BLOCKCHAIN-BASED IOT INTRUSION DETECTION SYSTEM WITH MULTI-AGENT SYSTEMS.

Submitted by

Meherab Hossain

ID: 202-35-650

Department of Software Engineering

Daffodil International University

Supervised by

Dr. Imran Mahmud

Associate Professor and Head

Department of Software Engineering

Daffodil International University

A thesis submitted in partial fulfillment of the requirement for the degree of Bachelorof Science in Software Engineering

Spring 2024

APPROVAL

This thesis titled on "**BLOCKCHAIN-BASED IOT INTRUSION DETECTION SYSTEM WITH MULTI- AGENT SYSTEMS.**", submitted by **Meherab Hossain Ope (ID: 202-35-650)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science inSoftware Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

Fazla Elahe 10.7.24

Chairman

Dr. Md. Fazla Elahe
Assistant Professor & Associate Head
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 1

Tapushe Rabaya Toma
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 2

Khalid Been Md. Badruzzaman Biplob
Lecturer (Senior Scale)
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

External Examiner

Md Mostafiz Khan
Managing Director
Tecognize Solutions Limited

DECLARATION

I announce that i am rendering this study document under Dr. Imran Mahmud, Associate Professor & Head, Department of Software Engineering, Daffodil International University. I therefore, state that this work or any portion of it was not proposed here therefore for Bachelor's degree or any graduation.

Supervised By

Dr. Imran Mahmud

Associate Professor & Head

Department of Software Engineering

Daffodil International University

Submitted by

Meherab Hossain

ID: 202-35-650

Department of Software Engineering

Daffodil International University

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to Almighty Allah who supported me throughout my bachelor's degree journey and the completion of this thesis. I would like to extend my heartfelt thanks to my family. To my parents, thank you for your endless love, patience, and belief in me. First and foremost, I would like to thank my supervisor, Dr. Imran Mahmud Associate Professor and Head of the Department of Software Engineering, for his invaluable guidance, patience, and support. His expertise and constructive feedback have been instrumental in shaping this thesis. I am deeply thankful to all the esteemed teachers who have guided me throughout my educational journey. I am truly fortunate to have had them as my teachers. I would also like to acknowledge the administrative staff at Daffodil International University for their assistance and for providing the resources needed for my research. I would like to express my profound gratitude to everyone who has contributed to completing this thesis. Your support has made this achievement possible.

ABSTRACT

With the increasing adoption of IoT technologies, securing IoT networks has become increasingly critical. Traditional intrusion detection systems face challenges in IoT environments due to resource constraints and the complexity of these networks. This tpes of research develops, evaluates, and implements an intrusion detection system incorporating deep learning algorithms, blockchain technology, and a hybrid placement approach within a multi-agent framework. The system comprises modules for response, analysis, data management, and data collection. Testing is conducted using the NSL-KDD dataset from the National Security Lab-Knowledge Discovery and Data Mining. The findings indicate that deep learning approaches are viable for intrusion detection in IoT networks.

Keywords: intrusion detection system, Internet of Things, blockchain, multi-agent system

TABLE OF CONTENT

Chapter 1(Introduction)

Introduction

IOT refers to a network of interconnected objects that communicate data using standardized protocols. This interconnected system is becoming more pervasive, as advancements in IoT are embedding intelligence into everyday works. IOT has rapidly emerged as one of the most dynamic sectors within contemporary computing and communication technology, influencing various industries, from automotive automation to agriculture. Due to its interaction with numerous connected devices in daily life, IoT is sometimes referred to as the Internet of Everything (IoE). It is estimated that by 2025, there could be up to 21.5 billion connected devices.

The IoT ecosystem comprises multiple layers, with the network layer being a critical component. The primary function of the NL is to facilitate data packet transfer between hosts, structured according to conventional Internet communication layers. However, the network layer is both complex and vulnerable, giving rise to various security challenges. While several security frameworks have been developed to address these concerns, they require integration into both the IoT architecture and the devices themselves. Unfortunately, many existing frameworks demand considerable storage and processing power, although solutions such as lightweight encryption and authentication methods help mitigate these limitations.

One significant factor contributing to security challenges in IoT is the sheer number of nodes—devices connected within the network—where a security breach in one node could compromise the entire system. Common security threats faced by IoT systems include botnets, ransomware, distributed denial of service (DDoS) attacks, remote surveillance, routing attacks, and data leaks. Although security systems are generally considered the first line of defense against such attacks, the complexity and diversity of IoT systems make this approach less effective.

Intrusion detection systems (IDSs) have gained prominence as a more resilient solution in recent years. The concept of IDS was first introduced by James in 1980. IDSs are designed to detect unauthorized access within a designated area, where any host attempting unauthorized access to other nodes within an IoT environment is classified as an intruder. A typical IDS comprises three key components: an agent, an analysis engine, and a response module. The response module then processes the information received from the analysis engine. Despite advancements in IDS performance and reliability, attackers have developed increasingly sophisticated methods to bypass these systems. Moreover, traditional IDSs are not well-suited for managing the diverse network layers found in IoT environments.

Recent advancements in intelligent systems have led to the integration of distributed IDSs with various machine learning techniques, including artificial neural networks (ANN), deep learning, and reinforcement learning. However, the complexity of IDSs presents significant challenges for traditional ANNs. To fully harness the potential of IDSs in real-world scenarios, these shortcomings must be addressed.

Purpose of the Study

Research indicates that the NK, or TL, is the important target for attacks. This study primarily aims to explore the feasibility of using blockchain technology within a multi-agent system (MAS) to enhance intrusion detection. The goal is to develop an intelligent IDS capable of detecting intruders and preventing attacks in IoT environments. This involves evaluating current IDS models based on machine learning techniques and analyzing previous attacks on IoT systems. Additionally, the study assesses the limitations of existing IoT devices and proposes a strategy that leverages machine learning to defend against unanticipated intrusions at both the network and transport layers. Various optimization techniques are explored to improve IDS efficiency.

The remainder of this document is structured as follows: Section 2 provides an overview of IDS and related technologies. Section 3 introduces the IDS test and details its modules. Section 4 discusses the experimental results. Finally, Section 5 presents the conclusions and suggestions for future research.

Chapter 2(Literature Review)

2. Literature Review

Find out more about the most recent studies on IDS, which for IOT devices, a comprehensive assessment of the literature has been done. A thorough examination of the potential risks in the Internet of Things domain and the IDS-based defences against them was conducted. Protocols for IoT system communication were also considered.

2.1. History of Intrusion Detection System

IDS is typically comprised of both hardware and software techniques, and it monitors systems and network environments in order to detect hostile activity. Put another way, intrusion detection systems (IDS) are used to identify intrusions and send out prompt notifications. All things considered, IDS protects the networks and systems. IDS is typically used in conjunction with an intrusion prevention system and placed subsequent to the firewall. In the domains of IoT security and privacy research, IDS is not a novel phrase. A substantial number of publications have been released in the last several years.

In order to combat cyberattacks, this has led to the development of the idea of IDS embedding into IoT designs and devices [11–13]. The main focus of research is developing novel models and strategies to thwart intrusions in established network protocols. Nevertheless, IoT devices connected via IPv6 and other intricate network architectures cannot be used with conventional IDS procedures [14]. To safeguard and preserve privacy in the Internet of Things, IDS has to conduct more thorough study on the application of machine learning techniques.

2.1.1. Classes of IDS

IDS is classified in two main categories as follows:

1. Host-based IDS
2. Network-based IDS

IDSs based on hosts go through audit and log records to identify signs of intrusion. To safeguard host security from all angles, this type of intrusion detection system is typically deployed on critical servers. Compared to network-based IDS, host-based IDS has the advantage of offering more specific information, a lower false alarm rate, and less complexity. On the other hand, it makes the application system less effective and overly dependent on the host's log data and monitoring capabilities [15].Network-based IDSs require attention due to the peculiarities of the Internet of Things and the large number of IoT devices that can be connected to the network. Network-based intrusion detection systems (IDS) are able to identify potentially malicious activity and irregular network data flow. It has no effect on the business system's performance and does not modify the host settings. Normal business

operations won't be impacted, even in the event that the network IDS fails. One issue is that, in order to detect threats, network-based intrusion detection systems only examine the segment's direct connection and ignore adjacent network segments. Using network-based IDS to process encrypted sessions is similarly challenging. [16].

2.1.1. Detection Methods

Intrusion detection techniques are divided into four main groups based on the types of intrusion attempts that occur (Figure 1)

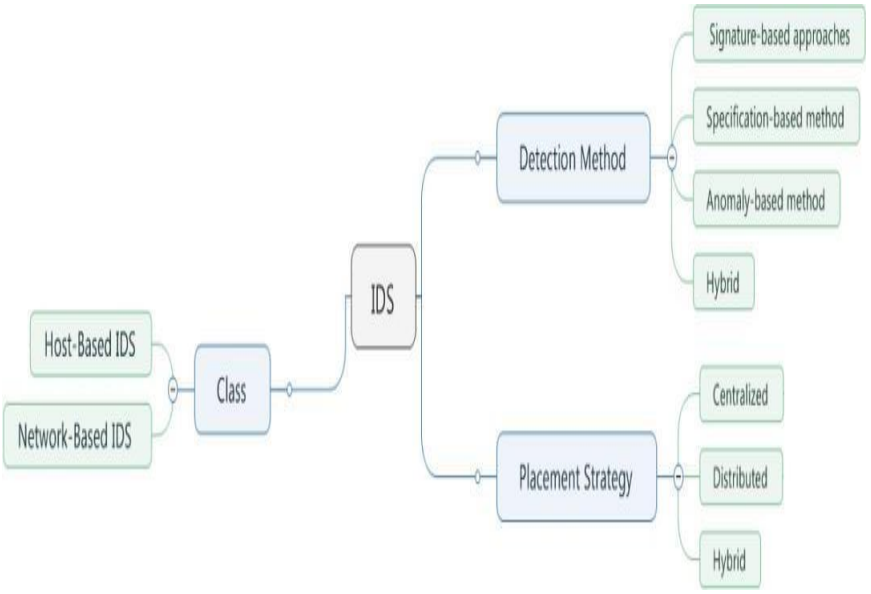


Figure 1 : Detection Method

Signature-based techniques start by scanning the network for data and then compare it to a feature database. The scanned data will be regarded as an intrusion if it is discovered to match the features in the signature database. The fact that it can precisely identify the kind of attack is a benefit. It is comparatively easy to use, and not a lot of resources are needed.

