electronics

MDPI

Article

Intrusion Detection System for the Internet of Things Based on Blockchain and Multi-Agent Systems

Chao Liang [1], Bharanidharan Shanmugam [1,*], Sami Azam [1], Asif Karim [1], Ashraful Islam [2], Mazdak Zamani [3], Sanaz Kavianpour [4] and Norbik Bashah Idris [5]

1 College of Engineering, IT and Environment, Charles Darwin University, Casuarina, NT 0810, Australia; liangchaotech@gmail.com (C.L.); sami.azam@cdu.edu.au (S.A.); asif.karim@cdu.edu.au (A.K.)
2 School of Computing and Informatics, University of Louisiana at Lafayette, Louisiana, LA 70504, USA; ashraful.islam1@louisiana.edu
3 School of Arts and Science, Felician University, Rutherford, NJ 07070, USA; ZamaniM@felician.edu
4 School of Design and Informatics, Abertay University, Dundee DD1 1HG, UK; s.kavianpour@abertay.ac.uk
5 Kulliyyah of Information and Communication Technology, International Islamic University Malaysia, Selangor 53100, Malaysia; norbik@iium.edu.my
* Correspondence: bharanidharan.shanmugam@cdu.edu.au

Received: 6 May 2020; Accepted: 19 June 2020; Published: 10 July 2020

Abstract: With the popularity of Internet of Things (IoT) technology, the security of the IoT network has become an important issue. Traditional intrusion detection systems have their limitations when applied to the IoT network due to resource constraints and the complexity. This research focusses on the design, implementation and testing of an intrusion detection system which uses a hybrid placement strategy based on a multi-agent system, blockchain and deep learning algorithms. The system consists of the following modules: data collection, data management, analysis, and response. The National security lab–knowledge discovery and data mining NSL-KDD dataset is used to test the system. The results demonstrate the efficiency of deep learning algorithms when detecting attacks from the transport layer. The experiment indicates that deep learning algorithms are suitable for intrusion detection in IoT network environment.

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

1. Introduction

The Internet of Things (IoT) can be considered as an interconnected system based on approved protocols which exchange information [1] among the devices operating via the internet. Recent advancements in IoT introduces the concept of smartness to devices, sensors, homes, streets and even cities. IoT is one of the leading and growing fields of modern computing and communication technology and has made major contributions to various domains, from agricultural sector to vehicles automation. Nowadays, IoT is also referred [2] as the Internet of Everything (IoE) as it deals with any type of connected device in daily life. It is anticipated that in 2025 [2] the number of connected devices may reach up to 21.5 billon.

IoT is a combination [3] of several layers including a network layer. The design of the network layer is based on the traditional layers of Internet communication and is mainly responsible for transferring data packets between hosts. Moreover, the network layer is complex and a vulnerable portion in IoT architecture leading to various security issues. Nevertheless, several security frameworks are in place to address the security issues [4]. These frameworks require installation in the IoT architecture and/or the devices in order the devices to operate effectively resolve security threats. Unfortunately, most security frameworks require considerable computational power, as well as storage [5]. However, some approaches such as light weighted encryption, authentication mechanisms can be employed to overcome the constraints [6].

The large number of nodes, i.e., hosts or devices connected to the IoT is a key reason for the security issues as a security breach occurring in a single node may lead to failing of the whole system. The most common security threats faced by IoT system are botnets, distributed denial of service (DDoS) attacks, ransomware, remote recording, routing attacks and data leakage [1]. To combat the attacks on IoT devices, utilizing a firewall is considered as a first line of defense but it is not an effective solution due to the variability and complexity of IoT architectures.

In recent years, intrusion detection systems (IDSs) have gained popularity due to their robustness. James [7] first introduced the concept of an IDS in 1980 and proposed a definition. The idea of IDSs is to discover intruders in an area. In an IoT environment, an intruder can be a host that is trying to access some other nodes without permission. An IDS is comprised of three main attributes: an agent, an analysis engine and a response module. The agent is solely responsible for compiling information from the data stream by the monitoring events. The analysis engine traces the signs of intrusion and generates alerts. Finally, the response module works on the outcomes it receives from the analysis engine. Over the decades, IDSs have become more reliable and efficient but attackers have also developed more diversified attack modes to defeat these detection systems. In addition, traditional IDSs cannot cope with the complex network layers in IoT [8]. The latest developments in intelligent systems has encouraged researchers to employ distributed IDSs in combination with various machine learning mechanisms, e.g., artificial neural network (ANN), deep learning and reinforcement learning. Ordinary ANNs have some limitations in dealing with the complexity of IDSs. Improving technology by resolving these shortcomings is a requirement for realizing the promise of IDSs promising in the real world. The main contribution of this work is to apply blockchain for a multi-agent system and test it with a well-known dataset for its effectiveness.

Purpose of the Study

Evidence shows that most of the attacks are generated from the network layer or the transport layer [9,10]. The primary aim of this research is to explore the opportunity to apply the concept of blockchain for a multi agent system (MAS).

The aim of this study is to develop a solution, building an intelligent IDS that can detect with the intruders and prevent attacks in IoT environments. To achieve this goal, firstly, the current state-of machine learning techniques-based IDS models is reviewed. To identify potential security and privacy issues related to IoT systems attacks which have already occurred are examined. In addition, the shortcomings of existing IoT devices are evaluated. Finally, a solution is proposed, utilizing the strength of machine learning mechanisms to combat sudden attacks from unauthorized intruders, on the network layer and transport layer. To improve the performance of IDS, various optimization techniques to improve the performance of IDS are explored.

The rest of the paper is organized as follows: Section 2 provides a brief introduction to IDS and its related technologies. Section 3 introduces the IDS tested in detail with the modules explained. A discussion about experimental results can be found in Section 4. Finally, the conclusions along with future work are presented in Section 5.

2. Literature Review

A thorough literature review has been conducted to investigate current research in the area of IDS performance for IoT devices. A detailed analysis of the possible threats in IoT space and their countermeasures using IDS was carried out. IoT system communication protocols were also taken in consideration.

2.1. History of Intrustion Detection System

Generally, IDS includes both software and hardware mechanisms and IDS is responsible for identifying malicious activities by monitoring network environments and systems. In other words, IDS is used for detecting cyber-attacks and providing immediate alerts. Overall, IDS acts like a safeguard to the networks and systems. IDS is normally deployed after the firewall and is used with an intrusion prevention system.

IDS is not a new term in the fields of IoT research regarding security and privacy. A significant number of publications have appeared in recent years. Cyber security experts have been concerned about the security and privacy of IoT environments for some time. This has led to the introduction of the concept of IDS embedding into IoT architectures and devices to deal with cyber-attacks [11–13]. Researchers are mostly interested in inventing new mechanisms and models to counter intruders in conventional network protocols. However, traditional IDS mechanisms are incompatible with IoT devices connected through IPv6 and other complex network structures [14]. More comprehensive research on the use of machine learning methods is essential for IDS to secure and protect privacy in IoT.

2.1.1. Classes of IDS

IDS is classified in two main categories as follows:

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

Host-based IDS detects intrusion behavior by scanning log and audit records. This kind of IDS is usually used on important hosts to protect the host security from all directions. The advantage of the host-based IDS is that it provides more detailed information, lower false alarm rates, and has less complexity than network-based IDS. However, it reduces the efficiency of the application system and relies excessively on the log data and monitoring capability of the host [15].

Because of the characteristics of the IoT and because many IoT devices can be connected to the network, network-based IDSs need attention. Network-based IDS can detect the abnormal behavior and data flow in the network, to find potential intrusions. It does not change the host configuration and does not affect the performance of the business system. Even if the network IDS fail, it will not affect normal business operation. A problem is that network-based IDS only checks its direct connection to the network segment without looking at other network segments. It is also difficult to process encrypted sessions with network-based IDS. [16]

2.1.2. Detection Methods

Based on the nature of various intrusion attacks (Figure 1), intrusion detection methods are classified into four major categories: signature-based methods, specification-based methods, anomaly-based methods and hybrid methods [16].

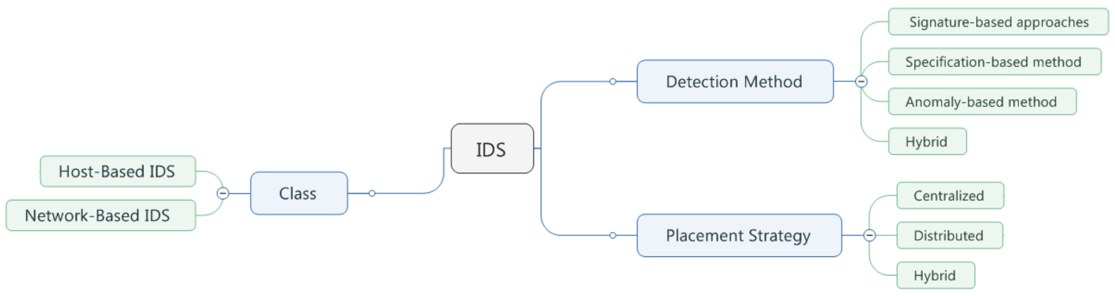

Figure 1. Types and placement of intrusion detection systems (IDS).

Signature-based methods first scan the data in the network and compare it with a feature database. If the scanned data is found to match the features in the signature database, the data will be treated as an intrusion. The advantage is that it can accurately determine the type of attack. It is relatively convenient to use, and the demand for resources is comparatively small.

Specification-based methods require the system administrators to set rules and thresholds in advance. IDS detect the status of the current system and network according to the rules and thresholds set by administrators [17]. If the threshold is exceeded or the rules are violated, the IDS will detect an abnormal situation and act accordingly.

