Department of Electrical and Electronic Engineering

MSc in Communication Networks and Signal Processing

Self-Supervised Security in Multi-Agent Systems

Wenbo Wu
io21103

September 2022

Abstract

In recent years, smart logistics and warehousing have gained the attention of many pioneering logistics companies. Multi-Agent System, the main force behind the realization of smart warehouses, continues to move towards a high degree of autonomy and is being deployed in warehouses to make them unmanned. However, developing a fully intelligent warehouse system requires the collaboration of multiple robots with different functions, which may come from different manufacturers. It is challenging to ensure that the various robots from different manufacturers in the system complete their tasks honestly. With this concern, we developed a multi-robot collaboration framework called Self-Supervised Multi-Agent Robotic System to prevent four main types of attacks: identity impersonation, malicious intruders, intra-fraud and Sybil attacks based on distributed ledger technology and asymmetric cryptography algorithms. We have validated the robustness and promptness of the framework in detecting malicious robots through a self-developed simulation platform. The batch simulation results show that the system can detect all the malicious intruders within the first 4% of the period of the system runtime and locate all the illegitimate robots which submitted fake transactions to the distributed ledger within the first 20% of the period of the system runtime.

Dedication and acknowledgements

This project is supported by both the University of Bristol and Toshiba BRIL. Many thanks to Marius Jurt, Research Engineer at Toshiba, for his great help throughout the project cycle, and Ben Holden, Senior Research Engineer at Toshiba, for his insightful suggestions. Secondly, I would like to thank Kevin Morris, a Professor at the University of Bristol, for his valuable guidance on school businesses. Also, it was a pleasure to meet several interns at Toshiba and to have their support in life and studies. Last but not least, I would like to thank my family for their all-around support over the years.

DECLARATION AND DISCLAIMER

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Postgraduate Programmes and that it has not been submitted for any other academic award.

Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted from the work of others, I have included the source in the references/bibliography.

Any views expressed in the dissertation are those of the author.

The author confirms that the printed copy and the electronic version of this thesis are identical.

SIGNED: ..... WENBO WU .....
DATE: ..... 10/09/2022 .....

Table of Contents

List of Tables

List of Figures

Figure **Page**

Nomenclature

CA Communication Amount

$DAG - DLT$ Directed Acyclic Graph based Distributed Ledger Technology

DLT Distributed Ledger Technology

ECC Elliptic Curve Cryptography

$ECDSA$ Elliptic Curve Digital Signature Algorithm

IoT Internet of Things

$IOTA$ An Open, Feeless Data And Value Transfer Protocol

LCP Linear Consensus Protocol

MAS Multi-Agent System

MRS Multi-Robot System

MT Merkle Tree

PoS Power of Stake

PoW Power of Work

ROS Robot Operating System

RSA Rivest Shamir Adleman

$SMARS$ Self-Supervised Multi-Agent Robotic System

UAV Unmanned Aerial Vehicle

$W - MSR$ Weighted-Mean-Subsequence-Reduce

WoT Web of Trust

Chapter 1

Introduction

This chapter introduces Multi-agent systems (MASs), focusing on their development, properties and applications. By analysing the applications of MAS in logistics and warehousing, it lists the potential threats and proposes a secure framework for self-supervised multi-robot collaboration based on distributed ledger technology (DLT) and cryptography.

1.1 Motivation

Multi-agent system (MAS) [6] is a relatively new technology that organizes a large number of robots to build an autonomous system to perform tasks that a single robot cannot complete. Compared to traditional robotics, MAS is more intelligent in terms of task execution and cooperation. Such a system has numerous applications ranging from agriculture (e.g., SwarmFarm [1]), military [7], healthcare [8], logistics [9] to environment monitoring [10]. Compared to other fields in the industry, MAS has been deployed by most pioneering logistics companies in the field of warehousing and delivering to substitute humans for performing tedious and dangerous tasks [11]. Each item has to go through different processes such as picking, packing and dispatching in a warehouse before it can be dispatched. In order to build a smart warehouse, there will be a variety of robots with different functions (e.g., parcel shipping robots, shelf shipping robots, unmanned aerial vehicle (UAV), robot arm) to be added to the MAS to perform these processes. However, robots with different functions may come from different manufacturers. For these robots to cooperate and achieve full autonomy securely in the warehouse, some potential security threads (e.g., malicious intruder, identity management, communication) should be considered when designing a MAS [9].

Ferrer et al., [2] introduced a novel method to encapsulate cooperative robotic missions in an authenticated data structure known as Merkle Tree. This approach allows the robot to complete the task as expected while the task information is confidential. Strobel et al. proposed a solution for managing Byzantine robots via Blockchain technology in a collective

[1] SwarmFarm Robotics website: `http://www.swarmfarm.com/`

decision-making scenario [3]. This approach can be used in consensus-reaching scenarios such as collective mapping. However, there is still a gap in research on ensuring that each robot in the MAS performs its respective task as expected. In this project, we build a Self-supervised Multi-Agent Robotic System (SMARS) for logistics/warehouse scenarios to validate if each robot completes its task as expected or not. This system is designed to prevent attacks such as impersonation, intra-fraud and malicious intruders by combining currently available technologies such as distributed ledger technology (DLT) [12] and public-key cryptography [1]. The study [13] gives a vital clue that directed acyclic graph (DAG) based DLT is feasible in a multi-agent system. Additionally, a reputation-based incentive strategy is integrated into SMARS to prevent Sybil attacks. Below are the brief definitions of all four attack scenarios aforementioned:

- **Intra-fraud** can be defined as a legitimate robot in the system claiming to have performed a task when it did not. If a robot is physically captured, tampered with, and re-introduced into the system, intra-fraud may occur.
- **Malicious intruders** can be robots without a valid identity in the MAS management system and move the objects in the warehouse randomly to disrupt the task executions of the legitimate robots.
- **Identity impersonation** is defined as the attempt by some malicious robots to use the identity of a legitimate robot and the public key to perform actions that reduce the system's efficiency or submit fake transactions.
- **Sybil Attacks** [14] is another way to reduce the efficiency of a robotic system by submitting a large number of invalid transactions without performing a task since the legitimate robots in the system need to verify if the transactions in the DLT graph are valid or not.

Besides these four kinds of attacks, problems such as malfunctioning and self-failure can also cause a robot not to work as expected.

1.2 Aim and Objectives

This project aims to design a Self-supervised Multi-Agent Robotic System (SMARS) using distributed ledger technology (DLT) and asymmetric cryptography to prevent multiple potential security attacks (e.g., identity impersonation, malicious intruder, intra-fraud, sybil attack) in warehousing and logistics scenarios.

The objectives of this project are:

- Develop a multi-robot simulation platform in Python to simulate robots transporting objects within a warehouse. Some key parameters can be customized to fulfil different test requirements.

- Design an asymmetric cryptography-based robotic identity management system to prevent identity impersonation and repudiation. In addition, robots can sign messages to ensure their integrity by using a private key.

- Develop a DAG-DLT-based distributed ledger system to record all the task execution records of robots in the form of transactions and supply a way for robots to verify the work of other robots.

- The timeliness and accuracy of the system in detecting malicious robots will be evaluated by comparing the presence or absence of malicious intruders or illegitimate robots in the warehouse.

1.3 Key Contributions

The following four main features have been developed and integrated into the SMARS simulator to prevent the four attacks mentioned in section 1.1.

Public Key Cryptography based Identity System Asymmetric cryptography algorithms [1] available for Internet of Things (IoT) devices were evaluated against their relative power consumption and security to select the right one for SMARS. Then we designed an identity management system based on the selected asymmetric cryptography algorithm to to supply identity authentication and transaction integrity verification that are desirable in SMARS. Each robot in a SMARS has a unique identity to execute a task and submit the task execution record (as a transaction) to the distributed ledger. The successive robots that submit transactions or the idle robots need to verify the task execution records recorded in the distributed ledger and ensure the reliability of the ledger under a certain incentive strategy.

DAG-DLT based Task Execution Ledger We analyzed different distributed ledger technologies available for IoT devices and, inspired by IOTA [15], which has several features desired in a robotic system, developed a distributed ledger system for SMARS. In our system design, Directed Acyclic Graph-based Distributed Ledger Technology (DAG-DLT) is mainly used to record the task execution records (in the form of a transaction) of a robot and provide an effective manner to verify whether a robot has completed the task execution record it submitted (as depicted in Fig. 1.1).

Fig. 1.2 depicts the form of a DAG-based distributed ledger with transaction contents. For instance, No.002 is the index of a transaction, 'h' represents hash calculation, 'R001' the code of the object carried by Robot2 and 'R' means an ordinary object which should be delivered to warehouse 'A' (in comparison, 'E' represents a different kind of object should be delivered to warehouse 'B'), 'PK' means the public key of a robot, time will be the real-time a transaction

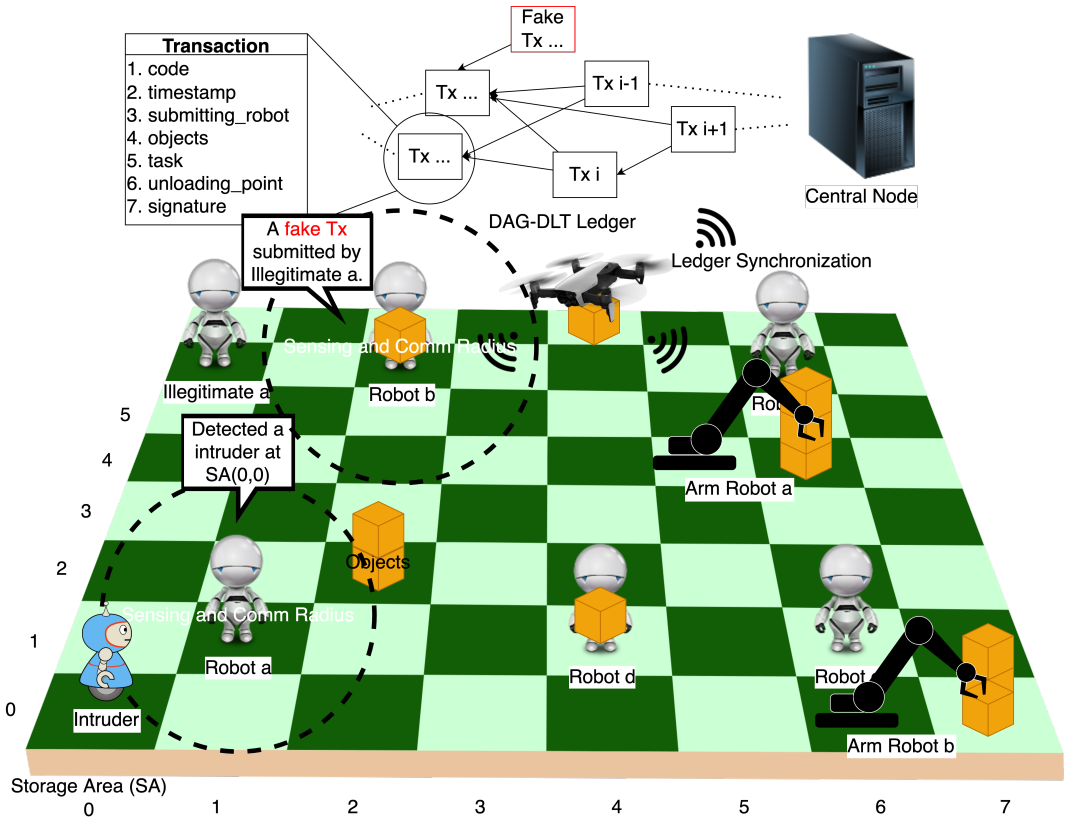


Figure 1.1: Self-Supervised Multi-Agent Robotic System (SMARS)

is submitted. With this verification system, intra-fraud in a MAS can be detected in time to ensure unpredictable losses.

Reputation based Incentive System A reputation-based incentive system has been developed to prevent robots from submitting an unlimited number of transactions to the distributed ledger without performing a task. The initial reputation of robots are fixed and a certain amount of reputation is spent when submitting a transaction to the ledger. If a transaction is verified as valid by another robot, the submitter will receive a reputation value equal to the task weight. Conversely, the reputation value spent on submitting the transaction is lost as a penalty.

Physical and Cryptographical Verification Processes As we developed a DAG-DLT ledger system to record transactions submitted by robots, we also needed a way for robots to validate these transactions. Therefore, we designed a piggyback verification approach after taking into account the resource constraints of the robot as well as the security of the verification. A robot can simultaneously scan objects in the surrounding storage areas during task execution and confirm whether the scanned items need to be verified by searching the local ledger. If there is a transaction in the ledger, the robot verifies that it is at the location recorded in the

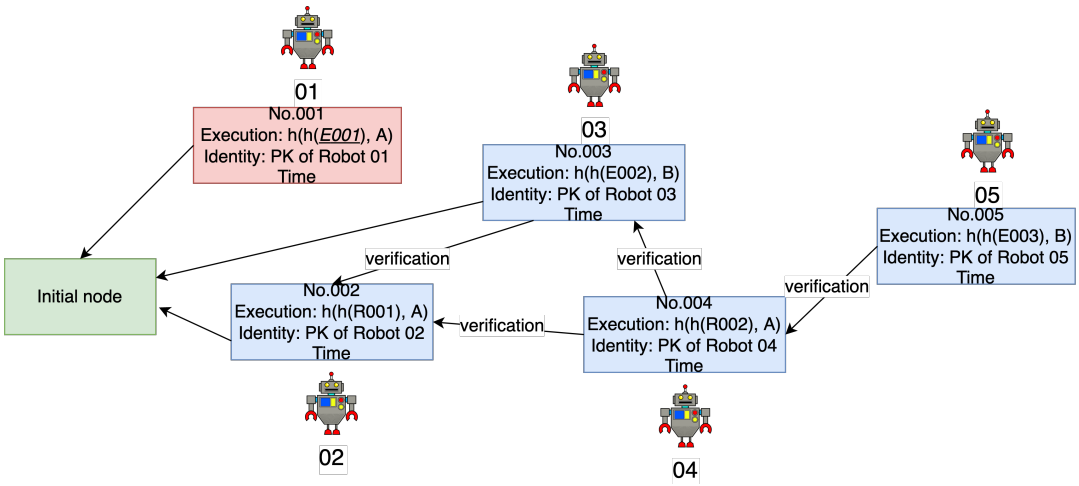


Figure 1.2: Directed Acyclic Graph (DAG) based Distributed Ledger Technology (DLT).

transaction and cryptographically validates the transaction using the public key contained in the transaction.

1.4 Outline

The following parts of this thesis are organized as follows:

Chapter 2 illustrates the research gap based on state-of-the-art literature and related works about security issues in MAS. Additionally, we will analyze the current available DLT technologies and asymmetric cryptography algorithms for IoT devices. Based on the analytic results, a feasible DLT will be selected as the prototype to design a distributed ledger system for SMARS, and an efficient asymmetric cryptography algorithm will be used to develop an identity management system.

Chapter 3 gives a detailed description of four kinds of attacks and how SMARS can prevent them. This is followed by the system's overview and key processes (e.g., intruder detection and verification processes).

Chapter 4 introduces the development tools and the dependencies we used to develop the SMARS simulator.

Chapter 5 conducts three different simulation scenarios (e.g., none attack, malicious intruder, intra-fraud) and collects the simulation results for performance analysis. For each kind of scenario, we draw a conclusion based on the results, which can show the robustness, efficiency, and stability of the SMARS.

Chapter 6 outlines the salient features of the SMARS system and the types of attacks that can be prevented. In addition, we summarise the strengths and weaknesses of the system and how it can be improved based on the contents in Chapter 5. Also, we develop future directions on how the SMARS can be enhanced and lists the techniques that can be used to enhance its

performance and robustness.

Chapter 2

Background and Literature Review

In this chapter, we will first discuss the current state of the art on the security challenges of MAS in Section 2.1. Then two studies are specifically analyzed regarding task allocation and consensus decision making. Their solutions' strengths and weaknesses are discussed in light of their findings. In Section 2.2, we will analyze the underlying security technologies that can be applied in the SMARS system.

2.1 Related Works

There are several published papers summarising the current security challenges facing MAS robots. Studies [5] [16] [17] categorized security challenges(e.g., resource constraints, physical capture, fake identity, and tampering) for robotic systems based on the different use cases(e.g., monitoring, disaster relief, identity, and authentication). Furthermore, they illustrated potential attack models for each security challenge. The study [9] focuses on the logistics industry and refers to several imminent challenges with robotic systems in terms of communication, navigation, and recognition. The security challenges of the MAS robots listed in the above studies demonstrate that the MAS is not yet mature enough to be deployed in various real-world applications. What is known about the challenges facing MAS robots and some of the overlooked security issues needs to be carefully considered to transition from academic to industrial use of this technology. Ferrer(2016) demonstrated a Blockchain-based framework to four emergent issues(e.g., security, decision-making, behavior differentiation) in MAS [18]. He explained how Blockchain technology can provide a reliable peer-to-peer communication channel from cryptography for security issues. The research [2] introduces an approach to encapsulate the mission's high-level objectives in an authenticated data structure known as Merkle Tree. The mission information is in ciphertext with this secure data object during the task executing process. In addition, data verification is separated from the data itself. We will discuss it in detail below in Section 2.1.1. Strobel et al., [3] proposed a solution for managing Byzantine robots (malicious robots) and Sybil attacks[19] via Blockchain technology

in a consensus decision-making scenario. It prevents Sybil attacks via the scarce cryptocurrency that limits the number of transactions a robot can submit. This research will also be discussed below in Section 2.1.2.