Setting rules and thresholds ahead of time is necessary for system administrators using specification-based methodologies. IDS monitors the state of the network and system in accordance with the administrator-established rules and thresholds [17]. When a rule is broken or the threshold is surpassed, the IDS will identify an unusual circumstance and take appropriate action.The approaches based on anomalies rely on comparing traffic patterns and recognizing anomalous patterns. The benefit of employing this technique is that it makes it possible to identify novel and unidentified invasions. The method's main drawback, meanwhile, is that it frequently produces significant false positive rates. To increase the robustness of anomaly-based intrusion detection techniques, research is currently concentrating on integrating machine learning algorithms into these techniques.

The employment of any combination of the aforementioned detection techniques within a single IDS is referred to as hybrid approaches. By addressing the limitations of a single technique, this strategy can improve the overall dependability of the Internet of Things system. The clear disadvantage is that the IDS as a whole will get extremely big and complicated. This will increase the resource requirements and make the system as a whole harder to run. Large amounts of time and resources will be required for the intrusion detection procedure, particularly when the IoT system involves numerous protocols.

2.1.2. Placement Strategy

Multiple monitoring nodes are used in centralized intrusion detection systems (Figure 1) to analyse host or network activities. The primary server or node will then get the data (either alerts, system logs, or network logs) for additional analysis. The central server load will grow as the system gets more complicated, which will lower system performance and possibly even lead to security issues. The majority of data interaction in an IoT environment takes place at the network and perceptual layers, and IoT facilities are dispersed and complexly distributed. For this reason, IoT device intrusions cannot be successfully detected by the centralized IDS [18].

Different duties are assigned to each of the components that make up Distributed IDS (DIDS). Usually, they divide up the monitoring duties. For instance, in an Internet of Things context, the risks that affect the network layer, perception layer, and application layer are distinct, necessitating the use of specialized intrusion detection modules for each. By doing this, the system's security can be significantly increased without requiring the modules to share and store a lot of data. Its tremendous scalability allows it to be tailored to future IoT contexts, which is its main advantage. The drawback is that there may be significant resource consumption and communication expenses [18].

Recent studies have offered a few hybrid placement options. The network is divided into distinct sections using the first hybrid placement technique. A node with intrusion detection capabilities will be present to keep an eye on other nodes. These nodes are also accountable for keeping an eye on the data package that is flowing from their nearby nodes. In the event that unusual activity is discovered, it is assumed that attackers have hacked the nearby nodes. After gathering data from the nodes, the border router decides whether or not there has been an intrusion. Also to be mentioned is the superior accuracy of hybrid IDS with dispersed components compared to centralized IDS. The hybrid system does, however, require a lot of resources .

2.2. Previous Research

A primary goal of this research is to enhance the IDS of IoT [17]. An summary of the advancements and enhancements of IDS for IoT is given in this section. These can be categorized according to the deployment plan, detection technique, and target threats. Bostani (2017) presented a hybrid and unsupervised intrusion detection method to identify sinkhole and selective-forwarding attacks for IoT [20]. In order to detect any discrepancies.

In accordance with an automation demonstration, suggested using a contemporary interruption finding technique for the Internet of Things [21]. Three kind of IoT attacks are distinguished by this strategy: reply-attacks, jam-attacks, and false-attacks. It is a named move framework expansion. Since the data collected by the organize hubs is provided to the interruption discovery module, which assists in creating the occasion databases, the arrangement of this IDS may take a centralized approach. Based on a strategy based on specifications, the IDS uses the occasion analyser to detect interruptions. In IoT networks, this intrusion detection system (IDS) can effectively

detect and identify reply, fake, and jam assaults .In 2017, Kapton et al. [22] also presented a technique to enhance communication between different by combining blockchain technology and multi-agent systems. In order to integrate communication via the they put out an architectural protocol. A protocol that might work with autonomous agents is the end product. The suggested protocol helps to predict the resilience of variable working states and guarantees the security of the communication process. This has prompted more research into the potential integration of multi-agent systems and blockchain technology.

According to Calvaresi et al. [23], the combination of MAS with blockchain technology (BCT) can distribute trust and eliminate. They created and put into use a system that integrated BTC, which was based on Hyperledger Fabric, with MAS, which was based on the Java agent development framework (JADE). The solution establishes a reliable community by extending the control of the agent identification to the Hyperledger Fabric membership service. Additionally, services utilizing the ledger are supplied to the association agent. Agent reputations are computed and stored transparently using mechanisms based on the interactions' evaluations and the agents' communication patterns. A variety of test cases are used to assess the system. Both user- dependent and autonomous actions make up agent behaviours. Actions are integrated with smart contract techniques. The solution is scalable and robust because to the available Hyperledger Fabric implementation. A traditional command-line interface makes testing simple and quick . MAS and BCT in practical settings raises additional ethical questions. The immutability of the ledger, and the achievement .Users' privacy may be jeopardized by intentional or inadvertent system malfunctions, as the framework entirely .Eliminate middlemen in disputes, there are still questions about the software's dependability and the BCT environment's verification procedure.

Calvaresi et al. [24] examined blockchain technology and multi-agent systems in 2018. Their analysis came to the conclusion that the difficulty of applying blockchain technology is comparable to the issues with multi-agent systems that have been previously discussed for other application domains. Many aspects were covered, including needs and motivations, application areas and processes, strengths and limits. The authors stressed that, in addition to discussing potential benefits, a thorough assessment of these systems is required.

Agents in frequently precarious situations where achieving "consensus" becomes unfeasible. It is not an easy process to find conflicts of interest among agent to resolve the ensuing sign. The model put forth by [25] looks at cooperative tendencies by taking into account the likelihood that each agent in the environment will cooperate. CP have the potential to increase the collective payoff. The best candidate action sets are then chosen using the NBS algorithm. It is anticipated that reducing memory footprint and "convergence time" will solve these problems. However, high dimensional settings with large state space are not yet suited for the existing solution. Compared to traditional IDS, this was shown to be more successful in recognizing IoT Fog orchestrate assaults.

Air Traffic Flow Management (ATFM) systems are crucial for the efficient functioning of air traffic control operations. These systems often require a high-fidelity, mathematically reliable model to support precise decision-making. However, due to the complexity of airspace management, designing such a model can be challenging. To address this issue, Ta et al. [27] developed Block Agent, an intelligent, multi-agent distributed ATFM system that integrates reinforcement learning and blockchain technology. The system consists of three layers: the local layer, the blockchain layer, and the global optimization layer. The local layer empowers regional stakeholders at various bases, while the blockchain layer, in contrast to conventional multi-agent ATFM architectures, provides decentralized coordination mechanisms. This layer manages and synchronizes a wide range of dynamic information, including airport operations, aircraft data, weather conditions, and route optimization.

Reinforcement learning is employed by the system to minimize aircraft delays, both during flight and prior to take-off. When evaluated in a constrained configuration, Block Agent performed better at handling delays than conventional ATFM systems. In order to make the system feasible for widespread adoption, the authors want to improve it.

A hybrid Internet of things architecture that enables decentralized data management via blockchain was presented by Casado-Vara et al. [28]. This architecture allows for the optimization of the blockchain-to-IoT link through a compute distribution based on the edge computing paradigm. Data management is another component of this hybrid architecture. This includes a big data ecosystem that makes managing massive volumes of data in the blockchain easier. To improve data quality and false data detection, a new game theory-based method is suggested and applied to the data gathered by IoT devices. The suggested hybrid architecture optimizes the current end-to-end designs.

A method for extracting features from IoT data were proposed by Li et al. in 2019 [29]. The suggested intrusion detection system (IDS) can handle sample misclassifications, spatial limits of data clustering issues, and lack of appropriate training sets by utilizing deep migration learning. While the results demonstrate that our suggested system outperforms existing approaches and efficiently shortens declines with compression. Le et al. proposed a new deep learning-based model in 2019 [30] for network intrusion detection. An intelligent intrusion feature selection . This step produced a dimension reduction that produced that were crucial for intrusion detection. Building numerous IDSs and training them with the chosen characteristics was the next step in the process. Several deep learning techniques, such as recurrent neural networks (RNN), were used in the learning process.

A novel intrusion detection approach for IoT devices with limited resources was suggested in another noteworthy study that Arshad et al. published in 2019 [31]. The goal of this approach is to keep edge router and Internet of Things device intrusion detection separate. In order to give the edge router a comprehensive overview of the network. This framework's drawback is that sensitive data may be included in the raw packets that the edge router receives from the host node.

A three-layer IDS approach was presented by Anthia et al. in 2019 [32] in order to differentiate between real-time harmful activity in home IoT devices. To detect and categorize threats, this architecture relies on identifying the kind and profile of each IoT device's typical activity. Scripts to launch multilevel attacks that mimic an attacker's activity and organized action data from a real test bed have been used to assess this IDS architecture. Future research should look into ways to get around the significant requirement for feature engineering and data labelling. Machine learning approaches are effective, according to responses to a survey done in 2019 by Chaabouni et al. [33]. The authors came to the conclusion that highly were shown using machine learning-based intrusion detection frameworks. They support more research and development, integrating IDSs with cutting-edge .The above highlight recent developments in IDS for IoT development.

2.3. Technology

Technology underlying our suggested IDS, which uses blockchain technology and a multiagent system, is explained in this section.