Anomaly-based methods depend on identifying abnormal patterns and by comparing traffic patterns. The advantage of using this method is that it enables the detection of new and unknown intrusions. However, the primary limitation is that the method tends to result in high false positive rates. Research is now focusing on applying machine learning algorithms in anomaly-based intrusion detection methods to improve the robustness of this kind of method. By employing machine learning algorithms, anomaly-based intrusion detection methods can monitor the ongoing intrusion footprints and compare them with existing datasets to be alert to potential future attacks.

Hybrid methods refer to the use of any combination of the above-mentioned detection methods in the same IDS. This approach can help to overcome the shortcomings of a single method thereby enhancing the reliability of the entire IoT system. However, the obvious drawback is that the entire IDS will become very large and complex. This will make the whole system more difficult to operate and will require more resources. Especially when there are many protocols involved in the IoT system, the intrusion detection process will have large resources and time demands.

2.1.3. Placement Strategy

Centralized IDS (Figure 1) consists of several monitoring nodes for understanding the behavior of the host or the network. The data (either alerts or system logs or network logs) will then be shared with the main server or node for further analysis. As the system becomes more complex, the central server load will increase, reducing system performance and potentially even causing security risks. In the IoT environment, most of the data interaction occurs in the network layer and the perceptual layer, and the distribution of the IoT facilities is complex and scattered. The centralized IDS can therefore not effectively detect intrusions of IoT devices [18].

Distributed IDS (DIDS) consists of several modules, each of which is responsible for different tasks. They usually share the monitoring tasks. For example, in the IoT environment, the threats faced by the network layer, the perception layer, and the application layer are not the same, so the corresponding intrusion detection modules need to be different as well. This can greatly increase the security of the system, without the need for excessive data storage and sharing among the modules. The primary advantage is high scalability, which means it can be adapted to future IoT environments. The disadvantage is that the resource consumption and communication costs can be high [18].

Some hybrid placement strategies have been presented in recent research. The first strategy for hybrid placement separates the network into different regions. There will be a node that has an intrusion detection function and monitors nodes in the same cluster. The responsibility of these nodes also is to monitor the data package coming from their neighbor nodes. If any anomalous behavior is found, it is concluded that the neighbor nodes were compromised by attackers. The border router collects information from the nodes and makes the final decision about the intrusion. It also should be noted that hybrid IDS with distributed components is more accurate than centralized IDS. However, the hybrid system has large resource demands [19].

2.2. Previous Research on IDS for IoT

Improving the IDS of IoT [17] is a major focus of this research. This section provides an overview of the improvements and the developments of IDS for IoT. These can be classified based on target threats, placement strategy and detection method.

An unsupervised and hybrid intrusion discovery framework to recognize selective-forwarding and sinkhole assaults for IoT was proposed by Bostani in 2017 [20]. It uses a half breed situation technique, pursuing an anomaly-based middle agent, separated with a few specification-based components which share their own location information to a centralized interruption location module in order to anticipate any inconsistencies. It applies an unsupervised optimum-path timberland (OPF) calculation with MapReduce engineering.

Yulong et al. proposed employing a modern interruption discovery technique for the IoT environment agreeing with an automation demonstration [21]. This strategy distinguishes three types of IoT assaults: false-attacks, jam-attacks and reply-attack. It is an expansion of named move frameworks. The arrangement of this IDS may be a centralized approach, since the information that is accumulated by arrange hubs is sent to the interruption discovery module and aids in constructing the occasion databases. The IDS utilize the occasion analyzer, based on a specification-based strategy, to identify interruptions. By comparing the pre-occupied activity streams, this IDS can proficiently identify jam-attacks, false-attacks, and reply-attacks in IoT networks.

Kapitnov et al. [22] proposed a method, also in 2017, that assembles multi-agent systems and blockchain technology in augmenting communication among various autonomous agents involved in unmanned aerial vehicles (UAV). They proposed an architectural protocol to incorporate the communication through an Ethereum blockchain technology in peer-to-peer connected networks. The result is a protocol that is potentially compatible with autonomous agents. The proposed protocol ensures security of the communication process and assists in anticipating the robustness of variable working states. This has led to further investigations in combining blockchain technologies with multi-agent systems.

Calvaresi et al. [23] stated that integration of blockchain technology (BCT) and MAS can achieve the distribution of the trust and can remove the necessity for trusted third parties (TTP) which are the main single points of failure. Calvaresi et al. in 2019, proposed implementing mechanisms for transparent reputation management via BCT, which addressed the challenge of enforcing trust in MAS. They designed and implemented a system integrating MAS, based on the Java agent development framework (JADE), and BTC, based on Hyperledger Fabric. By extending the management of the agent identity to the membership service of Hyperledger Fabric a trustworthy community is created by the system. Moreover, the association agent is offered services using the ledger. Mechanisms are implemented to transparently compute and store agents' reputations based on the communication patterns between the agents and the evaluations of the interactions. The system is evaluated with various test cases. Agent behaviors consists of autonomous and user-dependent behaviors. Smart contract mechanisms are combined with computing and monitoring the agents' reputation based on agent behaviors. Because of the available Hyperledger Fabric implementation, the system is robust and scalable. Testing can be done easily and fast because of a classic command-line interface (graphical interface eased the interactions). Although a prototype integrating BCT and MAS was implemented, technical limitations still exist. This includes prevention of single points of failure of the system by outlining a reasonable mapping between real entities and distributed components of the underlying blockchain technology, enforcing cryptographic solutions to enhance security and privacy, verifying the smart contract implementation precision, and approving the blockchain and agent technology in real-world systems. There are also ethical considerations for BCT and MAS integration in real-world scenarios. There is a direct relationship between the immutability property of the ledger and the transparency of reputation management and achieving an application area that enhances the system users' power and supports trustworthy interactions. Deliberate or accidentally malfunction of the system can compromise the users' privacy, and the framework relies completely on the smart contracts. Although the BCT enabled MAS would preclude intermediaries in conflict resolution, the reliability of the software and verification process in the BCT environment is an open question.

In 2018, Calvaresi et al. [24] reviewed multi-agent systems and blockchain technologies. Their review concluded that the challenge in using blockchain technology is similar to the previously described problems in multi-agent systems for different application domains. Various factors were discussed, such as motivations and requirements, mechanisms and application domains, strengths and limitations. The authors emphasized that a comprehensive evaluation of these systems is necessary, not just discussion of possible advantages. However, they also concluded that combining blockchain technology with multi-agent system-based platforms is serviceable and can be a robust mechanism for gathering information and data from a very large number of associated nodes in a network e.g., IoT. Multi-agent systems combined with blockchain technology could cope more effectively with the challenges of sophisticated interconnected environments.

In a multi-agent system, agents often may get stuck in a situation where reaching 'consensus' becomes impossible. Identifying conflicts of interest among agents and resolving the resulting stagnation is not a trivial task. The model proposed by [25] examines cooperative tendencies by considering the probabilities of cooperation for each agent in the environment. The collective payoff can be augmented by candidate policies (identifying the candidate action sets). The Nash bargaining solution (NBS) algorithm is then used to select the top candidate action sets. The solution to reduce these dilemmas is expected to decrease memory footprint and 'convergence time'. However, the current implementation is not yet suitable for high dimensional environments having extensive state space.

In 2018, Diro et al. proposed an IDS for IoT where deep learning is utilized for anomaly detection [26]. This was demonstrated to be effective in identifying IoT Fog organize attacks compared to conventional IDS.

Air traffic flow management (ATFM) systems plays a significant role in efficiently managing air traffic control. For high precision decision making, a high-fidelity and mathematically reliable model of ATFM is often required but this is quite difficult to design due to the complex nature of airspaces. To address the issue, Ta et al. [27] developed an intelligent, multi-agent and distributed ATFM solution, named BlockAgent which is based on blockchain and reinforcement learning. The system is comprised of three parts, the local layer, the blockchain layer and the global optimization layer. The local layer allows local stakeholders placed in regional bases. The blockchain layer provides distributed and decentralized coordination mechanisms to manage and coordinate a range of diverse information related to airports, aircrafts, varied and dynamic weather conditions, route optimization, etc., most often controlled from a central location in contrast to the usual multi-agent structure of ATFM systems. The global optimization layer: enables the execution of smart contracts to aggregate information that has been locally obtained. The system uses reinforcement learning to reduce the delays experienced by aircrafts both on air and before take-off. BlockAgent did outperformed traditional ATFM systems in handling delays when tested in a limited setup. The authors aim to enhance the system to make it viable for large-scale implementation.

Casado-Vara et al. [28] presented a hybrid IoT architecture that allows decentralized data management via blockchain. This architecture includes a computation distribution under the edge computing paradigm, which enables optimizing the connection between IoT and blockchain. Another part of this hybrid architecture is data management. This consist of a big data ecosystem that simplifies the management of large amounts of data in the blockchain. A new algorithm based on game theory is proposed and applied to the data collected by IoT devices to enhance data quality and false data detection. The existing end-to-end architectures are optimized by the proposed hybrid architecture.

In 2019, Li et al. proposed a way for IoT data feature extraction and a new IDS for smart cities based on deep migration learning [29]. By using deep migration learning, the proposed IDS can overcome the lack of a suitable training sets and resolve sample misclassifications and spatial constraints of data clustering problems, optimizing the intrusion detection model and improving efficiency. Although the experiments show that this proposed system has a better performance than conventional methods and that it reduces the clustering time effectively, the classification accuracy decreases when compressing.

Another deep learning-based model for intrusion detection in networks was put forward by Le et al. in 2019 [30]. A framework was developed for a feature engineering process, selecting intrusion features intelligently. The outcome of this stage was a dimension reduction resulting in the important features for intrusion detection, a subset of the original feature set. The next part of the process was building multiple IDSs and training them using the selected features. The learning process took place using several deep learning methods e.g., recurrent neural networks (RNN). The authors reported a very high classification accuracy over two datasets containing intrusion footprints.