2.1.1 Merkle Tree based Task Allocation

In Study [2], Ferrer et al. devise a Merkle Tree (MT) based task allocation scheme through a target discovery as well as a target delivering scenario, so-called foraging. In this scenario, the central operator needs to design and encode the complete sequence of operations included in the MT. This MT will be synchronized with each swarm robot before the mission starts. Under this scheme, any deviation from the original sequence included in the MT will be hard to adapt. After the swarm robots have synchronized the tasks stored in the MT, they need to discover the task first since the robots cannot analyze the specific task content in the obtained MT. When a robot senses a task, it first confirms by hash calculation whether the task is the one currently expected to be executed (as the 'match' operation depicted in Fig. 2.1). If it is, the robot will perform the task and exchange cryptographic proofs with the other robots in the swarm. In this way, all robots can begin to sense the next task. If not, the robot continues with the task discovering process. In addition, the study systematically analyses and compares the resources used for task execution in terms of memory, communication, and computation. The analysis results indicate that the task execution time decreases as the number of swarm robots performing the task increases. However, after a certain amount (e.g., 8 robots) of robots is reached, the task execution time will no longer decrease significantly. Although this approach efficiently protects raw data, if a robot discovers a valid task and generates the corresponding cryptographic proof, there is no guarantee that the robot fulfils the operation practically.

Pros. A subtle virtue of this solution is that the robots do not need to know the objectives of the task in order to cooperate. This approach ensures that the confidentiality of the task is maintained. They complete all the subtasks contained in the MT in exact order, as distributed by the central operator. Such a solution could minimize previously mentioned risks, like physical capture or tampered members, and could be applied to some data-sensitive industries.

Cons. However, as the goal of swarm robot systems is to maximize robot autonomy, the approach demonstrated in this study is only partially applicable to some scenarios with tasks with a strict execution sequence and requires confidentiality to be guaranteed. This scheme means we must consider the approach's scalability, reliability, and flexibility when applying it to industries such as logistics and warehousing. Flexibility requires the system to meet the dynamic variability of tasks, which this method cannot provide. Because the task allocation in this method is that the central operator divides it into subtasks before the task starts and stores them in the MT, the MT is immutable during the task execution. Reliability ensures that

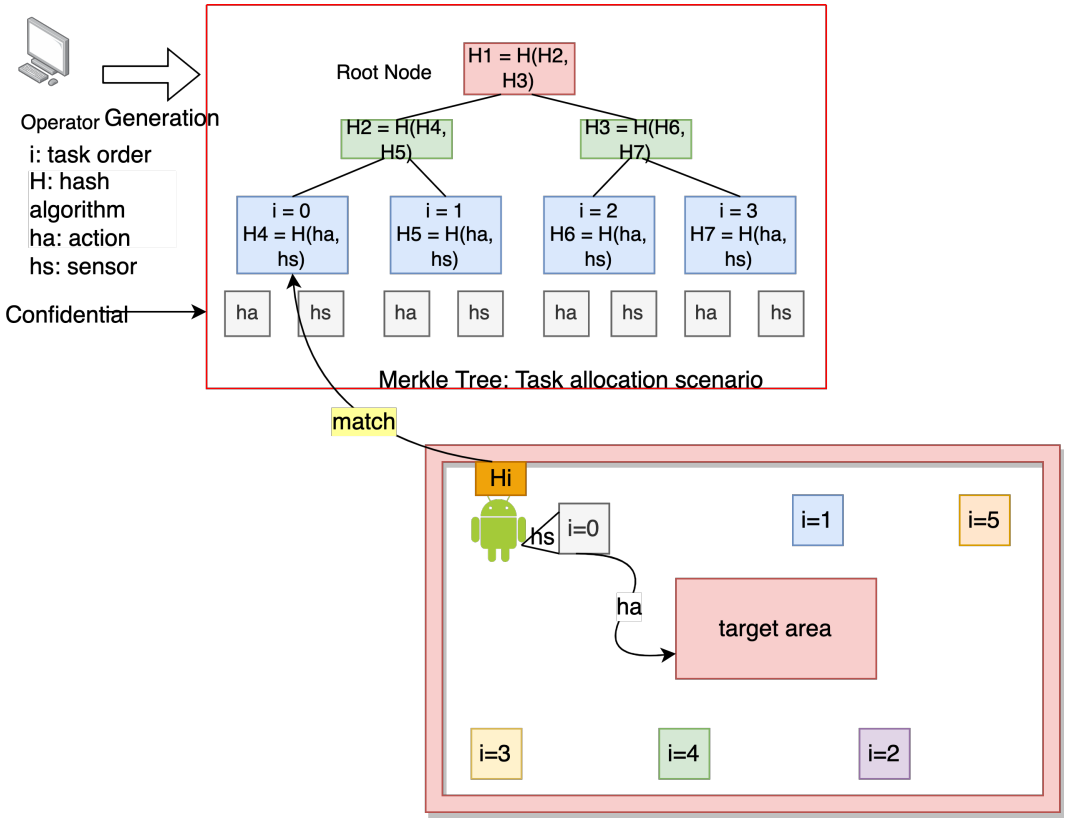


Figure 2.1: Secure and secret cooperation in robot swarms. Inspired by [2].

the robot completes the corresponding task before submitting the task performance credentials. As the method lacks a mechanism to verify the task execution credentials of a swarm of robots, there would be a significant security risk in terms of intra-fraud.

Conclusion. Since swarm robots autonomously perform tasks, their reliability must be guaranteed through effective measures. Our project aims to provide such a validation manner to check the validity of the task execution credentials of the robots in a swarm submitted. Blockchain technology is an optional candidate for providing features including authentication, message integrity, and fraud prevention, all of which are desirable features in current swarm robotic systems[18]. The task execution verification system based on DLT and asymmetric cryptography can verify the reliability of task execution credentials (as transactions in DLT technology) on the chain. The system can ensure that other robots will verify the task credentials submitted by the swarm robots to ensure that the tasks are completed. DLT technology stores task execution credentials and provide an efficient transaction integrity verification strategy. Asymmetric cryptography is used for identity authentication to prevent attacks, including physical hijacking or tampering with members. In addition, it can also ensure that the task execution credentials submitted by swarm robots are traceable.

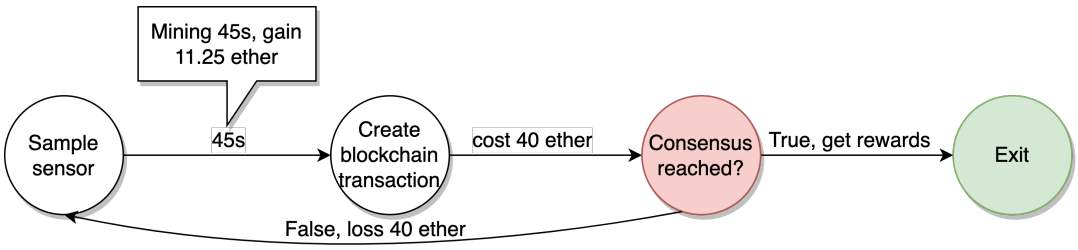


Figure 2.2: Blockchain Technology Secures Robot Swarms. Reproduced from [3]

2.1.2 Blockchain based Consensus Decision Making

Consensus achievement has many applications in swarm robotics, including path selection, pattern formation, and collective perception. However, the final consensus may be disturbed and lose accuracy due to malicious attacks and interference from Byzantine bots. Previous research on consensus achievement (e.g., Linear Consensus Protocol (LCP) [20] and weighted-mean-subsequence-reduced (W-MSR) [21]) mainly assumes that all participants can work as expected. Unfortunately, in the real world, there may be a variety of situations that can lead to consensus failure, including failure of robot perception components or malicious attacks (e.g., Byzantine robots) from outside. Research [3] demonstrates Blockchain technology to defend against Byzantine bots and Sybil attacks in a swarm robotic consensus decision-making scenario. Their proposed approach is based on blockchain technology (e.g., Ethereum) and smart contracts (decentralized protocols executed via blockchain technology) to determine the validity of each robot's decisions through a mining mechanism, as shown in Fig. 2.2. The simulation uses swarm robots to count the frequency of black and white squares in a cell to decide which colour squares are more frequent. Compared to LCP and W-MSR, the blockchain-based consensus achievement method is slightly unstable regarding statistical results in the absence of Byzantine robots. On the other hand, the W-MSR method is resilient when the number of Byzantine robots is low. The blockchain-based approach is also resilient even in the presence of a large number of Byzantine robots. The blockchain-based approach is inherently resistant to Sybil attacks, as swarms of robots need to use crypto tokens when submitting transactions. Such crypto tokens require robots to participate in the task before obtaining them. LCP and W-MSR, on the other hand, do not have this capability.

Pros. This research [3] combines blockchain technology and smart contracts with consensus-reaching methods to defend against Sybil attacks using blockchain technology and Byzantine bots using smart contracts. This method performs very well with or without malicious attacks compared to traditional consensus-reaching protocols. In addition, the authors illustrate in the conclusion section that the method can be used for other applications, including task assignment scenarios, reputation management, collective mapping, and robot-to-robot and robot-to-human economies.

Cons. While blockchain-based consensus-reaching methods can cope with Byzantine bots and Sybil attacks, applying Blockchain technology to swarm robotic systems entails a large resource overhead. Firstly, Blockchain synchronisation between robots in a swarm requires a large amount of data exchange, implying reliance on a stable communication network and a communication delay. Secondly, a Blockchain-based system design requires distributed Blockchain storage, with each bot requiring a copy of the Blockchain to be synchronised locally. That would imply a high reliance on storage capacity. In addition, the more bots involved will generate a large amount of Blockchain data, which also limits the scalability of the swarm robotic system.

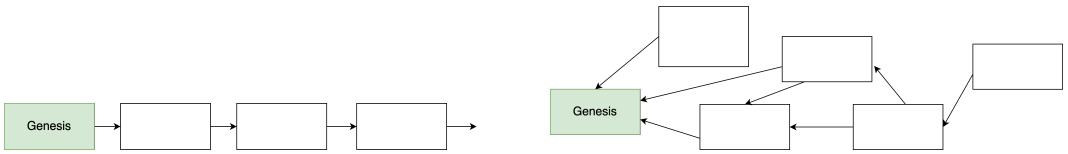
Conclusion. Blockchain technology (e.g., Ethereum) involves many participants conducting transactions and mining, and it is mostly used in some Internet applications such as cryptocurrencies [22]. To deploy this technology in MAS, it is necessary to consider factors such as communication, storage, and energy consumption of individual robots. In Section 2, we explicitly stated that the resources of swarm robots are limited. The resource constraints are an obstacle to applying Blockchain technology to MAS. Fortunately, a DAG-based DLT technology for IoT currently exists, called IOTA [15]. IOTA has adapted to IoT devices, and there is no concept of block and mining. When a user submits a transaction to IOTA's Tangle (as the counterpart of Blockchain in Ethereum), he/she needs to verify the previously submitted transaction and attach his/her transaction to the previous transaction. It is the reason why IOTA has high scalability and low resource consumption in the validation process. Traditional Blockchain technologies (e.g., Ethereum) require a Power of Work (PoW) or Power of Stake (PoS)-based mining process, which is the main procedure that causes resource consumption problems [23]. Considering that there is no research on combining DAG-DLT with MAS robots, this project attempts to combine these two technologies to build a SMARS to test the applicability of DAG-DLT in MAS.

2.2 Dependent Technologies of the SMARS Simulator

This section will compare the underlying dependent technologies deployed in the SMARS. We will analyze the characteristics, applications, pros and cons of Blockchain and DAG-based DLT in the current mainstream DLT technologies. Then we will compare Rivest Shamir Adleman (RSA) and Elliptic Curve Digital Signature Algorithm (ECDSA) in asymmetric cryptography and describe its application in a multi-agent system.

2.2.1 Distributed Ledger Technology

In order to build a DLT-based task execution verification system, we will briefly analyze and compare the existing DLT technologies in this subsection. Distributed ledger technology (DLT) typically refers to a distributed system based on a public or private network for storing,



(a) Blockchain (b) DAG-based DLT

Figure 2.3: Form of different DLT technologies

distributing, and exchanging data between users. In such a system, each participant is referred to as a node. A distributed ledger is generally stored on multiple distributed nodes. Based on the way the technology is implemented, distributed ledger technologies can be divided into three categories: Blockchain (see Fig. 2.3a), Directed Acyclic Graph (DAG) (see Fig. 2.3b), and hybrid DLT [23]. Blockchain has become widely known after the publication of a paper titled *Bitcoin: A Peer to Peer Electronic Cash System* [22]. The technology currently has many applications in healthcare, supply chain management, and the food industry. However, the Blockchain's poor scalability and small transaction throughput are unsuitable for deployment in highly scalable systems like MAS. In addition, Blockchain-based systems usually incur transaction fees and computationally intensive mining processes when transacting, which is another reason Blockchain is unsuitable for robots. Compared with Blockchain technology, DAG-based DLT has strong scalability and supports high transaction throughput, which are exactly the characteristics desired by MAS. The next two paragraphs will briefly introduce Blockchain technology and DAG-based DLT, respectively, and summarize why we chose DAG-based DLT for this project.

Blockchain for IoT This part will analyze potential issues applying Blockchain to MAS for task execution credential verification. When a robot completes a task, it must submit the task execution credentials (the transaction in the Blockchain). Miners will validate transactions submitted within a fixed period. This verification process based on PoW or PoS is called mining. Because the mining process requires many resources, not all robots can be miners. In other words, the verification efficiency of a Blockchain-based SMARS system will be very low. When the number of robots is large enough, there will be many task execution credentials that cannot be verified in time. This problem will limit the scalability of the MAS.

Another potential problem is related to the efficiency of the system. When two miners complete the mining process simultaneously, they generate corresponding blocks and append them to the previous block simultaneously. The Blockchain is a single chain structure that does not allow multiple branches. After a period, the longer branch will become the valid chain, and the shorter branch chain will be deserted [4] (as depicted in Fig. 2.4). This result indicates that even though both miners are legitimate, their work will not be accepted by all. This problem will lead to certain waste of resources. Since the resources of robotic systems are inherently limited, this will exacerbate the problem of resource constraints.

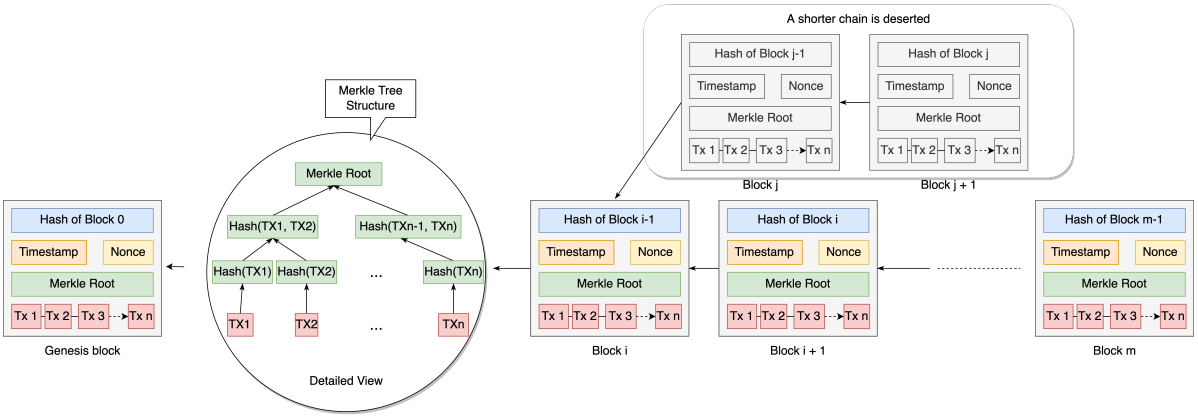


Figure 2.4: Blockchain Structure. Reproduced from [4]

DAG-based DLT The application of DAG-based DLT to SMARS will be analyzed in this paragraph. When a robot completes a task, it must attach the task execution credential (the transaction in DLT) to the previous transaction. An idle robot or a successive robot can verify the transaction. The verification process is different from the mining process in the Blockchain system, so it will not waste many resources. Therefore, every robot in the MAS can act as a validator. More robots can participate in the verification process when the number of robots is large enough. This phenomenon suggests that DAG-based DLT inherently supports the scalability of robots in a MAS.

The verification efficiency of miners in the DAG-based DLT system will also be better than that of the Blockchain-based system. Because DAG-based DLT is based on a directed acyclic graph (see Fig. 1.2), when multiple robots submit task execution transactions simultaneously, they can select different predecessor transactions for verification and attach their transactions to the corresponding transaction.

2.2.2 Asymmetric Cryptography

One of the obstacles to transferring MAS from academia to industry is the issue of authentication of the identity of robots [17]. Given the large number of robots performing tasks in full autonomy requiring extensive data sharing and task collaboration, we need to ensure that each robot in the MAS is reliable and that their behaviour is traceable. It is necessary for an individual robot in a MAS to determine whether the other robots are legitimate entities before interacting with them. Technology such as asymmetric cryptography can provide identity authentication and integrity verification to robots in a MAS.

