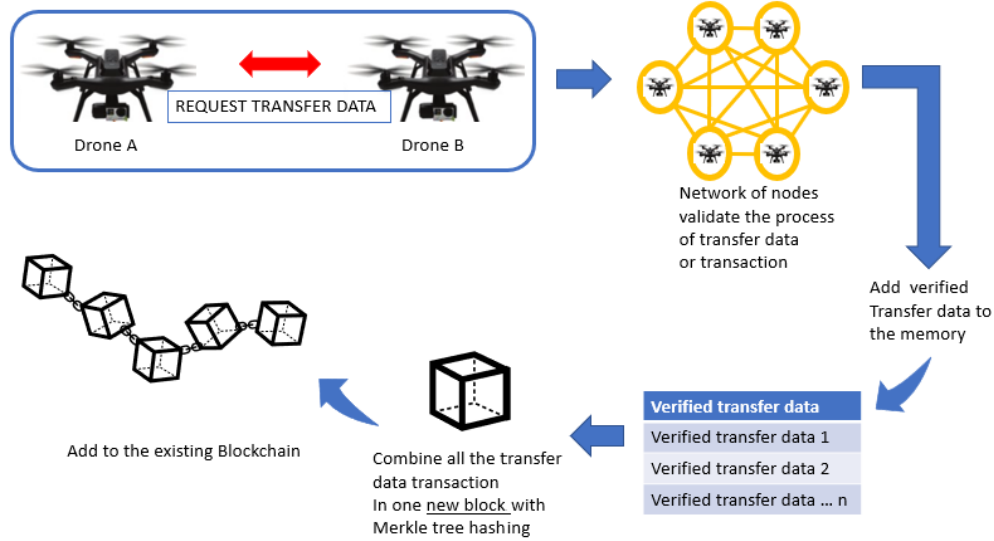


Figure 2. The concept of block generation in the UAVs swarm intelligence using blockchain technology

Given that a new block is added after the validation process is declared complete, who has the right to enter this new block into the blockchain? The one that has the right is a node that has been verified as the winner of a competition mechanism for that right. The winning node will be rewarded and have the right and obligation to insert new blocks into the blockchain thread in order. This verification mechanism prevents illegal nodes from entering the network. Figure 3 shows how an illegal drone attempted to enter the network [17]. This process is called mining, and because the mining process requires a large computing system, if the node does not know the mutually agreed algorithm, naturally this illegal node will fail or be unable to win the mining competition. Another thing that causes the failure of illegal nodes to insert fake blocks into the blockchain is that because a new block must be validated and verified by at least half of the network's resident nodes, an illegal node will need a lot of power to perform computations of algorithms that it does not know, and it would be a waste if illegal nodes enforce this. The mechanism of agreement that occurs between these nodes is called the consensus mechanism [17].

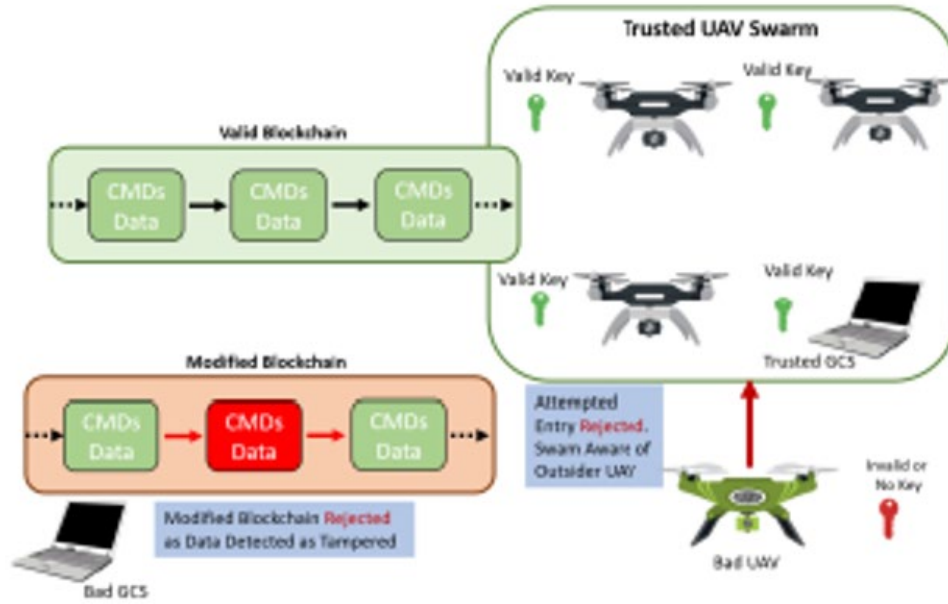


Figure 3. Leveraging blockchain technology to reject an illegal drone

3. Blockchain Components

In general, blockchain has four main components that make up the technology, which are, asymmetric cryptography and node applications, block generation and transactions, distributed ledger, and consensus mechanism [18]. See Figure 4.

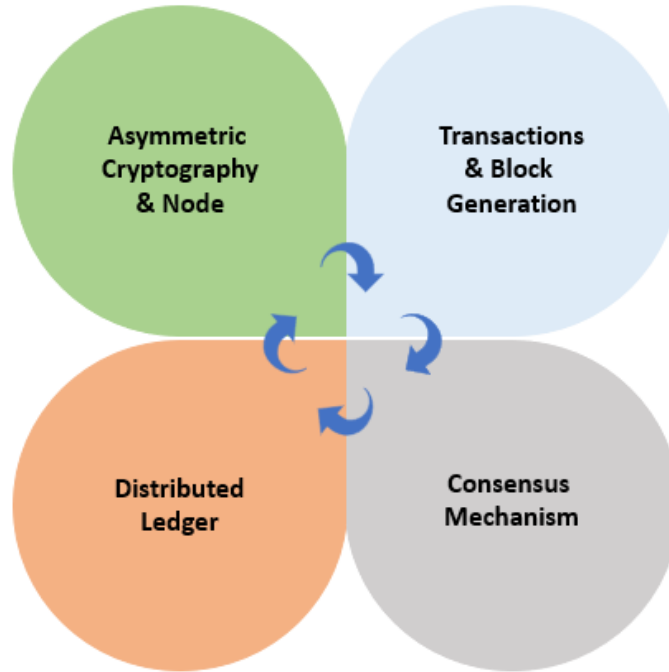


Figure 4. Four main components of blockchain

a. Asymmetric Cryptography and Node Applications

Blockchain technology provides a protection solution for communications that occur because in the blockchain there is an asymmetric key encryption component that is used to protect data on the network. Asymmetric cryptography is the second form of cryptography, besides symmetric cryptography.

As explained in Chapter I, UAV swarm intelligence will use the FANET technique. There are several examples given in references [19] and [20] The aim is to protect the communication lines which are based on secret-key cryptography or on public-key cryptography. However, given that UAVs that are mobile, it will be difficult to distribute shared secret keys and difficult to achieve the condition of ensuring a fully secure channel to send symmetric keys. If the colonies in the UAV swarm grows large and only rely on symmetric cryptography, then the number of secret keys required will be high as well. Therefore, to solve this problem asymmetric cryptography is offered for use by UAVs.

The advantage of using asymmetric cryptography is that it has two variants of keys. The private key, which is identified, and accessed by each individual node only, while the

other key. The public key is available to anyone and has a function as an address so that they can have transactions with each other. The public key can be changed anytime for security purposes [18].

Every existing node will be equipped with a private key and a public key for security. In general, the use of a private key is to digitally sign for every message transaction addressed to that node. What is special about blockchain is that every communication transaction that occurs is transparent so that it can be known by other nodes and can be accessed by other nodes through their public key. This is because the nodes in the blockchain use a mutual consensus. Failure to access transactions will make the nodes recognized as illegal nodes trying to enter the network, the man-in-the-middle attack.

In general, asymmetric cryptography has a scheme that requires high computational complexity because public-key cryptosystems are also heavyweight, it is concluded that asymmetric cryptography left a drawback which is not feasible to use in lightweight environments such as swarm intelligence UAVs. The solution is to use a combination of symmetric and asymmetric cryptography to be used as secret session keys. It will be used for authentication on all legal nodes in the swarm through an open communication network and exchange secret keys shared in peer-to-peer, peer-to-many, and many-to-peer exchanges, which can be used as session keys. Therefore, authentication at each node will use the Zero Knowledge Proof (ZKP) approach.

(1) Leveraging Zero Knowledge Proofs Cryptographic Techniques.

In UAV swarm intelligence, transparency and decentralization are not really needed because all drones in the network are known nodes. In fact, privacy, confidentiality, latency, and scalability are more prioritized. UAV swarm Intelligence prioritizes transaction security between node-to-node, node-to-all nodes, and all nodes-to-node in its network, thus encryption of data transactions from the sender to the recipient must be strong and guarantee that it will not be compromised.

When the focus falls on privacy and confidentiality then there are tradeoffs that will accompany this action, one of which is the loss of transparency, where transparency was the main thing about blockchain when it was first introduced. But in this thesis, our focus

is on how to develop and maintain privacy and confidentiality while at the same time achieving network stability, low latency, and scalability. Therefore, there is a way that can be taken legally even though each UAV in a swarm does not trust the other UAVs [21].

One way to secure privacy and confidentiality is to use advanced cryptographic techniques, where the blockchain will be treated with encryption, as Megan Kaczanowsky says in her article [22]. The purpose of the encryption is to operate an action of disguising plaintext. The data will be hidden, the encryption algorithm is used to encrypt the text, then the encryption algorithm will pair with the encryption key to encrypt the message. The algorithm is completed with a decryption algorithm that operates a decryption key, to reproduce the plaintext. This advanced cryptographic technique will provide a mathematically provable warranty to protect the confidentiality of data and every transaction that occurs. This technique will also maintain the confidentiality of the data or transactions when performing mathematical computations.

In the blockchain, there are times when a node just notifies other nodes about a change in its status without releasing and revealing all the information. For example when a drone X moves from point A to point B, this activity of course requires to be recorded and the node X as a prover needs to prove it to other drones (the verifier) as legal nodes in the network, but this drone X does not want to share other information on that node such as a piece of particular mission information which only node X knows about it, so the use of advanced cryptographic techniques such as Zero Knowledge Proofs (ZKP) is needed in this system.

ZKP is a cryptographic technique that involves two parties who act as a prover and a verifier, both of which will prove that a proposition is true without having to reveal all the information in the proposition. In the computing process, both parties will perform an interactive proof system through the soundness and completeness properties, where the use of false and true logic will prove the computation. and in the last property, zero knowledge. If the computation is correct, then both parties can only know the fact that this proposition is true from the legal node.

In ZKP, work trials and repeated responses for proof and verification involve the interaction of messages exchanged between the two parties with the need for a stable and continuous connection is inevitable. This is difficult to achieve in an environment where the connection is uncertain and not guaranteed to be stable, such as swarms of UAV. However, interactive message exchanges activities can fail. An alternative solution offered is the use of a non-interactive ZKP, which does not require back and forth interaction between two parties (the prover and the verifier).

Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge or commonly known as ZK-SNARK is one of the types of ZKP, which is offered as an alternative solution when users need a way for a prover to convince any verifier to validate and achieve computational zero knowledge without requiring interaction [23]. All trials will be compiled into one package sent in one message. This will optimize the time needed to exchange messages, even though these messages can be modified into a beacon in a peer-to-many scenarios [24].

ZK-SNARK is a cryptographic instrument implemented in forming the ZKP blockchains, which are Z-cash and Hawk, both of which have different goals. Z-cash aims to verify transactions that occur while Hawk is used to verifying smart contracts. The advantages of ZK-SNARK are that the inspection process can be carried out in a faster time. The level of evidence needed is low, and the integrity of the nodes is protected [25].

To prove and verify, ZK-SNARK works by performing computations expressed in Non-deterministic Polynomial (NP) statements. An example of using NP statements in zk-SNARK is as follows. There are 4 statements. The first is “The program P, when executed will return to exit code 1 if given the input value X and some additional input value Y.” The second is “the Boolean of P satisfies with a number of additional values of input Y.” The third is “the circuits of P accept the partial task of X, when extended into some full task Y.” The last is “the set limit of P is satisfiable by the partial task of X when extended into some full task Y.” Then, the prover with knowledge of the witness will produce a succinct proof, which will prove the truth of the NP statement. Then the verifier will verify this short proof.

Because it uses the property principle of zero knowledge, the verifier does not get from the proof anything other than the truth of the statement. For example, the value of Y is kept confidential. This proof is also short and easy to verify, and because the proof is in the form of a string, it does not require a two-way interaction between the prover and the verifier. It can also be ensured that the proof of the argument cannot be falsified and thus the soundness property is fulfilled. In addition, it can be guaranteed that not only does the prover know that the NP statement is true, but the prover also knows exactly the values of Y [23].

Besides computations, the difference between interactive and non-interactive is to assign a hash function to the proof so that when the online verifier will be able to verify it later, this is suitable to be implemented on swarms of UAVs whose connection is not guaranteed to always be stable [24] and with this ZKP scheme, can be used as a means of authentication where each node will show their identity to each other through a password. This is believed to result in lower processing and memory consumption [25].

(2) Hash Function for ZKP

A hash function is known as a function that is based on a deterministic mathematical function and works by mapping an input that has an arbitrary size into an output that has a fixed length. To illustrate, a simple example is a function that returns a numeric output value based on the input of the first letter of a person's name. So, when a person named "James Bond" is treated with a hash function and maps the initial letters of the name, the output of the function is a number between 1 and 26, for example, "James Bond" = 1002, which is a fixed-length number. Because this function is deterministic, every time the same input is entered it will produce the same output value. Problems arise when there is more than one name whose first letter in that name is the same. This will cause the output hash value for those people to be the same too. This event is known as a collision. When a collision occurs, we cannot distinguish the names of people based on the output hash value of their names because they have the same value. The information is too compressed.

One of the subclasses of the hash function family is the cryptographic hash function, which is usually found integrated with ZKP or symmetric key derivation. It is designed to assist in cryptographic schemes. The cryptographic hash function answers a collision event by creating a function for which it is not possible to have the same hash value for two different messages. It does not give any possibility to create a reversible process so that the given value of the original message recurs. If a small change in a message happens, it will modify the hash value. The previous hash value is not correlated with the new hash value [26].

Secure Hash Function (SHA) is the part of a cryptographic hash function that is intended to maintain data security. The way SHA works is by changing the data with a hash function, using an algorithm that has a compression function, bitwise operations, and modular additions. The resulting hash function product will be in the form of a string with a fixed size and without resembling the original message. SHA is usually used to encrypt passwords so that when an adversary hacks the database, they cannot find the plaintext password but instead only find the hash function result, which is the hash value. When they use it as a password it will not work. Instead, it will generate a new string hash value, and eventually illegal parties will not be able to access anything. SHA has an advantage called the avalanche effect which means that any change to the encrypted input, even if it is a minor change, will result in a major change to the encrypted output. In other words, the hash value will not give any information about the input [27]. This can be seen in Figure 5.

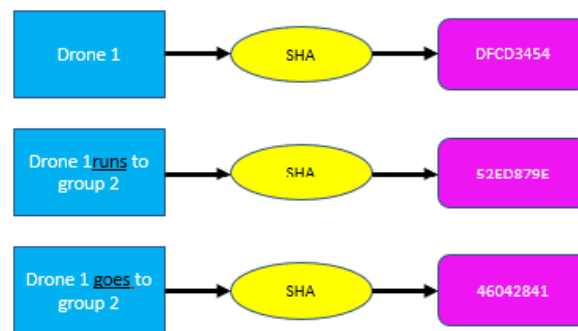


Figure 5. Avalanche effect. Source: [27].

