Islamic Azad University
South Tehran Branch
Faculty of Technical and Engineering

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of MSc in the Department of Computer Engineering

Title

Intrusion Detection and Identification of Attacks on the Internet of Things (IoT) Using a Combination of Machine Learning Methods

By
Hamid Bostani

Supervisor
Prof. Dr. Mansour Sheikhan

September 2015

Reviewers

Dr. Mansour Sheikhan, Full Professor of Electrical Engineering, IAU- South Tehran Branch
Dr. Ali Moeini, Full Professor of Computer Science, University of Tehran
Dr. Amir Sahafi, Assistant Professor of Computer Engineering, IAU- South Tehran Branch

Note: This thesis is translated to English in May 2019

Abstract

The Internet of Things (IoT) is a worldwide network including all identifiable heterogeneous objects around us such as smartphones, laptops or smart sensors that can connect to the Internet by using a wide range of technologies. IoT is able to provide accessibility to the Internet for all physical objects since it is a hybrid network of the Internet and diverse networks with heterogeneous nodes. Generally, due to the insecure nature of the Internet as well as Wireless Sensor Networks, which are the main components of IoT, implementing security mechanisms in IoT seems necessary. To deal with intrusions that may occur in IoT, a novel multi-faceted intrusion detection system is proposed in this thesis which can detect both cyber-attacks and insider-attacks of IoT.

This study proposes an offline misused-based technique based on a modified supervised Optimum-Path Forest (OPF) to detect the external (cyber) attacks of IoT. In this technique, k-Means algorithm, along with the social network analysis (i.e., centrality and prestige), is employed to overcome the problem of the scalability of the input large dataset and prune the training dataset with the aim of identifying the most informative samples. The proposed method uses a high-dimensional NSL-KDD dataset as the simulated traffic of the Internet; hence, this thesis presents a novel hybrid method based on Binary Gravitational Search Algorithm and Mutual Information to reduce the dimensions of the original input dataset. The experimental results show the superior performance of the proposed approach in detecting and identifying the types of cyber-attacks.

In addition, this study provides a hybrid of anomaly-based and specification-based real-time intrusion detection system to determine routing attacks in 6LoWPAN (the main effort to make the concept of real IoT) based on the unsupervised OPF. The proposed method, which is an efficient security technique not only can detect internal (insider) attacks of IoT but also can determine the malicious nodes as the cause of the IoT's insider-attacks. In addition, the presented model is developed based on the MapReduce architecture in order to obtain the ability of distributed detection.

Keywords

Internet of Things (IoT), Intrusion Detection System (IDS), machine learning, feature selection, Binary Gravitational Search Algorithm (BGSA), mutual information, social networks, Optimum-Path Forest (OPF), misuse-based intrusion detection, anomaly-based intrusion detection, specification-based intrusion detection.

Publications associated with this thesis

1. **Bostani, H.**, & Sheikhan, M. (2017). Hybrid of anomaly-based and specification-based IDS for Internet of Things using unsupervised OPF based on MapReduce approach. *Computer Communications*, *98*, 52-71.

2. **Bostani, H.**, & Sheikhan, M. (2017). Modification of supervised OPF-based intrusion detection systems using unsupervised learning and social network concept. *Pattern Recognition*, *62*, 56-72.

3. **Bostani, H.**, & Sheikhan, M. (2017). Hybrid of binary gravitational search algorithm and mutual information for feature selection in intrusion detection systems. *Soft Computing*, *21*(9), 2307-2324.

4. Sheikhan, M., & **Bostani, H.** (2017). A security mechanism for detecting intrusions in Internet of Things using selected features based on MI-BGSA. *International Journal of Information and Communication Technology Research*, *9*(2), 53-62.

5. Sheikhan, M., & **Bostani, H.** (2016). Binary Gravitational Search Algorithm (BGSA): Improved Efficiency. In *Encyclopedia of Information Assurance-4 Volume Set (Print)* (pp. 1-19).

6. Sheikhan, M., & **Bostani, H.** (2016, September). A hybrid intrusion detection architecture for Internet of things. In *2016 8th International Symposium on Telecommunications (IST)* (pp. 601-606). IEEE.

Acknowledgments

I am truly indebted to my dear wife, Marzieh that there is no way to properly acknowledge her. I am extremely thankful with love to her for her patience, encouragement, and motivation throughout this research work. Moreover, I would like to express my deepest appreciation to my supervisor, Professor Mansour Sheikhan for his valuable guidance, constant support and useful critiques which helped me to present the scientific results in an efficient manner in this thesis. Finally, I would like to thank my dear parents for their support and encouragement throughout my life.

Content

List of Figures

List of Tables

Chapter 1: Introduction

Internet of Things (IoT) is a worldwide network in which the identifiable heterogeneous physical objects can connect to the Internet by using a wide range of technologies. Actually, the main idea of IoT is developing an autonomous world including a series of smart objects, which are available from anywhere at any time, and they also are able to connect with each other, exchange information and even make decisions (Rghioui, Khannous, & Bouhorma, 2014). In the IoT's approach, the different heterogeneous objects (e.g., PCs or laptops, smartphones, smart sensors, electronic home appliances, and surveillance cameras) can be directly connected to the Internet or other networks by employing a wide range of different technologies such as radio frequency identification (RFID), embedded sensors, and miniature actuators without using a special interface tools such as PCs. Therefore, in the near future, IoT will be the base of many services that living in a developed country depends on the reliable and appropriate access to the obtained services by IoT. Wireless Sensor Networks (WSNs) are one of the important components of IoT in which the nodes can communicate with each other and also other smart systems, autonomously (Shahid, 2013). One of the main efforts to make the concept of real IoT is the IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs) that is proposed and standardized by the Internet Engineering Task Force (IETF) workgroup. 6LoWPAN is a wireless sensor network based on IPv6 that integrates the IP-based infrastructures and WSNs by specifying how IPv6 packets are be routed in constrained networks such as IEEE 802.15.4 networks (Shahid, 2013). As can be seen in Figure 1-1, the resource-constrained devices (from the aspect of radio range, processing power, and battery life) such as sensors can connect to the Internet or other networks through 6BR[1] (Wallgren, Raza, & Voigt, 2013).