Asymmetric cryptography (a.k.a. Public-key cryptography) is a cryptosystem that contains a pair of keys: a public key and a private key. The generation of such a key pair is based on cryptography algorithms and mathematical problems termed one-way functions. The main applications of asymmetric cryptography are encryption, digital signatures, and key exchange for

symmetric encryption. In the digital signature scenario, asymmetric cryptography can prevent security problems in terms of authentication, integrity, and non-repudiation. To deploy a digital signature system, the transmitter (e.g., Alice) generates a pair of keys locally and distributes the public key to the receiver (e.g., Bob). When Alice wants to send data to Bob, she first obtains a unique signature through mathematical operations through her private key and the message. Alice then sends the message with the additional signature to Bob. After receiving the message, Bob first extracts the signature and uses Alice's public key to perform corresponding mathematical operations with the signature to verify whether the message comes from Alice. The digital signature can ensure the non-repudiation and traceability of the message.

Asymmetric encryption algorithms such as RSA, which are widely used in computer systems today, can be used to encrypt data and digital signatures. A distinctive feature of traditional asymmetric encryption algorithms is that the longer the key length, the better the security. In the current era of accelerated information technology, computer performance is increasing following Moore's Law, so it has become easier for computers to decipher some encryption algorithms. In order to continue using the previous encryption algorithms, we have to increase the key length to improve data security. This strategy has resulted in some of the current lower-performance edge devices struggling to cope with the corresponding amount of computation for encryption and decryption. When the computational performance (hardware performance) of edge devices is not increasing rapidly, it becomes critical to protect data using more efficient encryption algorithms. Elliptic Curve Cryptography (ECC) is a subtle choice, as the key length of ECC is shorter than traditional asymmetric encryption algorithms while maintaining the same level of data encryption. The study [1] presents a comparison of the key length requirements for RSA and ECC encryption algorithms to achieve the same level of security on embedded systems. The authors concluded that the ECDSA has an overall shorter running time than the RSA in all three phases of digital signature application (e.g., key generation, signature generation, signature verification). Lenstra et al. have demonstrated the key length required for each cryptographic algorithm to achieve the same level of security when applying different cryptographic algorithms [24] (see Tab. 2.1). Saho et al. conclude that the system based on elliptic curves is an efficient alternative compared to RSA-based cryptosystems [1]. Due to embedded devices' limited memory and computing power, it is impossible to perform many RSA-based algorithms calculations. ECC would be a suitable choice for embedded systems today. Among the characteristics of robots in a MAS mentioned in the section 2, MAS robots also suffer from the same resource constraint problem, so ECC is better for identity authentication in MAS.

Identity Authentication and Data Integrity in MAS To introduce the concept of identity in a MAS, we first need to determine how to define the unique identity of the robot. In asymmetric cryptography, the device generates a pair of keys: a public key and a private key. The public key is generally publicly available and poses no impact on system security.

Algorithms Key lengths (bits)					
Key lengths label	1	2	3	4	5
Symmetric	80	112	128	192	256
ECC	163	233	283	409	571
RSA	1024	2240	3072	7680	15360

Table 2.1: The Key Lengths of Different Algorithms. Data in the table from [1].

The private key has to be stored locally and, if compromised, will cause unpredictable security risks. Therefore, using the public key, which is already publicly available, as the robot's identity ensures that the identity is unique within the MAS and allows secure interaction with other robots. As this project aims to build a SMARS based on DLT, all the robots will need to submit task execution records to the distributed ledger consisting of transaction nodes after executing a task. We will use the public key in asymmetric cryptography as the robot's identity to avoid fraud within the MAS. The robot uses the private key to sign the transactions, consisting of the task execution information (as depicted in Fig. 1.2) it wants to submit. In this way, when other robots synchronize the DLT graph of transactions, they can verify the transaction by the signature of the transaction information contained in the transaction node and the robot identity, i.e., the public key.

2.3 Summary

Study [2] encapsulates the robotic collaboration mission to ensure confidentiality of the global task throughout the task execution cycle, which is ideal for scenarios where multiple robots collaborate on large tasks. In contrast, in the smart storage domain, in most cases, the tasks performed by the robots are unrelated to each other, which means that this scenario is unsuitable in the smart warehousing area. Study [25] used a Blockchain-based consensus-reaching mechanism to prevent Byzantine robots, an approach that works well in MAS, but energy consumption and scalability are its fatal flaws. There is still a research gap in ensuring that each robot completes its task as expected.

In this project, we aim to design a Self-supervised Multi-Agent Robotic System (SMARS) using distributed ledger technology (DLT) and asymmetric cryptography to prevent multiple potential security attacks (e.g., identity impersonation, malicious intruder, intra-fraud, Sybil attack) in warehousing and logistics scenarios. The dependent technologies are analyzed above in section 2.2. In our system design, Directed Acyclic Graph-based Distributed Ledger Technology (DAG-DLT) (see Fig. 1.2) is mainly used to record the task execution records (in the form of a transaction) of the robot and provide an effective manner to verify whether the robot has completed the task execution record it submitted. Asymmetric cryptography algorithms [1] are applied in our system design to supply identity authentication and transaction integrity verification desirable in SMARS. Each robot in a SMARS has a unique identity to execute a

task and submit the task execution record (as a transaction) to the distributed ledger. The successive robots that submit transactions or the idle robots need to verify the task execution records recorded in the distributed ledger and ensure the reliability of the ledger under a certain incentive strategy.

Chapter 3

Methodology

This chapter introduces the four potential threats present in heterogeneous multi-robot systems (MRS) and explains how SMARS prevents these attacks by analysing their attack characteristics. This is followed by a description of the entities (e.g., the central node and the robots) included in SMARS, their respective functions, and how they collaborate for secure task execution. Finally, the robot's four key procedures (e.g., intruder detection, transaction verification, dynamic credential threshold calculation, distributed ledger synchronisation) during task execution are detailed.

3.1 Attack Scenarios

Study [5] categorizes five types of attacks (as shown in Fig. 3.1) that can be carried out through fake identities. We have redefined four categories of attacks by analyzing possible scenarios in intelligent warehouse systems, some of which may be a combination of the multiple attacks listed in Fig. 3.1. For each attack scenario, we will analyze the attack processes and present the approach of the SMARS to prevent it.

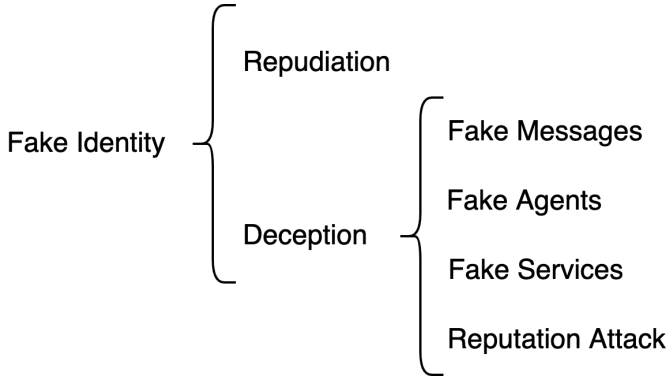


Figure 3.1: Taxonomy of Fake Identity Attacks. Reproduced from[5].

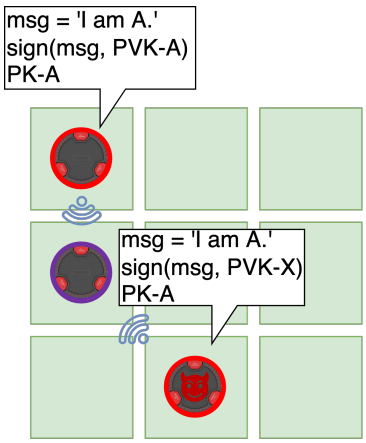


Figure 3.2: Attack Scenario: Impersonation

3.1.1 Identity Impersonation and Repudiation

In SMARS, each robot must use a unique identity to join the system before performing its tasks. Since the identity system is based on asymmetric cryptography algorithms, any robot can access the identity information of other robots and the public key. However, all the transactions in a DLT Ledger are digitally signed, which means that a malicious robot cannot use a legitimate robot's identity information. For example, suppose a malicious robot (fake agent) observes that a legitimate robot has successfully delivered an item to its destination. In that case, it can use the observed information to submit a fake transaction (fake message) to the distributed ledger (as depicted in Fig. 3.2). Since each transaction has a signature generated using the private key, the fake transaction can be easily verified and exposed by other robots through cryptographic methods. Therefore, asymmetric cryptography-based identity systems can inherently prevent identity impersonation attacks. In contrast, if a robot signs a transaction using its private key and denies that it submitted the transaction, we can verify it using its public key due to the non-repudiation nature of digital signatures.

3.1.2 Malicious Intruder

Each robot participating in task executions must have a legitimate identity registered with the central node. Otherwise, it will be considered a malicious intruder (fake agent). In Chapter 1, we introduced that since autonomous MRS requires a large number of robots with different functions to work together, it is essential to manage the identity of each robot. Suppose a robot already in the warehouse does not have a legitimate identity. It can move around the warehouse and deliver items to the wrong storage area (as depicted in Fig. 3.3). In this case, the items to be moved by a legitimate robot may be unloaded to another storage area, causing the robot to fail its task. If a malicious intruder were to carry out a large number of random item movements, this would severely impact the efficiency of the MAS and eventually lead

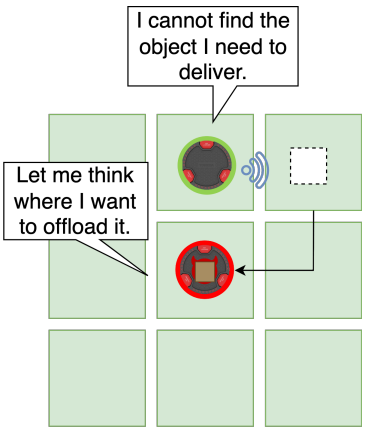


Figure 3.3: Attack Scenario: Malicious Intruder

to complete downtime. We developed an intruder detection function for robots in a MAS to prevent this attack. All the robots in the warehouse without a legitimate identity registered in the central node will be regarded as malicious intruders. If a robot detects a malicious robot, it will immediately report the information (e.g., identity and location of the malicious robot) to the central point. We will describe how a legitimate robot detects malicious robots in Section 3.2.

3.1.3 Intra-fraud

Intra-fraud is another attack that can severely affect the efficiency of the MAS. A legitimate robot could be physically captured, manipulated and re-introduced into the warehouse by an attacker, in which case an intra-fraud attack would occur. As the robot has a legitimate identity registered to the central node, it could obtain tasks from the task list and partially execute them (e.g., randomly unloading objects into a storage area in the middle of a delivery, as depicted in Fig. 3.4). It can then generate a transaction and tamper with the unloading location of the goods to complete the attack on the system. It can then generate a transaction and tamper with the unloading location of the goods to complete the attack on the system. As we develop a physical and cryptographic task execution verification procedure, the successive robots will sniff the actual location of the shipment and compare it with the information in the distributed ledger. If the information does not match, the verifier will report back to the central node an alert with the location of the object and the identity of the transaction submitter. The central node then uses the results of multiple robot verifications to make a judgement and intercept the manipulated robot in time.

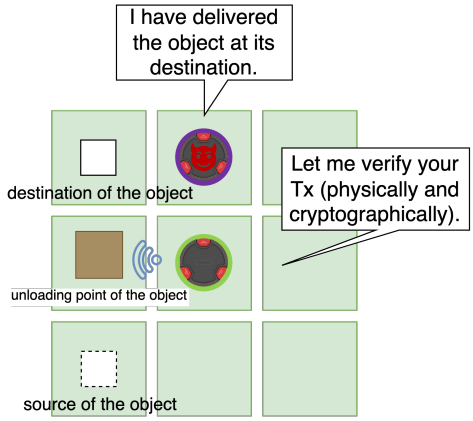


Figure 3.4: Attack Scenario: Intra-fraud

3.1.4 Sybil Attacks

Sybil attack [26] is a kind of attack that exists in DLT, which can severely impact the efficiency of the network. An attacker can subvert the distributed ledger by creating many virtual robot identities (i.e., fake agents) and registering on the central node. Then the attacker can submit lots of fake transactions using the fake identities without performing the tasks (as depicted in Fig. 3.5). Since the legitimate robots in a MAS need to verify all the transactions recorded in the distributed ledger, their efficiency will be affected by the verification processes as lots of the transactions are invalid. To prevent the Sybil attack, we introduced a reputation (e.g., credits) based incentive system to SMARS. All the robots will have a reputation value of 100 credits after joining the SMARS successfully. A robot completing a task must spend a reputation value equal to the task weight to submit the corresponding transaction to the distributed ledger. To ensure the robot itself has enough credit value to submit a transaction, it needs to check if it has enough credit value before executing a task. Because each robot registered to the central node has an initial reputation value of 100, a robot cannot submit an unlimited number of invalid transactions to the distributed ledger. In addition, if a robot submits a transaction verified as valid by 4 other legitimate bots, it receives 4 times the reputation value of the task weight as a bonus. Conversely, if a transaction is deemed invalid by 4 other legitimate robots, the transaction submitter will lose the reputation value it spent to submit the transaction as a penalty.

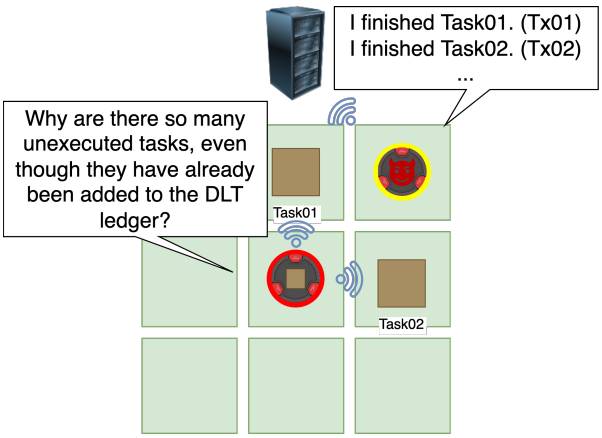


Figure 3.5: Attack Scenario: Sybil Attack

3.2 System Design

In order to verify the feasibility of our proposed approach, we integrated the DLT and cryptographic-based task execution verification system into an autonomous MAS in a smart warehouse scenario (as shown in Fig.3.6) and developed a simulation platform, SMARS. This section primarily provides an overview of the SMARS system and detailed descriptions of the key components (e.g., warehouse, task management system, identity system, distributed ledger system) separately.

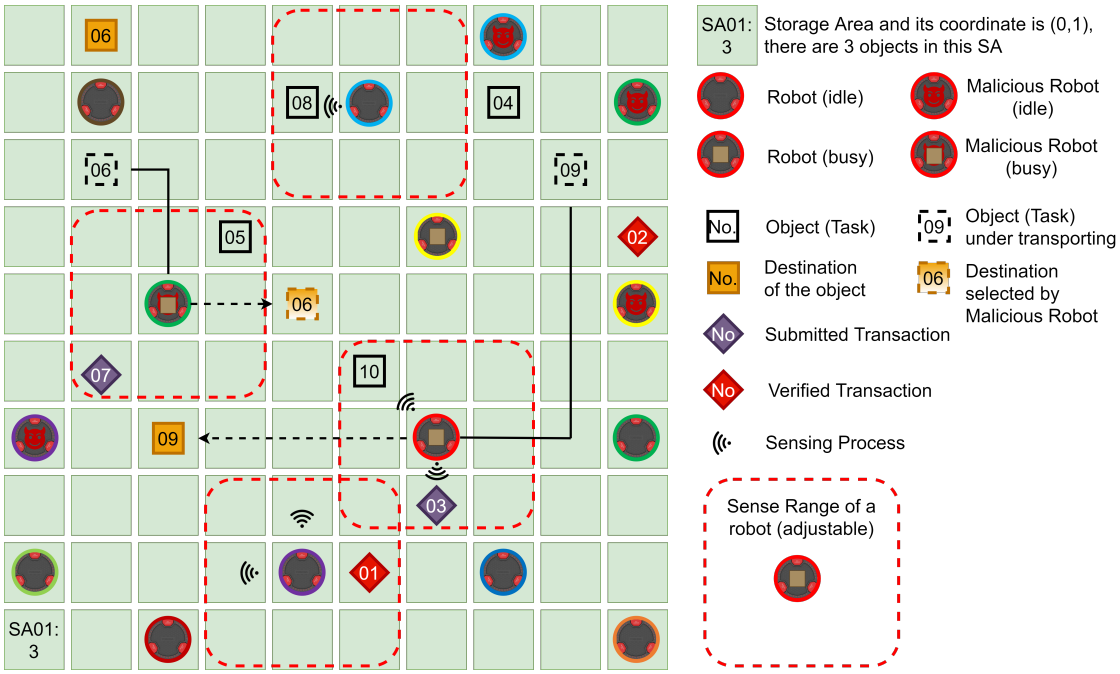


Figure 3.6: Demo of Self-supervised Multi-agent System

Properties of a transaction	**Value**
code (reference number)	Tx+count
timestamp	time the transaction generated
submitting robot	identity of the transaction submitting robot
object	code of the object (reference number)
task	code of the task (reference number)
unloading point	location where the object being unloaded
credentials	predecessors of this transaction node in the DLT graph
signature	calculated with the contents above
verifiers	successors of this transaction node in the DLT graph

Table 3.1: Contents of a transaction

The warehouse is a two-dimensional plane consisting of several individual storage areas, each with a unique code (e.g., SA01 means its coordinate is (0,1)) used to refer to its location. Each storage area contains an objects management system to record objects' inbound and outbound records and the corresponding carriers. There are two lists (e.g., objects list (in real-time), objects accessing records list) maintained by each storage area to assist the SMARS in task allocation and creation. The number of objects in a storage area will be shown in the warehouse map during the simulation (as shown in Fig. 5.1).