SHA has three fundamental safety characteristics: the first is the pre-image resistance character that makes it difficult for an adversary or illegal nodes to reveal the original message m , which is treated with a hash function and produces a hash value Hm , by providing a one-way function or infeasible to reverse back the process. This characteristic provides security guarantees to prevent brute force attacks. The second is the second pre-image resistance characteristic, which guarantees security from the possibility of two different messages m_1 and m_2 but having the same two hash values, for example, $Hm_1 = Hm_2$. This will not happen with SHA. In addition, if SHA does not have this character, an attacker will easily break into a database without requiring the original password. The last safety characteristic is collision resistance, which is already explained above. However, the absence of collision resistance will make it possible to hack digital signatures because two messages that produce the same hash value would take into consideration two documents were signed by two different users where in the other place a certain user is able to generate another document with the same hash value.

The latest SHA family of standards member is SHA-3, released by NIST in August 2015 [28], SHA-3 has a different structure from SHA-1 and SHA-2. It is still part of the cryptographic primitive family Keccak. The Keccak algorithm has a new approach named sponge construction, which is based on a wide random permutation. This makes the function able to absorb any amount of data while outputting any amount of data at the same time and plays the role of a pseudorandom function with all previous input data. Therefore, the sponge construction is highly flexible. In addition, in the Keccak algorithm there are several additional uses for the function, including a stream cipher, a “tree” hashing scheme to speed up hashing, and an authenticated encryption system.

There are several reasons why SHA-3 is recommended to be implemented as part of the hash function security design for a UAV swarm. SHA-3 has advantages in performance, which is less expensive to implement in specific hardware such as a Field-Programmable Gate Array (FPGA). It is quite fast on a circuit, which is critically important for low-power devices, such as the Internet of Things or drones. By comparison, SHA-3 will require fewer Joules per byte than SHA-2 when implemented in hardware [29]. Furthermore, because development for UAV swarm intelligence is still wide open the

advantage of using SHA-3 is that SH-3 is significantly different from SHA-2, because if there is a new variant of a malicious attack that defeats SHA-2. The recommended software design and protocol can support the implementation, so if a weakness is found in SHA-2 it will be easy to transition to SHA-3. Table 3. gives a comparison of the SHA functions table that shows the performance of all SHA families [30].

Table 3. Comparison of SHA functions. Source: [30].

Algorithm and variant	Output size (bits)	Internal state size (bits)	Block size (bits)	Rounds	Operations	Security against collision attacks (bits)	Security against length extension attacks (bits)	Performance on Skylake (median cpb) ^[1]		First published
								Long messages	8 bytes	
MD5 (as reference)	128	128 (4 × 32)	512	64	And, Xor, Or, Rot, Add (mod 2 ³²)	≤ 18 (collisions found) ^[2]	0	4.99	55.00	1992
SHA-0	160	160 (5 × 32)	512	80	And, Xor, Or, Rot, Add (mod 2 ³²)	< 34 (collisions found)	0	≈ SHA-1	≈ SHA-1	1993
SHA-1						< 63 (collisions found) ^[3]		3.47	52.00	1995
SHA-2	SHA-224	224	256 (8 × 32)	64	And, Xor, Or, Rot, Shr, Add (mod 2 ³²)	112	32	7.62	84.50	2004
	SHA-256	256				128	0	7.63	85.25	2001
	SHA-384	384	512 (8 × 64)	80	And, Xor, Or, Rot, Shr, Add (mod 2 ⁶⁴)	192	128 (≤ 384)	5.12	135.75	2001
	SHA-512	512				256	0 ^[4]	5.06	135.50	2001
	SHA-512/224	224				112	288	≈ SHA-384	≈ SHA-384	2012
SHA-3	SHA-512/256	256	1600 (5 × 5 × 64)	24 ^[5]	And, Xor, Rot, Not	128	256			
	SHA3-224	224				112	448	8.12	154.25	2015
	SHA3-256	256				128	512	8.59	155.50	
	SHA3-384	384				192	768	11.06	164.00	
	SHA3-512	512				256	1024	15.88	164.00	
	SHAKE128	d (arbitrary)	1344			min(d/2, 128)	256	7.08	155.25	
	SHAKE256	d (arbitrary)				min(d/2, 256)	512	8.59	155.50	

b. Transaction and Block Generation

When in a network there is a transfer process in any form, such as data transfer, information transfer, digital asset transfer, and is carried out by two or more nodes, this activity must go through a transaction process. In the transaction process, the node that sends the transfer will complete it with information about the transaction carried out. This information will be broadcast to all nodes in the network to be validated. In the validation process each node in the network will ensure whether the information broadcast is in accordance with the provisions. All are adjusted to the agreed consensus standards. After being validated, these transactions will be added to the previous blockchain thread as described earlier [18].

How to improve the performance of the blockchain transactions is important in terms of throughput and latency. The goals are to reduce the transaction confirmation latency to milliseconds and increase throughput by ensuring a reduction in the confirmation

delay. Both ways aim to get a high throughput and low latency, which is required for drones [31].

c. The Distributed Ledger

Another important component of a blockchain network is a distributed ledger, also known as shared ledger. Distributed ledger technology is a digital system that has a data structure that is maintained and stored in each user or participant so that every transaction that occurs will be identical to each other because each participant keeps a copy [18]. The distributed ledger does not have any central place to store transaction data. Therefore, it has a decentralization feature that provides security, transparency, and trust between users in its blockchain network.

Distributed ledger originated from peer-to-peer (P2P) network architecture, in which each user communicates with each other user without any centralized entity. There are several characteristics that make decentralization in a distributed ledger different from centralization. The first character is immutable, which is obtained from utilized cryptography, which guarantees that once data is stored it cannot be changed, falsified, or modified. Because of this characteristic, decentralization has an append-only character as the next character. The third character is distributed so that the data will not be stored in one place and each user will have a copy of the data. The last is the shared character which means that the existing data will be shared among users. However, a consensus algorithm mechanism will be used to make this decentralized system work.

d. The Consensus Mechanism

The consensus mechanism can be said to be a rule of thumb about how all users or nodes participate in the blockchain. All nodes agree on a protocol so that they are not only to be a part of the blockchain but also compulsory for updating the ledger [18]. In other words, the consensus mechanism running a process that makes a decision from participants in the network where each participant in the series will build and support the best decision for them all. It is a win-win solution between them. Consensus mechanism implementation depends on the type of network in which the consensus mechanism is created, which network type of the blockchain can be public or private, and performance metrics that

become parameters such as latency and throughput that also affect how the consensus mechanism operates.

As previously discussed, in the blockchain each node stores the same transaction history. Thus, the system must guarantee the integrity of the transactions. Suppose that in our network we have 10 drones that work in a swarm formation, and of course, all drones have the same database of all transactions that have previously occurred. Then GCS releases a new drone that carries a message to be broadcast to all drones (peer-to-many scenarios). From here the questions arises. Which drone will be responsible for verifying the transaction in the form of this message to form a new block into the blockchain? How to tell if a new drone is not a malicious drone? This is the reason that a consensus algorithm is needed that is mutually agreed upon by all drones, ensures that new blocks formed are not threats, and are verified by authorized drones.

There are many consensus mechanisms that could be utilized such as Proof of Work, Proof of Elapsed Time, Proof of Activity, Proof of Stake, Directed Acyclic Graph, Proof of Burn, Proof of Weight, Proof of Capacity, Practical Byzantine Fault Tolerance, and Proof of Importance [32]. However, commonly the consensus mechanism on the blockchain is performed in three ways, those being Proof of Stake, Proof of Work, or Practical Byzantine Fault Tolerance. In the world of cryptocurrency, Proof of Work is used for mining. It works by doing computation of mathematical equations on each node. Then each node that completes the computation first will have the right to enter the latest block into the blockchain. This mechanism proves to be safe for a massive scale public blockchain but requires a lot of computational power. Using Proof of Stake, only legal nodes can perform computations to complete the computational algorithm. When compared to Proof of Work, this results in reduced power consumption and eliminates competition [18]. On the other hand, Practical Byzantine Fault Tolerance is voting-based, requiring at least of one third of the authorized nodes be Byzantine, Practical Byzantine Fault Tolerance is suitable if the network type is private. It provides improvements in throughput and efficiency. Once again, it depends on the type of blockchain in which the consensus mechanism is utilized, because each mechanism has its own strengths and weaknesses.

4. Smart Contract

A smart contract in a blockchain is a computer program or a protocol in transactions that aims to execute, manage, document legally automatically, and act on an agreement agreed upon by all participants with applicable terms and conditions. Smart contracts allow transactions and agreements to be performed by different parties in the network without requiring intermediaries' involvement, so there is no time loss, and each participant will communicate only with the participant's identity that has been validated by the smart contract and considers other interactions to be illegal or malicious [33]. The code program contains an agreement that will be distributed into the blockchain network. Moreover, a blockchain smart contract determines who can receive the data payload, when and for a limited time, and some other properties for peer-to-peer, broadcast, and report scenarios. It provides a confirmation receipt info to sender or consensus participant as appropriate.

The smart contract based on ZKP verification in the blockchain architecture is used for blockchain data communication for UAV swarm intelligence. This smart contract sends a signal to all legal nodes, then the nodes trigger the script for verification and will perform the computations without revealing the passwords of the valid nodes that will do the proof. The output will release that proves true or false. If the verification script returns false, then it would be tagged as a malicious or illegal node. Else, it will accept the transaction and will be published on the blockchain [34]. Due to the mobility of the drones. The smart contract based on ZKP verification will be equipped with a generate random number as an OTP which will have a time expiration as part of the computational proof [24].

5. Blockchain Characteristics

Blockchain has several characteristics, which include Distributed Ledger, Decentralized, Consensus, Immutability, and security. It can be said that these characteristics are mandatory because without these characteristics the system in the blockchain will not function properly. The existing characteristics produce derivative characteristics, trust, auditability, tamper-proof, transparency, and anonymity.

Immutability is obtained from the trust of nodes in the network where they carry out transaction activities and conduct the exchange of information on the blockchain, which

is guaranteed not to be tampered with or guarantee to be tamper-proof. Once it proves robust and unbreakable from malicious then it presents security. With sequential hashing, cryptography, and its decentralized nature, it will be arduous to modify the data. The operation of a blockchain network is decentralized and based on a peer-to-peer network. Decentralized means that there is no central authority to control the system because transactions that occur are broadcast to all nodes so that transparency is created [35].

A ledger is a place to record all transactions that are already verified by the authorized nodes. A distributed ledger is a copied ledger, preserved by the legal nodes as a record of transactions so that they are auditable as every transaction can be inspected and traced back to the corresponding nodes [35]. The mechanism to regulate how each node agrees on transactions that have taken place is called consensus. Consensus distinguishes blockchain from other systems. Other derived characteristics are anonymity and privacy because there is no central entity and the asymmetric encryption makes the keys to any acting node remain anonymous, while confidentiality in the network is guaranteed. It means even though an illegal node could access the data inside of the blockchain network, it would not be able to read the data without the keys from the legal nodes.

B. IPV6

An important reason for using IPv6 protocol is to avoid the limitations of IPv4. The current development of device connectivity, where everything related to the Internet or the Internet of Things (IoT) will interconnect and connect to the Internet of Everything (IoE), meaning that each user has multiple Internet-enabled devices. IPv4 is limited to about 4.3 billion IP addresses. IPv4 uses Network Address Translation (NAT) for mapping IP addresses and sharing IP addresses to address limitation, but NAT cannot be used in transport mode or on mobile devices (like drones), in which a high-speed connection is required, and NAT introduces latency. Hence, Utilizing IPv6 is a necessity. There is some evidence that IPv6 has several advantages over IPv4.

1. IPv6 Header Format

The IPv6 does not included Internet header length. The length of the IPv6 header has a fixed length of 40 bytes and the header is divided into two parts which are a fixed

header and optional extension headers. IPv6 extends the space to 128 bytes. IPv4 has only 32 bytes, so IPv6 provides a much larger number of addresses. IPv6 also removed Identification, flags, fragment offset, header checksum, and options from IPv4. Instead, IPv6 added a new field such as a flow label and changed the name and position of several fields in the IPv4, IPv6 is completed with a Maximum Transmission Unit (MTU) of 1280 bytes, UDP mandatory checksum, and fragmentation. Table 4 below shows the IPv6 header format compared to the IPv4 header [36].

Table 4. IPv6 header format compared to IPV4 header. Source: [36].

IPv6 Fixed header format																																	
Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				Traffic Class				Flow Label																							
4	32	Payload Length												Next Header								Hop Limit											
8	64	Source Address																															
12	96																																
16	128																																
20	160																																
24	192	Destination Address																															
28	224																																
32	256																																
36	288																																

Legend:

	Field's name kept from IPv4 to IPv6
	Fields not kept in IPv6
	Name and position changed in IPv6
	New field in IPv6

a. Version

The version field consists of the version number of the IP header or the version that states the IP packet, hence the name IPv6 always has the same number of 6, using 4 bits.

b. Traffic Class

Traffic Class is holds two values with a total of 8 bits, of which 6 bits are used to classify packets, and the remaining 2 bits are used for an Explicit Congestion Notification (ECN), which provides for end-to-end notification of network congestion. In contrast with IPV4, which has Differentiated Services Code Point (DSCP) that has a function to classify packets and ECN that also has the same function with ECN in IPv6.

c. Flow Label

When packets are sent from one source and flow sequentially to one or more destinations, each packet will have a unique label as the identity given by the source address, which is a non-zero label. The labels of such flow packets are called flow labels. The label enables packages to go straight to their destination and prioritize delivery. By default, flow labels are represented by the symbol 0, indicating an unordered flow of packets, and flow labels have a 20-bit value.

d. Payload Length

The Payload length is 16 bits, it is the size of the payload following the main IP header, or the packet's data portion including any extension headers, which are used if the packet payload has higher octets. In addition, there are several options from the extension header that is available. The IPv6 payload length only consists of data and does not include the IPv6 header.

e. Next Header

This field function has the same function and value (8 bits) as the protocol field in IPv4. The next header field is where you find specifications from transport layer protocols such as UDP, TCP, and ICMPv6, which are used by the payload, meaning it specifies the protocol carried in the data portion of the IPv6 packet, and it also indicates that extension header follows.

f. Hop Limit

The Hop Limit field is 8 bits, and it shows the maximum number of links a packet will traverse before being discarded. If the Hop Limit is declared as 0 and the packet is discarded, the source address will receive an ICMPv6 time exceeded message. The Hop Limit field is almost identical to the time-to-live in IPv4, except that there is no longer any correlation in this field with the time queue of packets on the router.

g. Source Address and Destination Address

Both source and destination addresses are 128 bits. The source address is the packet sender's IPv6 address, and the destination address is the packet receiver's address. Contrast that with IPv4, which is only 32 bits.

The optional extension header in IPv6 follows the main IPv6 header in that it may be required. Extension headers are in the form of a series. It starts from the next header field in the fixed header, meaning that each upcoming next header field will be denoted by the previous next header, and so on until the last next header will point to the upper layer header. Along with the Hop-by-Hop Options and Routing in the extension headers, there are also some commonly used headers as options, namely, fragmentation, Destination options, Authentication, and Encapsulated Security Payload. At the same time, Host Identity Protocol, Mobility, and Reserved headers are rarely used. Table 5 lists the extension headers [37].

Table 5. Extension header options. Source: [37].