2.3.1. Multi-Agent System

In computer science and artificial intelligence (AI), an agent is defined as the intelligent behaviour of computer software. In general, are referred to as "multi- agent." A Multi-Agent System (MAS) comprises several agents and represents a development and utilization .Using a MAS, made simpler and more modular. Through their individual responsibilities, the agents are in charge of communication and coordination. Each agent in the MAS can be both an individual and a cluster because it is fully autonomous and form-independent [24,36,37]. Despite having diverse design patterns and being written in several languages, the agents should employ standard communication modes that encourage reciprocal communication, which is absent from single agent systems.

Every agent has a unique set of characteristics and guidelines for operation. They carry out tasks while the entire MAS is operating in accordance with these action rules. Humans are able to solve certain difficult phenomena Four fundamental qualities that agents should possess mostly demonstrated by the system's intellect. The ability of an application system to evaluate and comprehend various types of data and knowledge through learning, reasoning, and other methods is referred to as intelligence [38]. The term "agent capability" describes an agent's capacity to receive signals from the outside environment and respond on its own initiative based on its own understanding.

In order to evaluate the agent-based computing (ABC) IoT's feasibility for IoT developmen. Even though the authors claimed that there are no technological barriers preventing the IoT ecosystem from being fully realized in terms of computation, storage, and communication, more attention is still needed to address the ecosystem's many development challenges. According to the analysis, interoperability, automaticity, and distributed intelligence can be programmed into SO and IoT systems to varying degrees through the application of ABC. Compared to alternative strategies like component-oriented, object-oriented, and service-oriented computing paradigms, this modelling offers advantages. Moreover, the agent-based simulation can validate several design options before to the deployment stage. When used in combinatio, agent-based approaches can carry it out methodically and effectively. According to the report, many SOs and IoT systems find that using an agent-based development strategy is a wise decision. The suggested system in this study makes use of the JADE MAS platform [39], which offers a straightforward yet effective task execution and composition architecture. Asynchronous message passing is used to facilitate agent- to-agent communication. Because JADE is built on Java, it works with all platforms. JADE is compatible with several Java-oriented environments, including and Android devices. Additionally, a network setup with partial connectivity is supported by JADE.

2.3.2. Machine Learning

2.3.3. Deep Learning

The phrase "Deep Learning" has become more and more prominent in recent years. Similar to ANNs, deep learning draws inspiration from the overall architecture and operations of the brain. It creates a layered data flow network by having input and output layers. The terms deep neural network (DNN) and deep structured learning can also be used to characterize deep learning [40]. The hidden layers are the primary distinction between deep learning and ANN learning. The input and output layers are arranged hierarchically, followed by the hidden levels. It is more resilient than a normal ANN. The amount of data in the real world is increasing, which causes data complexity to rise. The inherent quality of deep learning is its capacity to learn from a huge body of data or from earlier layers. Because of this characteristic, deep learning is a very strong and potent contender for machine learning model selection, particularly when it comes to categorizing data from unlabelled sample datasets [41].

Multi-Agent Reinforcement Learning

Through ongoing training and learning from the beginning, the computer is able to complete the task thanks to reinforcement learning (RL) techniques [42]. Recent years have seen the development of DeepMind and its success at challenging challenges, such as Alpha Go, where RL has demonstrated significant research promise [43]. RL has a long history because research on it was first conducted in 1979 by Sutton et al. Although RL was first employed in the realm of intelligent control, it has since used areas as science and technology have advanced. In the fields of computational intelligence and machine learning, reinforcement learning is crucial.

The environment of multi-agent systems is controlled by the order in which the various agents act. Since MAS design rules dictate the RL method, concurrent isolated RL (CIRL) can be used in situations where a single agent solely uses an unsupervised learning strategy and does not communicate with other agents. Applying interactive reinforcement learning allows for the interaction learning of several agents. While multi-Agent RL

2.3.4. Blockchain

The primary driving force behind blockchain technology is decentralisation [45]. The blockchain's distributed and transparent ledger ensures that the failure of a single node doesn't impact the entire system. Blockchain transformed the transaction network's architecture from a star to a point-to-point (P2P) configuration. This updated framework, which uses encryption and security based on code and algorithm protection, enables direct communication between two parties. There is no need to know whether a party is trustworthy because all that is needed for the parties to create mutual trust in the transaction system is for them to trust the algorithm that is being used [46]. Moreover, since the algorithm handles all forms of authentication, the framework doesn't need the security endorsement of outside actors.

The two most widely used platforms for developing blockchain. They both use the same basic technologies. Ethereum provides access to the Ethereum virtual machine (EVM). Applications intended for general customers are the focus of public blockchains and smart contracts. The highly modular architecture of Hyperledger Fabric makes it better suited for corporate applications. It applies business logic more freely and offers a great deal of freedom. The objective is to use blockchain technology to streamline work and trade procedures, thereby resolving the inter-firm credit issue.

Chapter 3(Methodology)

3. SESS (Smart Efficient Secure and Scalable) System

Design science research and science and engineering research approaches are used in this work [49]. Through the integration of scientific study, engineering design, and implementation, this new intrusion detection system framework's viability and worth can be investigated.

3.1. System Design

3.1.1. System Perspective

This technology will be used to track the specifics of the IoT network status and identify assaults from the traffic on the network right now. The setup of response rules and the visualisation of detection results will be done using a web site. A blockchain smart contract module, a detection and analysis module, a response module, a data process module, and a collection module make up the five components of the system. The system will have to interact with Internet of Things devices and keep an eye on their data flow. The system needs a database to store data because it needs data to identify harmful activity on an IoT network. The web portal can only collect data; the system is able to add and edit data. Every database communication will take place via the Intranet or Internet.

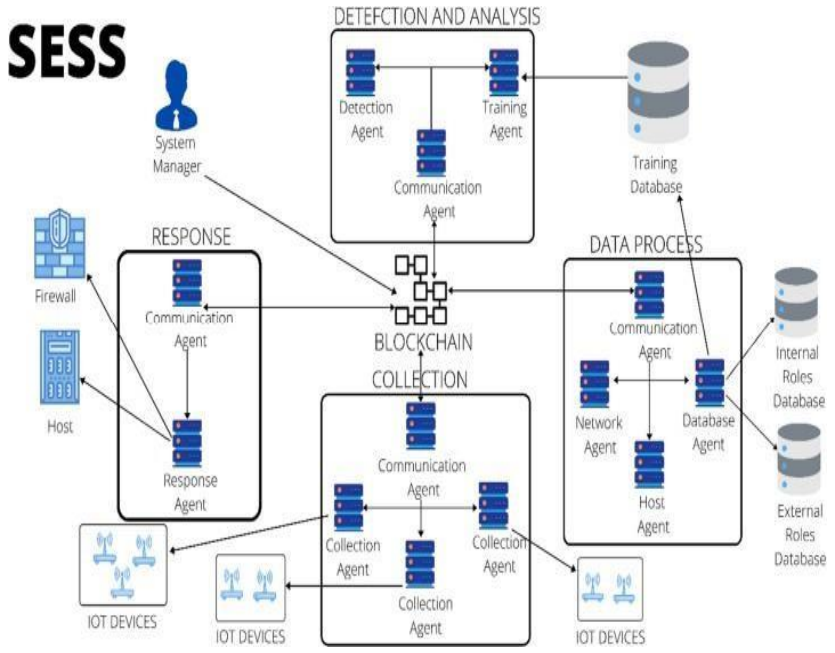
2.3.5. System Functions

The Smart, Efficient, Secure, and Scalable (SESS) system enables IoT network administrators to monitor IoT devices based on network traffic data [34]. The monitoring scope is determined by criteria set by the administrator. Administrators can customize by adjusting various parameters. The data process module then handles the gathered data, performing initial attack detection through feature classification. This data is subsequently divided into two categories: a training dataset and an unknown dataset. These datasets are forwarded while the unknown dataset is used to evaluate the model's performance. The response module processes the results according to the administrator’s predefined configuration. All activities are recorded in the blockchain ledgers of hosts that incorporate the SESS module.

2.3.6. User Characteristics

An IoT network administrator can utilise the system to detect intrusions, add data, modify the data processing and detection criteria, and, if needed, decrease the number of agents. The system components for expansive IoT networks will be installed and run on many hosts. This implies that an administrator is needed for every SESS module. The modules under their supervision ought to be configurable by these administrators.

This paper's [34] intrusion detection system (Figure 2) uses a multi-agent technique. Every agent is a comparatively autonomous entity. Modules for agents are divided into four categories. Through communication agents housed in every module, they are able to communicate with one another. Each module can function very independently in this fashion, which lessens the need for dependencies between components. A collection module, a data processing module, a detection and analysis module, and a reaction module make up the entire system. Because FIPA-ACL is supported by numerous



groups, SESS adopts it as the agent communication language for the Foundation of Intelligent Physical Agents (FIPA-ACL). Through interactive reinforcement learning, all four of the communication agents will be enhanced and changed. This implies that every agent has an impact on other agents.

Feedback is generated by each successful action taken by other agents and is communicated to the communication agent in the same module. Communication agents gather input from other agents in the module and compile it into a feedback report. Communication agents will receive training from the feedback reports. Communication agents will receive credit for their contributions to the feedback. Every move that agents of communication make will be regarded as a transaction. Most security concerns will be brought up by communication agents since they are the only agents with the authority to give directives. Only the system manager will have access to the blockchain's smart contract, or chain code, which will record transactions.

2.4. System Development

Traditional IoT configurations face several challenges, including weak device security, centralized data control, rigid architecture, compatibility issues with certain communication protocols, and difficulties in coordinating actions among diverse stakeholders. These problems must be addressed by the intrusion detection systems (IDS) implemented within IoT networks.

Blockchain technology, with its decentralized approach to storing and exchanging communication messages—essentially commands that direct system behavior—can significantly improve device security. By employing mutually agreed-upon smart contracts to govern agent behavior and system performance, blockchain also helps establish trust among different parties. Agents within the system allow it to scale flexibly, making it adaptable to various IoT topologies and communication protocols. As IoT networks continuously expand, the system must evolve rapidly. While blockchain ensures the security of the growing network, agents assist in adapting to these dynamic changes. The source code for this system is available on GitHub [50] and has been shared with the open-source community.