The task management system initializes a fixed number of tasks at the beginning of the simulation. A task contains information like the source and destination of the object, weight and volume, and weight of the task (calculated from the properties of the objects and shipment distance). The task management system records all the inbound and outbound objects with the corresponding carriers.

The identity management system is developed based on asymmetric cryptography (e.g., ECDSA) as analysed in Section 2.2. The robots have to generate a pair of keys (e.g., $public_key$ and $private_key$) during the initialization phase. The identity of a robot is obtained according to a public key based on the $SHA256$ hashing algorithm, which ensures its uniqueness. For a robot to participate in task execution, it must first request to join the SMARS system. After getting permission, the robot can access the task management system and synchronize the distributed ledger with the central node.

The transactions are the nodes in a DAG-DLT graph (as shown in Fig. 3.7). Each node represents a transaction. Specific information about each task execution is stored within a transaction (as listed in Tab. 3.1), and non-transaction submitters of a specific transaction can validate it to obtain permission to attach their transactions to it. A single transaction can only be validated once by the same robot.

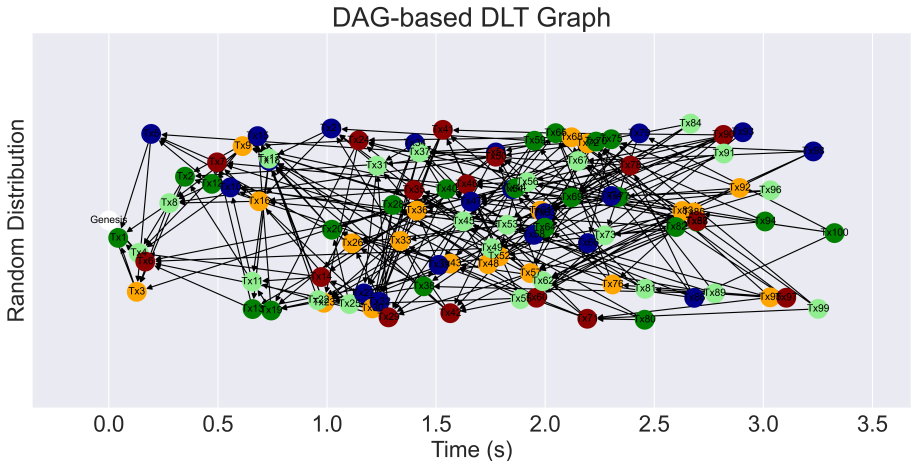


Figure 3.7: A distributed ledger generated with 5 robots and 100 tasks in a 6*6 warehouse

The credentials are needed to submit a transaction since a robot needs to attach its transaction to the transactions in the DAG-DLT ledger. After a robot has successfully validated a transaction, it will get the credential that permits it to attach its transaction to that transaction. The verifier's identity information is also added to the list of verifiers within the transaction.

The Tips are the transaction nodes in the DAG-DLT ledger that are verified less than four times. The number of Tips in the DAG-DLT reveals that the timeliness of a transaction is verified and the system's efficiency.

The distributed ledger system is used to record information about the task execution records submitted by robots in the form of transactions. During the initialization phase of the system, a Genesis node (e.g., the white node in Fig.3.7) will be initialized in the distributed ledger. After successfully executing a task, the robot needs to record the task execution information in a transaction and submit it to the distributed ledger. Before committing, each robot needs to verify the transactions submitted by other robots to obtain sufficient credentials. Since there are no verifiable transactions in the distributed ledger at the beginning, the robots can attach their transactions directly to the Genesis node. As we mentioned before, to prevent Sybil attacks, robots must spend the same amount of reputation value as the task weight when submitting a transaction. Fig.3.7 is a DLT ledger generated with the configuration of a 6*6 warehouse, 5 robots with a sensing and communication radius of 1, and 100 tasks. Transaction nodes in the DAG-DLT graph are distributed in chronological order of submission to the ledger on the x-axis and random values between -1 and 1 on the y-axis.

The robots are involved in task execution (e.g., item dispatching). The number of robots in a simulation and their sensing and communication range is individually adjustable (as shown in Fig.3.6), which facilitates comparative experiments. A single robot's related properties and functions can be found in Appendix A.2. Except for the $Transaction$ and $Task$ parameters, the assignment of the other parameters will be completed during the system initialisation phase. The colour of a robot in simulation visualisation and its corresponding transaction nodes in the DLT ledger are the same, which enables us to distinguish each entity in the system and the transactions it submitted. The trajectories of the robots will be displayed on the warehouse map, as in Fig.5.1. All the robots in the system are independent in terms of task acquisition, task execution and transaction submission. Section 3.8 illustrates the detailed work processes of a robot.

3.2.1 System Structure

For a simulation without attack scenarios, we need to input 4 parameters (e.g., warehouse size, number of robots, sensing and communication radius, and number of tasks) to start the system. The system will first initialize the warehouse with a customized warehouse size. The warehouse size is defined by two values (e.g., x and y) to denote the number of storage areas in each dimension. Then an amount of $x * y$ storage areas will be initialized. The identity management system initialization process follows this. There are two lists maintained by the identity system to store the legitimate robots (in the robot list) and illegitimate robots (on the blacklist) in the warehouse. Robots will generate a pair of keys (e.g., the public and private keys) first and use the public key and identity register to the identity system. After successful registration, its identity will be added to the legitimate robots list, so it can access the task list and synchronize the distributed ledger with the central node later. The task management system is another key component to support the SMARS since all the robots need to be organized to deliver objects to their corresponding destinations. The task management system records the identity of executors and timestamps when a task has been allocated or finished to make the task executions of robots traceable. The distributed ledger and corresponding DLT graph are also initialized at the beginning of a simulation.

Once all the management systems have been initialised, the robot starts the initialisation process. In the simulation environment, we use multi-threading programming to simulate the task execution for multiple independent robots in a warehouse. For a single robot, the initialisation process is carried out in the main process, and the subsequent task execution process is carried out in a separate thread (forked from the main process). The initialisation of the robot contains all parameters in the green box connected to the robot initialisation box in Fig. 3.8. After robots have completed the initialisation process, the main process forks a thread for each robot to start the task execution. A robot's task execution cycle is divided into seven main phases, the details of each of which are listed below:

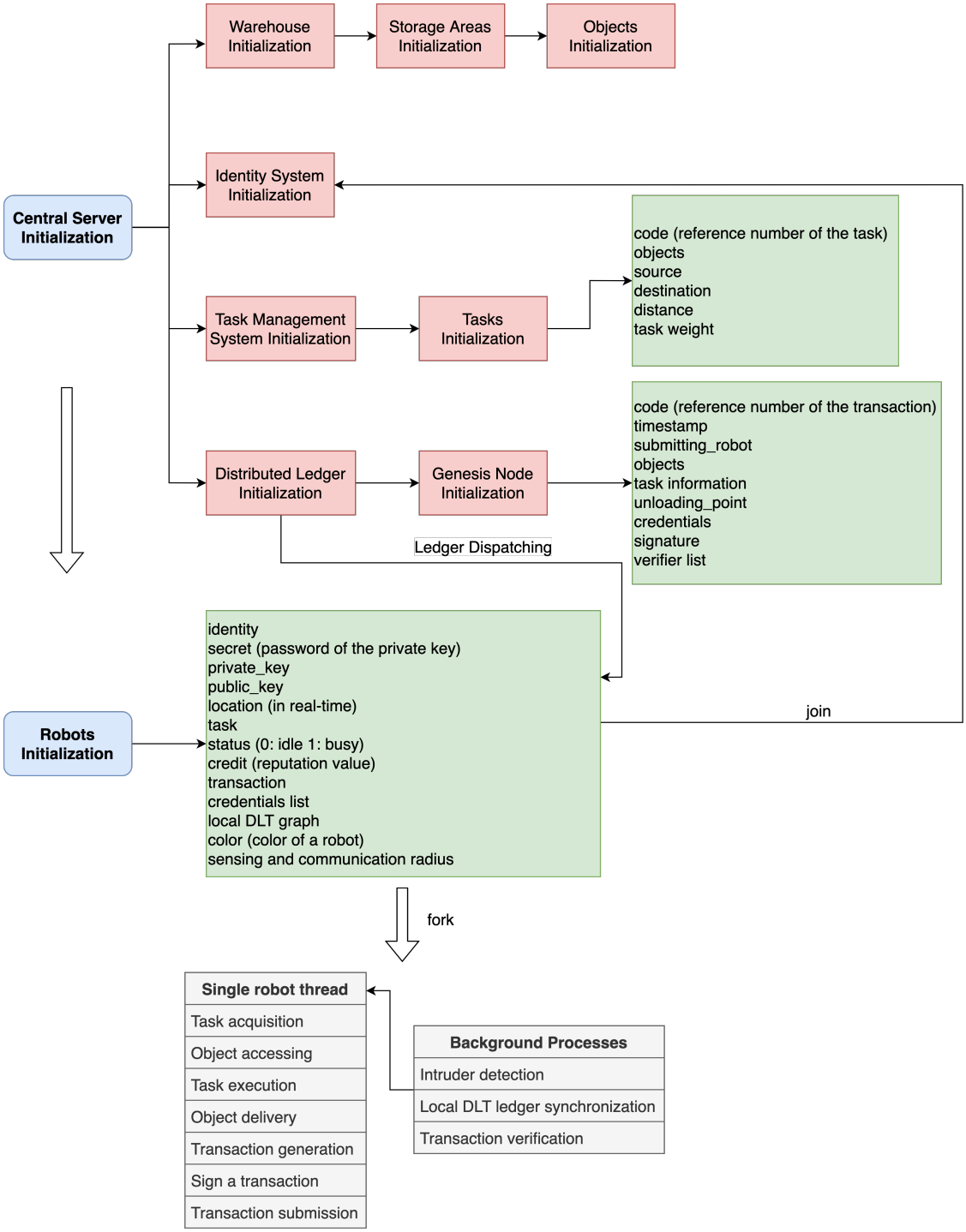


Figure 3.8: System Structure of Self-supervised Multi-agent Robotic System (SMARS)

- **Task acquisition** The robot first calculates the nearest item based on its real-time location and the source address of the item to be transported in the task list and requests the corresponding task from the central node.
- **Object accessing** The robot then moves to the source address of the item specified in the task to obtain it, while its identity is recorded in the inventory management system of the storage area.
- **Task execution** The robot moves to the destination of the object specified in the task while constantly sensing other robots around it to make sure there are no illegitimate robots in its vicinity.
- **Object delivery** The robot moves towards the destination of the item specified in the task while constantly sensing other robots around it to ensure that no illegitimate robots are in its vicinity. As it moves, the robot also senses items in the surrounding storage areas and matches them to items that have already been delivered to their destination by querying the transactions in the locally stored distributed ledger. If there is a corresponding transaction in the DLT ledger, the robot will first determine if the transaction has reached the upper limit of the number of times it has been verified, i.e. four. If the limit has not been reached, the robot first matches the item's physical location with the unloading location recorded in the transaction and then verifies the transaction's signature using the public key in the transaction. If all information matches, the verification is complete. The authenticator's identity is recorded in the transaction's verifier list, and he or she is given credentials to attach his or her transaction to the transaction. For each successful transaction validation, the transaction submitter will receive a reputation value equal to the task's weight, up to four validations for a single transaction.
- **Transaction generation** Immediately after a task has been successfully executed. The executor will generate a transaction (contents are listed in Tab. 3.1) to record the task execution information and store it in the cache.
- **Signature generation** Then, based on its private key and the first seven lines of contents in the transaction in Tab. 3.1, the robot will generate the corresponding signature with the ECDSA algorithm as analysed in section 2.2.
- **Transaction submission** A robot needs to ensure that its reputation value is greater than the weight of the task it is performing before it submits a transaction, which is why a robot needs to ensure that its reputation value is high enough before it can acquire a task. On the other hand, a robot needs to calculate the number of credentials it needs to submit the current transaction (obtained by completing the authentication of transactions submitted to the distributed ledger by other robots) using a dynamic threshold formula 3.1.

3.2.2 Key Processes

The outstanding feature of the SMARS system is its ability to detect intruders and some manipulated legitimate bots in time. For intruders, each robot in the MAS system is equipped with a sensor (e.g., a camera) to detect robots within a certain range (defined by the user) that do not have a legitimate identity and report back to the central node in real-time. However, a legitimate robot manipulated by an attacker may not be detected in time by physical methods. Nevertheless, suppose these robots try to disrupt the normal operation of the SMARS system (e.g. by dispatching items to the wrong destination but indicating in a transaction submitted to the distributed ledger that the items were dispatched to the correct destination.). In that case, these anomalies can be quickly detected through the transaction verification process.

We will then detail how legitimate robots detect malicious intruders and how the SAMRS system's transaction verification process works. Also, since the SMARS system is based on DAG-DLT, it is important to control a stable range of transactions that must be verified in the distributed ledger. We propose a dynamic threshold function to specify how many credentials are needed when a robot submits a transaction (i.e., how many predecessor nodes it needs to attach its transactions) so that the number of Tips can be kept within a small range. In addition, since the ledger is distributed, the robots need to synchronise the ledger with other robots in communication range in real-time while they are moving in the warehouse. This part also shows how ledger synchronisation is performed between robots and identifies the problems that may arise from large latency in synchronisation.

3.2.2.1 Intruder Detection

Each legitimate robot will be equipped with multiple sensors (e.g. cameras) to detect intruders in their sensory range. Malicious robot detection will always run in the background after the robot is started and constantly scan the environment as the robot moves. The robot will report the location of the intruder and the time of detection to the central node as soon as it detects the malicious intruder, as well as confirm whether it is under load and record information about the object. Since intruders can randomly place items in the wrong storage area, this can disturb the legitimate robot from finding an item at its source address. If this happens, the legitimate robot reports to the central node that it cannot locate the item that needs to be dispatched. The central node then disseminates the item's information to all robots in the system and asks them to assist in locating the item during the task execution. Once the item is found, the central node re-creates a delivery task for the item in the task management system to allow other robots to deliver it to its destination.

3.2.2.2 Physical and Cryptographical Verifications

To ensure that each robot in the warehouse has completed its respective task honestly, we develop a task execution verification process to allow the robots in the warehouse to verify their task completion (e.g., transactions in the distributed ledger) with each other during the task execution period. Up to four legitimate robots can verify a single transaction in the DLT graph. The verification process consists of two parts: physical verification and cryptographic verification.

Physical verification Each robot will be equipped with several sensors (e.g. cameras) to detect items in the surrounding storage areas dispatched to their destination. When an item is detected as being at its destination, the robot will check the transaction information associated with the item in the distributed ledger. The location of the item detected by the sensor is compared with the unloading point recorded in the transaction to confirm that the corresponding task has been successfully performed. If the verification result matches, the physical verification is complete.

Cryptographical verification Each robot generates a pair of keys locally before joining the system and registers its identity using the public key. Before a bot submits a transaction (e.g., a task execution record), it signs the transaction message and adds the signature and public key to the transaction for other robots to verify the integrity of the transaction. The other robots cryptographically verify the transaction by first accessing the contents of the public key and the signature and then using the signature verification algorithm to complete the integrity verification.

3.2.2.3 Dynamic Threshold Calculation

The DLT-based verification system needs to record the transaction verification relationships between robots, and a certain number of transactions need to be verified before a robot can submit a transaction. Each successfully validated transaction gives the robot the credential to attach to that transaction, and only after it has a sufficient number of credentials can its transaction be successfully submitted to the DLT graph. We propose a dynamic threshold formula (Formula 3.1) to specify the number of credentials required by the robot to submit a transaction. The threshold is calculated with the number of Tips in the DAG-DLT ledger and the number of local credentials. The Divisor and Base (e.g., default Divisor is 10, default Base is 2) can be customized to find the balance point for each simulation configuration. Using this dynamic threshold credential formula, the number of Tips in the DLT graph can be kept within a small range (e.g., less than 20).

$$Threshold = [(Tips + Credentials)//Divisor] + Base \tag{3.1}$$

3.2.2.4 Distributed Ledger Synchronization

Each robot that successfully registers with the system will first synchronise its ledger with the central node. During subsequent task execution, each robot maintains its ledger locally and synchronises the distributed ledger in real-time with other robots in its communication range. As the robots' communication range is limited, different robots may store different versions of the distributed ledger. We previously mentioned that a transaction could be verified up to four times by different robots, but due to the ledger synchronisation latency problem, a transaction may be verified more than four times, which will lead to a reduction in the efficiency of the system and we define this problem as an over-attachment problem. Because controlling that a transaction can be verified up to four times ensures that the number of transactions that need to be verified (aka Tips) in the system is less than a certain value.

Chapter 4

Experimental Setup

In this chapter, we describe the key components of the SMARS simulation platform, as well as the dependencies and key functions used during development. The hardware and software environments for the development of the SMARS system are summarised in Tab. 4.1. Please refer to Appendix A.1 for all the classes defined in the simulation program.

4.1 DAG-DLT Ledger and Visualization

As we analysed in section 2.2, we developed a DAG-based DLT ledger system to store all the transactions submitted by robots. The inspiration for this DAG-DLT ledger system is taken from IOTA tangle [15], which demonstrates three desired characteristics for MASs: scalability, low resource requirements, and distribution.

The development of the DAG-DLT ledger system is mainly based on the *networkx* library in *Python*. Since the ledger is a directed acyclic graph, we developed the ledger to succeed to *nx.DiGraph*. A Genesis node will be initialised into the ledger when the system starts. The transaction submitted to the ledger is added via *add_node*() function. The successive transaction nodes will be attached to the predecessor nodes in the ledger via the *add_edge*() function. To synchronise the DAG-DLT ledger on each robot, we defined *local_graph_synchronization*(). DAG-DLT ledger plots are first drawn using *nx.draw_networkx*() function in *networkx* and then displayed using *plot* function in *matplotlib* library.