Extension Header Options	Next Header Field Value	Description
Hop-by-Hop Options	0	Always appears in the first position in the field, and must be read by all devices on the network path
Routing	43	Methods for sending packets to a destination via a predefined route specification
Fragment	44	Contains about the parameters for datagram fragmentation, meaning it allows sender to divided the packet into several smaller packets
Authentication Header	51	Consist of cryptographic checksum calculation and information regarding authenticity of the packets
Encapsulating Security Payload	50	If used, it indicates that the payload data has been encrypted and will always be placed at the end of the extension header series
Destination Options	60	Contains padding options and information that must be examined by the destination device
Mobility	135	Parameters used with mobile IPV6
Host Identity Protocol	139	Used for Host Identity Protocol version 2
Shim6 Protocol	140	Used for Shim6 which has functions to detect failures and locates pair exploration
Reserved	253	Used for experimental purposes
Reserved	254	Used for experimental purposes

Some of the advantages of using IPv6 over IPv4 can be seen from the fact that IPv6 has auto-configuration or stateless autoconfiguration where hosts can self-generate a

routable address compared to IPv4 auto-configured addresses are usable only on the local subnet link-local and are never forwarded by a router, so that it eliminates the need for the NAT. Furthermore, IPv6 eliminates broadcasts. It uses solicited-node multicast addresses whereas, on IPv4 the Address Resolution Protocol (ARP) uses broadcasts, a fact that makes IPv6 more efficient than IPv4. IPv6 is also completed with tunneling and NAT as transition tools from IPv4 to IPv6 where tunneling makes IPv6 packets can deliver over IPv4 network only, NAT takes a different approach. In other words, the IPv6 has migration strategies that allow both protocols to coexist. Consider the reasons that IPv6 supports the growth of IoT and mobile devices and the possibility that drones based on drone swarm intelligence will proliferate. Hence, the use of IPv6 is recommended.

2. IPv6 Encrypted Data Payload and Lightweight Blockchain

IPv6 will be loaded with encrypted data payload and Lightweight Blockchain (LWBC). It will be used only once by using One Time Pad (OTP) for the protection of illegal drones, and the security of the UAV swarm. The encrypted data payload and the lightweight blockchain are each 64 bytes, so when both are added to IPv6, the total packet size is 128 bytes, equal to IPv6, computed as Encryption {LWBC (64 bytes of SHA3-256/384/512) + Encrypted Data Payload (64 bytes)} = Encrypted [128 Bytes].

C. ONE TIME PAD (OTP) BASED ON SHANNON'S OTP

For One Time Pad (OTP) a randomly generated private key is sent and used only once. To encrypt a message that will later be decrypted by the recipient, the recipient will use a matching OTP and key, a type of encryption that is guaranteed not to be cracked.

To achieve non-breaking encryption, OTP should meet several conditions such as the key has to be random, it has never been used before and after, meaning only one time. It also must remain completely secret by the connecting parties [38].

Claude Shannon proved that the message gave a truly uniformly random key that is only used one time and produces ciphertext and plaintext of the same length that can be translated and vice versa [38]. However, this theory leaves a weakness that must be overcome. It is difficult to achieve truly random data from a computer. Instead, the computer will generate

pseudorandom output, although it has the advantage of being fast in the process to get a random number generated, but the possibility of repetition will still exist. On the other hand, true random number generators tend to be slower. The second is the possibility of an adversary intercepting the OTP thus allowing them to decrypt the message. There need to be mechanisms to prevent the OTP from being reused from the computer's memory.

The Random Number Generator (RNG) can be distinguished either the Pseudo-Random Number Generator (PRNG) or the opposite, the True Random Number Generator (TRNG). Non-deterministic physical events and unpredictable events form the basis of TRNG generating random numbers. Such events include noise from drone sensor data like the multiple low-order bits (least significant bit, LSB) [39], and collecting data from accelerometers, gyroscopes, and barometers [38]. A TRNG is created based on hardware characteristics, requiring measurement, and utilized noise sources. to anticipate obtaining less randomness due to various environment dependencies. Hence, TRNG needs several processes the run which require some time. As an alternative solution, you can use a PRNG which produces a random number by deterministic algorithms, but it must be ensured that the generated random number is only used once, does not form a pattern, and is unpredictable. Treatment such as time expiration and deletion of value after the specified time runs out can assist in meeting those requirements.

Efforts to produce TRNG for drones can also be done by obtaining it from using sensors on drones such as accelerometers, gyroscopes, and barometers. The sensor controller collects data from sensors, then divides it into good bits and bad bits using a randomness test. After processing, the good bits will be converted into bytes, and then after that mixing and swapping [40].

However, to overcome the weaknesses of OTP as described above, this research thesis uses PRNG because the experiment was carried out by parametric analysis with drone simulation and did not use actual drones. The OTP on ZKP, which will be used works, with time expiration, so when the specified time runs out, the drone will regenerate a new random number. This process provides a faster, simpler process, and address the aforementioned weaknesses. On the other hand, TRNG requires a physical entropy source, and hardware device, along with timing information.

III. DESIGN OF EXPERIMENT

This chapter describes the design of the experiments conducted in this thesis. We start by describing the components of the IPv6 Lightweight Blockchain. In contrast to the typical use of blockchain on mobile devices. In this thesis, we use ZKP for verification between legal users, which will then communicate with each other via IPv6. We also use OTP to impose a time limit.

For this thesis, we simulate the operation of physical UAVs, applying parametric analysis to understand and characterize the performance of a swarm of drones utilizing IPv6 Lightweight Blockchain, specifically latency and scalability [41].

A. IPV6 LIGHTWEIGHT BLOCKCHAIN DESIGN

We set out to demonstrate that IPv6 lightweight blockchain is suitable for use with swarms of UAVs. Let's start with an overview of the framework, network topologies, and consensus model used in our experimentation with the UAV swarms.

1. Framework

In this thesis, the IPv6 lightweight blockchain framework created in this experiment uses the C++ programming language, which provides several advantages, such as having high sensitivity when implemented on low-power and memory-constrained mobile devices. Another advantage is that the same C++ code can be used beyond the simulations conducted in these, that is, the code can be installed on physical drones. The C++ also can be recompiled for use on different platforms [42].

In every UAV and the Ground Control Station (GCS) is treated as a node. Each node communicates with other nodes via TCP. All of the nodes are connected in a blockchain organization and use smart contracts that implement ZKP. Figure 6 illustrates how the blockchain guarantees that no illegal UAVs, or GCSs, will be able to interfere with the communication in the swarm's network.

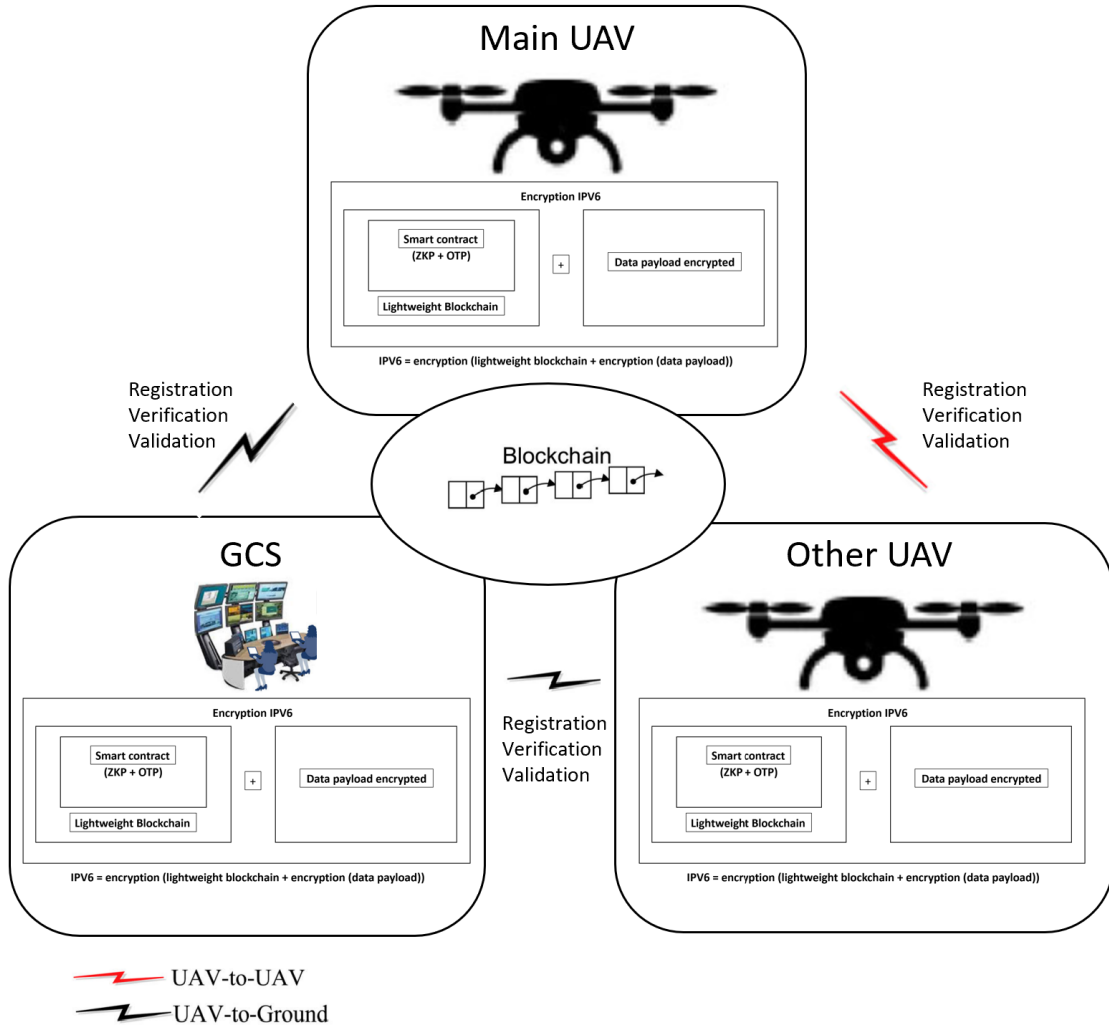


Figure 6. ZKP smart contract in blockchain guarantee the network security

UML diagrams provide space to describe the modeling of the structure, behavior, process, and architecture [43]. We use UML diagrams in this section to specify the hierarchies, networks, and relationships used in the simulations conducted for this thesis. Figure 7 shows the main class diagram consisting of several parent classes, GCS, Communication, Node, and Node Interaction.

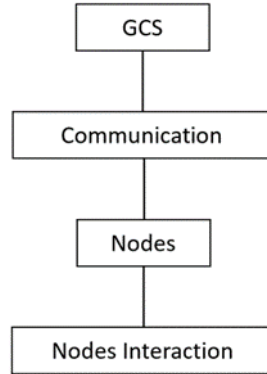


Figure 7. The class diagram shows only parent classes

The GCS class monitor and controls all drones. Each drone performs its assigned tasks and reports the completion of those tasks. The GCS authorizes each drone to register and verify the communication transactions that take the place. GCS can be both client and server nodes. The Communication class specifies how between nodes is established. The communication model uses socket programming, where the programs created include point-to-point (server-to-client) communication, as well as point-to-multipoint (server-to-multiple clients) with multithreading [44]. Figure 8 shows the send and receive data diagram between nodes.

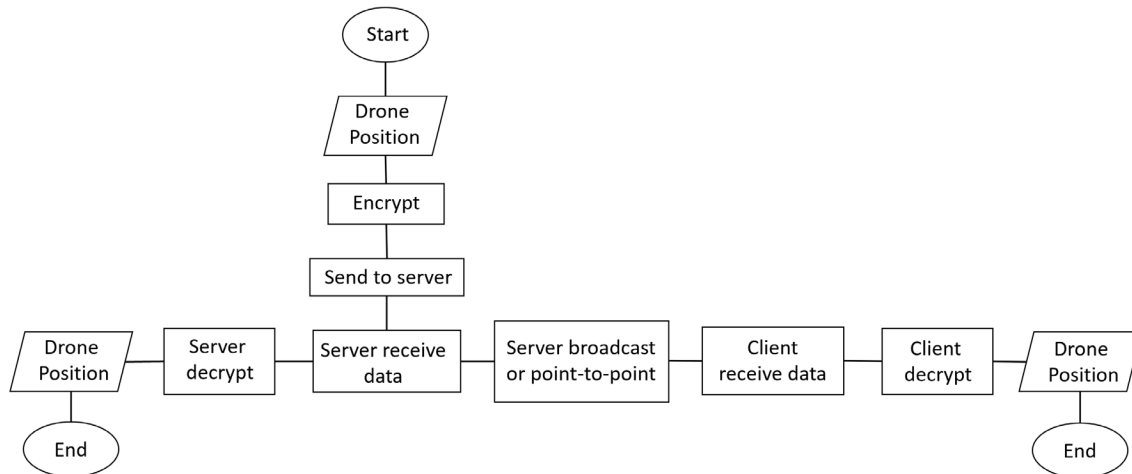


Figure 8. The send and receive data diagram

The class nodes in the class diagram are described as nodes involved in the experiment, namely GCS and UAVs. UAVs are divided into several types, namely, legal nodes that are on standby and not connected to other legal nodes, nodes that are ready for peer-to-peer connections, nodes which are ready for many-to-peer connections, nodes that are ready for broadcast mode or peer-to-many connections, and for the purposes of developing research and future testing, illegal nodes that do not belong to the network are added. The nodes interaction class specifies how the nodes interact or are connected. Furthermore, this class is divided into three sub-classes, namely peer-to-peer, peer-to-many, and many-to-peer, which correspond to the scenarios in the experiments that we conducted.

For the purposes of running the program and carrying out computations in the experiment, a CPU with 16GB RAM is used, and to anticipate the lack of computation performance, a supercomputing named Hamming [45] was used as back up planning.

2. Network Topologies

In this thesis, we experiment with several network topologies to explore the performance of communication between nodes in a blockchain-based UAV swarm intelligence network [46].

a. Star Topology

The Star topology will be used as a solution for describing the connection path between nodes. The design structure represents the positions where each node is located. Figure 9 shows the star network topology design used in this experiment. We model connections, between nodes for this topology as a wireless network.

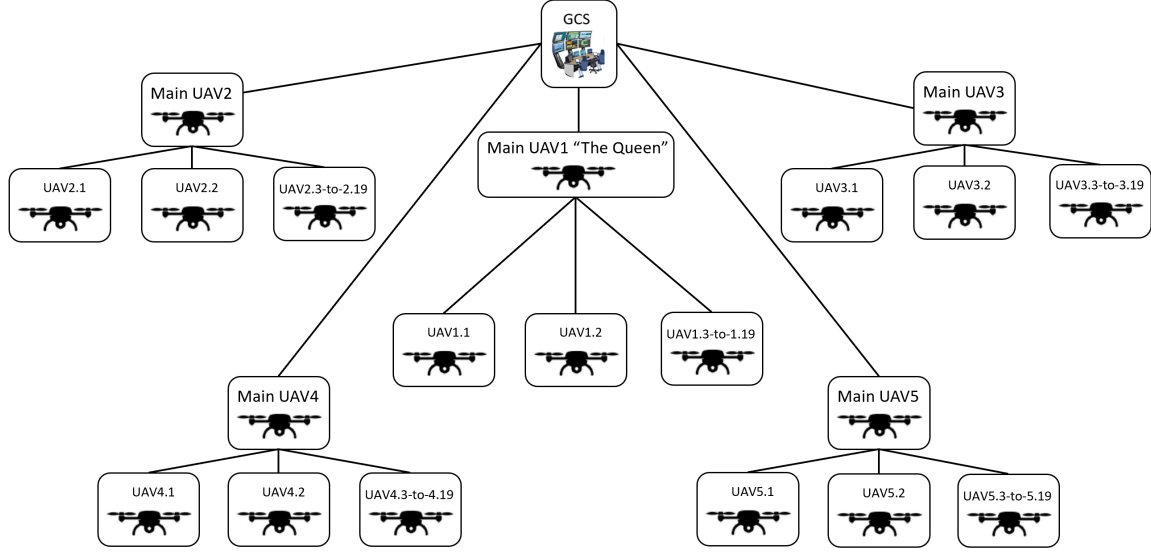


Figure 9. Star topology design

b. Point-to-point Topology

In this topology, two nodes are directly connected. This point-to-point connection describes a peer-to-peer scenario in the experiment. This connection can occur between GCS to main UAV1 (“the queen”), GCS to every Main UAV other than main UAV1 such as to main UAV2, and so on up to the main UAV5. Connections can also occur between any main UAV to one of the numbered UAVs such as main UAV2 with UAV2.1 or even among the member that comes from another main UAV division such as between UAV2.2 and UAV1.4. Many more connections can represent this peer-to-peer scenario. Figure 10 illustrates this type of topology.