Another notable work presented by Arshad et al. in 2019 proposed a new intrusion detection framework for resource-constrained IoT devices [31]. This framework aims to separate intrusion detection among the IoT devices and the edge router. IoT devices are used as IDS nodes to scan network packets so that the edge router can have an overall view about the network. The limitation of this framework is that the edge router will receive the raw packets from the host node which may contain sensitive information.

Anthi el al. in 2019 proposed a three-layer IDS design to distinguish genuine time pernicious behavior in domestic IoT gadgets [32]. This architecture focuses on classifying the type and profile of the normal behavior of each IoT device in order to detect and classify attacks. This IDS architecture has been evaluated by utilizing organized action information from a genuine test bed and scripts to dispatch multilevel attacks which speak to the behavior of an assailant. In future it should be investigated work, how the extensive need of feature engineering and data labelling could be circumvented.

The response to a survey conducted by Chaabouni et al. in 2019 indicate that machine learning techniques are successful in terms of network security and privacy [33]. The authors concluded that machine learning based frameworks for intrusion detection showed very promising results (up to 99% detection accuracy with a 0.01% false positive rate). They advocate further experimentation, combining IDSs with novel machine learning models.

The research summaries above show current progress in the development IDSs for IoT. In the following section, the technological aspects of IDS models for IoT are further explored [34].

2.3. *Technology*

This section explains the technology behind our proposed IDS, involving a multiagent system and the application of blockchain technology.

2.3.1. Multi-Agent System

An agent is described as the intelligent behavior of computer software in the fields of artificial intelligence (AI) and computer science. The term "multi-agent" generally refers to MAS or multi-agent technology (MAT). A MAS is a set of multiple agents and it is an extension and application of distributed artificial intelligence (DAI) [35]. Complex systems can be simplified and modularized using a MAS. The agents are responsible for coordination and communication through their respective tasks. Being thoroughly autonomous and form-independent, every agent can be an individual and a cluster in the MAS [24,36,37]. While the agents are developed in different languages and have divergent design patterns, they should use standard communication modes that invokes mutual communication which is not present in a single agent system.

Each agent works by having its respective set of properties and operation rules. Based on these action rules, they execute tasks during the operation of the whole MAS. The cooperation between agents in MASs helps humans to resolve some complex phenomena. Agents should have the following four basic characteristics: autonomy, responsiveness, initiative, and sociality. This is mainly manifested in the intelligence and agency ability of the system. Intelligence refers to the ability of an application system to use reasoning, learning, and other techniques to analyze and interpret all kinds of information and knowledge [38]. Agent capability refers to the ability of an agent to perceive messages from the outside world and react autonomously according to its own knowledge.

Savaglio et al. [35] provided a survey, reporting the most relevant contemporary contributions in the agent-based computing (ABC) IoT in order to assess the suitability of the ABC for IoT development. Although the authors stated in terms of computing, storage, and communication, there are no technological limitations which would prevent the full realization of the IoT ecosystem, its multifaceted development issues still require further attention. The analysis indicated that by applying ABC both SO and IoT system modeling and programming of interoperability, automaticity, and distributed intelligence can be done at to various degrees. This modeling has advantages over other approaches, such as object-oriented, service-oriented, and component-oriented computing paradigms. Furthermore, validation of multiple design choices can be performed by the agent-based simulation before the deployment phase. In synergy with other paradigms such as cloud and edge computing, it can be performed systematically and efficiently by agent-based methodologies. The survey confirmed that an agent-based development approach signifies an effective choice for many SOs and IoT systems.

In this research the proposed system uses a MAS platform JADE [39], which provides simple but powerful task execution and composition model. It uses asynchronous message passing for communication between agents. JADE is based on Java, so it is platform-independent. JADE can be used on different Java-oriented environments such as Android devices and J2ME-CLDC MIDP 1.0 devices. Moreover, JADE allows a configuration of networks characterized by partial connectivity.

2.3.2. Machine Learning

Deep Learning

In recent years, the term "deep learning" has gained popularity among researchers who work with artificial neural network (ANN) based machine learning techniques. Like ANNs, deep learning is inspired by the general structure and functionality of the brain. Deep learning involves a collection of multiple ANN layers which are stacked. It consists of both an input and output layers, building a layered data flow network. Deep learning is also described by the term deep neural network (DNN) or deep structured learning [40]. The key difference between ANN learning and deep learning is in the hidden layers. The hidden layers are placed hierarchically between the input and output layers. Deep learning is more robust than typical ANN as it processes and computes the given inputs to a further extent than an ANN. In the real world, the volume of data is growing leading to more data complexity. The intrinsic character of deep learning is the ability to learn from the previous layers or a set of large data. This feature makes deep learning a very strong and powerful candidate in the selection of machine learning models especially in classifying data of unlabeled sample datasets [41].

Multi-Agent Reinforcement Learning

Reinforcement learning (RL) methods [42] enable the computer to finish the task through continuous training and learning from the start. In recent years, as Alpha Go, in which DeepMind developed and has excelled in complex tasks, RL has shown great research potential [43]. RL has a long history as Sutton et al. already began to study RL in 1979. The method was initially used in the field of intelligent control, but with the further development of science and technology, RL has now been widely applied in autopilots, voice interactions and many other fields. RL is of great importance in the area of machine learning and computational intelligence.

The sequence of actions of multiple agents regulates the environment of multi agent systems. If an agent fails to perceive the interrelation between its own actions and environmental changes, it may generate a non-standard Markov environment. As the approach of RL is defined by the design patterns of MASs, concurrent isolated RL (CIRL) can be employed where a solo agent does not interact with other agents and only relies on an unsupervised learning technique. Interaction learning of multiple agents is achievable by applying interactive RL. Single agent RL only needs to consider temporary credit assignment problems, while multi-Agent RL needs to consider structural credit assignments [44].

2.3.3. Blockchain

Decentralization is the main motivation behind blockchain technology [45]. The distributed and transparent feature of the ledger of the blockchain means that the failure of one node does not affect the whole system. Blockchain changed the framework of the transaction network from a star to a point to point (P2P) layout. This transformed framework allows two parties to deal with each other directly with the help of encryption and security based on code and algorithm protection. Since the parties engaged in the transaction system are only required to trust the employed algorithm for establishing mutual trust, there is no necessity for knowledge about the trustworthiness of parties [46]. Moreover, the framework does not require any security endorsement by third party agents as the algorithm is fully responsible for all kinds of authentication.

Ethereum [47] and Hyperledger Fabric [48] are the two most popular blockchain application development platforms. Their underlying technologies are the same. The fundamental difference between Ethereum and Hyperledger lies in the way they are designed and in the target users. The Ethereum virtual machine (EVM) is available in Ethereum. Smart contracts and public blockchains are targeted at applications that are used by general consumers. Hyperledger Fabric has a very modular architecture that is more suited to business applications. It provides great flexibility and applies business logic more freely. The goal is to simplify work and trade processes using blockchain technology, that is, to solve the problem of inter-firm credit.

3. SESS (Smart Efficient Secure and Scalable) System

This paper uses Design Science Research and Science and Engineering Research methodologies [49]. By combining science research with engineering design and implementation, the feasibility and value of this new purposed intrusion detection system framework can be explored.

3.1. System Design

3.1.1. System Perspective

This system will be used to discover attacks from current network traffic and monitor the details of the IoT network condition. A web portal will be used for visualization of detection reports and configuration of response rules. The system consists of five parts: a blockchain smart contract module, a detection and analysis module, a response module, a data process module and a collection module. The system will need to communicate to IoT devices and monitor the dataflow of these devices. Since the system needs data to detect malicious behavior of IoT network it requires a database for data storage. The system will add and modify the data while the web portal can only gather the data. All the database communication will occur over the Internet/Intranet.

3.1.2. System Functions

The smart, efficient, secure, and scalable (SESS) system [34] will enable the IoT network administrator to monitor IoT devices based on network traffic data. The area that will be monitored is based on criteria defined by the administrator. There are several options for administrator to adjust the monitoring process of the system collection module. Collected data will be processed by the data process module which does the first attack detection, based on feature classification. These data will then be divided to two parts: an unidentified dataset and a training dataset. These two datasets will be moved to the detection and analysis module. The training dataset will be used to train the detection agent. The unidentified dataset will be used to analyze the performance of the model. All results will be sent to response module which acts based on configuration rules set by the administrator. All processes will be stored in blockchain ledgers of hosts which have a SESS module.

3.1.3. User Characteristics

IoT network administrator can use the system for intrusion detection and can add data and configure the data process and detection criteria and reduce agents when desired. For large scale IoT networks, the system modules will be installed and executed in different hosts. This means that each SESS module requires its own administrator. These administrators should be able to configure the modules that they manage.

The intrusion detection system described [34] in this paper (Figure 2) adopts a multi agent technology. Each agent is a relatively independent unit. Agents are categorized into four different types of modules. They communicate with each other through communication agents residing in each module. In this way, each module can work relatively independently which reduces dependencies between modules. The whole system consists of a collection module, a data processing module, a detection and analysis module and a response module. SESS uses Foundation of intelligent physical agents-agent communication language FIPA-ACL as agent communication language because FIPA-ACL is supported by many communities. The four communication agents will be modified and improved through interactive reinforcement learning. This means that each agent is affected by the other agents.