4.2 Identity Management System

According to the comparison results of the asymmetric cryptography algorithms in section 2.2, we developed the identity management system based on the *cryptography* library in *Python*. If a robot want to participate into task execution, it needs firstly generate a pair of keys (e.g., public key and private key) and identity using functions: *private_key_generation*(*Secret*), *public_key_generation*(*Private_key*), *identity_generation*(*Public_key*). During the task execu-

tion, a robot can use function $private\_key\_loading().sign(Transaction)$ to generate a signature for a transaction. To verify the other robots' transactions, a robot can using function $public\_key\_loading().verify(Signature, Transaction)$.

4.3 Visualization of the SMARS Simulation

The simulation for the SMARS system focuses on the smart warehousing scenario, which is demonstrated by moving the robots in a two-dimensional plane (as shown in Fig. 5.1). The visualisation development is based on the *Turtle* library in *Python*, and the trajectory of the robots will follow the movements recorded during the simulation. Each robot will have an independent *Turtle* object to show its movements. The track of a robot will be displayed on the warehouse map.

The simulation configuration and results will be delivered to a webpage using the *pywebio* library. We can check the object distribution and transaction contents (see Appendix B) after a simulation is finished.

4.4 Development Environment

The development environment of the SMARS system is summarised in Tab. 4.1

Operating System	Ubuntu 20.04
Language	Python 3.9.5
Libraries	Cryptography, Networkx, Turtle, Time, Pywebio, Pandas, Random, Pyecharts, Pandasgui
IDE	PyCharm 2021.3.1 (Professional Edition)
Memory	16 GB 2133 MHz LPDDR3
CPU	2.9 GHz Dual-Core Intel Core i5

Table 4.1: Development environment of the SMARS simulation platform

Chapter 5

Simulation Results and Performance Analysis

In this chapter, we first introduce five metrics for evaluating system performance. This is followed by an analysis of how the system performs in different scenarios by demonstrating the results of system simulations in three scenarios (e.g., without attacks, with the malicious intruder, with intra-fraud). Because the identity management system was developed based on asymmetric cryptography (e.g., ECDSA), it can inherently prevent identity impersonation problems, and therefore we will not simulate this scenario. Furthermore, the Sybil attack, which essentially submits many fake transactions to the distributed ledger, is similar in principle to the intra-fraud scenario, and we will omit the simulation of this attack scenario here. The reliability and scalability of the system are then verified through batch testing. Finally, we will discuss the limitations of the SMARS system.

5.1 Metrics

The following are five metrics that can be used to measure the efficiency of a system.

- **Communciation Amount** In Chapter 3, we mentioned the need for real-time synchronisation of the distributed ledger between robots to keep the ledger updated. The ledger synchronisation process is conducted by data transmitting, and the amount of transmission will significantly impact the work time of a robot. As the size of the ledger constantly changes during the system's operation, we will use the final ledger size to calculate the maximum bandwidth required for inter-robot communication. The maximum communication bandwidth is calculated with the average communication amount (CA) per robot per second multiply data size.
- **Time elapsed to finish all the tasks** Another parameter that highlights the system's performance is the time taken for all tasks to be completed. To evaluate the system's

performance in terms of time, we will use the time it takes to complete all tasks in the absence of an attack as a benchmark and test how long the system takes to complete all tasks in the presence of malicious intruders and illegitimate robots, respectively.

- **Time to detect all the malicious robots** Our system's primary goal is to detect illegitimate robots in the system promptly. We will record the time it takes for the system to locate all the malicious robots by batch testing in two attack scenarios (e.g., malicious intruder, intra-fraud).

- **Number of overattachment nodes in DAG-DLT** We specify that a transaction in the DAG-DLT system is validated at most four times by different robots, but a transaction may be validated more than four times due to ledger synchronisation delays. A transaction node validated more than four times is named an over-attachment node. Since too many over-attachment nodes will result in newly submitted transactions not being validated promptly, we will analyse the severity of this problem through batch testing. We also discuss solving this problem through a consensus mechanism or a central coordinator in the section 6.2.

- **Number of Tips in DAG-DLT** The Tips is already defined in section 3.2. The total number of Tips in the ledger must be kept in a small range to ensure that a transaction can be verified in time. We will record how the number of Tips changes throughout the task cycle for different attack scenarios.

5.2 Simulation Results

We conducted simulations in three different scenarios (e.g., without attacks, with the malicious intruder, with intra-fraud) with the same configuration settings to analyse the reliability of the SMARS system.

5.2.1 Multi-agent System without Attacks

The SMARS system operates strictly following the steps described in chapter 3.2 without an attack. The simulation results in this scenario will be used as a reference criterion to evaluate the system's performance in other scenarios.

Simulation Configuration Tab. 5.1 shows the configuration of the simulation.

Simulation Configuration	**None**
Size_x of Warehouse	8
Size_y of Warehouse	8
Number of Robots in the Swarm	6
Sensing and Communication Radius of robots	2
Number of Tasks	200

Table 5.1: Simulation Configuration of No-attack scenario

Simulation Visualization Fig. 5.1 is captured during the system running. We set different colours for different robots in the warehouse. The DAG-DLT ledger (as shown in Fig. 5.4) is generated with the coloured transaction nodes, and the colour of the nodes in the ledger matches the colour of the robots. Each node in the ledger represents a transaction, and the edges represent the attachment relationships between nodes.

Simulation Results Tab. 5.2 is the simulation results of the no-attack scenario. According to the data below, we can calculate that the maximum communication bandwidth for robots is 907.26 Bytes per second (Approximately 8 Mbps). The study [27] reviewed several communication protocols for IoT devices, and there is no one can reach the data rate of 8 Mbps. Later, we will analyze how to compact the ledger in terms of codec, compression or update frequency.

Metric	**Value**
Time eclapsed	10.74
Communication Amount (48 Bytes)	203.0
Number of Tips	10.88
Number of credentials needed in average	4.0
Over attachment nodes	9

Table 5.2: Simulation results without an attack

Trends of Credentials Required Fig. 5.2 depicts the dynamic credentials needed for each transaction submitted to the ledger. The blue line represents the changing process of the credentials needed to submit a single transaction. The red line is the simple moving average with a window size of 30. The red line indicates that the number of credentials required increases at the beginning of the system and then stabilizes. At the beginning of the system, there are not enough Tips in the ledger for the robots to validate, but with the number of tasks are performed increases and the amount of transactions in the ledger increases, the number of credentials that each robot needs to submit a transaction increases. However, as the system reaches mid-run and the number of Tips within the system stabilizes, the number of credentials required to submit a transaction will also become stable.

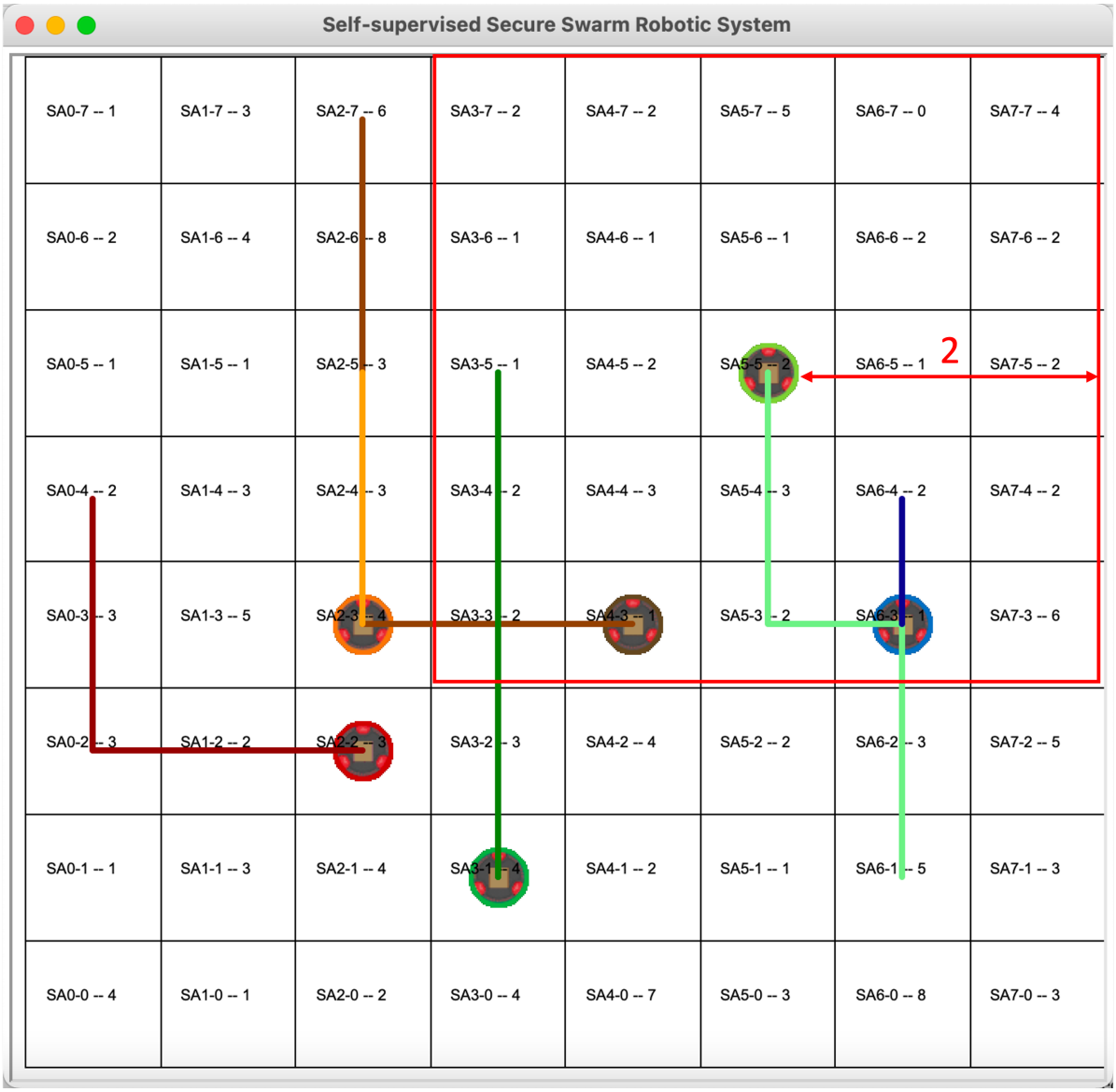


Figure 5.1: Simulation of SMARS warehouse (8*8) scenario with 6 robots

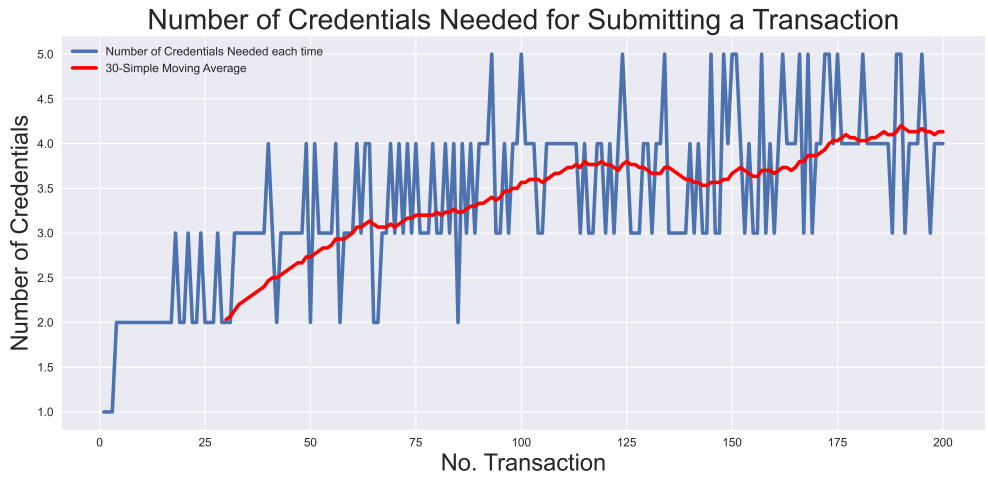


Figure 5.2: Number of credentials needed for each transaction submission

Trend of Tips in DAG-DLT The trend for Tips (as depicted in Fig. 5.3) is similar to credentials, increasing initially but becoming stable as the system runs into the middle stages. However, the number of tips decreases at the end stage of the simulation since the number of tasks decreases, but the number of credentials required to submit a transaction does not change.

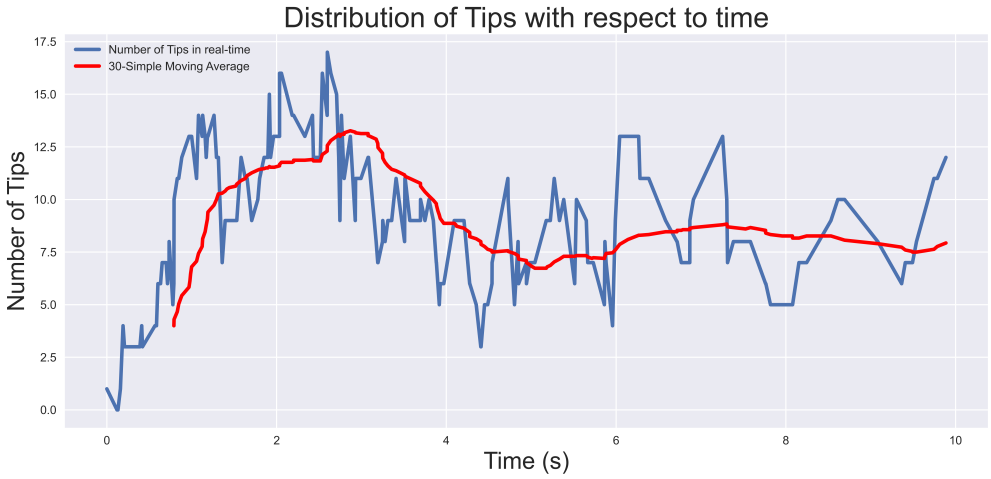


Figure 5.3: Number of Tips in real-time

DAG-DLT Ledger Visualization The DAG-DLT ledger generated in the simulation is shown in Fig. 5.4.

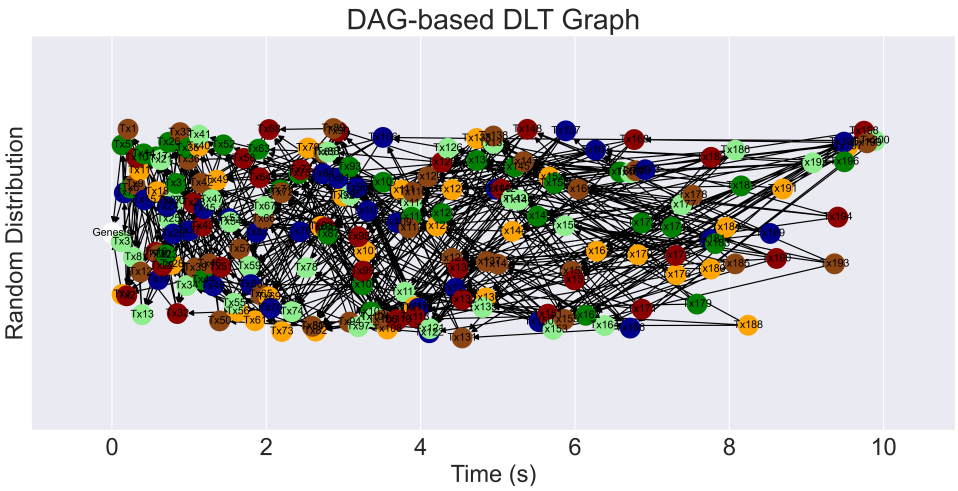


Figure 5.4: Distribution of transaction nodes in DLT graph

Credits of each Robot According to the results in Tab. 5.3, it can be noticed that the reputation values between robots vary considerably at the end of the simulation. The task weight, the timeliness of authentication, and the dynamic credential threshold all affect the reputation value of the robots. Our original intention of deploying the incentive system was to prevent Sybil attacks, but the current one did not perform as we expected. A robot has an initial reputation value of 100 and needs to spend a reputation value equal to the task weight value to submit the corresponding transaction after the task has been executed. If a robot submits several transactions and none of them is validated by other robots, this will result in the robot remaining idle and thus reducing the overall efficiency of the system. It will be a future research direction to improve the incentive system's reward and punishment mechanism to improve the efficiency of the system.

Reference	Identity	Credits
Robot0	097ce1ddac81b9119544e6baba0...87956b36a58361916ab1968256	391.5
Robot1	bf5830698d536ebd0b05c17807a...4ce46dc65dc22f34b3f20a5d14b	129.5
Robot2	2d1cddb88eefecfb5e99905dbead...e54663b1217b0615dcd38b803f	367.0
Robot3	f9fa0216b82d2ea63ccfbbdba8d8...0365a370e859fd09c4e3f06712f	154.0
Robot4	7bb19fde9f10c9044d3fdb34d5f5...91f8cef4f561b8f569aa42e016b7	383.0
Robot5	cda7459a363d0c69098a3bf65bd...2fc9a58f47be210dd15ddea62e91	251.0

Table 5.3: Reputation Value of each robot after tasks finished

5.2.2 Multi-agent System with Malicious Intruders

The SMARS simulation with malicious robots is conducted in this section.