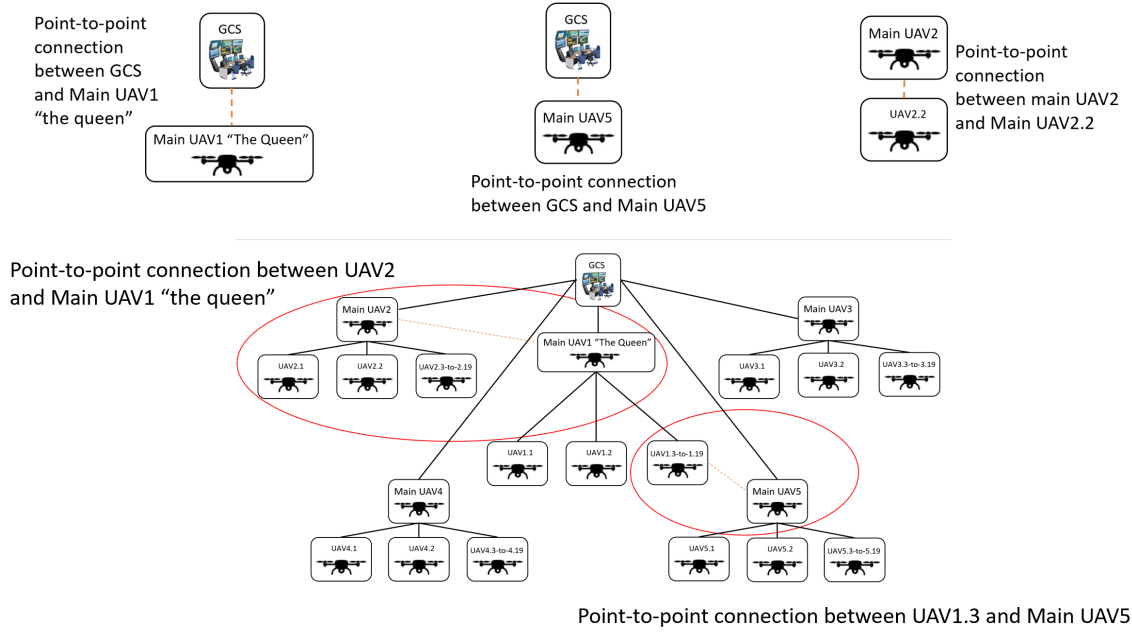


Figure 10. Point-to-point connection examples from the experiment

c. *Point-to-multipoint Topology*

In this thesis, we also experiment with a point-to-multipoint topology, where a node can connect directly with several other nodes in the network. In Figure 11, after Main UAV1 ("the queen") establishes a connection with GCS and receives data transactions in the form of a peer-to-peer scenario connection, Main UAV1 then broadcasts the existing data to other main UAVs. This communication occurs in the form of point-to-multipoint or peer-to-many, which in actual conditions this situation can occur when the GCS cannot reach the other main UAV for reasons of being out of range.

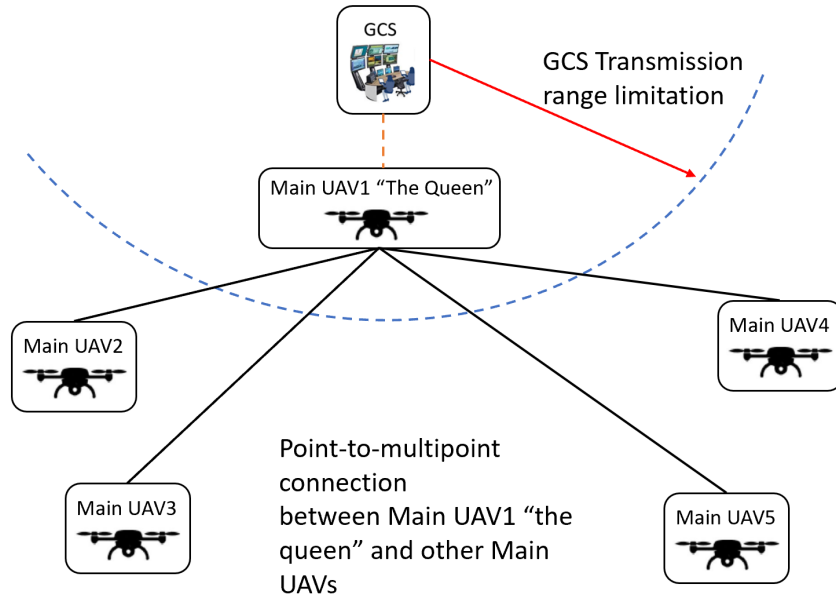


Figure 11. Point-to-multipoint topology example from the experiment

d. Mesh Topology

Mesh topology is a type of topology that describes the multiple connection paths that occur between nodes. Like the point-to-multipoint topology, where a node can establish connections with several other nodes, the difference is that these other nodes also have the same capabilities, can have multiple connection paths to several other nodes as well so that they are connected to each other in the network. For example, Main UAV2 opens multiple connection paths with UAV2.1, UAV2.2, UAV2.3, and UAV2.4. Furthermore, in this network all numbered UAVs also do the same thing, namely doing multiple path connections. To calculate how many connections are established in a network compared to the number of nodes, it can be calculated using the formula [46]:

$$\text{Number of connections} = ((\text{nodes amount}) * (\text{nodes amount} - 1))/2$$

As an example:

$$\text{Number of connections} = (5 * (5-1))/2 = 10$$

Based on the illustration above, it can be seen that the mesh topology can represent the depiction of many-to-peer and peer-to-many scenarios in the experiment at the same time. Figure 12 illustrates the mesh topology.

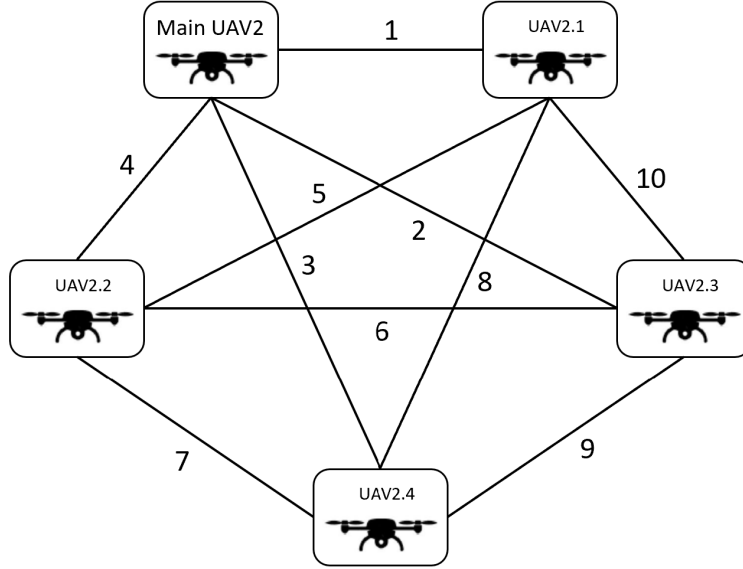


Figure 12. Mesh topology example for the experiment

e. *Hybrid Topology*

A UAV swarm can also utilize a hybrid topology, but it is not a part of the current experiment but instead is left as a future research topic. In a hybrid topology, there is a merger of two or more forms of topologies. For example, in the future, there is a development, which involves a larger organization such as the headquarters. Of course, this idea will cause the addition of a new network to accommodate the connection needs between the headquarters and the existing network. Figure 13 illustrates how a wired network in the headquarters network is connected to a wireless network from the UAV swarm, and the topology built for this new network includes a star topology and a point-to-multipoint topology so that when two networks are connected a hybrid network is created [46].

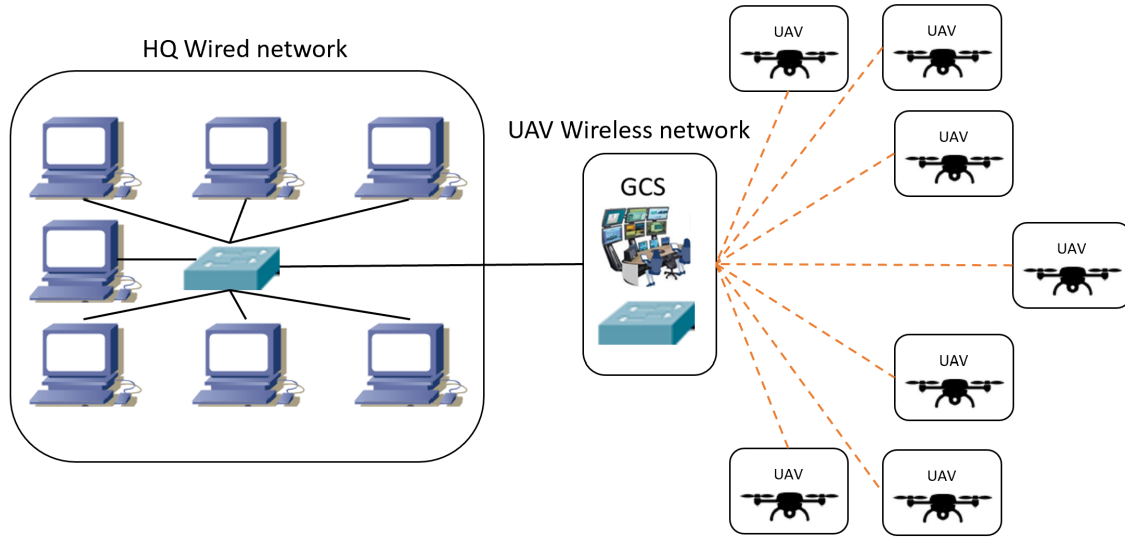


Figure 13. Hybrid topology example

3. Consensus

One way to secure communication between UAVs is to utilize a lightweight blockchain solution. In this thesis, the first conceptual model of consensus in a lightweight blockchain was developed using the Proof of Work process. Only registered nodes can solve the challenge problem. The mechanism developed in the program uses the ZKP method, which is used as a smart contract in the lightweight blockchain.

All process mechanisms are simulated in the initial settings. Each node in the network registers its username and password. Only the username is stored in memory. Each registered node will be authorized only with its username.

When a transaction occurs in the network, each server node will provide a challenge in the form of mathematical calculations to get a value of Y as a sign that the client node also knows the value of X without mentioning what the value of X is. The client node will calculate the challenge. The server node matches the calculation results from the client node by entering the Y value from the client node calculation results. It then ensures that the two calculated values are the same. From these results, the client node knows the value

of X without the need to share information with the server node. On the other hand, the server node also does not know the password of the client node.

This process aims to protect the network from interference from illegal nodes. In addition, the simple computational process coupled with OTP makes calculating latency for this experiment straightforward. Figure 14 depicts the sequence of the ZKP process in the program.

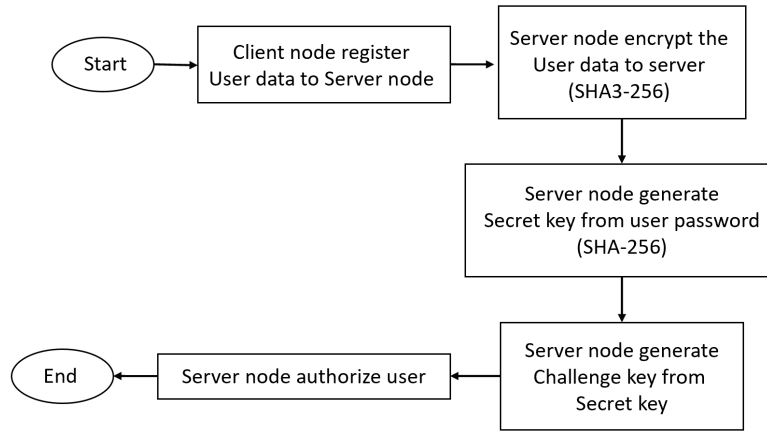


Figure 14. The ZKP process in the program

B. DESIGN IMPLEMENTATION

This section describes the development of the software for simulating the operation of UAV swarms that utilize an IPv6 lightweight blockchain.

1. Software and Tool Installation

The program code is stored in GitLab NPS which can then be retrieved via a clone git repository, and as a backup, also stored in a folder on the local computer. Since the local computer is based on Windows 10, one option is to manually install the Windows Subsystem for Linux feature, which provides a way to install the Ubuntu terminal environment on Windows and use it without leaving Windows. Then, the installation process through Windows PowerShell terminal to release Ubuntu OS version Ubuntu 20.04.4 LTS and creating additional Virtual Machine (VM) in VMware Workstation with

the same specification as a backup. Figure 15 shows the specification of Ubuntu 20.04.4 LTS.

```
dymas@DESKTOP-AB4F21R: ~  
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 4.4.0-19041-Microsoft x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
System information as of Sun Aug  7 11:55:06 PDT 2022  
  
System load:          0.52  
Usage of /home:        unknown  
Memory usage:         78%  
Swap usage:           35%  
Processes:            7  
Users logged in:       0  
IPv4 address for eth2: 172.20.208.232  
IPv4 address for eth3: 169.254.212.244  
IPv4 address for eth4: 169.254.2.204  
IPv4 address for eth5: 192.168.56.1  
IPv4 address for eth6: 172.31.176.1  
IPv4 address for wifi0: 10.0.0.243  
IPv6 address for wifi0: 2601:642:c205:35a0:4cab:6961:f93d:eb19  
IPv6 address for wifi0: 2601:642:c205:35a0::76b9  
IPv6 address for wifi0: 2601:642:c205:35a0:ac19:9259:b585:1d3f
```

Figure 15. Ubuntu 20.04.4 LTS specification

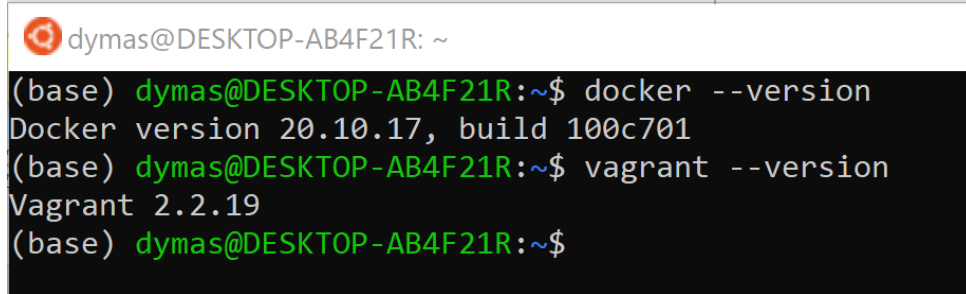
The implementation of the simulation or experiment is supported by a visual code that functions as a coding editor and code debugging for the simulation program, besides that the visual code can connect to a container and run a different operating system.

2. Computing Platforms

To obtain scalability and latency measurements for the IPv6 lightweight blockchain program, simulations were carried out on predetermined scenarios, with testing 1 to 10000 iterations mimic the transactions that occur in each experimental scenario plan. Running the tests requires a development environment that automates calculations consistently in a VM environment.

For this thesis we used Ubuntu and Visual Studio Code for hosting and developing the modelling and simulation of swarms of drones. We used references [47] and [48].to

guide our installation of Docker and Vagrant. Figure 16 is a screenshot of the information about the version of Docker and Vagrant used for the experiments conducted for this thesis.