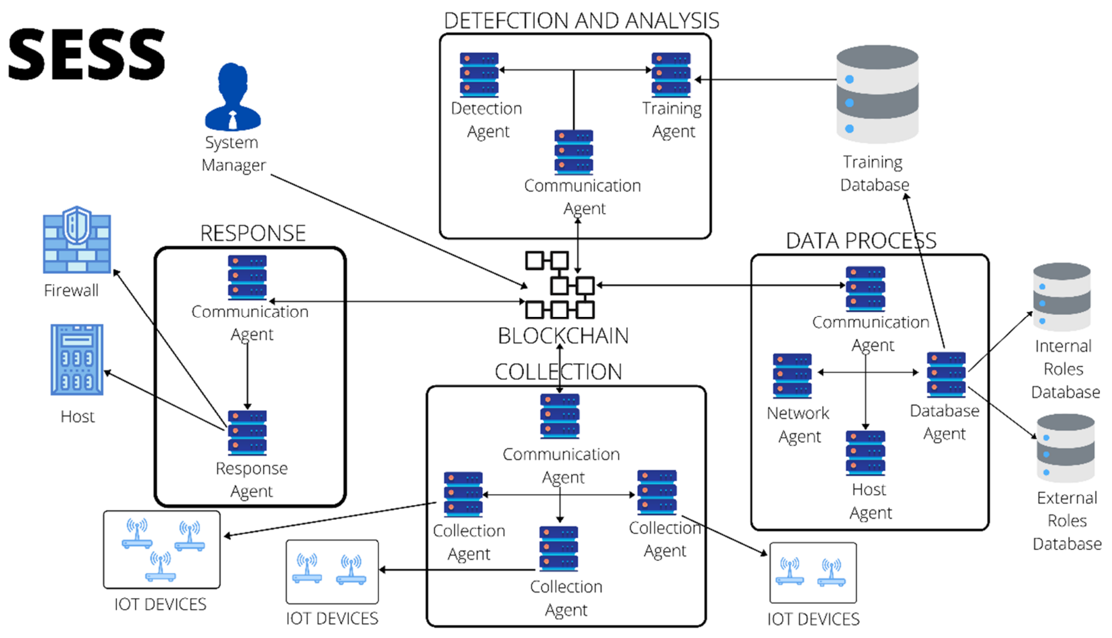

Figure 2. Intrusion detection system discussed [34].

Every successful action achieved by other agents creates feedback which is sent to the communication agent in same module. Communication agents create a feedback report after collecting feedback from other agents in the module. The feedback reports will be used for training communication agents. The credit for feedback will be assigned to communication agents based on their contributions. Each action that communication agents make will be considered a transaction. Because communication agents are the only agents which have the right to make commands, most security issues will be raised by the communication agents. Transactions will be recorded on blockchain and only the system manager has access to the smart contract (chaincode) of this blockchain.

3.2. System Development

Traditional IoT environments face many issues, including a lack of device security, centralized control of data, rigid architecture, lack of communication compatibility and difficulties in cooperation between multiple parties. The IDS used for IoT networks needs to deal with these challenges.

Blockchain can improve device security by its decentralized way to save and pass communication messages, which are commands that control the behavior of the system. Blockchain also helps to build trust between multiple parties by using an agreed smart contract to regulate system operation and behaviors of agents. Agents allow the system to scale down and scale up, making it more suitable for coping with different types of IoT environment structures and communication protocols. As the scale of IoT networks increases rapidly, IoT networks change very fast. Agents can help the system to adopt to these changes, while blockchain can ensure the security of the large scale IoT network. The source code of this system has been released for the open source community and can be downloaded from Github [50].

3.2.1. Blockchain Component

In this system, private chain is used to secure communication between agents. The system integrates the embedded database and the blockchain node in the same agent, starting and stopping at the same time. The latest interactions will be added to the cache area, reducing the search time, and speeding up the usage and storage of disk data by agents. Storing a copy of the database (DB) in all the agents will reduce the overhead of the communication agent by performing a local search of the newly updated information. The communication agent of the detection and analysis module is a super node for the blockchain module. It contains the entire blockchain ledger and is used to make sure all nodes in the network can receive the correct copy and connect with each other. All other communication agents will be full nodes. They contain and distribute the entire blockchain ledger and are essential for validating every communication record in the blockchain. Other agents in each module can be enabled as light nodes and connect to the communication agent which is their parent node. These light nodes only contain block headers of the block and used to check if their parent node gets tampered The blockchain module is not only used to regulate and validate behaviors of communication agents and to ensure the safety of the system, but also to build trust in IoT networks in which multiple parties are involved which each have their own agents. In addition, data from communication messages can be used to analyze attacks and improve agents by applying reinforcement learning. The following describes the Java classes we have used to implement the blockchain.

AgentAccount

The AgentAccount object will store the agent name, the agent publicKey and the agent privateKey. These keys will be generated by the generate PublicAndPrivateKey method. The keys are encrypted by the Elliptic curve digital signature algorithm (ECDSA) cryptographic algorithm, which is also used for encrypting Bitcoin accounts. This ensures that only the correct agent can generate certain communication messages. By random generation of the public key and the private key for each agent, based on the ECDSA cryptographic algorithm, agents can secure their information during communication.

BlockChainUtil

This class contains common methods that need to be used in blockchain products. These methods are mainly are used to deal with encryption and decryption based on cryptographic algorithms.

Communication

The Communication class is used as message container among communication agents. Because communication agents are administrators of the system, their communication has a crucial impact on the system performance. It is therefore important to ensure integrity and security of information which is shared between communication agents. Communication objects contain a hash, data, a previous communication hash, the sender agent's privateKey signature, the sender's publickKey and the recipient agent's publicKey. The hash argument is the identifier of this communication object. The data argument contains information that the sender agent wants to send to the recipient agent. The prevCmHash argument is the hash value of previous communication agents, which is used to track the order of relevant communication objects. The signature argument is created based on the sender agent's privateKey and can be verified by the sender agent's publicKey. It is used to make sure the information is sent by sender itself. The sender and recipient arguments are publicKeys of the sender agent and the recipient agent.

Block

Blockchain is a chain consisting of different blocks. A block contains the previous block hash, its own hash value, the generation time, MerkleRoot and the max block height. MerkleRoot is calculated in a recursive way. The first tree layer is calculated according to hash of communications in the block, and the each subsequent tree layer is calculated based on the previous tree layer, until the tree root is calculated. The MerkleRoot is an efficient way to track target communication and verify its integrity. The previous hash is used to identify the position of this block, while the hash is used as an identification value. Once the size of the list has reached the MAX_BLOCK_HEIGHT, the MerkleRoot and hash will be calculated and the block will be added in blockchain. The, updated blockchain then will be saved on the local host and broadcast to other modules. An example of a block generated using blockchain is shown on the next page.

Contract

The contract class contains all the business logic of the communication process. Permissions and validations are defined. They are related with specific communication messages the communication agent can send to other specific communication agents and determine when this communication action can be valid. Communication agents can only trigger the communication function by using this smart contract. Because this contract only regulates communication agents, other agents that have new functions and are under control of communication agents can still be added to or removed from the system. The contract is only used to secure communication between communication agents within the management layer, to maintain both security and scalability. When the detection and analysis module is started, blockchain will be initialized. Block genesis and communication will be saved in the blockchain and the blockchain will be saved in the local host and broadcast to other modules. The reason why this blockchain is initialized by the detection and analysis module is because this module is the core module of the system. The type of communications that can be sent and saved is decided by the Contract. Smart contracts can only be edited by system managers before initialization. They cannot be changed once the system is initialized.

Example of a block:

Box 1

```
{"generateTime":"2019-05-22
15:53:54:342","hash":"cKbh/b3rZr3J9BZpswQUTUkZcsH+9hUKXWykboItV7U\u003d","prevBHash":"jsYdYWS+SuBFKHgR+bf3bUNjZ
nVNuH5z/Ng1EC+J0AU\u003d","merkleRoot":"Athz4br2Wg2XJnD6/+X9rijYLBX8fs9OFwnuGGvv/rg\u003d","MAX_BLOCK_HEIGHT
":3,"communications":[{"prevCmHash":"MA\u003d\u003d","signature":"MDQCGGtbJbmxtPiIyKFasoz1Gm5n1oQBN9igTwIYUDlegk
AUTORIB1I+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":"MEkwEwYHKoZIzj0CAQYIKoZIzj0DA
QEDMgAEb4ZjAt85ZODTgqSvTDR/USW78eu/dw26AlsVjz/FpP3d9O5+iYriNTrHJc/y+/zf"},{"prevCmHash":"MA\u003d\u003d","signa
ture":"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":"MEkwEwYHKoZIzj0CAQYIKoZIzj0DA
QEDMgAEb4ZjAt85ZODTgqSvTDR/USW78eu/dw26AlsVjz/FpP3d9O5+iYriNTrHJc/y+/zf"},{"prevCmHash":"MA\u003d\u003d","signa
ture":"MDQCGHl8JOKqw1nHN3Shwy4HgX0voyAR8h8tQAIYF2VDUj6kiMpzj3CtC+NMgYHUlCRx3m1E","hash":"rHIaK0ZbmkH2SQ
1+6Kv5hgMEis+X6HwtLxin/hj0K6g\u003d","data":"{\"detectionDate\":\"2019-05-22
15:53:54:327\",\"attackNumber\":546,\"totalDataNumber\":1200}","sender":"MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAEDO+
Qsl5H3N3M5cJUmdWkF9rlK8qKfWH1Bb5xTLXYF5xwY8F8VumLgqssdSG0YjpO","recipient":"MEkwEwYHKoZIzj0CAQYIKoZIzj0DA
QEDMgAEb4ZjAt85ZODTgqSvTDR/USW78eu/dw26AlsVjz/FpP3d9O5+iYriNTrHJc/y+/zf"}]}
```

Each communication agent has an AgentAccount which will store the agent name, the agent publicKey and the agent privateKey. These keys are encrypted by the ECDSA cryptographic algorithm, to ensure that only the correct agent can generate and access certain communication messages. When agents in system are initialized, agent accounts are generated. When communication agents need to send a communication message (both command and feedback), these communication messages can only be processed by methods defined in the smart contract. Receivers can only check the message after verifying the integrity of the data by using the sender public key, the data and the signature. Smart contracts also decide which message can be sent or received by which communication agents. Each communication agent contains information of prevCmHash, hash, signature, data, sender and receiver. The signature is calculated by using the sender's private key and raw data. Each block contains the previous block hash, its own hash value, the generation time, merkleRoot, the max block height and communications. Communications contain the details of each communication message issued by the communication agents. MerkleRoot is calculated in a recursive way. The first tree layer is calculated according to hash of communications in the block, and then each subsequent tree layer is calculated based on the previous tree layer until the tree root is calculated. MerkleRoot is an efficient way to track target communication and verify its integrity. The previous hash is used to identify the position of this block, while the hash that is generated by Sha256 is used as identification value. Once the size of list of Communication reaches the MAX_BLOCK_HEIGHT, MerkleRoot and hash will be calculated, and the block will be added in blockchain. The updated blockchain will then be saved on local the host and broadcast to other modules.