Simulation Configuration Tab. 5.4 shows the configuration of the simulation. Compared to the no-attack scenario, we added five malicious intruders to the system to analyse the robustness and reliability of the SMARS system.

Simulation settings	**None**
Size_x of Warehouse	8
Size_y of Warehouse	8
Number of Robots in the Swarm	6
Sensing Radius of robots	2
Number of Malicious robots	5
Number of Tasks	200

Table 5.4: Simulation Configuration of Malicious intruders scenario

Simulation Results According to the simulation results in Tab. 5.2, it is clear that almost all the results differ very little from the no-attack scenario. The largest deviation of the simulation results from the no-attack scenario is the over-attachment points, approximately 30%. In terms of task completion time, the deviation was approximately 4%. The results suggest that even in the case of the malicious intruder scenario, robots equipped with intruder detection sensors can quickly detect all intruders and report them to the central node to prevent them from affecting the system's normal operation.

Metric	**Value**
Time eclapsed	10.27
Communication Amount	200.33
Number of Tips	13.1
Number of credentials needed in average	3.9
Over attachment nodes	13

Table 5.5: Simulation results with malicious intruders

Malicious Intruders, Credentials, Tips, DAG-DLT Graph Since all of the malicious robots were detected less than a second after the system started (as shown in Tab. 5.6), their impact on the system's normal operation was almost non-existent. Figure 5.5, 5.6, 5.4 present the change in credentials required to submit a transaction, the change in Tips in the ledger, and the DAG-DLT graph generated after all tasks are completed, respectively.

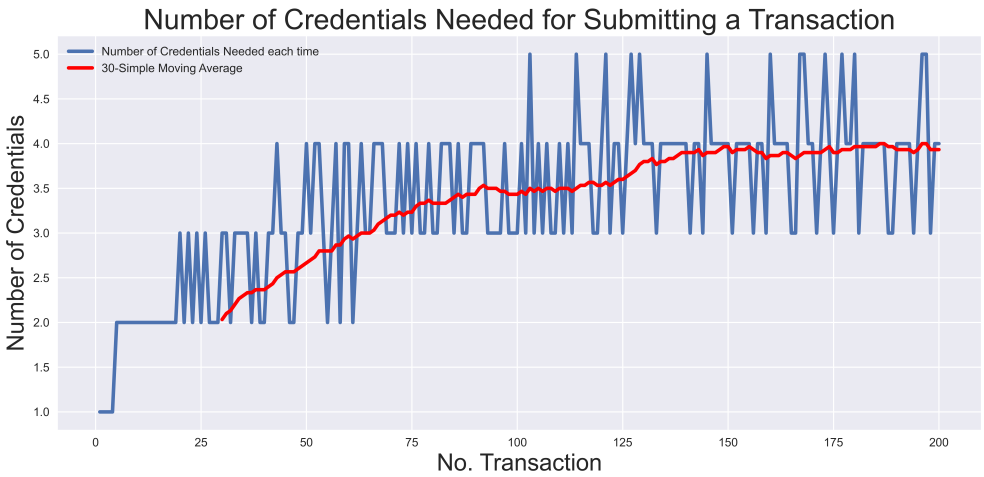


Figure 5.5: Intruder - Number of credentials needed for each transaction submission

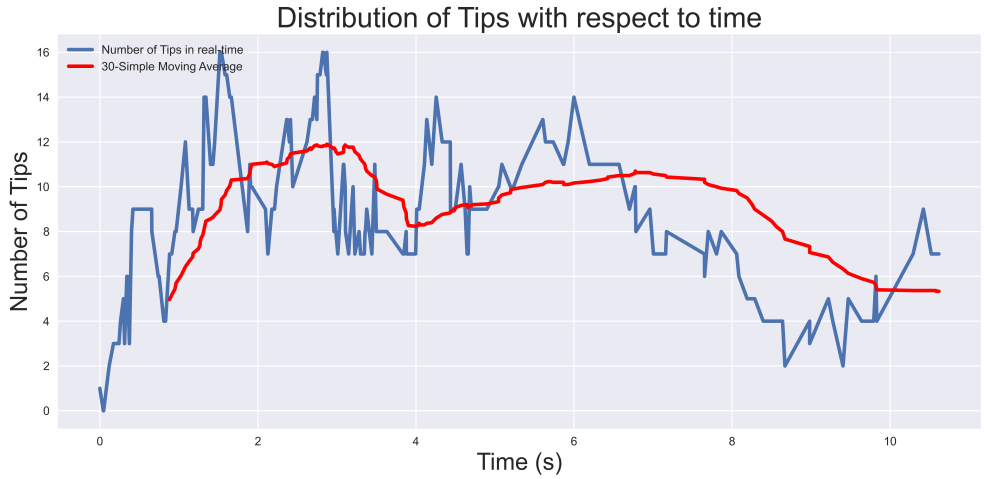


Figure 5.6: Intruder - Number of Tips in real-time

Reference	**Identity**	**Location**	**Time**
MRobot0	75b3d5d5c4460707c4820...4878173a413cbe826ddfde66	(7, 2)	0.01
MRobot1	f1913548bfd6589cedc784...521d8e8775e0da395d14af6	(0, 5)	0.01
MRobot2	483598f0daedda3bcfc4bb..4f1b265f38799b0dd2f3c640	(1, 2)	0.03
MRobot3	287429e470c6d2483094c...b8955e37d325e7948830d89	(7, 2)	0.03
MRobot4	a09d8bc24747854a35063...b74b3b3846e8143b53f73f4b	(1, 2)	0.03

Table 5.6: Detected malicious intruders with their corresponding location and detected time

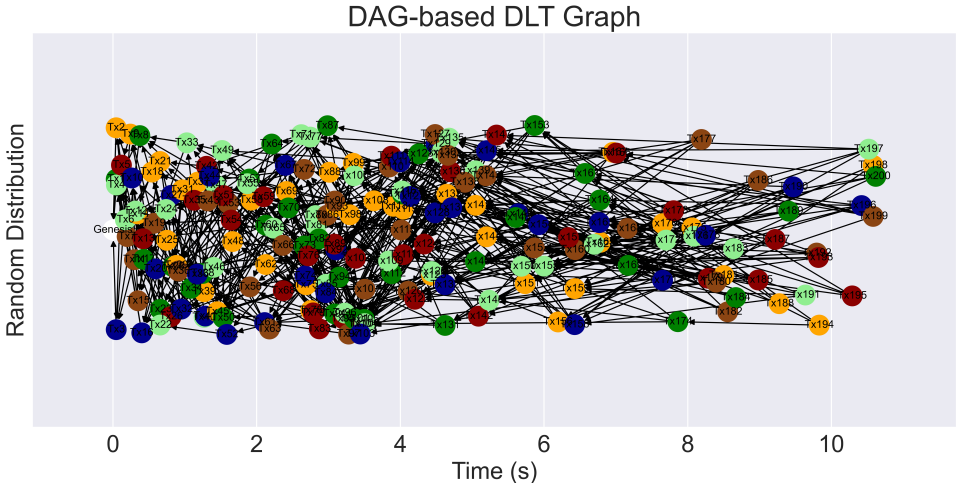


Figure 5.7: Intruder - Distribution of transaction nodes in DLT graph

5.2.3 Multi-agent System with Intra-fraud

Simulation Configuration The intra-fraud simulation configuration (in Tab. 5.7) was identical to the two previous experiments, except for the five illegitimate robots.

Simulation settings	**None**
Size_x of Warehouse	8
Size_y of Warehouse	8
Number of Robots in the Swarm	6
Sensing Radius of robots	2
Number of Malicious robots	5
Number of Tasks	200

Table 5.7: Simulation Configuration of Intra-fraud scenario

Simulation Results Tab. 5.8 shows that the task execution times are almost identical to the simulation results of the first two scenarios. The communication amount increases by about 20% in this scenario since the illegitimate robots are also involved in the distributed ledger update process during the task execution. Over-attachment nodes represent less than 5% of all nodes in the distributed ledger, which has virtually no impact on the system's operational efficiency.

Metric	Value
Time eclapsed	9.72
Communication Amount	240.67
Number of Tips	8.3
Number of credentials needed in average	4.0
Over attachment nodes	4

Table 5.8: Simulation results with Intra-fraud scenario

Credentials, Tips Similarly, Fig. 5.8, 5.9 present a change in credentials required to submit a transaction and the change in Tips in the ledger, respectively.

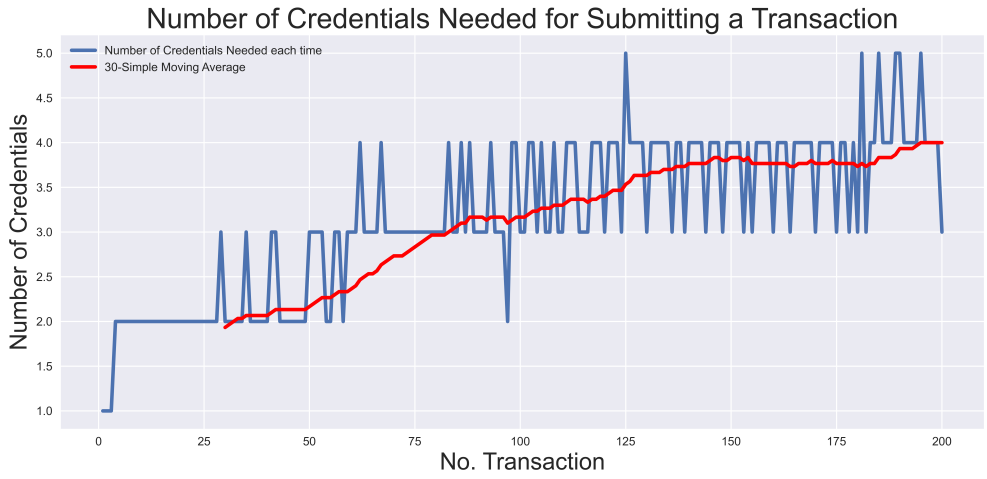


Figure 5.8: Intra-fraud - Number of credentials needed for each transaction submission

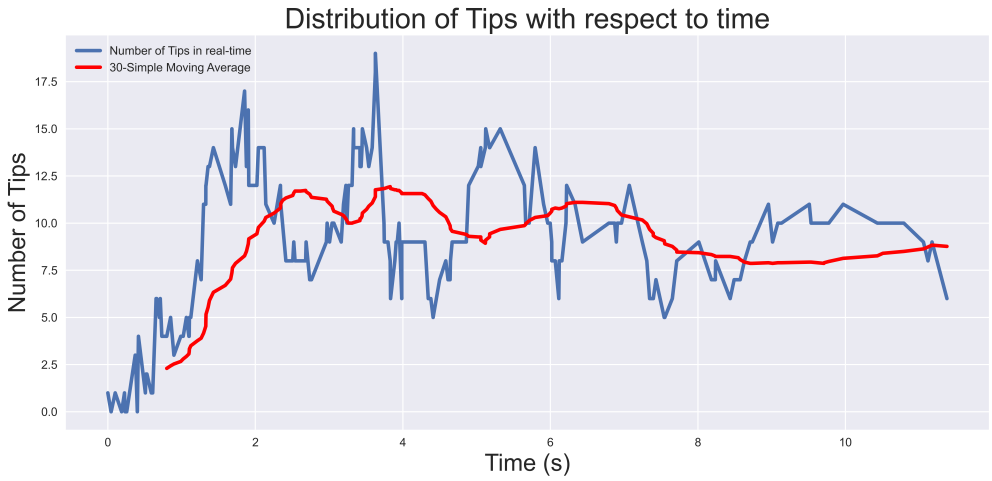


Figure 5.9: Intra-fraud - Number of Tips in real-time

DAG-DLT Ledger Visualization The DAG-DLT graph generated in this scenario differs slightly from the previous two simulations. Since illegitimate robots can participate in task execution and submit corresponding transactions, the fake transactions are shown in the red rectangular in Fig. 5.10. All these transactions were orphaned after 4 times of failed verification as no other transaction was attached to them. Also, the central node will block the corresponding submitters. Some of the robot's time was wasted trying to verify these fake transactions, so the transaction density was slightly lower in this period than in other parts.

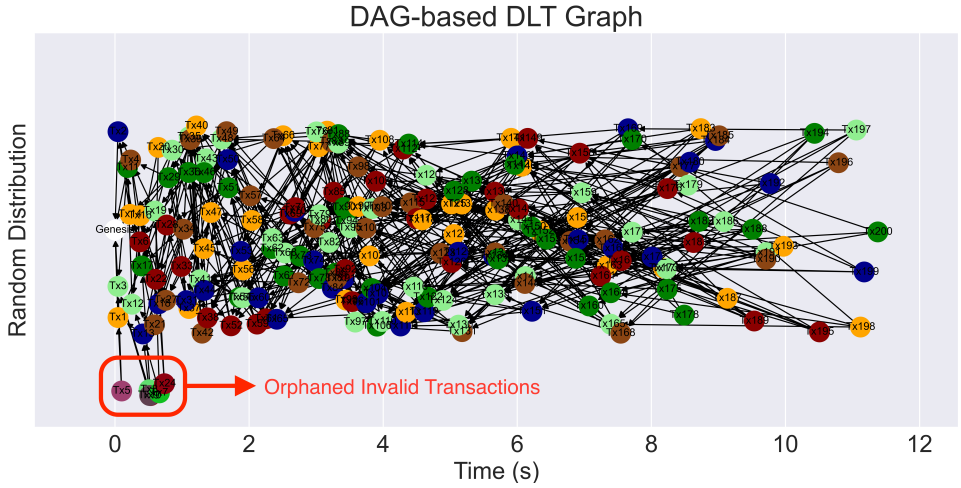


Figure 5.10: Intra-fraud - Distribution of transaction nodes in DLT graph

Illegitimate Robots detected Tab. 5.9 listed the locations and detected times of all 5 illegitimate robots in the system. It costs more time in the intra-fraud scenario to detect all the illegitimate robots since we can only confirm these robots as illegitimate robots via transaction verification processes.

Reference	Identity	Location	Time
MRobot0	32fa748c878881d0bc235c1133d...77a9c7baed0562c98980d	(6, 6)	0.18
MRobot1	856bd3f31471218313ed398a2f4...742378c272ddba933e8e	(5, 7)	0.2
MRobot2	6772287ac595954dea2926fd5c8...89252dfebd78e47f8ab4	(5, 6)	0.4
MRobot3	1eb775b77cf9d4f284e574565aaa...0073794171fe7659f3e	(1, 5)	0.8
MRobot4	aae113982e9c6bd3536769935a9...4189e01463d8c23a58e	(2, 4)	0.77

Table 5.9: Detected illegitimate robots with their corresponding location and detected time

5.3 Batch Testing

In this section, we will outline the overall performance of the SMARS system based on batch tests (e.g., 10 times simulation for each configuration setting) results. First, we will test the change in task execution time and communication amount with increasing task number in the

no-attack scenario. We will also analyse the change in communication amount as the number of robots increases. The second part will focus on a heatmap to analyse how the number of over-attached nodes changes as the number of robots and tasks increases. Finally, we will test the average time distribution of malicious robots detected in the two attack scenarios.

5.3.1 System Performance with the No-attack Scenario

We conducted three groups of batch tests (e.g., 10 times for each configuration) without attacks. The first set of tests was based on the configuration in Tab. 5.10, fixing the number of robots at 5 and setting the number of tasks in the range of (100, 1000) with an interval of 100 (e.g., the red text in Tab. 5.10). For each configuration combination, we ran ten times to get the average value. The line in Fig. 5.11 indicates the average time elapsed to complete a fixed number of tasks. It can be seen that the time to complete all the tasks does not increase linearly as the number of tasks grows. With the number of tasks increases, the verification process will take more time to ensure that newly submitted transactions are verified promptly.