2.4.1. Blockchain Component

Private chains are employed in this system to protect agent-to-agent communication. The technology synchronises the start and stop times of the blockchain node and embedded database within the same agent. Agents' uses and storage of disc data will be accelerated by the addition of the most recent interactions to the cache area, which will also shorten search times. By conducting a local search of the recently updated data, storing a copy of the database (DB) in each agent will lower the overhead of the communication agent. The blockchain module's super node is the communication agent of the detection and analysis module. It is used to ensure that every node in the network may receive the full blockchain ledger. Complete nodes will be all other agents of communication. They are necessary for verifying each communication record in the blockchain and hold and distribute the complete blockchain ledger. Each module has additional agents that can be turned on as light nodes and connected to their parent node, the communication agent. These light nodes are meant to detect any tampering with their parent node and simply carry the block headers of the block. In addition to regulating and validating communication agent behaviours and guaranteeing system security, the blockchain module is utilised to foster trust in multiple-party IoT networks, each with its own set of agents.

Agent Account

The agent’s name, agent public Key, and agent privateKey will all be kept in the Agent Account object. The create Public and Private Key function will produce these keys. The cryptographic technique known as the Elliptic Curve Digital Signature Algorithm (ECDSA), which is also used to encrypt Bitcoin accounts, encrypts the keys. This guarantees that the appropriate agent can only generate specific communication messages. Agents can safeguard their data during communication by randomly generating their unique public and private keys using the ECDSA cryptographic technique.

Blockchain

Common methods that are required for blockchain products are contained in this class. These techniques are mostly applied to cryptographic algorithm-based encryption and decryption.

Communication

Among communication agents, the Communication class serves as a message container. Communication agents have a significant influence on the system's operation since they are the b

information exchanged between communication agents. A hash, data, a prior communication hash, the sender agent's private Key signature, the sender's public Key, and the recipient agent's public Key are all contained in communication objects. This communication object's identifier is the hash argument. The information that the sender agent wishes to convey to the destination agent is contained in the data argument. The hash value of earlier communication agents, which is used to keep track of the sequence of pertinent communication objects, is the preaches parameter. The sender agent's public Key can be used to validate the signature argument, which is generated using the sender agent's private Key. It is employed to confirm that the sender is the one who submitted the information. The public Keys of the sender and receiver agents are the sender and recipient arguments, respectively.

Block

A blockchain is a chain made up of various blocks. A block hash includes its own hash value, the hash of the preceding block, the generation time, Merkle Root, and the maximum block height. The calculation of Merkle Root is done recursively. Up to the calculation of the tree root, each successive tree layer is determined using the hash of the communications in the block as a basis. The first tree layer is determined in this manner. Target communication may be efficiently tracked and its integrity confirmed with the Merkle Root. This block's location is determined by the preceding hash, and the hash serves as an identification value. The block will be added to the blockchain and the Merkle Root and hash computed whenever the list's size reaches MAX_BLOCK_HEIGHT. After that, the modified blockchain will be broadcast to further modules and stored on the local host. The following page displays an illustration of a block created with blockchain technology.

Contract

All of the communication process's business logic is contained in the contract class. There are defined validations and permissions. They have to do with the particular communication messages that an agent can send to other agents and the timing of when this communication activity can be considered legitimate. Only this smart contract can be used by communication agents to initiate the communication function. Other agents that have new functions and are governed by communication agents may still be added to or withdrawn from the system, as this contract solely governs communication agents. To preserve scalability and security, the contract is exclusively utilized to protect communication between communication agents inside the management layer. Blockchain will initialize when the detection and analysis module is launched. Communication and block genesis will be recorded in the blockchain, which will then be disseminated to other modules and stored locally on

why it initializes this blockchain. The Contract specifies the kinds of communications that can be sent and stored. Only system managers have the ability to modify smart contracts prior to their initiation. Once the system has been initialized, they cannot be modified.

Example of block:

BOX 1

{"generateTime":"2019-05-22 15:53:54:342","hash":"cKbh/b3rZr3J9BZpswQUTUkZcsH+9hUKXWykboItV7U\u003d","prevBHash":"jsYdYWS+SuBFKHgR+bf3bUNjZnVNuH5z/Ng1EC+J0AU\u003d","merkleRoot":"Athz4br2Wg2XJnD6/+X9rijYLBX8fs9OFwnuGGvv/rg\u003d","MAX_BLOCK_HEIGHT":3,"communications":[{"prevCmHash":"MA\u003d\u003d","signature":"MDQCGGtbJbmxtPiIyKFasoz1Gm5n1oQBN9igTwIYUDlegkAUTORIB1I+rC1z/jMspVAUzH5K","hash":"8L/sR/YkZsbKPlDyFy/+50k1t08/s+BE+u+Y1k1XNqM\u003d","data":"{\"detectionDate\":\"2019-05-22 15:53:41:295\",\"attackNumber\":546,\"totalDataNumber\":1200}","sender":"MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAEDO+Qsl5H3N3M5cJUmdWkF9rlK8qKfWH1Bb5xTLXYF5xwY8F8VumLgqssdSG0YjpO","recipient":"MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAEb4ZjAt85ZODTgqSvTDR/USW78eu/dw26AIsVjz/FpP3d9O5+iYriNTrHJc/y+/zf"},{"prevCmHash":"MA\u003d\u003d","signature":"MDUCGQDblB6LqMRGDyD7NR9DyHTXfT813BaxIJMCGAI49gqxO8rgQTbFJ9qADByEuvEWkZQclQ\u003d\u003d","hash":"oes7At5QfwwN/gv91M7pwNfg3HZr/02OaWHB+9DdkXU\u003d","data":"{\"detectionDate\":\"2019-05-22 15:53:49:069\",\"attackNumber\":546,\"totalDataNumber\":1200}","sender":"MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAEDO+Qsl5H3N3M5cJUmdWkF9rlK8qKfWH1Bb5xTLXYF5xwY8F8VumLgqssdSG0YjpO","recipient":"MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAEb4ZjAt85ZODTgqSvTDR/USW78eu/dw26AIsVjz/FpP3d9O5+iYriNTrHJc/y+/zf"},{"prevCmHash":"MA\u003d\u003d","signature":"MDQCGHl8JOKqw1nHN3Shwy4HgX0voyAR8h8tQAIYF2VDUj6kiMpzj3CtC+NMgYHUlCRx3m1E","hash":"rHIaK0ZbmkH2SQ1+6Kv5hgMEis+X6HwtLxin/hj0K6g\u003d","data":"{\"detectionDate\":\"2019-05-22 15:53:54:327\",\"attackNumber\":546,\"totalDataNumber\":1200}","sender":"MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAEDO+Qsl5H3N3M5cJUmdWkF9rlK8qKfWH1Bb5xTLXYF5xwY8F8VumLgqssdSG0YjpO","recipient":"MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAEb4ZjAt85ZODTgqSvTDR/USW78eu/dw26AIsVjz/FpP3d9O5+iYriNTrHJc/y+/zf"}]}

The agent’s name, agent public key, and agent private key are stored in each communication agent's Agent Account. The ECDSA cryptographic technique encrypts these keys to guarantee that only the authorized agent is able to create and access specific communication messages. Agent accounts are created during the initialization of agents in the system. Only the procedures specified in the smart contract may be used to process communication messages that communication agents need to transmit, including commands and feedback. Only after confirming the data's integrity with the sender public key, the data, and the signature can recipients examine the message. Additionally, communication agents' ability to send and receive messages is determined by smart contracts. Every communication agent has data, data, sender and recipient, hash, and previous Hash information.The sender's private key and unprocessed data are used to compute the signature. Every block has the hash of the previous block, its own hash value, the Merkle Root, the maximum block height, the creation time, and communications. Every communication message sent by the communication agents is detailed in the communications. The calculation of Merkle Root is done recursively. Each successive tree layer is computed based on the preceding tree layer until the tree root is determined. The first tree layer is determined using the hash of the communications in the block.

Target communication may be efficiently tracked, and its integrity confirmed with Merkle Root. This block's position is determined by the previous hash, and its identification value is determined by the hash produced by Sha256. The block will be added to the blockchain after the size of the Communication list reaches MAX_BLOCK_HEIGHT, after which Merkle Root and hash will be computed. After that, the modified blockchain will be broadcast to further modules and stored locally on the host.

Starting and halting simultaneously, the solution combines an embedded database and a blockchain node in the same agent. Cache will be expanded to include interactions, which will speed up data usage and storage in agents and decrease search times.

2.4.2. Detection and Analysis Module

Malicious traffic and patterns are identified and analysed by this module. The Java classes that we utilized to construct this module are described below.

Communication Agent

The detection and analysis module's communication agent is called the Communication agent. Figure 3 provides an explanation of this procedure. The Communication agent will instruct the detection agent to detect a new data package upon receipt of a communication from the data process module. The detection report will be sent to the Communication agent by the detection agent after the detection process is complete. The Communication agent will get the detection report packaged into a communication object. The Communication agent will send a command to the training agent in the event that it receives a communication object indicating that the training set has been changed. A training report will be forwarded to the Communication agent and the Communication agent once the training agent has completed training the model.

Detection Agent

The Detection agent will begin utilizing the DNN model to examine the new data package and identify assaults as soon as it gets instructions from the Communication agent. A detection report will be prepared and forwarded to the DACommunication agent once the process is complete.

Training Agent

The Training agent will begin training the DNN model using the revised training dataset as soon as

it receives instructions from the DACommunication agent. A training report will be prepared and forwarded to the DACommunication agent upon completion of the process.