A terminal window with a black background and white text. The prompt is 'dymas@DESKTOP-AB4F21R: ~'. The first command is '(base) dymas@DESKTOP-AB4F21R:~\$ docker --version', which outputs 'Docker version 20.10.17, build 100c701'. The second command is '(base) dymas@DESKTOP-AB4F21R:~\$ vagrant --version', which outputs 'Vagrant 2.2.19'. The prompt is then '(base) dymas@DESKTOP-AB4F21R:~\$' again.

```
dymas@DESKTOP-AB4F21R: ~  
(base) dymas@DESKTOP-AB4F21R:~$ docker --version  
Docker version 20.10.17, build 100c701  
(base) dymas@DESKTOP-AB4F21R:~$ vagrant --version  
Vagrant 2.2.19  
(base) dymas@DESKTOP-AB4F21R:~$
```

Figure 16. Docker and Vagrant version information

3. Time and Scalability Calculations

We integrated the IPv6 lightweight blockchain program into the UAV swarm intelligence network used in this thesis, then ran experiments by simulating 1 to 10,000 transactions carried out for each scenario. We then compared the resulting average time and scalability to the results of the findings in LtCdr Dwijayanto's thesis [49].

The time-taking mechanism and counter scalability measurement start immediately after transaction execution occurs and ends after the client node receives new transaction data. The calculation is expressed in script form from the coding chain and becomes a part of the coding file Server.c and is displayed on the terminal.

IV. EXPERIMENT RESULTS

This chapter discusses the implementation of the experimental design described in Chapter III and the results of conducting simulations. The results of the latency and scalability measurements are analyzed, then compared with the findings reported in [49].

A. SCENARIO PLANNING

The implementation of the simulation is divided into two stages, namely the stage of preparation for the simulation and the stage of starting the implementation of the simulation based on the scenario and its objectives. All obstacles found and solutions to overcome them are also documented in here.

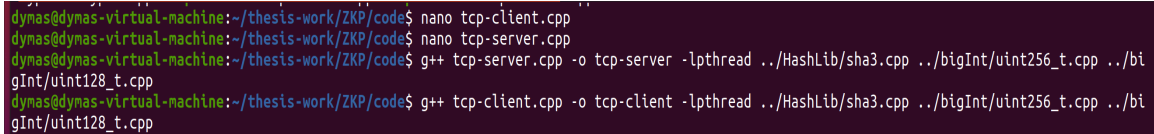
1. Simulation Preparation Stages

The preparation stage begins by downloading all the required simulation program data from Gitlab [50] into the Ubuntu for Windows folder and into the VMs that have been prepared. After the program data is ready, the next preparation step is to compile the program using the following command:

```
“g++ tcp-server.cpp -o tcp-server -lpthread ../HashLib/sha3.cpp ../bigInt/uint256_t.cpp ../bigInt/uint128_t.cpp.”
```

```
“g++ tcp-client.cpp -o tcp-client -lpthread ../HashLib/sha3.cpp ../bigInt/uint256_t.cpp ../bigInt/uint128_t.cpp.”
```

Figure 17 is a screenshot of the compiler command in the one of the VM terminals.

A screenshot of a terminal window from a virtual machine. The terminal shows a series of commands being entered and executed. The first command is 'nano tcp-client.cpp'. The second is 'nano tcp-server.cpp'. The third is 'g++ tcp-server.cpp -o tcp-server -lpthread ../HashLib/sha3.cpp ../bigInt/uint256_t.cpp ../bigInt/uint128_t.cpp'. The fourth is 'g++ tcp-client.cpp -o tcp-client -lpthread ../HashLib/sha3.cpp ../bigInt/uint256_t.cpp ../bigInt/uint128_t.cpp'. The terminal output shows the files being created or modified.

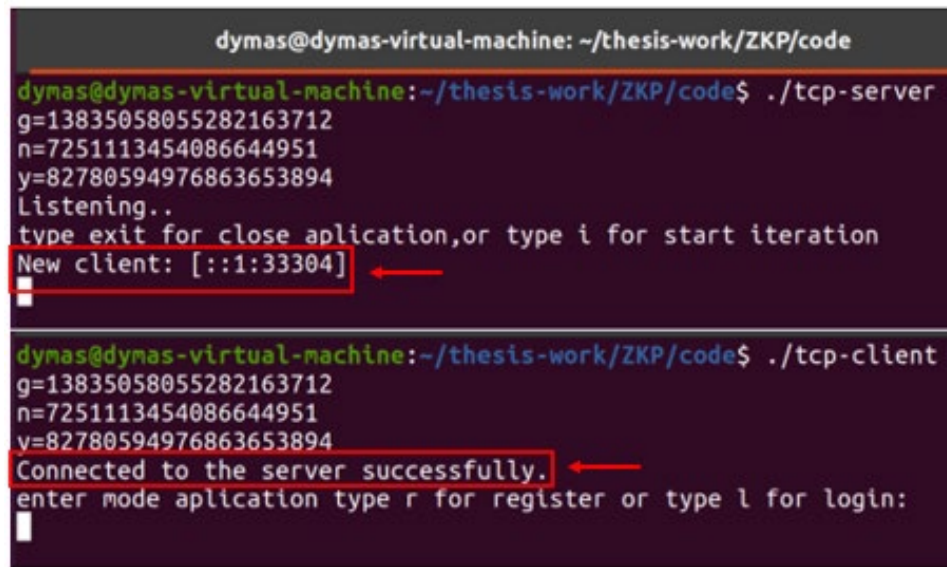
```
dynas@dynas-virtual-machine:~/thesis-work/ZKP/code$ nano tcp-client.cpp
dynas@dynas-virtual-machine:~/thesis-work/ZKP/code$ nano tcp-server.cpp
dynas@dynas-virtual-machine:~/thesis-work/ZKP/code$ g++ tcp-server.cpp -o tcp-server -lpthread ../HashLib/sha3.cpp ../bigInt/uint256_t.cpp ../bigInt/uint128_t.cpp
dynas@dynas-virtual-machine:~/thesis-work/ZKP/code$ g++ tcp-client.cpp -o tcp-client -lpthread ../HashLib/sha3.cpp ../bigInt/uint256_t.cpp ../bigInt/uint128_t.cpp
```

Figure 17. Compiler command in the VM terminal

To ensure the program works correctly, test the connection by opening two terminals on Ubuntu for Windows or using the following command from inside the VM:

- a. “./tcp-server” on terminal 1
- b. “./tcp-client” on terminal 2

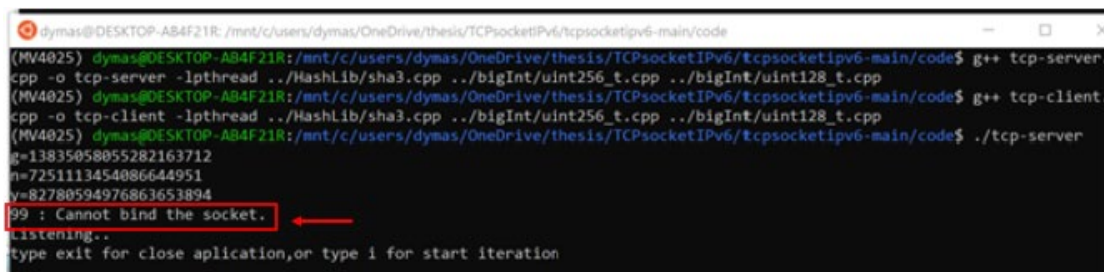
If successful, the terminal will so indicate, as shown in Figure 18.



```
dymas@dymas-virtual-machine: ~/thesis-work/ZKP/code
dymas@dymas-virtual-machine:~/thesis-work/ZKP/code$ ./tcp-server
g=13835058055282163712
n=7251113454086644951
y=82780594976863653894
Listening..
type exit for close application,or type i for start iteration
New client: [::1:33304]
dymas@dymas-virtual-machine:~/thesis-work/ZKP/code$ ./tcp-client
g=13835058055282163712
n=7251113454086644951
y=82780594976863653894
Connected to the server successfully.
enter mode application type r for register or type l for login:
```

Figure 18. Successful connection between two terminals in the VM

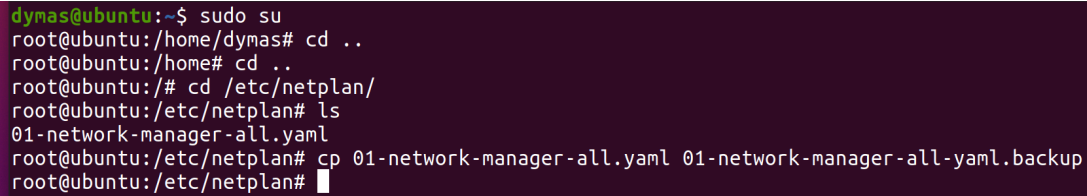
Due to the connection problem of not being able to bind socket on Ubuntu for Windows 20.04.4, the simulation was instead run using a VM. The obstacle is shown in Figure 19.



```
dymas@DESKTOP-AB4F21R: /mnt/c/users/dymas/OneDrive/thesis/TCPsocketIPv6/tcpsocketipv6-main/code
(MV4025) dymas@DESKTOP-AB4F21R:/mnt/c/users/dymas/OneDrive/thesis/TCPsocketIPv6/tcpsocketipv6-main/code$ g++ tcp-server.
cpp -o tcp-server -lpthread ../HashLib/sha3.cpp ../bigInt/uint256_t.cpp ../bigInt/uint128_t.cpp
(MV4025) dymas@DESKTOP-AB4F21R:/mnt/c/users/dymas/OneDrive/thesis/TCPsocketIPv6/tcpsocketipv6-main/code$ g++ tcp-client.
cpp -o tcp-client -lpthread ../HashLib/sha3.cpp ../bigInt/uint256_t.cpp ../bigInt/uint128_t.cpp
(MV4025) dymas@DESKTOP-AB4F21R:/mnt/c/users/dymas/OneDrive/thesis/TCPsocketIPv6/tcpsocketipv6-main/code$ ./tcp-server
g=13835058055282163712
n=7251113454086644951
y=82780594976863653894
99 : Cannot bind the socket.
Listening..
type exit for close application,or type i for start iteration
```

Figure 19. Connection failure in Ubuntu for Windows

In order for the simulation to meet the criteria for the planned scenario, each Ubuntu 20.04 VM is given an IPv6 address. The addition and setup of IPv6 addresses are done manually in the network interfaces [51]. The first thing to do is to go into the root system and add a new configuration in the 01-network-manager-all.yaml file, preceded by a backup action so that the old files remain. Sequence details can be seen in Figure 20.

A terminal window with a dark purple background and light green text. The user 'dymas' is at the prompt 'dymas@ubuntu:~\$' and runs 'sudo su' to become root. The prompt changes to 'root@ubuntu:/home/dymas#'. The user navigates through directories: 'cd ..' to '/home', 'cd ..' to '/', and 'cd /etc/netplan/'. The prompt is now 'root@ubuntu:/etc/netplan#'. The user runs 'ls' and sees '01-network-manager-all.yaml'. Finally, the user runs 'cp 01-network-manager-all.yaml 01-network-manager-all.yaml.backup' and the prompt returns to 'root@ubuntu:/etc/netplan#'.

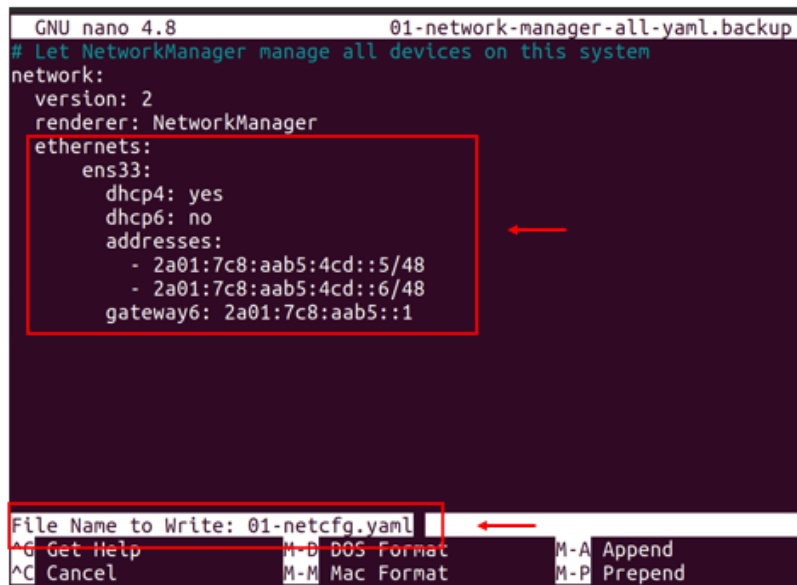
```
dymas@ubuntu:~$ sudo su
root@ubuntu:/home/dymas# cd ..
root@ubuntu:/home# cd ..
root@ubuntu:/# cd /etc/netplan/
root@ubuntu:/etc/netplan# ls
01-network-manager-all.yaml
root@ubuntu:/etc/netplan# cp 01-network-manager-all.yaml 01-network-manager-all.yaml.backup
root@ubuntu:/etc/netplan#
```

Figure 20. Back up file 01-network-manager-all.yaml

Then open the backup file 01-network-manager-all.yaml.backup with the following command:

“sudo 01-network-manager-all.yaml.backup”

Adjust the values, by providing additional IPv6 addresses and gateway6 from the additional IPv6 addresses, exit and save the changes by changing the file name to 01-netcfg.yaml as shown in Figure 21.



```
GNU nano 4.8                                01-network-manager-all-yaml.backup
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernet:
    ens33:
      dhcp4: yes
      dhcp6: no
      addresses:
        - 2a01:7c8:aab5:4cd::5/48
        - 2a01:7c8:aab5:4cd::6/48
      gateway6: 2a01:7c8:aab5::1

File Name to Write: 01-netcfg.yaml
^G Get Help      M-B DOS Format  M-A Append
^C Cancel        M-M Mac Format  M-P Prepend
```

Figure 21. Add IPv6 address and gateway 6

To guarantee that the files are actually stored in the directory, ensure that the newly added IP addresses have been entered into the configuration by issuing the following command:

- a. “ls”
- b. “sudo netplan apply”
- c. “ip a”

The order of commands can be seen in Figure 22.

```

root@ubuntu:/etc/netplan# ls
01-netcfg.yaml 01-network-manager-all.yaml
root@ubuntu:/etc/netplan# sudo netplan apply
root@ubuntu:/etc/netplan# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:8b:cd:df brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.174.135/24 brd 192.168.174.255 scope global dynamic noprefixroute ens33
        valid_lft 1798sec preferred_lft 1798sec
    inet6 2a01:7c8:aab5:4cd::6/48 scope global noprefixroute
        valid_lft forever preferred_lft forever
    inet6 2a01:7c8:aab5:4cd::5/48 scope global noprefixroute
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe8b:cddf/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
root@ubuntu:/etc/netplan#

```

Figure 22. Check updating new IP addresses

The final step in preparing for the simulation is to test the new IP address by performing a ping test from the other VM. Since this is IPv6, a ping test is added to the number 6 after the ping command, as shown in Figure 23.