Simulation settings	**None**
Size_x of Warehouse	5
Size_y of Warehouse	5
Number of Robots in the MAS	5
Sensing and Communication radius	1
Number of Tasks	(100,1000)

Table 5.10: Simulation Configuration with no attacks (batch test 10)

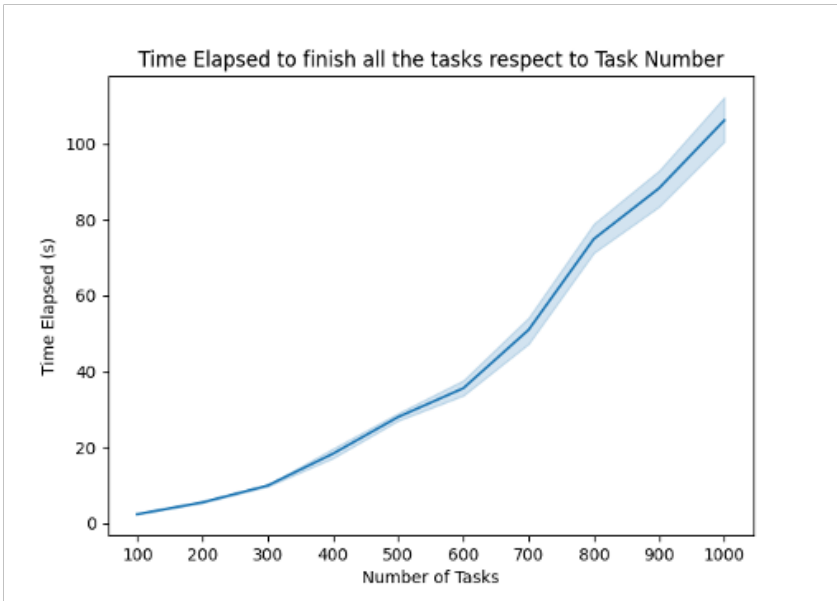


Figure 5.11: Time elapsed to finish all the tasks respect to task number

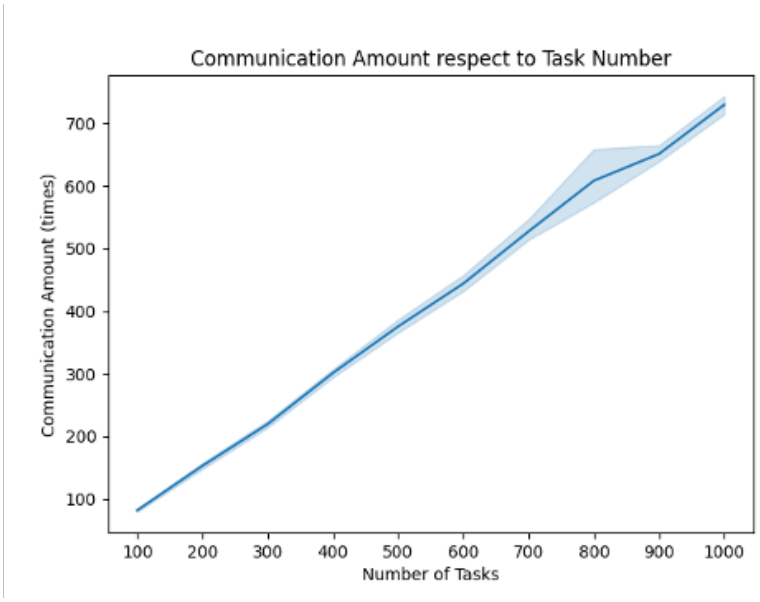


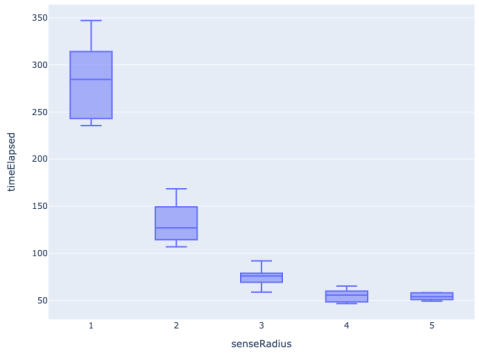
Figure 5.12: Communication Amount respect to task number

5.3.2 System Performance with different Sensing and Communication Radius

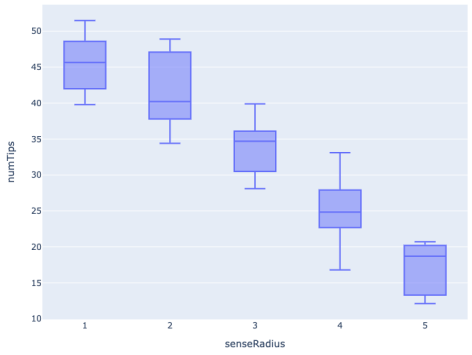
The robot's sensing and communication range can be personalized in the simulation program. A large-scale batch test was carried out with the configuration in Tab. 5.11 to explore this parameter's impact on the system's overall performance. With all other parameters fixed, the number of Tips in the DAG-DLT ledger (as shown in Fig. 5.13b) and the amount of inter-robot communication (as shown in Fig. 5.13e) significantly decreased as the sensing radius increased. Also, the time required to finish all the tasks initialized was reduced, as shown in Fig. 5.13a. On the other hand, the number of over-attachment points 5.13c increases from 0 to 3. This has a negligible impact on the system's performance, as almost all transactions are still verified an average of 4 times. The number of credentials required (as shown in Fig. 5.13d) for each robot to submit a transaction fluctuates slightly as the sensing and communication range increases, but the average value is around 4. Taken together, this means that a larger sensing and communication range benefits the system's overall performance.

Simulation settings	
Size x of Warehouse	25
Size y of Warehouse	25
Number of Robots in the MAS	6
Sensing and Communication radius	(1,5)
Number of Tasks	300

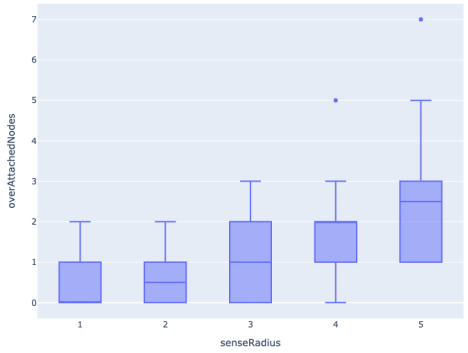
Table 5.11: Simulation Configuration with a range of Sensing and Communication Radius (batch test 10)



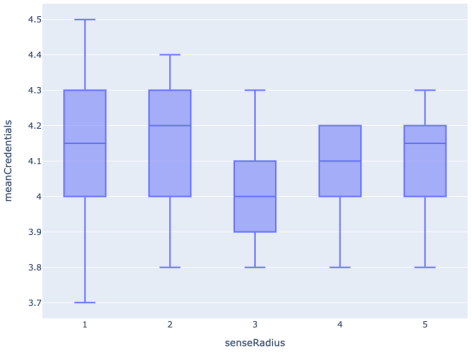
(a) Time Elapsed to finish all the Tasks



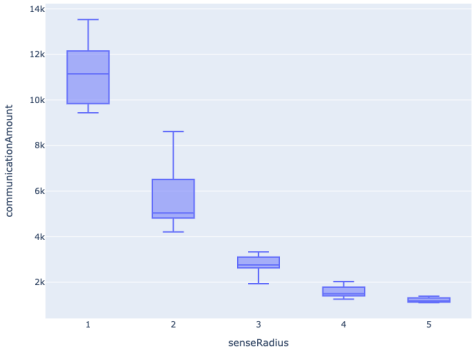
(b) Tips in DAG-DLT Ledger



(c) Over-attachment Nodes



(d) Credentials Needed for each Transaction



(e) Communication Amount

Figure 5.13: Simulation Results respect to Sensing and Communication Radius Changing

5.3.3 Number of Over-attachment Nodes in the DAG-DLT Graph

To test the severity of the over-attachment problem in the currently distributed ledger synchronization without a consensus mechanism, we conducted batch tests (e.g., 10 times for each configuration) to verify the system's performance in two dimensions. Fig. 5.14 is a heatmap of

the growth of over-attachment points, which shows a linear growth of over-attachment nodes as the number of tasks grows or as the number of bots grows. However, when increasing both the number of tasks and the number of robots, the growth of over-attachment points tends to increase almost exponentially (values on the diagonal in the graph).

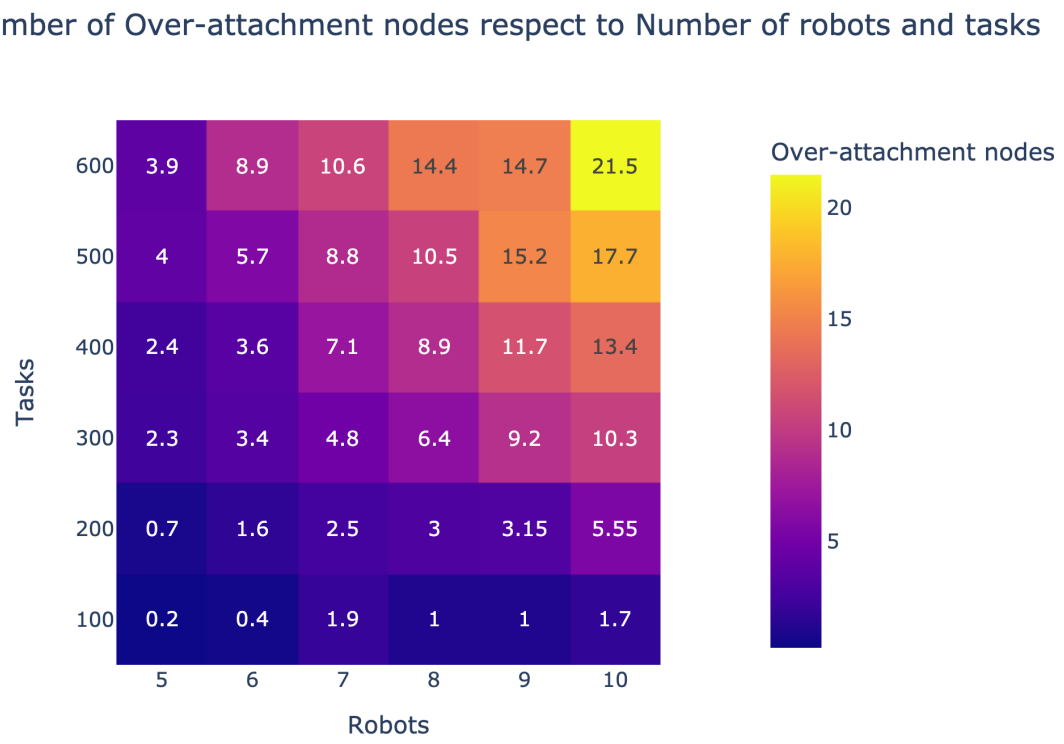


Figure 5.14: Number of Over-attachment nodes respect to Number of robots and tasks

5.3.4 Time Distribution of Malicious Intruders Detected

To verify the ability of the SMARS system to detect malicious intruders, we also conducted batch tests (e.g., 10 times for each configuration) and obtained a graph (Fig. 5.15) of the average time distribution of each malicious robot being detected. The simulation was conducted with the configuration in Tab. 5.12. By analysing the graph, we found that in a fixed scenario, all malicious robots can be detected in less than one second. This time (0.14s) is negligible compared to the system runtime (3.5s), which means that the malicious robots will not severely impact the system's normal operation. The time to detect five malicious robots is trending upwards mainly because we are using $multi-threading$, and since $multi-threading$ in $Python$ is concurrent rather than parallel, each robot sub-thread is not running simultaneously.

Simulation settings	**None**
Size_x of Warehouse	5
Size_y of Warehouse	5
Number of Robots in the MAS	5
Sensing and Communication radius	1
Number of Malicious intruders	5
Number of Tasks	100

Table 5.12: Simulation Configuration with malicious intruders (batch test 10)

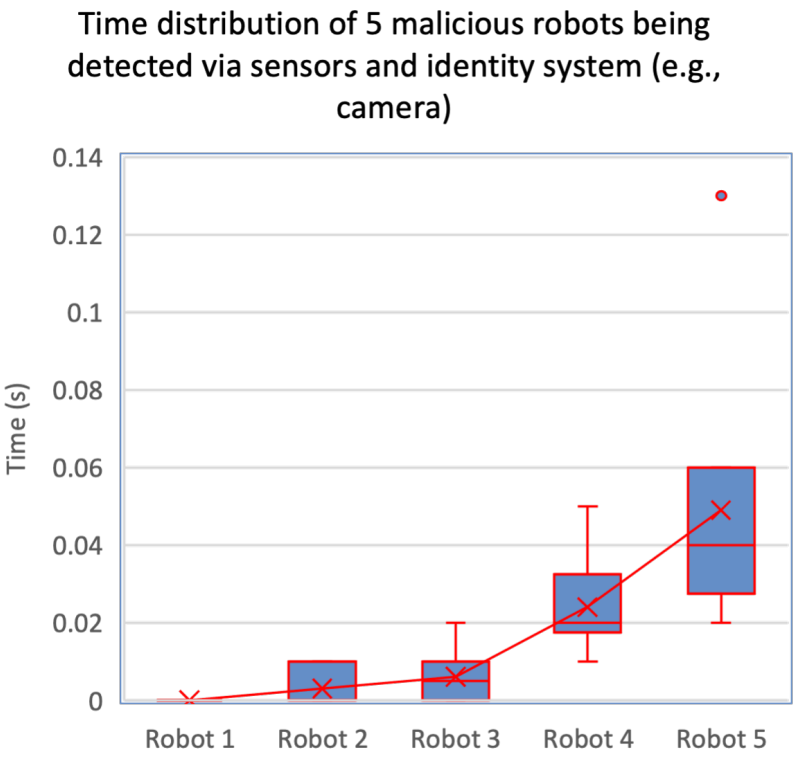


Figure 5.15: Time distribution of malicious intruders detected

5.3.5 Time Distribution of Illegitimate Robots Detected

With the same configuration and batch tests size (e.g., 10 times for each configuration) above in Tab. 5.12, the time distribution to detected each illegitimate robot (in Intra-fraud scenario) is shown in Fig. 5.16. The time to detect an illegitimate robot will cost more than the malicious intruder scenario since there must be a transaction verification process before an illegitimate robot can be located. Fortunately, the time distribution of each robot detected is less than 1 second. Compared to the system runtime (4.7s), it is still substantial. The upward trend in the time distribution for detecting five illegitimate robots is also mainly due to $multi-threading$ programming.

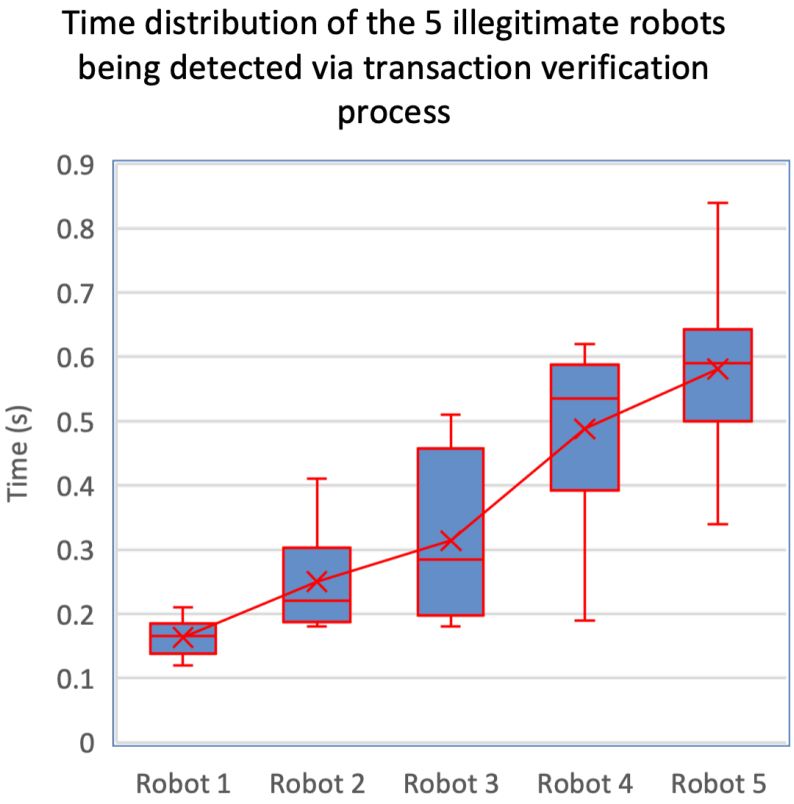


Figure 5.16: Time distribution of illegitimate robots detected

Chapter 6

Summary

In this Chapter, we draw the conclusion based on the analysis in Chapter 5. We also discuss the prospects for applying this approach and the directions for its improvement by relating it to the current applications of MAS in warehousing scenarios. Furthermore, we have presented several future directions for work on later.

6.1 Conclusion

In this project, we proposed a Self-Supervised Multi-Agent Robotic System (SMARS) focusing on the current potential security issues in the field of smart warehousing. SMARS was originally designed to prevent malicious intruders, identity impersonation, intra-fraud and Sybil attacks in current unmanned smart warehouses. The system primarily uses distributed ledger technology to record task completion information of robots (in the form of transactions) and provide the other robots with a way to verify it. Since we use a piggyback verification approach to physically and cryptographically validate the transactions in the ledger, the additional resource overhead incurred is small. In addition, we design a robot identity management system based on asymmetric cryptographic algorithms (e.g., ECDSA) and a message integrity verification method to ensure the authenticity and reliability of robot identities in the system.

Statistical-based Analysis From the batch simulation results in Chapter 5, it is apparent that SMARS can locate all of the malicious intruders in a very short period of time compared to the system runtime (e.g., within the first 4% of the time period of the system runtime). Also, robots in the SAMRS system can locate illegitimate robots submitting fake transactions in a timely manner through transaction verification (e.g., within the first 20% of the system's runtime). In addition, the overall performance of the system will improve significantly as the sensing and communication range of the robots increases (as analysed in subsection 5.3.2).

System Deficiencies While the system can meet expectations in terms of preventing malicious attacks, improvements in system design are still needed to improve the overall efficiency of the system. The current system requires extensive synchronization of ledgers between robots to keep each robot's locally stored ledger up to date, which requires high communication bandwidth and consumes many system resources. Another problem is related to the reward and punishment mechanism of the incentive system, where a robot is currently rewarded with 4 times the reputation value of the task weight for a transaction that is certified as valid, while submitting a fake transaction only loses reputation value equal to the task weight. The vulnerability in the reward and punishment mechanism used will cause the system to degrade significantly over time to the point of complete downtime. Since a transaction needs to be validated by four different robots, if some robots consider it as a valid transaction while others consider it invalid, this would cause another problem called Byzantine Fault. We will discuss the possible solutions to these problems in Section 6.2.

6.2 Limitations and Future Directions

As the ultimate goal of this project was to build a fully distributed MAS, we need distributed identity and task management systems. In addition, experimental simulation results and analysis reveal that distributed ledger synchronisation delays between robots will lead to over-attachment problems, and the rationality of rewards and penalties in the incentive system will affect the overall performance of the system. This chapter will based on the limitations of the SMARS and the ultimate goal of this research to list several possible future directions that we can work on.