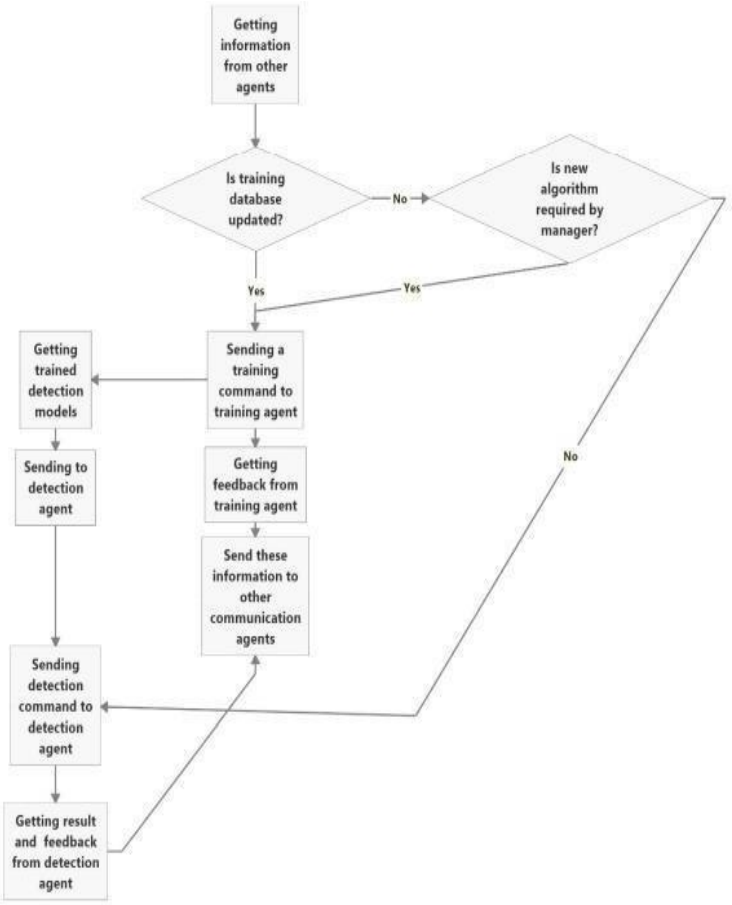


Figure 3 Process for detection and analysis.

DNNUtil

The methods that are pertinent to DNN model management are included in this class. Using normalization techniques, the new raw dataset will be standardized and normalized. The transform Process function and file allow for the examination and modification of the modified structure.

2.4.3. Response Module

RCommunication Agent

The response module's communication agent is the RCommunication agent. The RCommunication agent will instruct the response agent after receiving the Communication message with the training report or the detection report. The local host will store these reports.

Response Agent

Following an instruction from the RCommunication agent, the reaction agent will begin creating data visualization charts for the users based on the administrator's reaction plan. It will also alter host and firewall security in response to orders from the RCommunication agent. The following is the sequence of procedures for the intrusion detection system SESS (Figure 4):

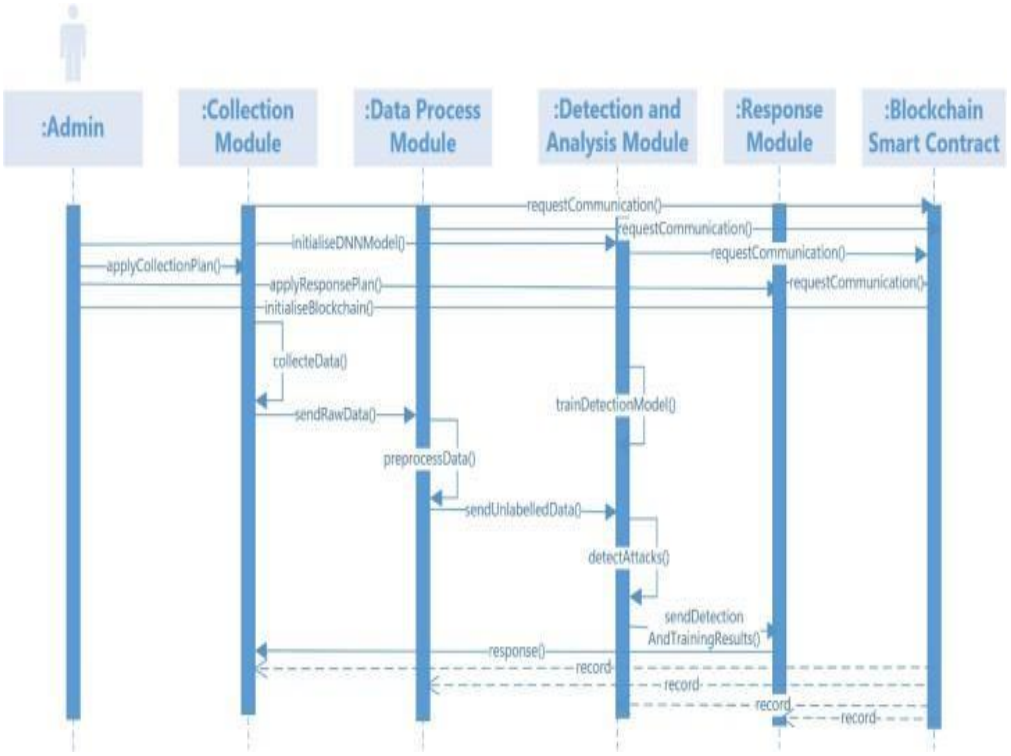


Figure 4 Smart, efficient, secure, and scalable (SESS) system sequence diagram.

Data Visualization

Using the BarChartUtil and LineChartUtil classes, data visualization is managed. Chart dimensions and themes are examples of parameters that can be changed.

2.4.4. Data Process Module

This module's (Figure 5) main goal is to process data for the host, network, and other agents.

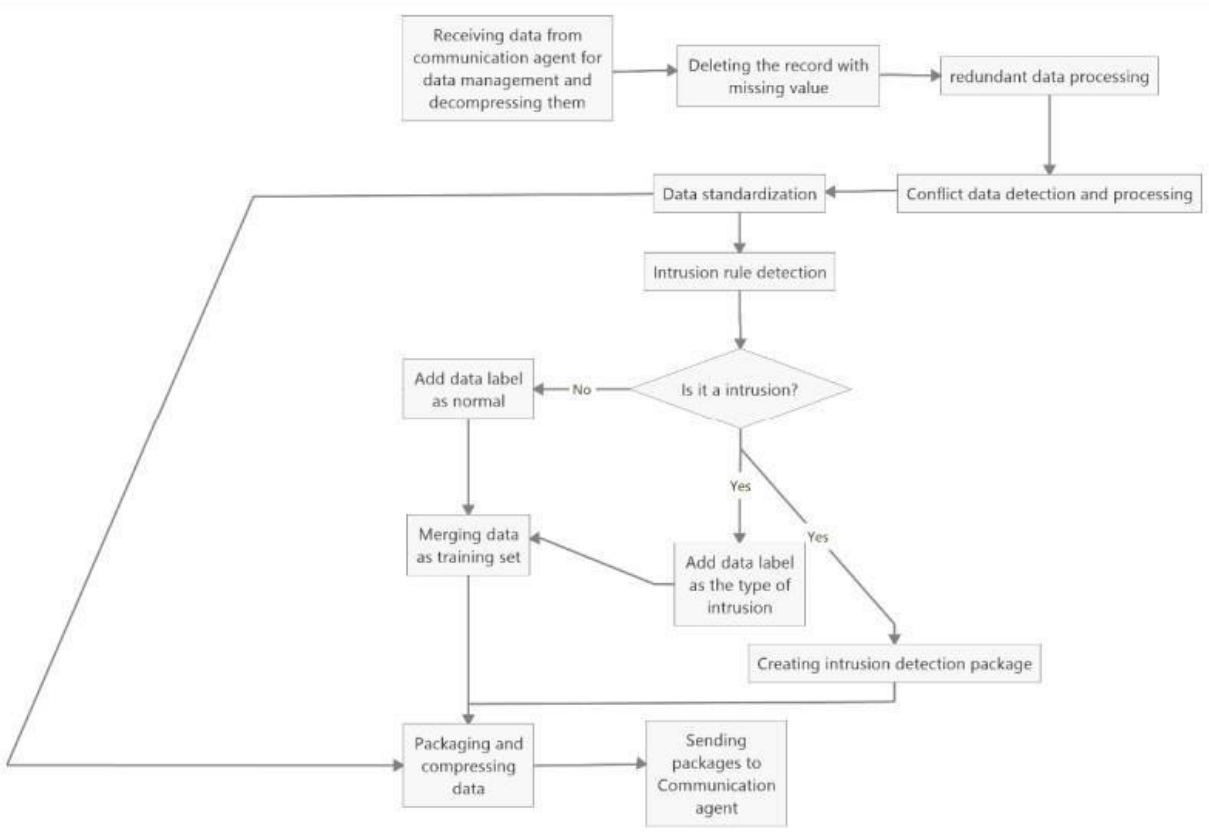


Figure 5 Process of data management agents.

Data from the network and the host are handled by the network data management agent and the host data management agent, respectively (Figure 5). Pre-processing of the data, such as missing value processing, data integration, and data standards, is required of both agents. The data will be scanned following the pre-processing phase in accordance with intrusion detection criteria created by other open-source IDS communities and/or system manager regulations. The pre-processed data will also be compressed and packed. The intrusion detection package keeps track of the identified intrusion behaviors and the associated processing steps. The intrusion detection package is forwarded to the response agent once this has been compared to the detection agent's result. To increase the detecting agent's accuracy, the labeled data are combined into a training set package.

The only agent that has the ability to change the database is the database agent (Figure 6). Only with a command from the communication agent can it alter the database. Based on the information supplied by the communication agent, the database agent modifies the database upon receiving the update instruction from the communication agent. Following database modifications, the database agent notifies the communication agent of any feedback.