The system integrates an embedded database and a blockchain node in the same agent, starting and stopping at the same time. Interactions will be added to cache, reducing the search time, and speeding up the usage and storage of data in agents.

3.2.2. Detection and Analysis Module

This module is responsible for the detection and analysis of malicious traffic/patterns. The following describes the Java classes we have used to implement this module.

DACommunication Agent

The DACommunication agent is the communication agent of the detection and analysis module. This process is been explained in Figure 3. Once the data process module sends a communication which says that a new data package needs to be detected, the DACommunication agent will send a command to the detection agent. When the detection agent has finished detection, the DACommunication agent will receive the detection report from the detection agent. The detection report will be encapsulated into a communication object and sent to the RCommunication agent. If the DACommunicaiton agent gets a communication object which says that the training set is updated, the DACommunication agent will send a command to the training agent. After the training agent has finished training the model a training report will be sent to the DACommunication agent as well as to the RCommunication agent.

Detection Agent

When the Detection agent receives commands from the DACommunication agent, it will start using the DNN model to analyze the new data package and detect attacks. After the process is finished, a detection report will be generated and sent to DACommunication agent.

Training Agent

When the Training agent receives commands from the DACommunication agent, the Training agent will start training the DNN model by using the updated training dataset. After the process is finished, a training report will be generated and sent to the DACommunication agent.

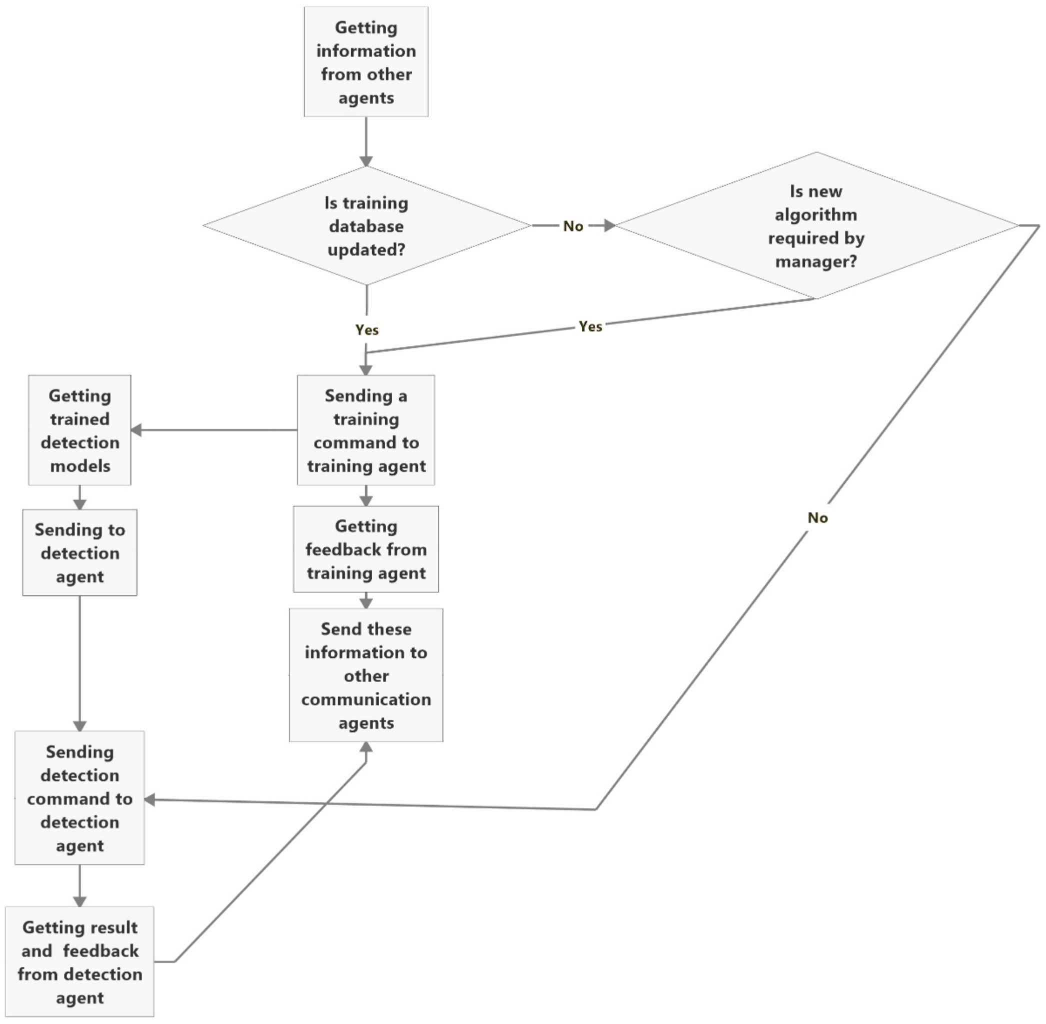
Getting information from other agents
Is training database updated?
No
Is new algorithm required by manager?
Yes
Yes
No
Getting trained detection models
Sending a training command to training agent
Sending to detection agent
Getting feedback from training agent
Send these information to other communication agents
Sending detection command to detection agent
Getting result and feedback from detection agent

Figure 3. Process for detection and analysis.

DNNUtil

This class contains the methods that are relevant for DNN model management. The new raw dataset will be standardized and normalized using normalization methods. The transformed structure can be reviewed and adjusted via the transformProcess function and file.

3.2.3. Response Module

RCommunication Agent

The RCommunication agent is the communication agent of the response module. Once the RCommunication agent receives the Communication message that contains the detection report or the training report, the RCommunication agent will send a command to the response agent. These reports will be saved on local host.

Response Agent

After the Response agent receives a command from the RCommunication agent, the response agent will start generating data visualization charts for the users, based on the administrator's response plan, and make firewall and host security changes based on the RCommunication agent's commands.

The procedure sequence of the intrusion detection system SESS is as follows (Figure 4):

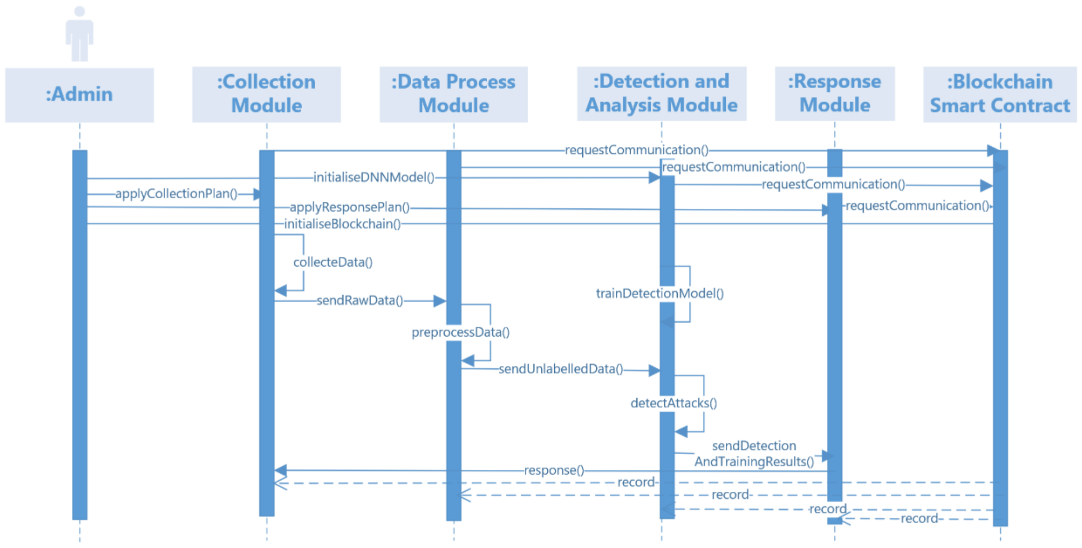

Figure 4. Smart, efficient, secure, and scalable (SESS) system sequence diagram.

Data Visualization

BarChartUtil and LineChartUtil classes are used to manage data visualization. Parameters, like the size of charts or the theme of the charts can be adjusted.

3.2.4. Data Process Module

The primary aim of this module (Figure 5) is to process the data for network, host and other agents.