Distributed Task Management System & Identity System The security management and control of the SMARS system are distributed, but task allocation and task creation still need to be done through a central node. This means the system is still not fully autonomous or distributed for task execution. We could then design a distributed task management system [28] for task creation and distribution, which would increase the robustness and flexibility of the system. Similarly, the identity system relies on a central node. A distributed identity management system (e.g., Web of trust (WoT) [29]) could replace the current central identity management node and make the system fully distributed.

Smart Contract & Consensus Mechanism There are two problems can be solved with smart contract [30] and consensus mechanism [31]. The first is the number of times a single transaction can be validated in DLT. In the current system, a single transaction is validated up to four times, but due to DLT synchronisation delays (e.g., propagation delays, out-of-communication range problems), a single transaction may be validated more than four times, which is named the over-attachment problem. In this case, more than four other transaction

nodes may be attached to a transaction node. This problem can be optimised by adding a consensus mechanism. Secondly, a single transaction node verified by multiple robots may get different verification results. In a detailed explanation, half of the verifiers consider a transaction valid while the other half consider it invalid. A problem will be raised, so called Byzantine fault. An additional auditing strategy is needed to resolve this disagreement. Research [3] has found that this problem can be dealt with by a consensus mechanism or a central coordinator [15].

Incentive System Optimization In the SMARS, we use an incentive system and reputation values as criteria for judgement. The current SMARS system needs to be improved concerning the initial reputation value setting and the criteria for reward or penalty. Since after a single robot has performed enough tasks and submitted the corresponding transactions, if other robots do not verify the transactions in time, the robot cannot continue to perform the tasks. This problem would lead to a loss of system efficiency.

Reaction to Malicious Robots Currently, the SMARS system can prevent four different types of attacks, detect malicious robots, and report them to the central node. The assumption of the system is that the system administrator will take precautions based on alerts, and this approach still relies on human intervention. We could then add an eviction function to each robot so that if a malicious robot is detected and identified, the legitimate robots in the system can take action to capture or evict it.

Multi-Processing or ROS based Simulation The project simulation is based on multi-threaded programming in Python. Since Python multi-threading is done in a concurrent rather than parallel form, the robot's task execution scenario is somewhat in error with the real scenario. This problem can be solved using multi-process programming or simulating through the ROS platform [1].

[1]ROS2 foxy documentation: `https://docs.ros.org/en/foxy/index.html`

Appendix A

Classes defined in the SMARS Simulation Platform

A.1 Classes Overview

Tab. A.1 lists all the classes defined in the simulation platform with a brief description for each class.

Classes	**Definition**
Warehouse	A 2-dimentional (e.g., x and y) plane that has x*y storage areas.
Storage_area	Each storage area has an independent inventory.
Object	Object has properties of weight and volume.
Task_management_system	A system to manage all the tasks and responsible for task creation and task allocation.
Task	Task has the delivery information (e.g., source and destination) of an object.
Identity_system	Supply multiple functions for robots to generate public/private keys and identities.
Robot	Legitimate robots in the system.
Robot_intruder	Malicious intruders that move the objects randomly in the warehouse.
Robot_intrafraud	Illegitimate robots that deliver objects to wrong storage areas.
DLT_graph	Distributed ledger is maintained by all the legitimate robots in the system.
Credential	A credential is obtained by a robot when it verified a transaction successfully.
Transaction	All the task execution information will be recorded in the form of a transaction.

Table A.1: Classes defined in the SMARS system simulation program

A.2 Robot Class

The properties of the robot and its corresponding functions are shown in Fig. A.1.

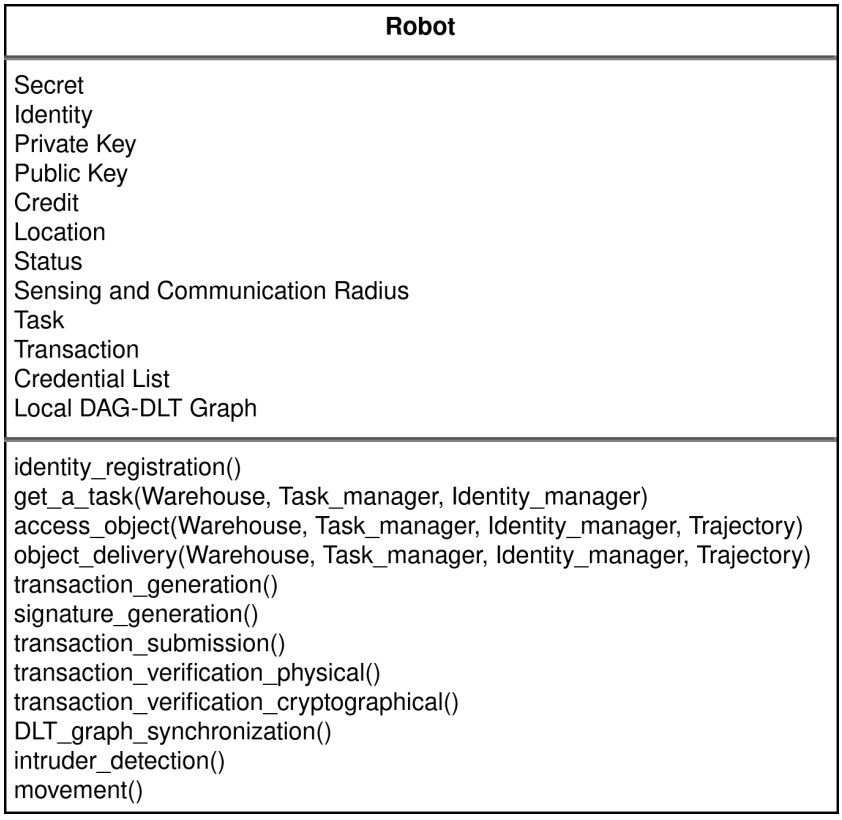


Figure A.1: Robot class defined in simulation program

Appendix B

Simulation Results on a Webpage

B.1 Distribution of Objects in the Warehouse

Here are two figures showing the distribution of the objects in a warehouse before (as shown in Fig. B.1) and after (as shown in Fig. B.2) being transported. The simulation is with the configuration of a 6*6 warehouse, 5 robots, sensing and communication range of 1, and 200 tasks.

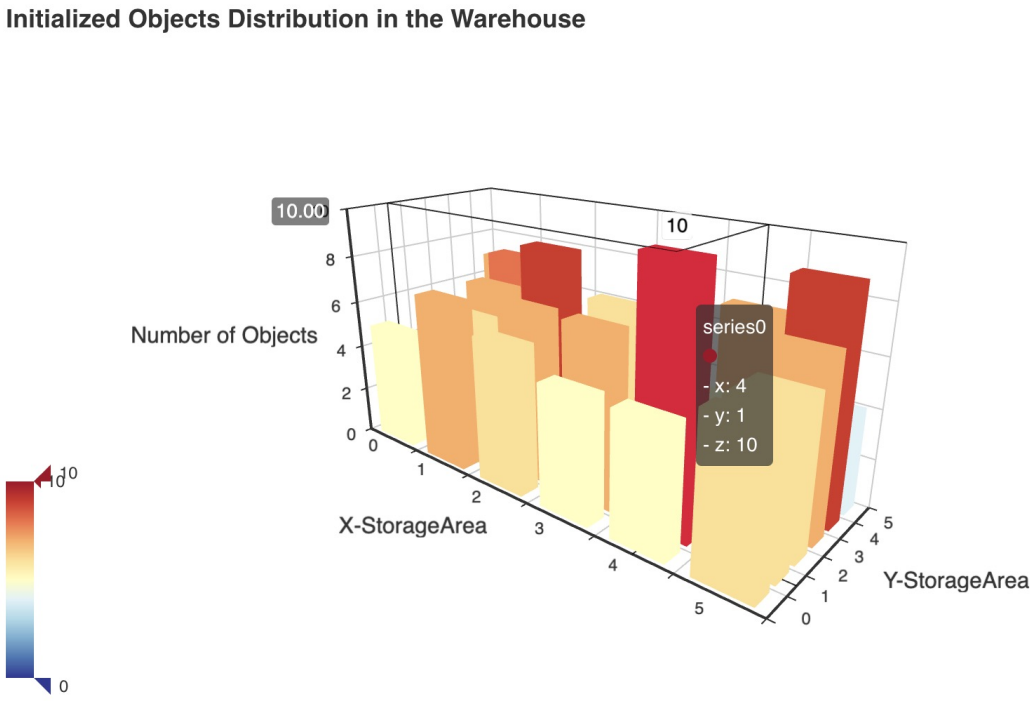


Figure B.1: Object distribution before tasks being executed

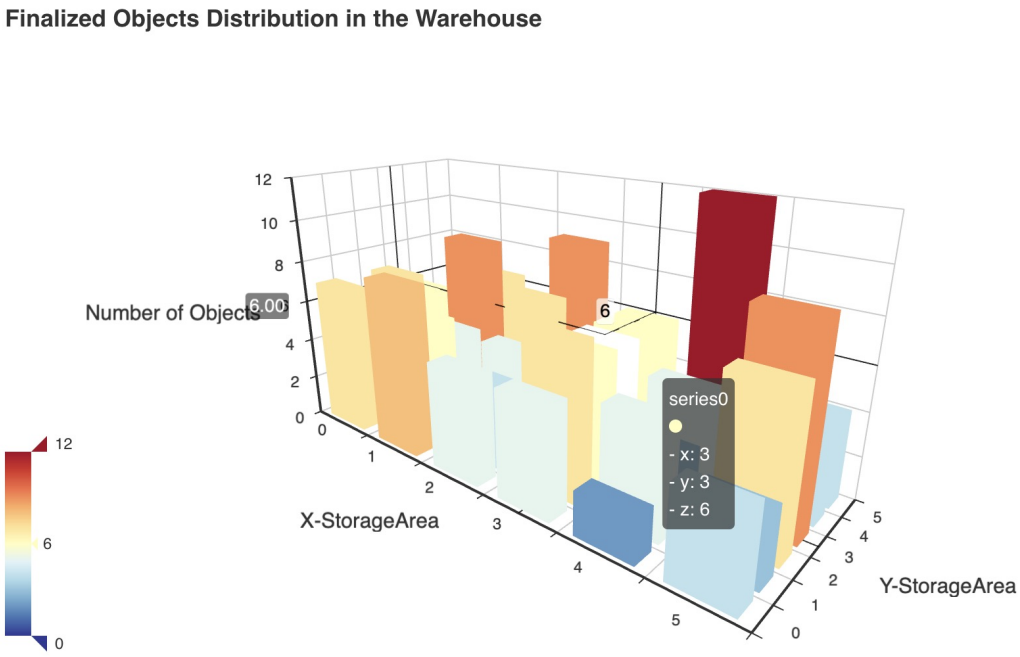


Figure B.2: Object distribution after all the tasks have been finished

B.2 Contents of a Transaction

Fig. B.3 shows a real transaction generated during a simulation.

▼ Tx23

Timestamp	**2022-09-08 17:23:18.597789**
Link to	['Tx11', 'Tx3']
Task	Task15
Submitting Robot	77fb1c93303bd256a7dda6e21faa3d929a7491da3d3b91fd824bc0db04b7d687
Credit of Submitting Robot	262.0
Object	OBJECT15
Source	(0, 0)
Destination	(4, 5)
Distance	9
Unloading Point	SA4-5
Task Weight	15.5
Public Key of Submitting Robot	b'-----BEGIN PUBLIC KEY----- \nMHYwEAYHKoZIzj0CAQYFK4EEACIDYgAEHM6haYoWhFgiN/ERwtssT6PFVVGYLr+b\nXWwIH9r7 -END PUBLIC KEY-----\n'
Signature	b'0e\x020\x0em\xbauc\xa8\xf8\xab\xac\xc0M\xb3\xd8\xc7a\x83\xf5\x18\xfe\xfc1\xea\xaf2\x8fU\x (V\x9d:i\x89\xa4-8;\x1f\x93%\xa3\x9eG\xe2\xfa\xe0\xe5\xda\x07\r<\xea\xcd\xc3\xb6\xd6z\x1b\x8
Verifier	['e12159ac8e8b2ed22c8b02cbbcda03b8a2a6c9459c90ae4b865ae6d221f2152f', '7d51324fdb7 '269a49dfb231530fab8fa1013a49c17aa2f2bbe5ca3ff0a0891a9af81854b845', '601bc885375796

Figure B.3: Transaction generated by a legitimate robot

Bibliography

[1] N. J. G. Saho and E. C. Ezin, "Survey on asymmetric cryptographic algorithms in embedded systems," *IJISRT*, vol. 5, pp. 544–554, 2020.

[2] E. C. Ferrer, T. Hardjono, A. Pentland, and M. Dorigo, "Secure and secret cooperation in robot swarms," *Science Robotics*, vol. 6, no. 56, p. eabf1538, 2021.

[3] V. Strobel, E. Castelló Ferrer, and M. Dorigo, "Managing byzantine robots via blockchain technology in a swarm robotics collective decision making scenario," 2018.

[4] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8076–8094, 2019.

[5] S. Bijani and D. Robertson, "A review of attacks and security approaches in open multi-agent systems," *Artificial Intelligence Review*, vol. 42, no. 4, pp. 607–636, 2014.

[6] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *Ieee Access*, vol. 6, pp. 28 573–28 593, 2018.

[7] T. H. Chung, "Offensive swarm-enabled tactics (offset)." DARPA, 2021.

[8] S. Kumar, A. Nayyar, and A. Paul, *Swarm intelligence and evolutionary algorithms in healthcare and drug development*. CRC Press, 2019.

[9] J. Wen, L. He, and F. Zhu, "Swarm robotics control and communications: Imminent challenges for next generation smart logistics," *IEEE Communications Magazine*, vol. 56, no. 7, pp. 102–107, 2018.

[10] A. Jeradi, M. M. Raeissi, A. Farinelli, N. Brooks, and P. Scerri, "Focused exploration for cooperative robotic watercraft." in *AIRO@ AI* IA*, 2015, pp. 83–93.

[11] R. Bogue, "Growth in e-commerce boosts innovation in the warehouse robot market," *Industrial Robot: An International Journal*, 2016.

[12] Q. Zhu, S. W. Loke, R. Trujillo-Rasua, F. Jiang, and Y. Xiang, "Applications of distributed ledger technologies to the internet of things: A survey," *ACM computing surveys (CSUR)*, vol. 52, no. 6, pp. 1–34, 2019.

[13] M. Bottone, F. Raimondi, and G. Primiero, "Multi-agent based simulations of block-free distributed ledgers," in *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. IEEE, 2018, pp. 585–590.

[14] M. Kedziora, P. Kozlowski, and P. Jozwiak, "Security of blockchain distributed ledger consensus mechanism in context of the sybil attack," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 2020, pp. 407–418.

[15] S. Popov, "The tangle," *White paper*, vol. 1, no. 3, 2018.

[16] A. T. Fiona Higgins and K. M.Martin, "Security challenges for swarm robotics," 2008.

[17] A. T. Fiona Higgins and K. M. Martin, "Threats to the swarm: Security considerations for swarm robotics," 2009.

[18] E. Castelló Ferrer, "The blockchain: a new framework for robotic swarm systems," in *Proceedings of the future technologies conference*. Springer, 2018, pp. 1037–1058.

[19] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," in *Third international symposium on information processing in sensor networks, 2004. IPSN 2004*. IEEE, 2004, pp. 259–268.

[20] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on automatic control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[21] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram, "Resilient asymptotic consensus in robust networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 4, pp. 766–781, 2013.

[22] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[23] X. Liu, B. Farahani, and F. Firouzi, "Distributed ledger technology," in *Intelligent Internet of Things*. Springer, 2020, pp. 393–431.

[24] A. K. Lenstra and E. R. Verheul, "Selecting cryptographic key sizes," *Journal of cryptology*, vol. 14, no. 4, pp. 255–293, 2001.

[25] D. Dolev, "The byzantine generals strike again," *Journal of algorithms*, vol. 3, no. 1, pp. 14–30, 1982.

[26] P. Swathi, C. Modi, and D. Patel, "Preventing sybil attack in blockchain using distributed behavior monitoring of miners," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 2019, pp. 1–6.

[27] S. Al-Sarawi, M. Anbar, K. Alieyan, and M. Alzubaidi, "Internet of things (iot) communication protocols," in *2017 8th International conference on information technology (ICIT)*. IEEE, 2017, pp. 685–690.

[28] T. Kreifelts, E. Hinrichs, and G. Woetzel, "Sharing to-do lists with a distributed task manager," in *Proceedings of the Third European Conference on Computer-Supported Cooperative Work 13–17 September 1993, Milan, Italy ECSCW'93*. Springer, 1993, pp. 31–46.

[29] Z. Li, X. Xu, L. Shi, J. Liu, and C. Liang, "Authentication in peer-to-peer network: Survey and research directions," in *2009 Third International Conference on Network and System Security*. IEEE, 2009, pp. 115–122.

[30] W. Zou, D. Lo, P. S. Kochhar, X.-B. D. Le, X. Xia, Y. Feng, Z. Chen, and B. Xu, "Smart contract development: Challenges and opportunities," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2084–2106, 2019.

[31] J. Huang, L. Kong, G. Chen, M.-Y. Wu, X. Liu, and P. Zeng, "Towards secure industrial iot: Blockchain system with credit-based consensus mechanism," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3680–3689, 2019.