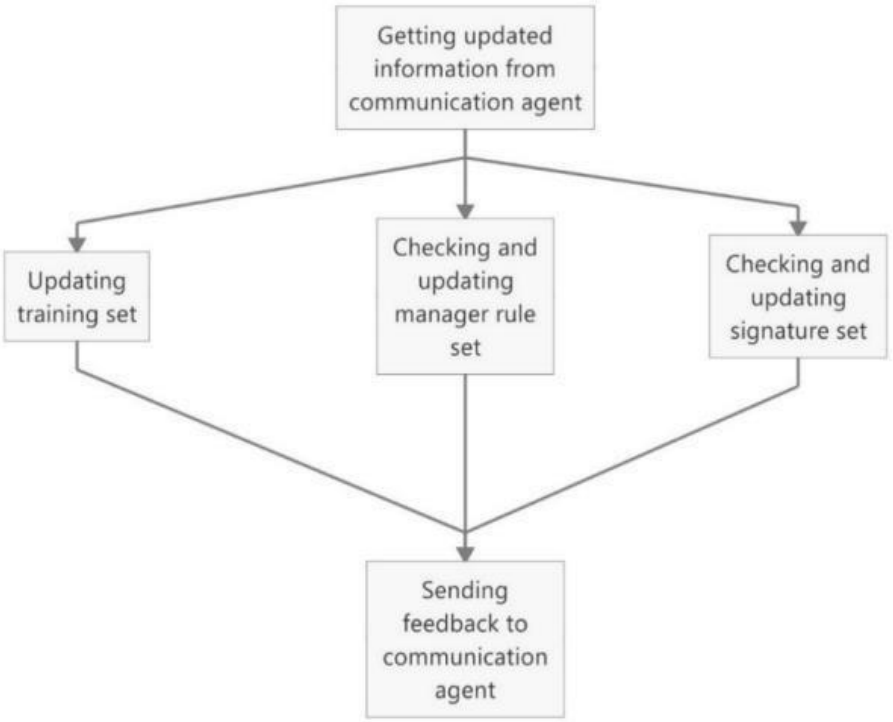


Figure 6 Process of database agent.

The host data process agent, the network data process agent, and the database agent are all under the management and control of the data process communication agent (Figure 7). It gives data process agent commands, receives responses from the agents, and transmits data packages from the collecting module to the agents. To ensure that the system is current, it requests that the database agent do routine updating checks. Additionally, this agent interacts with other communication agents and modifies its workflow in response to system conditions.

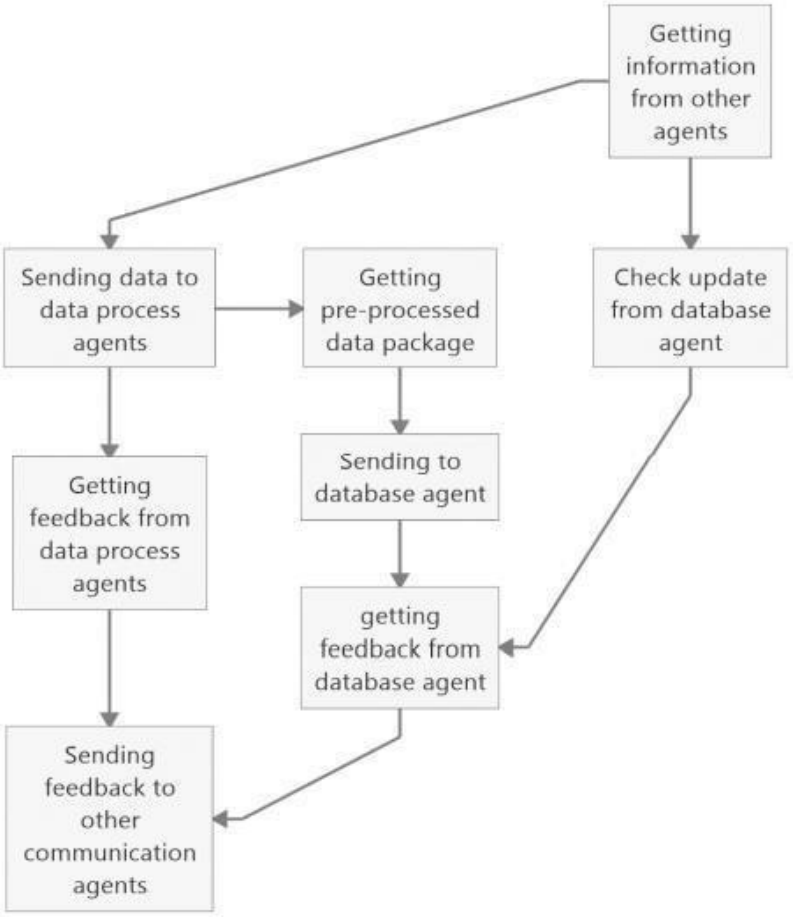


Figure 7 Communication agent for data process.

Chapter 4(Experimental Analysis)

3. Experimental Analysis

3.1. Dataset

The Defense Advanced Research Projects Agency (DARPA) Intrusion Detection Assessment initiative, a major project was undertaken by the MIT Lincoln Laboratories in the year 1998 has established to identify and assess the intrusion detection performance of system. This evaluation project's creation of a data set for modeling different threats is a significant accomplishment. But the dataset is too big, making it difficult to compare various intrusion detection techniques fairly. Furthermore, the recorded feature information is excessively complicated, utilizing disparate protocols and formats that are incompatible with the process of selecting feature qualities.Only network traffic data was included in the network security audit data set KDDCUP99, which was released by Professor Stolfo of Columbia University's Intrusion Detection Laboratory and colleagues in 1998 after it was examined and assembled from the IDS data set of MIT Lincoln Laboratory [51].

Some of the KDD99 data sets' intrinsic issues are resolved by the NSL-KDD dataset. Many intrusion detection systems are developed using the NSL-KDD data set. Despite its ongoing shortcomings, this data collection can serve as a useful baseline for academics comparing various intrusion detection techniques. The NSL-KDD training and test sets have appropriate sizes, and the evaluations conducted by various researchers will yield consistent and comparable findings. Three distinct protocols provide the data in the NSL-KDD data set: Internet Control Message Protocol (ICMP), User Datagram Protocol (UDP), and Transmission Control Protocol (TCP). There are 39 distinct attack types and four (4) attack classes (DoS, Probe, R2L, and U2R) in the NSL-KDD data set [52]. The NSL-KDD data collection is the source of data used in this study's intrusion detection simulation.

The protocols and attacks included in this dataset are useful for IoT environment intrusion detection research, even though the data from NSL-KDD is not specifically for IoT environment testing. The attack types in this dataset are also real-world occurrences. TCP and other IoT networks and protocols are becoming increasingly significant since cloud platform connections and traditional application layer connections are crucial components of IoT environments. For this reason, the NSL-KDD dataset

is utilized as an IoT dataset. The NSL-KDD dataset's samples are classified as anomalies or normals. 30% of the "Train + 20%" dataset is utilized as a validation dataset, and the remaining 70% of the subset is used as a training dataset. As a test, 1200 data samples from the "Test+ dataset" subset are employed as dataset (Tables 1 and 2).

Table 1: Train Attack Types + 20%

Major Attack Types	**Types Attack Types**
DOS	Reverse, Terra, Neptune, Pod, smurf, dropped tear
Probe	Nmap, portsweep, Satan, and IPsweep
R2L	imap, spy, guess_passwd, multihop, ftp_write, phf, warezmaster, and warezclient
U2R	load module, buffer overrun, and rootkit

Table 2: Forms of attacks on the Test+ subset of data.

Major Attack Types	**Types Attack Types**
DOS	Back, Land, Neptune, Pod, mailbomb, processtable, udpstorm, apache2, smurf, and teardrop
Probe	mscan, saint, Satan, IPsweep, Nmap, and portsweep
R2L	Sendmail, httptunnel, xsnoop, snmpguess, snmpgetattack, Warezmaster, Guess_passwd, named
U2R	rootkit, buffer_overflow

3.2. Evaluation Measures

Accuracy, precision, recall, and F1 score will be used to gauge intrusion detection ability. The number of samples in the prediction pair divided by the total number of samples is the accuracy. Prediction results provide the basis of the precision [53]. It shows the proportion of genuine positive samples among the positive prediction samples. The recall for the original sample shows the proportion of accurately anticipated positive cases in the sample. The precision rate and recall rate of the model are calculated and taken into account in full by the F1 score. The higher the F1-score, the higher the model quality. It is also necessary to take overfitting into account.

3.3. Pre-Processing.

One-hot coding, z-score measurements, and the same schema are used to standardize all of the raw data for the training, validation, and test datasets. Rather than employing categorical variables based on individual datasets, one-hot encoding employs the categorical variables from all training, validation, and test datasets for this experimental design. To avoid the dummy variable trap, one category variable per parameter that requires one-hot encoding is eliminated.

The output label is the final field, "class." One hot encoding must be used to convert multiple-valued variables, such as protocol_type, service, and flag, to numerical values. Protocol_type, service, and flag all had one category value eliminated in order to prevent the dummy variable trap.Additionally, the "class" field is encoded with numerical values.

3.4. Findings and Discussions

The detection and analysis module's performance has been determined by comparing its results in various scenarios (Tables 3–5).

Table 3: Accuracy of intrusion detection throughout varying epochs

Batch Size	Batch Number	Epoch	Accuracy on Validation Set	Accuracy on Test Set
32	300	50	98.46%	80.33%
32	300	100	98.51%	82.92%
32	300	150	98.47%	81.92%
32	300	200	98.52%	82.42%
32	300	250	98.44%	82.25%
32	300	300	98.45%	82.67%

Table 4: Accuracy of intrusion detection across various batch numbers.