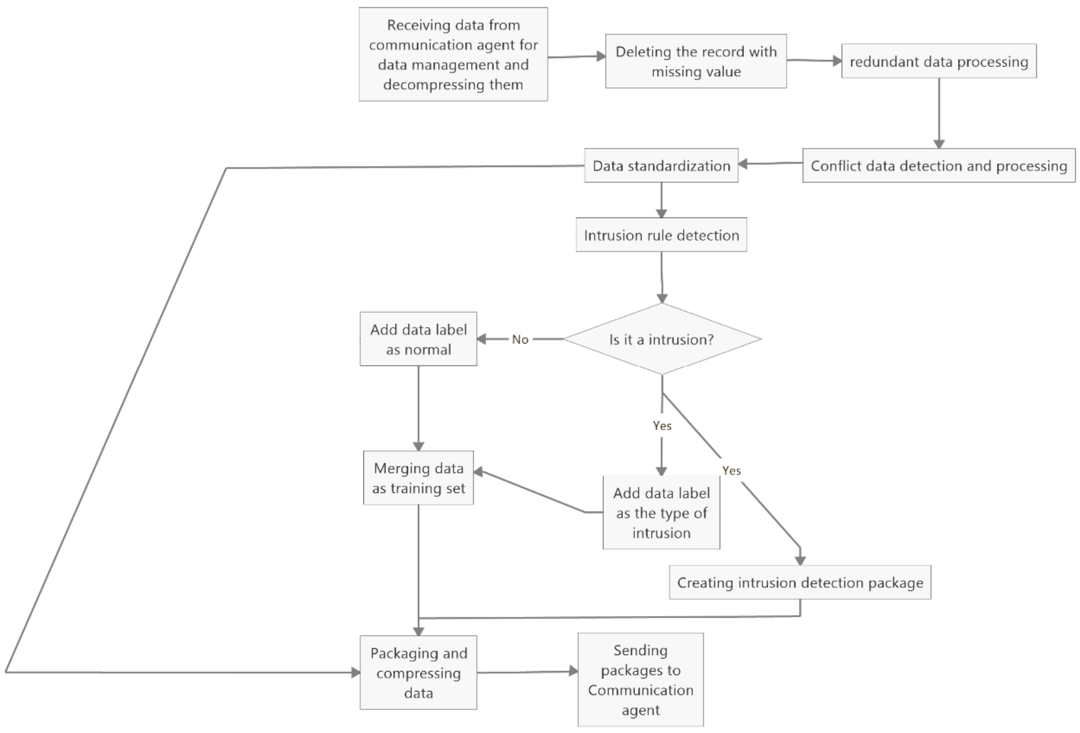
Receiving data from communication agent for data management and decompressing them
Deleting the record with missing value
redundant data processing
Conflict data detection and processing
Data standardization
Intrusion rule detection
Is it a intrusion?
No
Add data label as normal
Yes
Yes
Merging data as training set
Add data label as the type of intrusion
Creating intrusion detection package
Packaging and compressing data
Sending packages to Communication agent

Figure 5. Process of data management agents.

The network data management agent and the host data management agent deal (Figure 5) with data that come from the network and the host respectively. Both agents need to do data pre-processing, including missing value processing, data integration and data standardization. After the pre-processing stage, the data will be scanned, based on intrusion detection rules generated by other open source IDS communities and or rules of system manager. In addition, the pre-processed data will be packaged and compressed. The discovered intrusion behaviors and the corresponding processing measures are recorded in the intrusion detection package. After this is compared with the result of detection agent, the intrusion detection package is sent to the response agent. The labelled data are merged into a training set package, which is used to improve the accuracy of the detection agent.

The database agent (Figure 6) is the only agent which can alter the database. It can change database only if it has a command from the communication agent. Once the database agent receives the update command from the communication agent, it changes the database based on the data provided by the communication agent. After the database agent has changed the database, it sends a feedback message to the communication agent.

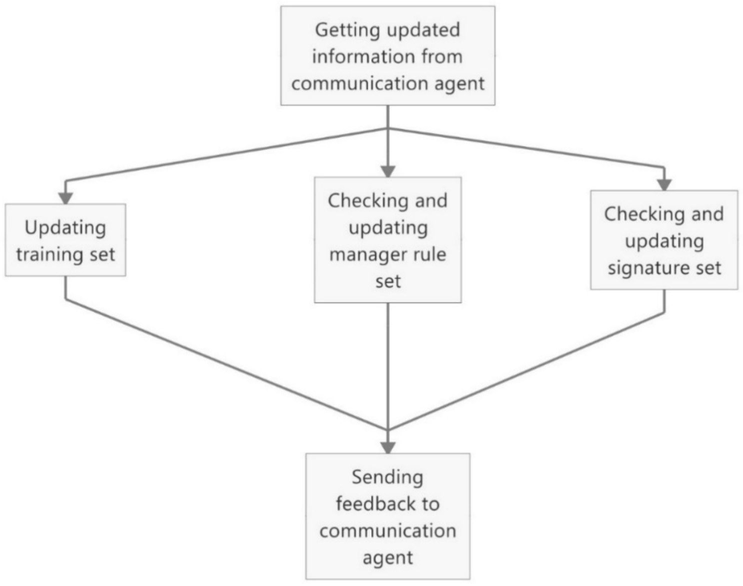
Getting updated information from communication agent
Updating training set
Checking and updating manager rule set
Checking and updating signature set
Sending feedback to communication agent

Figure 6. Process of database agent.

The communication agent for the data process (Figure 7) manages and controls the host data process agent, the network data process agent and the database agent. It sends the data packages from the collection module to the data process agents, gives data process agent commands and gets their feedback. It asks the database agent to run regular updating checks, making sure the system is up to date. This agent also communicates with other communication agents and adjust its work pattern according to the state of system.

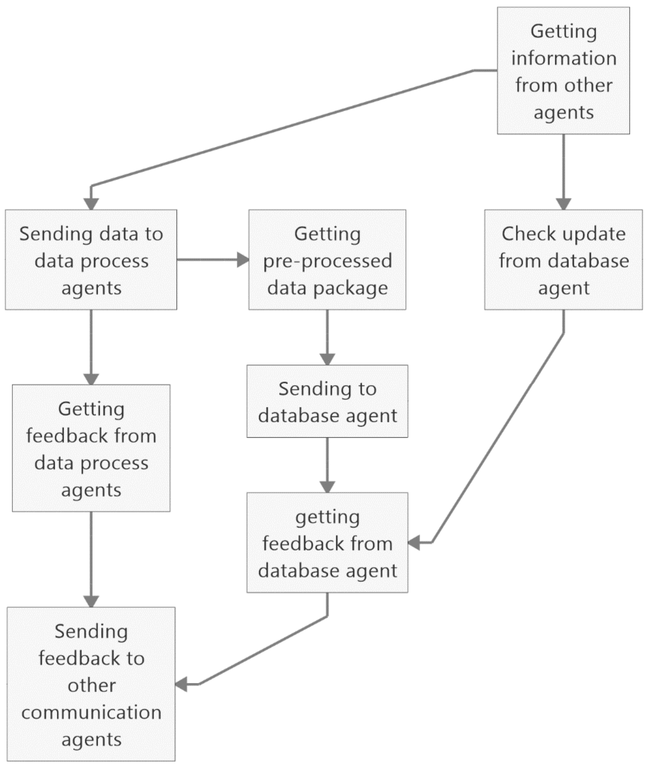
Getting information from other agents
Sending data to data process agents
Getting pre-processed data package
Check update from database agent
Getting feedback from data process agents
Sending to database agent
getting feedback from database agent
Sending feedback to other communication agents

Figure 7. Communication agent for data process.

4. Experimental Analysis

4.1. Dataset

In 1998, MIT Lincoln Laboratories undertook an important project, the Defense advanced research projects agency (DARPA) Intrusion Detection Assessment Project, which was designed to detect and evaluate the performance of intrusion detection systems. An important achievement of this evaluation project is the establishment of a data set for simulating various attacks. However, the dataset is too large, which is not conducive to a fair comparison of different intrusion detection algorithms. In addition, the feature information recorded is too complex, with different protocols using different formats which is not conducive to the selection of feature attributes.

The network security audit data set KDDCUP99, published by Professor Stolfo at Columbia University's Intrusion Detection Laboratory and others, was analyzed and compiled from the IDS data set of MIT Lincoln Laboratory in 1998, but it contained only network traffic data [51].

The NSL-KDD dataset solves some of the inherent problems of the KDD99 data sets. The NSL-KDD data set is widely used in the development of intrusion detection systems. Although this data set still has some flaws, it can be used as an effective benchmark data set to help researchers compare different intrusion detection methods. The size of the NSL-KDD training set and test set are reasonable, and the evaluation results of different researchers will be consistent and comparable. Data of the NSL-KDD data set comes from three different protocols (Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP). The NSL-KDD data set contains four (4) attack classes (DoS, Probe, R2L, U2R) and 39 different attack types [52]. This research uses the NSL-KDD data set as data source for intrusion detection simulation.

Although data from NSL-KDD is not specific for IoT environment testing, the protocols and attacks involved in this dataset are valuable for IoT environment intrusion detection research. Attack types in this dataset also happen in real world IoT networks and protocols like TCP are becoming more important because both the traditional application layer and cloud platform connections an important part of IoT environments. The NSL-KDD dataset is therefore used as dataset for IoT. Samples from the NSL-KDD dataset are labelled as normal or anomaly. A 70% of subset named "Train + 20% dataset" is used as a training dataset, while 30% of "Train + 20%" dataset is used as a validation dataset. A subset of "Test+ dataset" that contains 1200 data samples is used as a test dataset (Tables 1 and 2).

Table 1. Attack types on Train + 20%.

Major Attack Types	Attack Types
DOS	Back, Land, Neptune, Pod, smurf, teardrop
Probe	Satan, IPsweep, Nmap, portsweep
R2L	ftp_write, phf, multihop, warezmaster, guess_passwd, warezclient, imap, spy
U2R	buffer_overflow, loadmodule, rootkit

Table 2. Attack types on sub dataset of Test+.