In this simulation, four VMs are prepared to support the planned scenario. Each VM will play a flexible role as a node in the UAV swarm network, which means that each VM can act as either a UAV or a GCS, depending on usage in the current scenario.

```

dymas@dymas-virtual-machine:~$ ping6 2a01:7c8:aab5:4cd::1
PING 2a01:7c8:aab5:4cd::1(2a01:7c8:aab5:4cd::1) 56 data bytes
64 bytes from 2a01:7c8:aab5:4cd::1: icmp_seq=1 ttl=64 time=0.020 ms
64 bytes from 2a01:7c8:aab5:4cd::1: icmp_seq=2 ttl=64 time=0.035 ms
64 bytes from 2a01:7c8:aab5:4cd::1: icmp_seq=3 ttl=64 time=0.031 ms
64 bytes from 2a01:7c8:aab5:4cd::1: icmp_seq=4 ttl=64 time=0.043 ms
64 bytes from 2a01:7c8:aab5:4cd::1: icmp_seq=5 ttl=64 time=0.038 ms
^C
--- 2a01:7c8:aab5:4cd::1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4075ms
rtt min/avg/max/mdev = 0.020/0.033/0.043/0.007 ms
dymas@dymas-virtual-machine:~$

```

Figure 23. Ping test from another VM

2. The Implementation of the Simulation Based on Scenarios

The simulation implementation is divided into three scenarios, namely peer-to-peer, peer-to-many, and many-to-peer. The simulations of these three scenarios are run to obtain measures of latency and scalability.

a. *Peer-to-peer Scenario*

The peer-to-peer scenario illustrates the communication that occurs between two nodes in a UAV swarm. When two nodes have registered and been recorded in the memory of each participant in the network, the nodes can communicate and carry out transactions securely.

A VM with IPv6 address 2a01:7c8:aab5:4cd::1 acting as node 1 or UAV1 and a VM with IPv6 address 2a01:7c8:aab5:4cd::3 act as node 2 or UAV2. In the simulation, node 2 takes the initiative to communicate and send transactions to node 1. The number of transactions carried out is calculated in the form of iterations of 1, 10, 100, 1000, and 10000 iterations. Figure 24 illustrates the position of communication between two UAVs or peer-to-peer scenarios in the network.

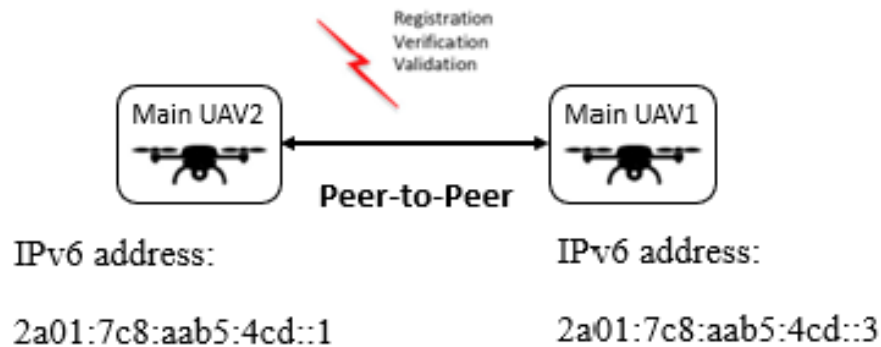


Figure 24. Illustration of peer-to-peer communication between two UAVs

After activating the two VMs that act as UAV1 and UAV2, the next step is to run the command on the terminal on the VM of UAV1:

“./tcp-server”

Then run the command on the terminal in the VM of UAV2:

“./tcp-client 2a01:7c8:aab5:4cd::1”

After connecting, there is a choice of application mode that must be selected, namely register or log in. If previously registered, the choice taken is log in. Node 2 will then enter its username and provide information on the number of transactions made. The program will display that node 2 has been authorized and validated by node 1. Figure 25 shows the terminal display results, with the two nodes connected successfully and the transaction has been successfully completed.

According to the sequence of scenarios, the first number of transactions is in one iteration. When finished and to start counting the new transaction, which is on ten iterations, there is no need to exit and reconnect. Instead, type “i” at the terminal prompt, then node 2 or UAV 2 will be ready to send transaction data again. After being given information on the number of iterations in ten times, execution begins. In the same way, by providing input iterations 100 times, the execution starts again. This applies equally to 1000 and 10000 iterations.

```
dyna@ubuntu:~/thesis-work/tcpsocketipv6-main/code$ ./tcp-client 2a01:7c8:aab5:4cd::1
g=13835058055282163712
n=725111345488644951
y=82780594976863653894
Address : 2a01:7c8:aab5:4cd::1Connected to the server successfully.
enter mode application type r for register or type l for login:
l
enter your username:
UAV2
enter number iteration data:
1
send to address -> 2a01:7c8:aab5:4cd::1
encrypted message -> l;UAV2
send encrypted message -> bDtVQVyy
type exit for exit application:
Received encrypted message ->bDtVQVyy01N1Y2Nlc3M7dGhpcyBpcyB2ZXJ5IHNLy3JldCBpbmZvIGF1dGhvcnZzZWQgdXNlcuMgb25seSEhCg==
Decrypted message ->bDtVQVyy01N1Y2Nlc3M7dGhpcyBpcyB2ZXJ5IHNLy3JldCBpbmZvIGF1dGhvcnZzZWQgdXNlcuMgb25seSEhCg==
Message from the Server: l;UAV2;Success;this is very secret info authorised users only!!

ndata[0] => l
ndata[1] => UAV2
ndata[2] => Success
ndata[3] => this is very secret info authorised users only!!

Screet Data :this is very secret info authorised users only!!

send to address -> 2a01:7c8:aab5:4cd::1
send encrypted message -> aTsxMTIuMzQ1Ozc4Ljk4Nm==
Received encrypted message ->aTTPSW==
Decrypted message ->aTTPSW==
Message from the Server: l;OK
ndata[0] => l
ndata[1] => OK
```

Figure 25. Terminal displays that show the peer-to-peer connection has succeeded

b. Peer-to-many Scenario

In this scenario, the simulation consists of a node sending transaction data to all nodes simultaneously, such as GCS sending operating commands to all existing UAVs. GCS will initiate communication first as a client and carry out data transmission transactions. In other words, broadcast a message to all nodes, which in this case involves simulating as many as 1, 10, 100, 1000, and 10000 iterations.

This simulation uses four VMs, one with IPv6 address 2a01:7c8:aab5:4cd::1 acting as the GCS node or client and the other three VMs with IPv6 addresses; 2a01:7c8:aab5:4cd::3, 2a01:7c8:aab5:4cd::5, and 2a01:7c8:aab5:4cd::7 acting as servers or the UAVs. Figure 26 illustrates the position of all nodes in peer-to-many scenarios in the network.

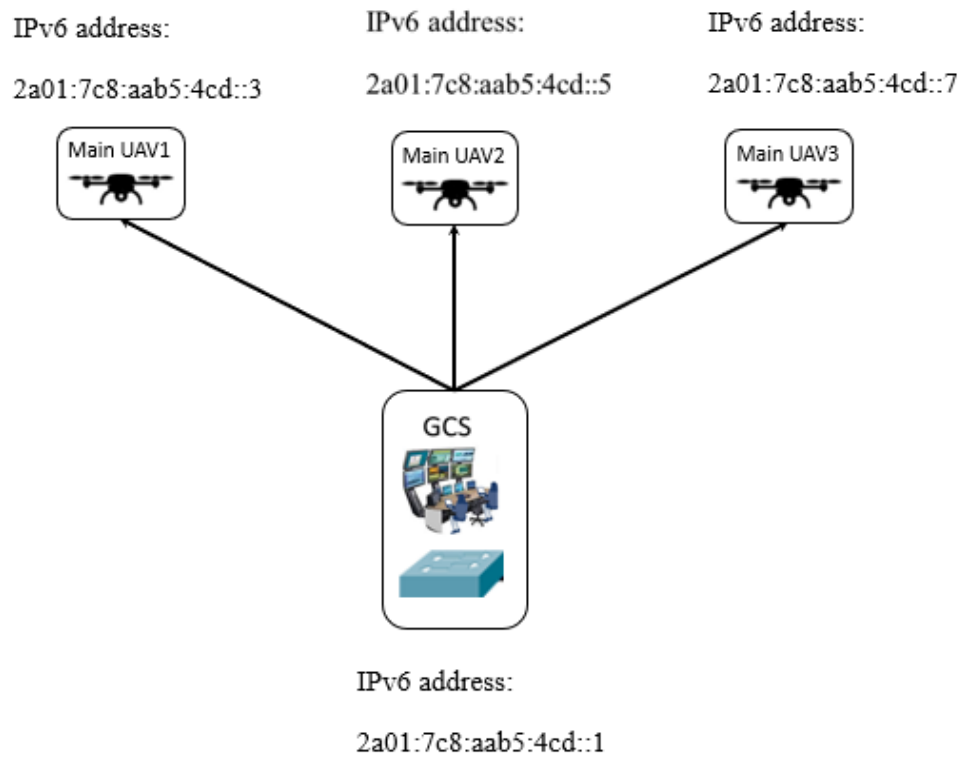


Figure 26. Illustration of the position of all nodes represented in peer-to-many scenario

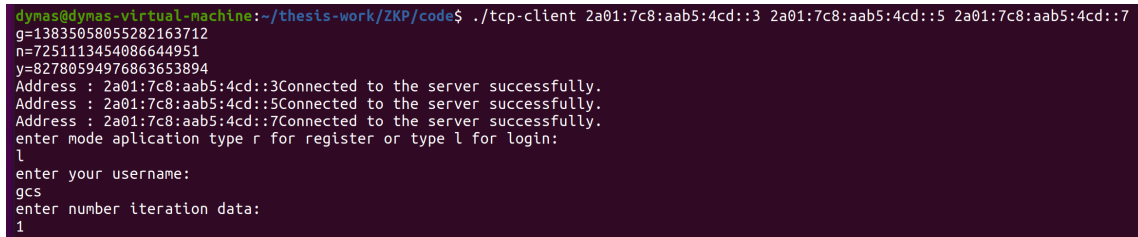
After activating the four VMs that act as GCS, UAV1, UAV2, and UAV3, the next step is to run the command on the terminal in the VM of UAV1, UAV2, and UAV3:

```
“./tcp-server”
```

Then run the command on the terminal in the VM of GCS:

```
“./tcp-client 2a01:7c8:aab5:4cd::3 2a01:7c8:aab5:4cd::5 2a01:7c8:aab5:4cd::7”
```

Figure 26 shows the command of the communication on the GCS node that acts as the party that broadcasts messages to all nodes in the network.



```
dynas@dynas-virtual-machine:~/thesis-work/ZKP/code$ ./tcp-client 2a01:7c8:aab5:4cd::3 2a01:7c8:aab5:4cd::5 2a01:7c8:aab5:4cd::7
g=13835058055282163712
n=7251113454086644951
y=82780594976863653894
Address : 2a01:7c8:aab5:4cd::3Connected to the server successfully.
Address : 2a01:7c8:aab5:4cd::5Connected to the server successfully.
Address : 2a01:7c8:aab5:4cd::7Connected to the server successfully.
enter mode aplcation type r for register or type l for login:
l
enter your username:
gcs
enter number iteration data:
1
```

Figure 27. The command to initiate the communication on the GCS node to other nodes

In this scenario, the GCS is the client node that broadcasts messages and initiates communication. When it finishes doing one iteration sequence, the GCS node continues to do the following iteration sequence. After one initial transaction or 1 iteration occurs, the program will offer to start the transaction with the following amount, as in the previous scenario, which are in sequence 10 iterations, 100 iterations, 1000 iterations, and 10000 iterations. All transactions must complete before starting the following transaction amount sequence.

c. *Many-to-peer Scenario*

The many-to-peer scenario is the opposite scenario to peer-to-many scenarios. In this scenario a node will receive transaction communication data from other nodes at the same time. The risk is that the node cannot handle this traffic which creates high latency and can even result in the possibility of losing data sent to it. This kind of situation can happen to a UAV swarm network, for that reason many-to-peer scenarios are also tested in

this simulation. The number of iterations carried out is the same, namely 1, 10, 100, 1000, and 10000.

The GCS receives communication data from UAV1, UAV2, and UAV3. This simulation uses four VMs, one with IPv6 address 2a01:7c8:aab5:4cd::1 acting as the GCS node or server and the other three VMs with IPv6 addresses; 2a01:7c8:aab5:4cd::3, 2a01:7c8:aab5:4cd::5, and 2a01:7c8:aab5:4cd::7 acting as clients or the UAVs. Figure 28 illustrates the position of all nodes in many-to-peer scenarios.

The command on the terminal VM of GCS:

“./tcp-server”

The command on the terminal VMs of UAV1, UAV2, and UAV3:

“./tcp-client 2a01:7c8:aab5:4cd::1”

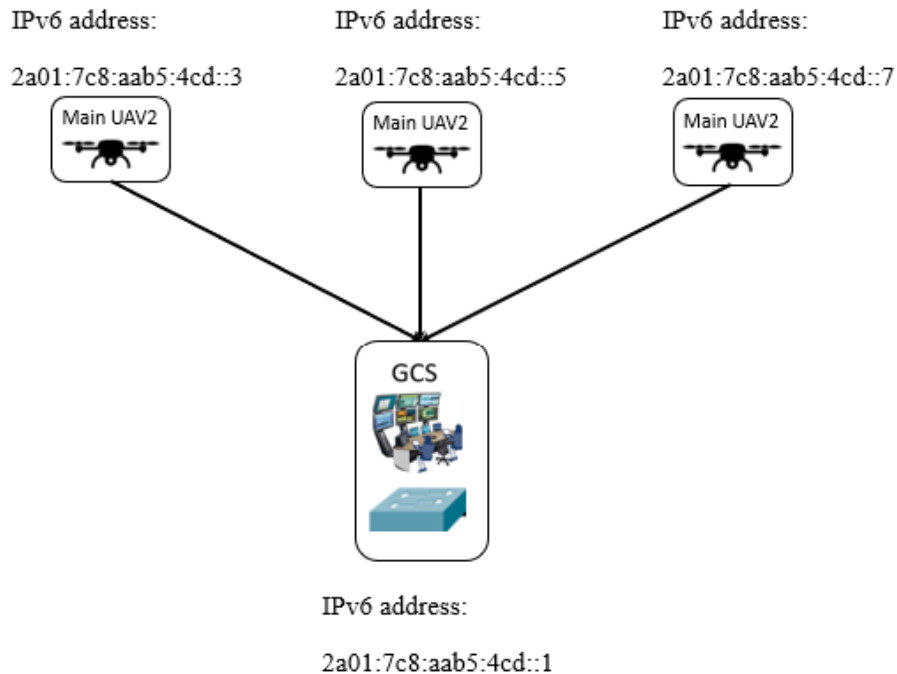


Figure 28. The illustration of the position of all nodes in many-to-peer scenarios

B. RESULTS AND ANALYSIS

The simulation of the three scenarios was successful. The results obtained from each scenario were analyzed and then compared with the results reported in [49]. The results here help us answer question of whether the implementation of the IPv6 Blockchain on the UAV swarm intelligence system can be realized.

1. Simulation Results

The first simulation results given here are those for the peer-to-peer scenario. After the two nodes are successfully connected, the following process is to send the execution of the transaction, which is iterated 1, 10, 100, 1000, and 10000 times in sequence. Figure 29 shows the results of all the iterations performed in the peer-to-peer scenario.

```

send to address -> 2a01:7c8:aab5:4cd::1
send encrypted message -> aTsxMTIuMzQ10zc4Ljk4Nw==
Received encrypted message -> aTtPSw==
Decrypted message -> aTtPSw==
Message from the Server: i;OK
mdata[0] => i
mdata[1] => OK
Time to complete 1 iteration
101(ms)
Average Time to complete 1 iteration
101(ms)

send encrypted message -> aXM7MTEyLjM0NTs30C450Q==
Received encrypted message -> aXM7T0s=
Process decrypted message from client
2a01:7c8:aab5:4cd::4:59634 => mdata[0] : is
2a01:7c8:aab5:4cd::4:59634 => mdata[1] : OK
Time to complete 10 iteration
4535(ms)
Average Time to complete 10 iteration
453(ms)

send encrypted message -> aXM7MTEyLjM0NTs30C450Tk=
Received encrypted message -> aXM7T0s=
Process decrypted message from client
2a01:7c8:aab5:4cd::4:59634 => mdata[0] : is
2a01:7c8:aab5:4cd::4:59634 => mdata[1] : OK
Time to complete 100 iteration
46926(ms)
Average Time to complete 100 iteration
469(ms)

send encrypted message -> aXM7MTEyLjM0NTs30C450Tk5
Received encrypted message -> aXM7T0s=
Process decrypted message from client
2a01:7c8:aab5:4cd::4:59634 => mdata[0] : is
2a01:7c8:aab5:4cd::4:59634 => mdata[1] : OK
Time to complete 1000 iteration
523273(ms)
Average Time to complete 1000 iteration
523(ms)

send encrypted message -> aTsxMTIuMzQ10zc4Ljk50Tk5
Received encrypted message -> aTtPSw==
Decrypted message -> aTtPSw==
Message from the Server: i;OK
mdata[0] => i
mdata[1] => OK
Time to complete 10000 iteration
8023861(ms)
Average Time to complete 10000 iteration
802(ms)

```

Figure 29. The results of all the iterations performed in the peer-to-peer scenario

The latency calculation results above are summarized in Table 6. Figure 30 shows the graph of average transactions per milliseconds for the peer-to-peer scenario.