Batch Size	Batch Number	Epoch	Accuracy on Validation Set	Accuracy on Test Set
32	50	100	97.36%	77.58%
32	100	100	97.83%	78.83%
32	150	100	98.07%	80.25%
32	200	100	98.29%	81.00%
32	250	100	98.34%	80.17%
32	300	100	98.51%	82.92%
32	350	100	98.44%	80.67%
32	400	100	98.47%	80.92%

Table 5: Accuracy rate of intrusion detection for varying batch sizes

Batch Size	Batch Number	Epoch	Accuracy on Validation Set	Accuracy on Test Set
16	300	100	98.09%	81.25%
32	300	100	98.51%	82.92%
48	300	100	98.27%	83.17%
64	300	100	98.54%	80.42%
80	300	100	98.67%	79.83%

It is demonstrated that the DNN model performs better for the batch sizes of 48, 300, and 100 epochs than for the other values examined by evaluating the detection and training module with these batch sizes, numbers, and epochs. While the accuracy of the test set is 83.17%, the accuracy rate using the validation set is 98.27%. The test set's precision is 83.53%. The test set's recall is 84.14%, and its F1 Score is 83.94%. These indicate that the system's DNN model can effectively differentiate between legitimate data flow and assaults. These tests also show that, even in cases when the training set contains fewer samples, taking an adequate number of epochs can guarantee that the model performs well. The performance of intrusion based on various datasets is displayed in Table 6. They are shown as mentioned: F1 score on validation set as FV, accuracy on test set as AT, precision on test set as PT,

recall on test set as RT, and F1 score on test set as FT. Accuracy on validation set as AV, precision on validation set as PV, recall on validation set as RV, and F1 score on test set as FV. The accuracy of detection under various data-splitting settings is summarized in Table 7. It's possible to modify the model to handle more complicated datasets.

Table 6: Intrusion detection performance with various training sets.

Training set	AV	PV	RV	FV	AT	PT	RT	FT
Train+	98.27%	98.27%	98.24%	98.12%	83.17%	83.53%	84.14%	83.94%
Test+	97.48%	97.69%	97.22%	97.81%	91.50%	91.53%	91.77%	91.21%

Table 7: Accuracy rate of intrusion detection considering various circumstances for data splitting

Data Splitting Conditions	Accuracy on Validation Set AV%	Accuracy on Test Set AT%
• 70% of Train + 20% as training set • 30% of Train + 20% as validation set • Subset of Test+ dataset containing 1200 data samples as test set	98.27	83.17
• 70% of Test+ as training set • 30% of Test+ as validation set • Subset of Train + Test dataset containing 1200 samples.	97.48	91.50

A brief summary of the parameters of the data derived from NSL-KDD is provided in Table 8 below, along with the data's serial number. 41 parameters were used in all, as listed below.

Table 8: Utilized NSL-KDD settings [35]

SL#	Attribute Name	Description
1	Duration	Length of time duration of the connection
2	Protocol_type	Protocol used in the connection
3	Service	Destination network service used
4	Flag	Status of the connection—Normal or Error
5	Src_bytes	Number of data bytes transferred from source to destination in single connection
6	Dst_bytes	Number of data bytes transferred from destination to source in single connection
7	Land	If source and destination IP addresses and port numbers are equal then, this variable takes value 1 else 0
8	Wrong_fragment	Total number of wrong fragments in this connection
9	Urgent	Number of urgent packets in this connection. Urgent packets are packets with the urgent bit activated
10	Hot	Number of 'hot' indicators in the content such as: entering a system directory, creating programs and executing programs
11	Num_failed_logins	Count of failed login attempts
12	Logged_in	Login Status: 1 if successfully logged in; 0 otherwise
13	Num_comp romised	Number of 'compromised' conditions'
14	Root_shell	1 if root shell is obtained; 0 otherwise
15	Su_attempted	1 if 'su root' command attempted or used; 0 otherwise
16	Num_root	Number of 'root' accesses or number of operations performed as a root in the connection
17	Num_file_creations	Number of file creation operations in the connection
18	Num_shells	Number of shell prompts
19	Num_access_files	Number of operations on access control files
20	Num_outbo und_cmds	Number of outbound commands in an FTP session
21	Is_hot_login	1 if the login belongs to the 'hot' list i.e., root or admin; else 0
22	Is_guest_login	1 if the login is a 'guest' login; 0 otherwise
23	Count	Number of connections to the same destination host as the current connection in the past two seconds
24	Srv_count	Number of connections to the same service (port number) as the current connection in the past two seconds
25	Serror_rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count (23)
26	Srv_serror_rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in srv_count (24)

Table 8: Continued.

SL#	Attribute Name	Description
27	Rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in count (23)
28	Srv_rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in srv_count (24)
29	Same_srv_rate	The percentage of connections that were to the same service, among the connections aggregated in count (23)
30	Diff_srv_rate	The percentage of connections that were to different services, among the connections aggregated in count (23)
31	Srv_diff_host_rate	The percentage of connections that were to different destination machines among the connections aggregated in srv_count (24)
32	Dst_host_count	Number of connections having the same destination host IP address
33	Dst_host_srv_count	Number of connections having the same port number
34	Dst_host_same_srv_rate	The percentage of connections that were to the same service, among the connections aggregated in dst_host_count (32)
35	Dst_host_diff_ srv_rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32)
36	Dst_host_same_src_port_rate	The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count (33)
37	Dst_host_srv_diff_host_rate	The percentage of connections that were to different destination machines, among the connections aggregated in dst_host_srv_count (33)
38	Dst_host_serror_rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_count (32)
39	Dst_host_srv_s error_rate	The percent of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_srv_count (33)
40	Dst_host_erro r_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_count (32)
41	Dst_host_srv_r error_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_srv_count (33)

The performance of the detection and analysis module on various parameter counts is displayed in Table 9. It is crucial to understand how varying parameter counts impact the system's performance because the SESS intrusion detection system is intended to be utilized on various IoT network scales and some parameters are challenging to gather on certain IoT networks.

Table 9: Performance of intrusion detection using various parameters

Number of Parameters	Parameters Included	AV	PV	RV	FV	AT	PT	RT	FT
41	[1–41]	98.27%	98.27%	98.24%	98.12%	83.17%	83.53%	84.14%	83.94%
40	[1–2], [4–41]	98.85%	98.84%	98.85%	98.76%	82.33%	84.04%	84.00%	82.30%
25	[1–2], [4–6], [12], [23–41]	97.82%	97.84%	97.77%	97.63%	80.75%	82.56%	82.45%	80.64%
17	[1–2], [4–6], [12], [23–24], [29], [32–39]	97.88%	97.92%	97.81%	97.69%	81.33%	83.13%	83.03%	81.24%
7	[1–2], [4–6], [23–24]	95.10%	95.21%	94.96%	94.62%	78.08%	78.19%	78.75%	79.37%
6	[1–2], [5–6], [23–24]	93.53%	93.87%	93.25%	92.75%	72.67%	79.45%	68.74%	80.15%
5	[2], [5–6], [23–24]	92.67%	93.16%	92.32%	91.71%	72.33%	79.23%	68.35%	79.95%
4	[2], [5–6], [23]	90.40%	91.45%	89.84%	88.80%	73.92%	75.68%	71.31%	79.71%
3	[2], [5–6]	58.15%	69.92%	54.95%	20.87%	71.75%	78.83%	67.67%	79.62%
2	[5–6]	84.27%	86.25%	85.16%	85.12%	68.50%	71.95%	64.68%	76.89%
1	[6]	53.72%	53.72%	50.00%	0.00%	43.00%	43.00%	50.00%	0.00%

Table 9 above utilizes an epoch of 100, 300 as the batch number, and 48 as the batch size. The NSL-KDD dataset has forty-one parameters in all.

Because different IoT networks have different types of services, which have a significant impact on the normalization process, the first parameter to be deleted is service. The DNN model has an accuracy of 82.33% on the test set when the service parameter is removed, which is 0.84% less than when all 41 parameters are used. The DNN model's accuracy on the test set is 80.75% when the data sets use 25 parameters, and it rises to 81.33 when we use just 17 parameters.

Because some parameters must be paired with other parameters in order to detect attacks, the accuracy for 17 parameters is higher than the accuracy for 25 parameters. Some parameters lose their combination parameters when the number of parameters is decreased from 40 to 25, which negatively affects the weighting procedure. The accuracy rises when these combination characteristics are eliminated.

The DNN model has an accuracy rate of 78.08% when only seven parameters are employed, namely duration, protocol_type, flag, src_bytes, dst_bytes, count, and srv_count. Thus, even if the collection module is unable to obtain a large amount of information from the IoT network, the DNN model still has a respectable level of detection capability. With the following parameters only: duration, protocol_type, src_bytes, dst_bytes, count, and srv_count. The DNN model performs with 72.68% accuracy and 68.74% recall on the test set. In this instance, the accuracy rate has decreased due to the parameter flag's removal. This demonstrates how crucial the flag parameter is. The flag argument, however, further muddies the normalization process. The flag parameter should be eliminated in order to examine the system's performance on an IoT network that offers limited details, as it can be challenging to record in specific situations.With just five parameters included in the data sets, the DNN model's accuracy on the test set is 72.33%. The DNN model has 73.92% accuracy on the test set if the data sets include protocol_type, src_bytes, dst_bytes, and count as parameters. This indicates that a greater performance is obtained using four parameters rather than five or six.

The DNN model has a 71.75% accuracy on the test set but only a 58.15% accuracy on the validation set if the data sets have three parameters. This is due to Dropout's use in the DNN model. During the training phase, certain weak classifiers will be randomly removed using this regularization strategy. In this case, applying dropout to the validation set will have an impact on accuracy. This issue can be fixed by increasing the batch size or doing away with the dropout technique, albeit doing so can cause an overfitting issue as well. The DNN model has an accuracy of 89.23% on the validation set and 70% on the test set if dropout is eliminated. The DNN model exhibits 68.5% accuracy on the test set if the data sets only employ the parameters src_bytes and dst_bytes.

The accuracy rate on the test dataset is 43% when dst_bytes is the only parameter used, yet the F1 Score is 0% for both the validation and test datasets. It indicates an overfitting issue when every sample in the data sets is categorized into a single class. This is a result of some hyper-parameters being too high for one parameter datasets, such as the number of epochs.