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

4.2. Evaluation Measures

The performance of intrusion detection will be measured based on an accuracy, precision, recall and F1 score. The accuracy is the ratio of the number of samples in the prediction pair to the total number of samples. The precision [53] is based on prediction results. It indicates how many of the positive prediction samples are true positive samples. The recall is for original sample, which indicates how many positive examples in the sample are predicted correctly. The F1 score comprehensively considers the calculation results of the model's precision rate and recall rate. The larger the F1-score, the better the quality of the model. Overfitting needs to be taken into consideration as well.

4.3. Pre-Processing

All raw data for the training, validation and test datasets are standardized by one-hot coding and z-score measures and normalization is done using same schema. For this experimental design, one-hot encoding uses the categorical variables from all training, validation and test dataset, instead of using categorical variables based on individual datasets. One categorical variable in each parameter that needs to be encoded by one-hot encoding is removed to prevent dummy variable trap.

The last field "class" is the output label. Variables as protocol_type, service and flag contains multiple values, so they need to be transferred to numerical values by using one hot encoding. In order to avoid the dummy variable trap, one categorical value of protocol_type, service and flag is removed. The "class" field is encoded to numerical values as well.

4.4. Findings and Discussions

To identify the performance of the detection and analysis module, the performance in different situations have been compared (Tables 3–5).

Table 3. Intrusion detection accuracy of different number of 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. Intrusion detection accuracy of different batch number.

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. Intrusion detection accuracy rate of different batch size.

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%

By testing the detection and training module with different batch sizes, batch numbers and epochs, it is shown that the DNN model has a better performance for the batch size of 48, the batch number of 300 and a number of epochs of 100 then for the other values tested. The accuracy rate using the validation set is 98.27%, while the accuracy of the test set is 83.17%. Precision on test set is 83.53%. Recall on test set is 84.14%, and the F1 Score on the test set is 83.94%. These mean that DNN model in this system has a good ability to distinguish attacks from normal data traffic. These experiments also indicate that taking enough epochs can ensure that the model has a good performance, even if the training set has fewer samples. Table 6 shows the performance of intrusion based on different datasets. They are denoted as follows: accuracy on validation set as AV, precision on validation set as PV, recall on validation set as RV, 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: Table 7 summarizes the accuracy of detection when using different data-splitting conditions. The model could potentially be upgraded for more complex datasets.

Table 6. Performance of intrusion detection by using different training set.

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 by using different data splitting conditions.

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

The table below (Table 8) shows a brief description of the parameters of the data extracted from NSL-KDD along with their serial number. In total 41 parameters used as mentioned below.

Table 8. NSL-KDD parameters [35] used.

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. *Cont.*

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)

Table 9 shows the performance of the detection and analysis module on different numbers of parameters. Because the SESS intrusion detection system is designed to be used on different scales of IoT networks and some parameters are difficult to collect on some IoT networks, it is important to know how the performance of the system is affected by different numbers of parameters.

Table 9. Intrusion detection performance by using different 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%

The table above (Table 9) uses 48 as batch sizes, 300 as batch number and 100 as epoch. The total number of parameters of the NSL-KDD dataset is 41.

The first parameter that is removed is service, because the type of service is different for different IoT networks which has a large impact on the normalization process. When excluding the service parameter only, using 40 parameters the DNN model has an 82.33% accuracy on the test set, 0.84% less than when using the full number of 41 parameters.

When the data sets use 25 parameters, the DNN model has an accuracy of 80.75% on the test set and the accuracy increases to 81.33 if we use only 17 parameters.

The reason why the accuracy for 17 parameters is higher than when using 25 parameters is because some parameters needs to be combined with other parameters to detect attacks. When the number of parameters is reduced from 40 to 25, some parameters lose their combination parameters, which has an adverse impact on the weighting process. After removing these combined parameters, the accuracy increases.

When only seven parameters i.e., duration, protocol_type, flag, src_bytes, dst_bytes, count and srv_count are used, the DNN model has a 78.08% accuracy rate. This means that the DNN model still has a decent detection ability even if the collection module cannot get many details from the IoT network. When duration, protocol_type, src_bytes, dst_bytes, count and srv_count only are used as parameters, the DNN model has a 72.68% accuracy and a 68.74% recall on the test set. In this case the removal of the parameter flag has reduced the accuracy rate. This shows the importance of flag parameter. However, the flag parameter also complicates normalization process. In order to investigate the performance of this system on an IoT network that does not provide many details, the flag parameter should be removed, because it can be hard to capture on certain circumstances.

If the data sets use only five parameters, the DNN model has a 72.33% accuracy on the test set. If the data sets use protocol_type, src_bytes, dst_bytes and count as parameters, the DNN model has 73.92% accuracy on test set. This means that using only four parameters results in a better performance than using five or six parameters.

If the data sets use three parameters, the DNN model has a 71.75% accuracy on the test set, but it only has a 58.15% accuracy on the validation set. This is because the DNN model uses Dropout. This regularization technique will randomly remove some weak classifiers during training process. For this situation, using dropout will affect accuracy when applied to the validation set. Increasing the batch size or removing the dropout technique can resolve this problem, although this may in turn lead to an overfitting problem. If dropout is removed from the DNN model, the DNN model has an accuracy of 89.23% on the validation set, and 70% on the test set. If the data sets use src_bytes and dst_bytes as parameters only, the DNN model has 68.5% accuracy on test set.

By using dst_bytes as the only parameter, accuracy rate is 43% on the test dataset, but the F1 Score on both the validation and the test dataset is 0%. It means that all samples in the data sets are classified as one class, an overfitting problem. This is because some hyper-parameters, such as the number of epochs, are too high for one parameter datasets.

As shown in Figure 8, this model works well on TCP and ICMP datasets (accuracy rate is 99.1% and 98.1% respectively). However, the accuracy rate on UDP protocol is only 93.7%. One reason is that the data set contains very few data with UDP protocols. The features of the UDP protocol are another reason. For a dataset which uses class for different attack types as values, DNN has a good performance (accuracy rate is 97%). Some attack types which occur only in a relatively small amount of the data, however, cannot be distinguished very well.

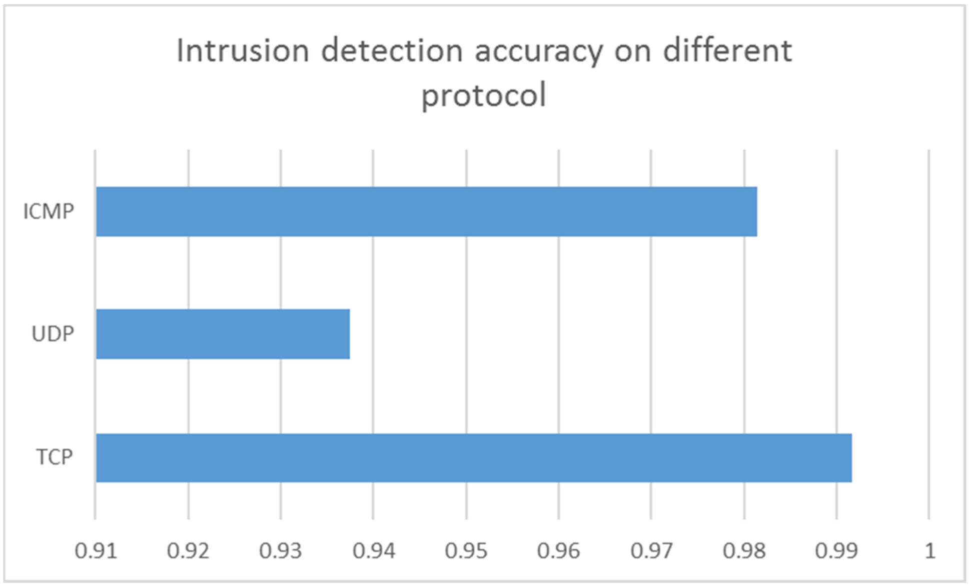

Figure 8. Accuracy on different protocols.

In conclusion, these experiments show the usability of this SESS system on different scales of IoT network. During the experiments, the detection time is around 0.18 s, which means that the system can detect attacks and respond to them in real time. The timing was recorded based on various stages including initializing and loading data model, training time, and the detection time. This timing may vary based on various circumstances including the hardware, other resources utilizing the server, memory needs etc. In fact, these measurements need further investigation and research to make sure the proposed model can perform in a high-speed live environment. The findings from these experiments will also be used to define statuses for the multi agent reinforcement learning process.

4.5. Limitations

The results of these experiments show that this system can be used on different scales of IoT networks, accommodating even more complex networks. However, more research still needs to be done on this area. Although the accuracy rate of DNN model on distinguishing different attack types is quite high, some rare attack types cannot be detected accurately. The lengthy learning period of agents and the need for large training sets will need to be resolved in future. How multiple agents can cooperate properly in large IoT network needs to be further investigated. Large computing power requirements is also an issue which needs to be addressed in future designs.

5. Conclusions

This paper proposed an intrusion detection system based on a multi-agent system, blockchain and deep learning. We have described the working mode of each component of the model and the operation of the whole system in detail. The flexibility of multi-agent systems means that this new IDS can be applied across IoT environments of various sizes. All actions caused by communication agents will be recorded on blockchain, which makes the system secure from threats, including information tampering, information disclosure and so on. Use of a multi-agent reinforcement algorithm can help the system to improve its performance continuously.