Table 6. The latency calculation results of peer-to-peer scenario

No	Iteration	Total time (ms)	Average Time (ms)
1	1	101	101.00
2	10	4538	453.80
3	100	46926	469.26
4	1000	523273	523.27
5	10000	8023861	802.39

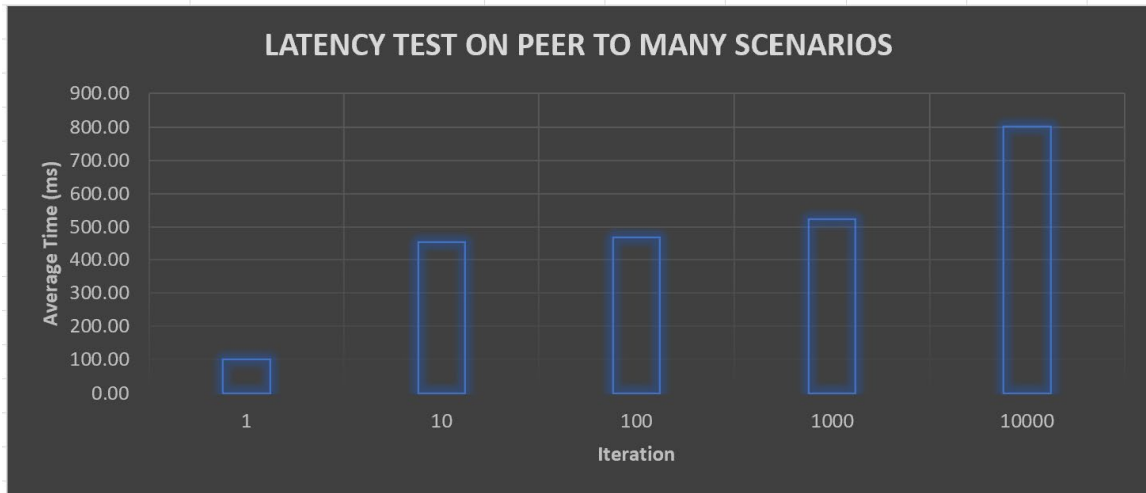


Figure 30. The average transactions per milliseconds of the peer-to-peer scenario

For the peer-to-many scenarios, after the client node is successfully connected with three server nodes, the following process is to send the execution of the transaction from the client node to the server nodes, which is iterated 1, 10, 100, 1000, and 10000 times in sequences. Figure 31 shows the results of all the iterations performed in the peer-to-many scenarios.

```

Message from the Server: i;OK
mdata[0] => i
mdata[1] => OK
Time to complete 1 iteration
667(ms)
Average Time to complete 1 iteration
667(ms)

Message from the Server: i;OK
mdata[0] => i
mdata[1] => OK
Time to complete 10 iteration
8253(ns)
Average Time to complete 10 iteration
825(ms)

Message from the Server: i;OK
mdata[0] => i
mdata[1] => OK
Time to complete 100 iteration
369227(ms)
Average Time to complete 100 iteration
3692(ms)

Message from the Server: i;OK
mdata[0] => i
mdata[1] => OK
Time to complete 1000 iteration
3482732(ns)
Average Time to complete 1000 iteration
3482(ms)

Message from the Server: i;OK
mdata[0] => i
mdata[1] => OK
Time to complete 10000 iteration
36449352(ms)
Average Time to complete 10000 iteration
3644(ms)

```

Figure 31. The results of all the iterations performed in the peer-to-many scenarios

The latency calculation results above are summarized in Table 7. Figure 32 shows the graph of average transactions per second of the peer-to-many scenarios.

Table 7. The latency calculation results of peer-to-many scenarios

No	Iteration	Total time (ms)	Average Time (ms)
1	1	667	667
2	10	8253	825.3
3	100	369227	3692.27
4	1000	3482732	3482.732
5	10000	36449352	3644.9352

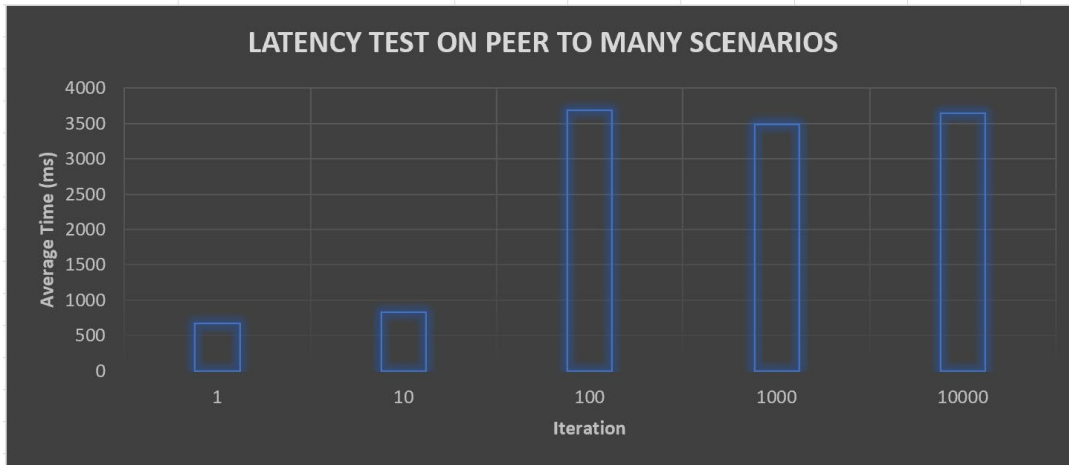


Figure 32. The average transactions per seconds of the peer-to-many scenarios

The final simulation results are from the many-to-peer scenario. In this case, each client node is connected to the server node at the same time, so the resulting latency results vary. Figure 33 shows the latency calculations for three client nodes at 1 and 10 transaction iterations, and Figure 34 shows the latency calculation results for 100, 1000, and 10000 transaction iterations.

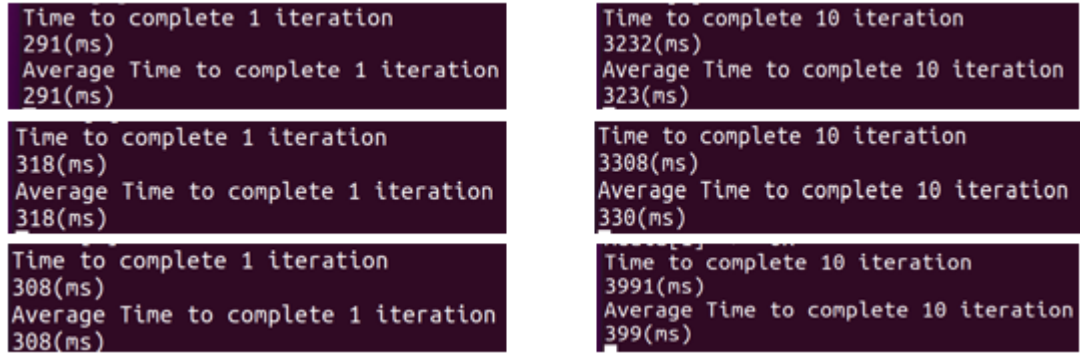


Figure 33. The latency calculation results for three client nodes at 1 and 10 transaction iterations

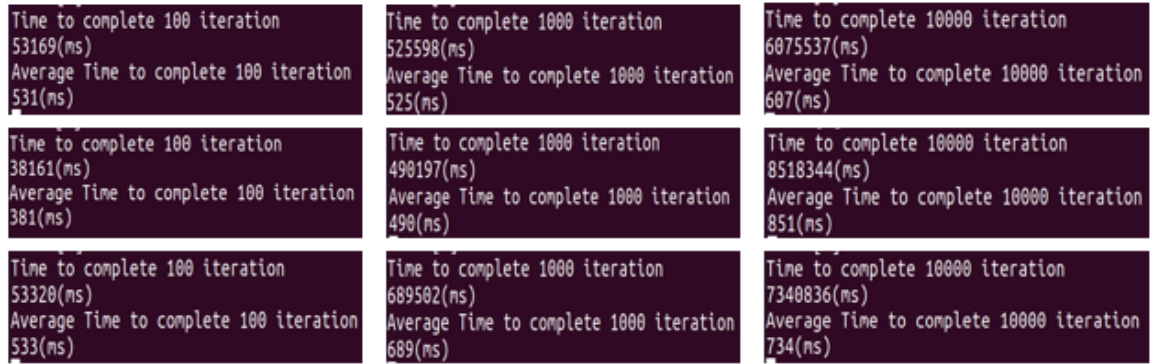


Figure 34. The latency calculation results for three client nodes at 100, 1000, and 10000 transaction iterations

The latency calculation results are summarized in Table 8. Figure 35 shows the graph of average transactions per millisecond of the many-to-peer scenario.

Table 8. The latency calculation results of many-to-peer scenario

No	Iteration	UAV1 time (ms)	UAV2 time (ms)	UAV3 time(ms)	Total time (ms)	Average Time (ms)
1	1	291	318	308	917	305.67
2	10	3232	3308	3991	10531	351.03
3	100	53169	38161	53320	144650	482.17
4	1000	525598	490197	689502	1705297	568.43
5	10000	6075537	8518344	7340836	21934717	731.16

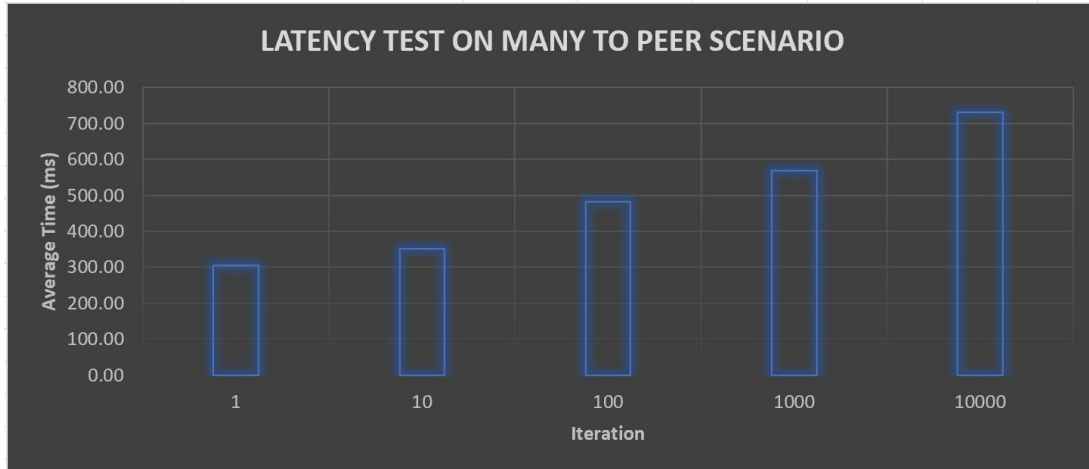


Figure 35. The average transactions per milliseconds of the many-to-peer scenario

2. Simulation Analysis

Based on the results of the simulations of the three scenarios, we found that for the peer-to-peer scenario where two nodes communicate with each other, the average transaction per millisecond increases with the number of transactions executed. This is expected because the data will be queued from one node to another, and the validation and authorization process of transactions will take time. On average, the latency for all transaction numbers is still less than one second per transaction.

In a peer-to-many scenarios, where a node broadcasts message transactions to all nodes, the average transaction per second not only increases with the number of transactions executed but also results in higher latency than for the peer-to-peer scenario. This is because the communication happens not only between two nodes, but also from one node to many nodes.

There is only a slight difference in latency results from many-to-peer scenarios compared to peer-to-peer scenarios. The results of the average transaction per millisecond can be said to be almost the same because the time difference is not much different. A significant difference is only seen in one transaction iteration, where the average number of transactions per millisecond is higher for the many-to-peer scenario due to the larger number of nodes interacting with each other than in the peer-to-peer scenario.

From the results of each of the three scenarios, we observe that the latency increases with the number of transactions, and the number of interacting nodes also causes the latency to increase, ranging from 0.1 to 3.69 seconds. However, peer-to-many, and many-to-peer scenarios result in different latency values, whereas in the many-to-peer scenario the number of nodes interacting with one node in the different number of transactions does not have a significant effect on latency changes. In addition, the results of each iteration of three scenarios vary from time to time depending on the load received from the computer at that time. Figure 36 displays a summary of the simulation results in three scenarios using the IPv6 lightweight blockchain program for UAV swarms.

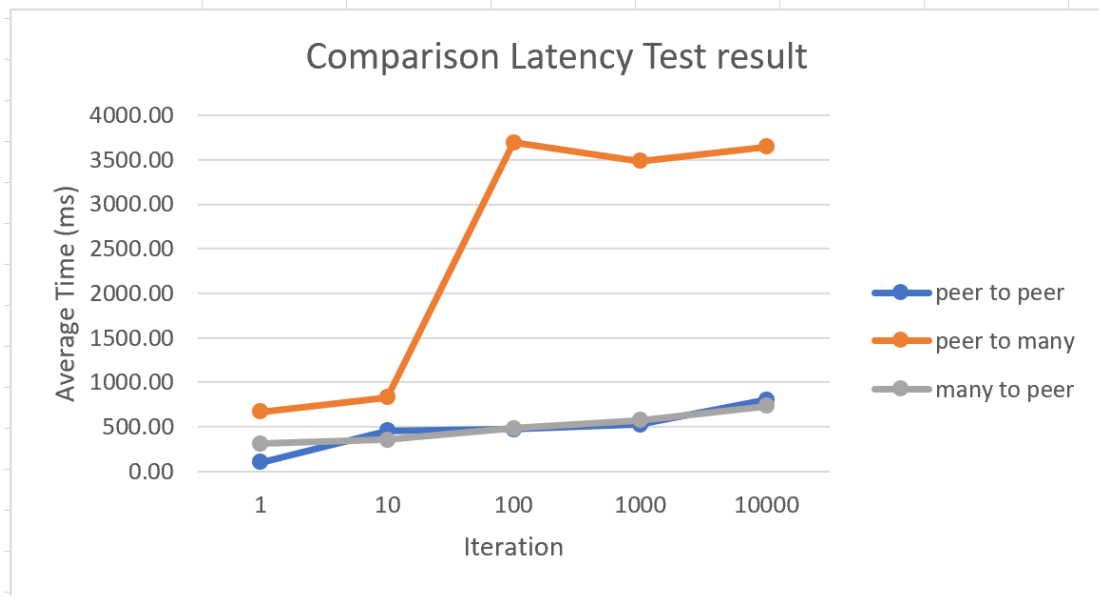


Figure 36. Comparison summary of latency test results

3. Comparison Results

The thesis reported in [49] compares the results of transactions that occur on two types of networks, namely the private IBM Hyperledger Fabric network, and the private Ethereum network. From the two networks, it is known that the addition of nodes that interact on the private IBM Hyperledger does not affect latency, but the range is around 3.4 to 3.7 seconds. On the other hand, as more nodes interact and enter transactions, the

latency of the private Ethereum network increases, ranging from 0.1 to 8 seconds. Based on the results of the simulations of the three scenarios presented in this thesis. The IPv6 Lightweight Blockchain program will have an increase in latency when running peer-to-many scenarios but does not have a significant increase when running many-to-peer scenarios, despite the latency ranges tend to be lower (0.1 to 3.69 seconds) than the two types of networks used in [49].