This model performs well on the TCP and ICMP datasets, as demonstrated in Figure 8 (accuracy rate is 99.1% and 98.1%, respectively). On the other hand, the UDP protocol's accuracy rate is only 93.7%. One explanation for this is that there aren't many UDP protocols in the data set. Another factor is the peculiarities of the UDP protocol. DNN does well (accuracy rate of 97%) on a dataset where the values are classes for various attack kinds. Nonetheless, some assault types that are present in only a small percentage of the data are difficult to differentiate.

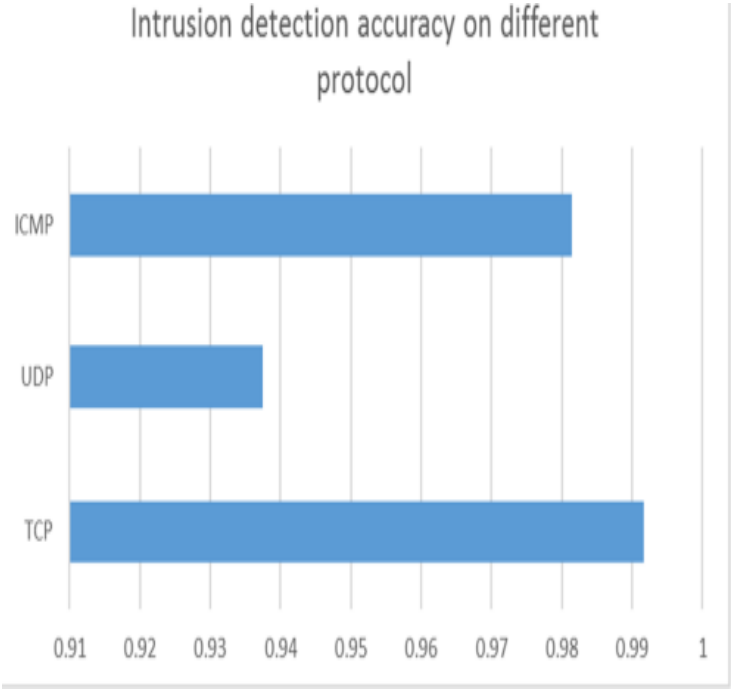


Figure 8: Precision across several methods

To summarize up, these tests demonstrate the SESS system's usefulness across a range of IoT network sizes. With a detection time of about 0.18 s during the experiments, the system can identify attacks and take action against them instantly. The training time, the detection time, and the stages of initializing and loading the data model were all taken into account while recording the timing. This timing could change depending on a number of factors, such as the server's hardware, other resources being used, memory requirements, etc. To ensure that the suggested model can function in a fast-paced living context, more study and research are actually required for these metrics. The multi-agent reinforcement learning process will also use the results of these experiments to establish statuses.

3.5. Limitations

The outcomes of these tests demonstrate that this method is applicable to various IoT network sizes, supporting even more intricate networks. However, additional study in this field is still required. Even though the DNN model has a fairly good accuracy rate in identifying various attack kinds, it is unable to effectively recognize certain rare assault types. Future work will need to address the long learning curve of agents and the requirement for huge training sets. More research is needed to understand how several agents might work together effectively in a big IoT network. Future solutions must also address

Chapter 5(Conclusion)

4. Conclusions

In this paper, a deep learning, blockchain, and multi-agent system-based intrusion detection system was developed. We have provided a detailed description of how each model component functions as well as how the entire system operates. Because multi-agent systems are flexible, this novel IDS can be used in a range of IoT scenarios with different scales. Since every activity taken by communication agents will be documented on blockchain, the system is protected against dangers such as information manipulation and disclosure. By using a multi-agent reinforcement algorithm, the system's performance can be continuously enhanced. Neural network application is researched. According to the simulation results, on the same kind of IoT network, the deep learning algorithm performs better than conventional techniques. Because blockchain technology secures ACL communication and smart contracts control agent communication, the deployment of blockchain technology guarantees that this system can be dispersed to various remote hosts. The NSL-KDD dataset studies show that DNN can detect intrusions with high accuracy on the transport layer of an IoT context. When it comes to differentiating anomalous from normal, the DNN model performs better than other machine learning techniques like decision trees. This illustrates how deep learning methods can be used for IoT device IDS. The tests show that the system performs well in a variety of settings, including networks with varying degrees of complexity and attack kinds.

Future Work

The article proposes an intrusion detection system model for the Internet of Things based on the research of multi-agent technology, block chains, and neural networks. The next stage is to execute the system in an actual environment and continuously develop the modules so that every agent can collaborate effectively with every other agent. The next step is to gather IoT network data sets so that the detection model may be trained, which will enhance system performance. Further research into the creation of datasets featuring innovative assault kinds is also a possibility. More dependable and quick multi-agent algorithms might be employed to train the communication agents and streamline the feedback training procedure, which would enhance system performance. It may be possible to investigate more complex deep learning methods to enhance system performance. We have tested the block chain on a smaller version, therefore in the next iteration, we may use multi agent reinforcement

blockchain. Implementing an unspent communication output pool is a good idea for larger-scale IoT networks in order to control communication order. Another option is to test against UNSW-BN15 datasets. An IoT data simulator [54] or any other tools of a similar nature could be utilized to produce data in order to further assess the correctness of the IDS in future IoT system testing.

REFERENCES

[1] Adat, V.; Gupta, B. Security in Internet of Things: Issues, challenges, taxonomy, and architecture. Model. Anal. Des. Manag. 2018, 67, 423–441. [CrossRef]

2. Statista Research Department. IoT: Number of connected devices worldwide 2012–2025. Available online: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ (accessed on 20 May 2020).

3. Tahaei, H.; Afifi, F.; Asemi, A.; Zaki, F.; Anuar, N.B. The rise of traffic classification in IoT networks: A survey. J. Netw. Comput. Appl. 2020, 154. [CrossRef]

4. Hajiheidari, S.; Wakil, K.; Badri, M.; Navimipour, N.J. Intrusion detection systems in the Internet of things: A comprehensive investigation. Comput. Netw. 2019, 160, 165–191. [CrossRef]

5. Samaila, M.; Neto, M.; Fernandes, D.; Freire, M.; Inácio, P. Challenges of securing Internet of Things devices: A survey. Secur. Priv. 2018, 1. [CrossRef]

6. Bhattarai, S.; Wang, Y. End-to-End Trust and Security for Internet of Things Applications. Computer 2018, 51, 20–27. [CrossRef].

7. Spafford, E.; James, P. Anderson: An Information Security Pioneer. IEEE Secur. Priv. 2008, 6, 9. [CrossRef]

8. Alnaghes, M.S.; Gebali, F. A Survey on Some Currently Existing Intrusion Detection Systems for Mobile Ad Hoc Networks. In Proceedings of the Second International Conference on Electrical and Electronics Engineering, Clean Energy and Green Computing (EEECEGC2015), Antalya, Turkey, 26–28 May 2015; Volume 12.

9. Hari, P.B.; Singh, S.N. Security Attacks at MAC and Network Layer in Wireless Sensor Networks. J. Adv. Res. Dyn. Control Syst. 2019, 11, 82–89. [CrossRef]

10. Pacheco, J.; Hariri, S. IoT Security Framework for Smart Cyber Infrastructures. In Proceedings of the IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W), Augsburg, Germany, 12–16 September 2016; pp. 242–247. [CrossRef]

11. Liu, C.; Yang, J.; Chen, R.; Zhang, Y.; Zeng, J. Research on immunity-based intrusion detection technology for the Internet of Things. In Proceedings of the 2011 Seventh International Conference

on Natural, Shanghai, China, 26–28 July 2011.
12. Roman, R.; Zhou, J.; Lopez, J. On the features and challenges of security and privacy in distributed internet of things. Comput. Netw. 2013, 57, 2266–2279. [CrossRef]
13. Wallgren, L.; Raza, S.; Voigt, T. Routing Attacks and Countermeasures in the RPL-Based Internet of Things. Int. J. Distrib. Sens. Netw. 2013, 9, 794326.
14. Raza, S.; Wallgren, L.; Voigt, T. SVELTE: Real-time intrusion detection in the Internet of Things. Ad Hoc Netw. 2013, 11, 2661–2674. [CrossRef]
15. Zegzhda, P.; Kort, S. Host-Based Intrusion Detection System: Model and Design Features. In Proceedings of the International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security, St. Petersburg, Russia, 13–15 September 2007; pp. 340–345.
16. Chakravarthi, S.S.; Veluru, S. A Review on Intrusion Detection Techniques and Intrusion Detection Systems in MANETs. In Proceedings of the International Conference on Computational Intelligence and Communication Networks, Bhopal, India, 14–16 November 2014.
17. Santos, L.; Rabadao, C.; Goncalves, R. Intrusion detection systems in Internet of Things: A literature review. In Proceedings of the 13th Iberian Conference on Information Systems and Technologies (Cisti), Caceres, Spain, 13–16 June 2018.
18. Mohamed, A.B.; Idris, N.B.; Shanmugum, B. A Brief Introduction to Intrusion Detection System. In Trends in Intelligent Robotics, Automation, and Manufacturing, Proceedings of the IRAM 2012, Kuala Lumpur, Malaysia, 28–30 November 2012; Communications in Computer and Information Science; Ponnambalam, S.G., Parkkinen, J., Ramanathan, K.C., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 330.
19. Zarpelão, B.B.; Miani, R.S.; Kawakani, C.T.; Alvarenga, S.C. A survey of intrusion detection in Internet of Things. J. Netw. Comput. Appl. 2017, 84, 25–37. [CrossRef]

202-35-650

ORIGINALITY REPORT

14%	9%	9%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	rke.abertay.ac.uk Internet Source	3%
2	Submitted to Middlesex University Student Paper	2%
3	Submitted to University of Northumbria at Newcastle Student Paper	2%
4	Submitted to Curtin University of Technology Student Paper	1%
5	Submitted to islamicuniversity Student Paper	1%
6	Submitted to University of North Texas Student Paper	1%
7	Submitted to University of Derby Student Paper	1%
8	www.techscience.com Internet Source	<1%
9	Al-Sakib Khan Pathan. "The State of the Art in Intrusion Prevention and Detection",	<1%