The application of neural networks is studied. The simulation results show that the deep learning algorithm has a better performance than traditional methods on the same type of IoT network. The implementation of blockchain technology ensures that this system can be distributed to different remote hosts, because ACL communication is secured by blockchain and communication among agents is regulated by the smart contract. The experiments using the NSL-KDD dataset demonstrate a high accuracy of intrusion detection on the transport layer of the IoT environment for DNN. The performance of the DNN model in distinguishing anomaly from normal is better than the performance of other machine learning methods, such as decision trees. This demonstrates the potential of using deep learning algorithms for IDS of IoT devices. The experiments demonstrate a high performance of the system in different scenarios, such as networks of different complexity and different attack types.

Future Work

The work is based on the study of multi-agent technology, block chains and neural networks to propose an intrusion detection system model for the IoT. The next step is to continuously improve the modules by running the system in an actual environment so that each agent can work well with each other. The following stage is to collect IoT network data sets to train the detection model in order to improve the performance of the system. In addition, creating datasets of novel attack types is something that could be explored further. To improve the performance of the system, more reliable and faster multi-agent algorithms could be used to train the communication agents and to optimize the feedback training process. Advanced deep learning algorithms could be explored to improve the performance of system. Since we have tested block chain on a smaller variant, block max height of blockchain and hyper-parameters of DNN model can be adjusted by using multi agent reinforcement learning in the next version. For a larger scale IoT network, unspent communication output pool should be implemented, to manage the order of communications. Testing against UNSW-BN15 datasets could also be explored. For future IoT system testing, an IoT data simulator [54] or any similar tools could be used to generate data to further evaluate the accuracy of the IDS.

Author Contributions: Conceptualization, C.L. and B.S.; methodology, C.L., B.S. and S.A.; software, C.L., and A.K.; validation, C.L., A.K., N.B.I. and A.I.; literature preparation A.I., M.Z. and C.L.; data curation, C.L., S.A. and A.I.; writing—original draft preparation, C.L., S.K. and M.Z.; writing—review and editing, S.A., B.S., S.K., A.I., N.B.I.; supervision, B.S. and S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

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]
20. Bostani, H.; Sheikhan, M. Hybrid of anomaly-based and specification-based IDS for Internet of Things using unsupervised OPF based on MapReduce approach. *Comput. Commun.* **2017**, *98*, 52–71. [CrossRef]
21. Fu, Y.; Yan, Z.; Cao, J.; Koné, O.; Cao, X. An Automata Based Intrusion Detection Method for Internet of Things. *Mob. Inf. Syst.* **2017**, *2017*, 1750637. [CrossRef]
22. Kapitonov, A.; Lonshakov, S.; Krupenkin, A.; Berman, I. Blockchain-based protocol of autonomous business activity for multi-agent systems consisting of UAVs. In Proceedings of the Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), Linkoping, Sweden, 3–5 October 2017; pp. 84–89. [CrossRef]
23. Calvaresi, D.; Calbimonte, J.P.; Dubovitskaya, A.; Mattioli, V.; Piguet, J.G.; Schumacher, M. The Good, the Bad, and the Ethical Implications of Bridging Blockchain and Multi-Agent Systems. *Information* **2019**, *10*, 363. [CrossRef]
24. Calvaresi, D.; Dubovitskaya, A.; Calbimonte, J.P.; Taveter, K.; Schumacher, M. Multi-Agent Systems and Blockchain: Results from a Systematic Literature Review. In Proceedings of the International Conference on Practical Applications of Agents and Multi-Agent Systems, Toledo, Spain, 20–22 June 2018.
25. Shi, H.; Zhai, L.; Wu, H.; Hwang, M.; Hwang, K.; Hsu, H. A Multi-tier Reinforcement Learning Model for a Cooperative Multi-agent System. *IEEE Trans. Cogn. Dev. Syst.* **2020**. [CrossRef]
26. Diro, A.A.; Chilamkurti, N. Distributed attack detection scheme using deep learning approach for Internet of Things. *Future Gener. Comput. Syst.* **2018**, *82*, 761–768. [CrossRef]
27. Duong, T.; Todi, K.K.; Chaudhary, U.; Truong, H. Decentralizing Air Traffic Flow Management with Blockchain-based Reinforcement Learning. In Proceedings of the IEEE 17th International Conference on Industrial Informatics (INDIN), Helsinki, Finland, 23–25 July 2019; pp. 1795–1800. [CrossRef]
28. Casado-Vara, R.; Prieta, F.D.L.; Prieto, J.; Corchado, J.M. Blockchain framework for IoT data quality via edge computing. In Proceedings of the BlockSys'18: 1st Workshop on Blockchain-enabled Networked Sensor System 2018, Shenzhen, China, 4 November 2018. [CrossRef]
29. Li, D.; Deng, L.; Lee, M.; Wang, H. IoT data feature extraction and intrusion detection system for smart cities based on deep migration learning. *Int. J. Inf. Manag.* **2019**, *49*, 533–545. [CrossRef]

30. Le, T.-T.-H.; Kim, Y.; Kim, H. Network Intrusion Detection Based on Novel Feature Selection Model and Various Recurrent Neural Networks. *Appl. Sci.* **2019**, *9*, 1392. [CrossRef]
31. Arshad, J.; Azad, M.A.; Abdeltaif, M.M.; Salah, K. An intrusion detection framework for energy constrained IoT devices. *Mech. Syst. Signal Process.* **2020**, *136*, 106436. [CrossRef]
32. Anthi, E.; Williams, L.; Slowinska, M.; Theodorakopoulos, G.; Burnap, P. A Supervised Intrusion Detection System for Smart Home IoT Devices. *IEEE Internet Things J.* **2019**, *6*, 9042–9053. [CrossRef]
33. Chaabouni, N.; Mosbah, M.; Zemmari, A.; Sauvignac, C.; Faruki, P. Network Intrusion Detection for IoT Security Based on Learning Techniques. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2671–2701. [CrossRef]
34. Liang, C.; Shanmugam, B.; Azam, S.; Jonkman, M.; Boer, F.D.; Narayansamy, G. Intrusion Detection System for Internet of Things based on a Machine Learning approach. In Proceedings of the International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), Vellore, India, 30–31 March 2019; pp. 1–6. [CrossRef]
35. Savaglio, C.; Fortino, G.; Ganzha, M.; Paprzycki, M.; Badica, C.; Ivanovic, M. Agent-based Internet of Things: State-of-the-art and research challenges. *Future Gener. Comput. Syst.* **2019**, *102*. [CrossRef]
36. Pipattanasomporn, M.; Feroze, H.; Rahman, S. Multi-agent systems in a distributed smart grid: Design and implementation. In Proceedings of the IEEE/PES Power Systems Conference & Exposition, Seattle, WA, USA, 15–18 March 2009.
37. Fortino, G.; Russo, W.; Savaglio, C. Agent-oriented modeling and simulation of IoT networks. In Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS), Gdansk, Poland, 11–14 September 2016; pp. 1449–1452.
38. Wang, S.; Wan, J.; Zhang, D.; Li, D.; Zhang, C. Towards smart factory for industry 4.0: A self-organized multi-agent system with big data based feedback and coordination. *Comput. Netw.* **2016**, *101*, 158–168. [CrossRef]
39. Bellifemine, F.L.; Caire, G.; Greenwood, D. *Developing Multi-Agent Systems with JADE*; Wiley: Hoboken, NJ, USA, 2007.
40. Nguyen, L.C.; Nguyen-Xuan, H. Deep learning for computational structural optimization. *ISA Trans.* **2020**, in press. [CrossRef] [PubMed]
41. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef] [PubMed]
42. Saleh, A.J.; Karim, A.; Shanmugam, B.; Azam, S.; Kannoorpatti, K.; Jonkman, M.; Boer, F.D. An Intelligent Spam Detection Model Based on Artificial Immune System. *Information* **2019**, *10*, 209. [CrossRef]
43. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
44. Zilberstein, S. Book Review: "Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence", Gerhard Weiss. *Int. J. Comput. Intell. Appl.* **2001**, *1*, 331–334. [CrossRef]
45. Vokerla, R.R.; Shanmugam, B.; Azam, S.; Karim, A.; Boer, F.D.; Jonkman, M.; Faisal, F. An Overview of Blockchain Applications and Attacks. In Proceedings of the International Conference on Vision Towards Emerging Trends in Communication and Networking (Vitecon), Vellore, India, 30–31 March 2019.
46. Kosba, A.; Miller, A.; Shi, E.; Wen, Z.; Papamanthou, C. Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. In Proceedings of the IEEE Symposium on Security and Privacy (Sp), San Jose, CA, USA, 22–26 May 2016.
47. Sun, T.; Yu, W. A Formal Verification Framework for Security Issues of Blockchain Smart Contracts. *Electronics* **2020**, *9*, 255. [CrossRef]
48. Košt'ál, K.; Helebrandt, P.; Belluš, M.; Ries, M.; Kotuliak, I. Management and Monitoring of IoT Devices Using Blockchain. *Sensors* **2019**, *19*, 856. [CrossRef]
49. Carstensen, A.-K.; Bernhard, J. Design science research—A powerful tool for improving methods in engineering education research. *Eur. J. Eng. Educ.* **2019**, *44*, 85–102. [CrossRef]
50. Github Code. Available online: https://github.com/aymwxbb2012/intrusion_detection_system_SESS/tree/develop (accessed on 30 May 2020).
51. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009.

52. Dhanabal, L.; Shantharajah, S.P. Intrusion Detection and Classification Using Hybrid Support Vector Machine and Dynamic Ant Colony Algorithm. *Aust. J. Basic Appl. Sci.* **2015**, *9*, 328–335.
53. Manjula, B.; Balachandra, M. Performance Evaluation of Supervised Machine Learning Algorithms for Intrusion Detection. *Procedia Comput. Sci.* **2016**, *89*, 117–123. [CrossRef]
54. IoT Data Simulator. Available online: https://assetwolf.com/learn/iot-data-simulator (accessed on 30 May 2020).