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION AND FUTURE WORK

This chapter contains the conclusions of the thesis and provides suggestions for future work.

A. SUMMARY OF FINDINGS

Based on the development of the IPv6 Lightweight Blockchain program which aims to simulate the operation of physical UAVs and the results of the comparison of latency and scalability that occur in the three scenarios in the simulations, we conclude the following.

The choice of the C++ programming language as the basis of the IPv6 Lightweight Blockchain program in this parametric modeling or simulation was good because it can support for real-time simulations and provides adequate execution speed. Moreover, the C++ program can be reused on a physical UAV.

In addition, based on the comparison of latency and scalability results for each scenario in the simulation, we found that for peer-to-peer scenarios and many-to-peer scenarios, there is no significant increase in latency even though in many-to-peer scenarios, the number of interacting nodes increased. In the peer-to-many scenarios where a node conducts a transaction in the form of broadcast messages to several nodes at the same time, this results in increased latency. This difference in results occurs because in the peer-to-many scenarios the execution of sending transactions from a node to all nodes is carried out concurrently, while in the many-to-peer scenario, the execution of sending transactions may not be carried out simultaneously.

Furthermore, we conclude that the use of IPv6 Lightweight Blockchain on the UAV swarm intelligence system allows for it to be used as an alternative solution that addresses the challenges of latency and scalability in data transmission communication between UAVs. This is also a keystone for opening opportunities for the further development of this program.

B. FUTURE WORK POSSIBILITIES AND SUGGESTIONS

The experiment carried out in this thesis is limited to obtaining the results of latency and scalability parameters in the context of utilizing the IPv6 Lightweight Blockchain for a UAV swarm data communication architecture. It is possible to explore other parameter in the future.

As discussed in Chapter II, the reason for using blockchain and smart contracts based on ZKP verification is to guarantee the security of communication data transactions on the UAV swarm intelligence network. However, the scope of this thesis does not include testing the security of this system. Thus, a next step would be to conduct testing to ensure the security of the communication.

As mentioned earlier, the experiments in this thesis were performed using Ubuntu VMs to host and develop the modeling and simulation of the UAV swarm. Therefore, the possibility remains that the experiments can be performed from another computer platform, such as one that consist of a Field Programmable Gate Arrays (FPGAs), or that the experiments can be rerun with real drones.

Experimental implementation can also be developed for greater scalability by utilizing docker or vagrant to maximize the calculation results, evaluating whether this approach would generalize for use on the Internet of Things (IoT).

Since the utilization of blockchain for UAV swarm communication architecture is relatively unexplored we recommend investigating the use of blockchain for other mobile device communication architecture designs.

LIST OF REFERENCES

- [1] V. Hassija *et al.*, “Fast, Reliable, and Secure Drone Communication: A Comprehensive Survey,” *IEEE Communications surveys and tutorials*, vol. 23, no. 4, pp. 2802–2832, 2021.
- [2] “A Not-So-Short History of Unmanned Aerial Vehicles (UAV),” *Consortiq*. <https://consortiq.com/uas-resources/short-history-unmanned-aerial-vehicles-uavs>.
- [3] MSW, “1845: Austria Drops Balloon Bombs on Venice,” *Weapons and Warfare*, Jun. 16, 2019. <https://weaponsandwarfare.com/2019/06/17/1845-austria-drops-balloon-bombs-on-venice/>.
- [4] M. Uddin, *Drone 101: A Must-Have Guide For Any Drone Enthusiast*. 2020.
- [5] “The dronfather,” *The Economist*. Accessed: Aug. 12, 2022. [Online]. Available: <https://www.economist.com/technology-quarterly/2012/12/01/the-dronfather>.
- [6] “General Atomics MQ-1 Predator,” *Wikipedia*. Aug. 12, 2022. Accessed: Aug. 12, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=General_Atomics_MQ-1_Predator&oldid=1104014699
- [7] S. Keerthi, A. K, and V. M.V, “Survey Paper on Swarm Intelligence,” *IJCA*, vol. 115, no. 5, pp. 8–12, Apr. 2015, doi: [10.5120/20145-2273](https://doi.org/10.5120/20145-2273).
- [8] H. Ullah, N. Gopalakrishnan Nair, M. Adrian, C. Nugent, P. Muschamp, and M. Cuevas, “5G Communication: An Overview of Vehicle-to-Everything, Drones, and Healthcare Use-cases,” *IEEE Access*, vol. 7, no. 6287639, pp. 37251–37268, Mar. 2019, doi: [10.1109/ACCESS.2019.2905347](https://doi.org/10.1109/ACCESS.2019.2905347).
- [9] M. Champion, P. Ranganathan, and S. Faruque, “Notice of Removal: A Review and Future Directions of UAV Swarm Communication Architectures,” in *2018 IEEE International Conference on Electro/Information Technology (EIT)*, May 2018, pp. 0903–0908. doi: [10.1109/EIT.2018.8500274](https://doi.org/10.1109/EIT.2018.8500274).
- [10] R. Gupta, A. Kumari, S. Tanwar, and N. Kumar, “Blockchain-Envisioned Softwarized Multi-Swarming UAVs to Tackle COVID-I9 Situations,” *IEEE Network*, vol. 35, no. 2, pp. 160–167, Mar. 2021, doi: [10.1109/MNET.011.2000439](https://doi.org/10.1109/MNET.011.2000439).
- [11] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, “Blockchain,” *Bus Inf Syst Eng*, vol. 59, no. 3, pp. 183–187, Jun. 2017, doi: [10.1007/s12599-017-0467-3](https://doi.org/10.1007/s12599-017-0467-3).
- [12] G. Iredale, “What Are The Different Types of Blockchain Technology?,” *101 Blockchains*, Jan. 06, 2021. <https://101blockchains.com/types-of-blockchain/>

- [13] D. Chong, “Three Main Types of Blockchain Technology Explained,” *Blockthatchain*, Sep. 29, 2021. <https://blockthatchain.com/three-main-types-of-blockchain-technology-explained/>
- [14] “TOP CRYPTOCURRENCIES WITH THEIR HIGH TRANSACTION SPEEDS.” <https://www.blockchain-council.org/cryptocurrency/top-cryptocurrencies-with-their-high-transaction-speeds/>.
- [15] “Transactions Speeds: How Do Cryptocurrencies Stack Up To Visa or PayPal?.” *HowMuch*. <https://howmuch.net/articles/crypto-transaction-speeds-compared>.
- [16] R. Bowden, H. P. Keeler, A. Krzesinski, and P. Taylor, “Block arrivals in the Bitcoin blockchain,” *undefined*, 2018, Accessed: Aug. 12, 2022. [Online]. Available: <https://www.semanticscholar.org/reader/0eecebe4aef723dd7f35c2aabaec623aebe87a038f>
- [17] I. J. Jensen, D. F. Selvaraj, and P. Ranganathan, “Blockchain Technology for Networked Swarms of Unmanned Aerial Vehicles (UAVs),” in *2019 IEEE 20th International Symposium on “A World of Wireless, Mobile and Multimedia Networks” (WoWMoM)*, Jun. 2019, pp. 1–7. doi: [10.1109/WoWMoM.2019.8793027](https://doi.org/10.1109/WoWMoM.2019.8793027).
- [18] D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, and G. Das, “Everything You Wanted to Know About the Blockchain: Its Promise, Components, Processes, and Problems,” *IEEE Consumer Electronics Magazine*, vol. 7, no. 4, pp. 6–14, Jul. 2018, doi: [10.1109/MCE.2018.2816299](https://doi.org/10.1109/MCE.2018.2816299).
- [19] A. C.-F. Chan, “Distributed symmetric key management for mobile ad hoc networks,” in *IEEE INFOCOM 2004*, Mar. 2004, vol. 4, pp. 2414–2424 vol.4. doi: [10.1109/INFCOM.2004.1354663](https://doi.org/10.1109/INFCOM.2004.1354663).
- [20] S. Capkun, L. Buttyan, and J.-P. Hubaux, “Self-organized public-key management for mobile ad hoc networks,” *IEEE Transactions on Mobile Computing*, vol. 2, no. 1, pp. 52–64, Jan. 2003, doi: [10.1109/TMC.2003.1195151](https://doi.org/10.1109/TMC.2003.1195151).
- [21] “The Trend Towards Blockchain Privacy: Zero Knowledge Proofs - CoinDesk.” <https://www.coindesk.com/markets/2016/09/11/the-trend-towards-blockchain-privacy-zero-knowledge-proofs/>.
- [22] “Encryption Algorithms Explained with Examples,” *freeCodeCamp.org*, May 01, 2020. <https://www.freecodecamp.org/news/understanding-encryption-algorithms/>
- [23] H. Wu, “libsnaark-tutorial.” [Online]. Available: <https://github.com/howardwu/libsnaark-tutorial>

- [24] F. Martín-Fernández, P. Caballero-Gil, and C. Caballero-Gil, “Authentication Based on Non-Interactive Zero-Knowledge Proofs for the Internet of Things,” *Sensors (Basel)*, vol. 16, no. 1, p. 75, Jan. 2016, doi: [10.3390/s16010075](https://doi.org/10.3390/s16010075).
- [25] J. Hasan, “Overview and Applications of Zero Knowledge Proof (ZKP),” vol. 8, no. 5, p. 5, 2019.
- [26] “Decrypting Cryptography: Hash Functions,” *ZK Podcast*, Apr. 12, 2021. <https://zeroknowledge.fm/decrypting-cryptography-hash-functions/>.
- [27] “Secure Hash Algorithms | Brilliant Math & Science Wiki.” <https://brilliant.org/wiki/secure-hashing-algorithms/>
- [28] “Announcing Approval of Federal Information Processing Standard (FIPS) 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, and Revision of the Applicability Clause of FIPS 180–4, Secure Hash Standard,” *Federal Register*, Aug. 05, 2015. <https://www.federalregister.gov/documents/2015/08/05/2015-19181/announcing-approval-of-federal-information-processing-standard-fips-202-sha-3-standard>.
- [29] X. Guo *et al.*, “Pre-silicon Characterization of NIST SHA-3 Final Round Candidates,” in *2011 14th Euromicro Conference on Digital System Design*, Aug. 2011, pp. 535–542. doi: [10.1109/DSD.2011.74](https://doi.org/10.1109/DSD.2011.74).
- [30] “Secure Hash Algorithms,” *Wikipedia*. Jun. 25, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Secure_Hash_Algorithms&oldid=1094940176.
- [31] H. Huang, W. Kong, S. Zhou, Z. Zheng, and S. Guo, “A Survey of State-of-the-Art on Blockchains: Theories, Modelings, and Tools,” *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–42, Mar. 2022, doi: [10.1145/3441692](https://doi.org/10.1145/3441692).
- [32] H. Anwar, “Consensus Algorithms: The Root of Blockchain Technology,” *101 Blockchains*, Aug. 25, 2018. <https://101blockchains.com/consensus-algorithms-blockchain/>.
- [33] K. Khacef and G. Pujolle, “Secure Peer-to-Peer Communication Based on Blockchain,” in *Web, Artificial Intelligence and Network Applications*, Cham, 2019, pp. 662–672. doi: [10.1007/978-3-030-15035-8_64](https://doi.org/10.1007/978-3-030-15035-8_64).
- [34] “CoPS-Cooperative Provenance System with ZKP using Ethereum Blockchain Smart Contracts,” *International Journal of Distributed Systems and Technologies*, Available: <https://dlnext.acm.org/doi/10.4018/IJDST.2018100103>

- [35] J. Moubarak, E. Filiol, and M. Chamoun, "On blockchain security and relevant attacks," in *2018 IEEE Middle East and North Africa Communications Conference (MENACOMM)*, Apr. 2018, pp. 1–6. doi: [10.1109/MENACOMM.2018.8371010](https://doi.org/10.1109/MENACOMM.2018.8371010).
- [36] O. Team, "IPv6 Header Explained," *Open4Tech*, Jun. 15, 2020. <https://open4tech.com/ipv6-header-explained/>.
- [37] "IPv6 packet," *Wikipedia*. Jul. 25, 2022. Available: https://en.wikipedia.org/w/index.php?title=IPv6_packet&oldid=1100437694
- [38] C. E. Shannon, "Communication theory of secrecy systems," *The Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, Oct. 1949, doi: [10.1002/j.1538-7305.1949.tb00928.x](https://doi.org/10.1002/j.1538-7305.1949.tb00928.x).
- [39] F. S. Alshaya, "Leveraging the NPS Femto satellite for alternative satellite communication networks." Monterey, California: Naval Postgraduate School.
- [40] S.-M. Cho, E. Hong, and S.-H. Seo, "Random Number Generator Using Sensors for Drone," *IEEE Access*, vol. 8, pp. 30343–30354, 2020, doi: [10.1109/ACCESS.2020.2972958](https://doi.org/10.1109/ACCESS.2020.2972958).
- [41] "Parametric Analysis vs. Optimization," *Default*. <https://www.altair.com/newsroom/articles/Parametric-Analysis-vs-Optimization>.
- [42] "Why use C++ to build smart IoT app development projects." <https://evontech.com/component/easyblog/why-use-c-to-build-smart-iot-app-development-projects.html?Itemid=159>
- [43] "Unified Modeling Language," *Wikipedia*. Jun. 13, 2022. Available: https://en.wikipedia.org/w/index.php?title=Unified_Modeling_Language&oldid=1092876785
- [44] "Handling multiple clients on server with multithreading using Socket Programming in C/C++," *GeeksforGeeks*, Aug. 10, 2021. <https://www.geeksforgeeks.org/handling-multiple-clients-on-server-with-multithreading-using-socket-programming-in-c-cpp/>.
- [45] "Information Technology and Communications Services at the Naval Postgraduate School." <https://nps.edu/web/technology/hpc>.
- [46] "Network Topologies Explained with Examples," *ComputerNetworkingNotes*. <https://www.computernetworkingnotes.com/networking-tutorials/network-topologies-explained-with-examples.html>.
- [47] "How To Install Docker on Ubuntu 20.04," *Knowledge Base by phoenixNAP*, Sep. 08, 2020. <https://phoenixnap.com/kb/install-docker-on-ubuntu-20-04>.

- [48] “How to Install Vagrant and use it with VirtualBox on Ubuntu 20.04,” *HowtoForge*. <https://www.howtoforge.com/how-to-install-vagrant-on-ubuntu-20-04/>.
- [49] N. E. Dwijayanto, “Comparison research for OS Ethereum blockchain and IBM enterprise-level hyperledger technology applied in autonomous Navy unclassified software distribution based on block time and scalability,” Naval Postgraduate School, Monterey, CA, 2020. [Online]. Available: <https://calhoun.nps.edu/handle/10945/66068>
- [50] “Dymas, Dymas (FORNATL, ID) / Thesis Work GitLab,” *GitLab*. <https://gitlab.nps.edu/dymas.dymas.id/thesis-work>
- [51] “Adding an IPv4 or IPv6 address in Ubuntu 20.04 | TransIP.” https://www.transip.eu/knowledgebase/entry/3412-adding-ipv4-ipv6-address-ubuntu/#adding_an_ipv6_address_in_ubuntu_20_04

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California