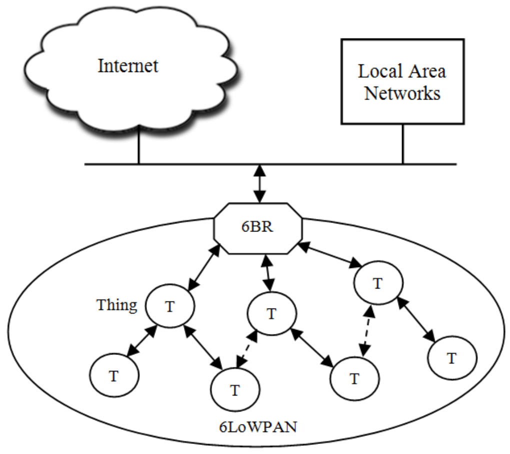


Figure 1-1: The architecture of Internet of Things

One of the important issues in IoT is security. Like other kinds of networks, providing security for IoT can be done by using the prevention and the intrusion detection mechanisms. The prevention mechanism is based on different methods such as cryptography. On the other hand, in the intrusion

[1] 6LoWPAN Bourder Router

detection mechanism (as the second layer of defense that this thesis focuses on it), identifying threats is done by using intrusion detection systems (IDSs). Intrusion detection systems are software/hardware systems that are able to detect malicious behaviors in networks or host systems. It is worth noting that the purpose of this thesis is providing a special IDS for IoT based on the machine learning algorithms, which can detect intrusions and identify the types of attacks in IoT, as well.

1.1 Problem Statement

With the fast growth of technologies in computer networks, especially the Internet, which has led to a dramatic increase in computer networks' applications, security has become the main challenge of different types of computer networks such as 6LoWPAN (a real IoT's deployment). IPv6 in IoT provides Internet connectivity for all the physical objects. Because IoT is a hybrid network that is consisted of the Internet and some kind of networks including heterogeneous and constrains objects (e.g., WSNs), working on the IoT's security for employing the standardized security mechanisms in IoT is a necessary approach (Shahid, 2013). By using the standard techniques (e.g., encryption and access control), the communications in IoT are to some extent safe. But due to the unsafe nature of wireless communications, these prevention mechanisms are not able to cope with all the attacks, especially insider-attacks (e.g., routing attach). Moreover, in IoT, the heterogeneous and resource-constrained objects can directly connect to the Internet through IPv6 and 6LoWPAN networks. Despite the confidentiality and integrity of 6LoWPAN messages, which are provided by employing the communication security, there are many attacks that can be happened for disrupting the normal functionality of a network, consequently countless problems in access and security services of a network (Wallgren et al., 2013). In addition, due to the wireless nature of IoT, every physical object can be easily threatened through various attacks (e.g., Denial of Service (DoS), which makes network or machine resource unavailable) by both of the Internet and WSNs. Therefore, another security mechanism should be employed for identifying some kind of attacks that cannot be detected by the standard prevention mechanism. Vividly, the intrusion detection systems (IDSs) as the most effective solution can be engaged to detect attacks or malicious behaviors in networks or machines.

The Internet of Things as an emerging concept has a novel architecture in which the physical heterogeneous objects, which are mostly resource-constrained devices, can be detectable and accessible from anywhere through an IP address. It seems providing a novel security mechanism by introducing an appropriative IDS for IoT is necessary. There are several intrusion detection systems that have been introduced for WSNs as well as IP-based networks. But most of them have not proper performance for IP-based WSNs such as 6LoWPAN. Roughly speaking, since the number of IDSs that were specifically designed for IoT is limited, there are few solutions for overcoming the security challenges of IoT. For instance, despite the efficiency of cryptography in the security mechanism, it is necessary to develop a specific IDS for IoT for confronting with the network or machine threats; therefore, employing an efficient intrusion detection technique for the Internet of Things is a hot research area.

1.2 Goal and Research Scope

Today, 6LoWPAN is still known as a new approach in computer networks that has a high capacity for researching. The purpose of this thesis is providing a novel method for intrusion detection in 6LoWPAN as a real IoT. For achieving this goal, the main features of 6LoWPAN should be considered to provide a suitable structure for analyzing IoT's behaviors as compared to the other kinds of networks' behaviors. Because the 6LoWPANs consist of a hybrid of IEEE 802.15.4 and IPv6 networks, the traffic patterns of them are different in comparison with to the regular networks. Indeed, the traditional solutions that were used for detecting intrusions in regular networks cannot be useful for IoT. Therefore, the IDSs for 6LoWPAN should be designed and developed regarding various traffic patterns of WSNs and IP-based networks.

Roughly speaking, every layer of 6LoWPAN stack can be threatened by diverse attacks. The routing attacks threaten the networks by controlling the information flow of networks. Hence, most of the attacks focus on sabotaging the network layer through routing attacks. Note the 6LoWPANs use a specific routing protocol named RPL[2] for routing the packets. This protocol is a distance-vector protocol introduced and standardized by the ROLL[3] workgroup as a private routing protocol for 6LoWPAN. On the other hand, because the 6LoWPAN network is WSN that its objects can be identified by the IP addresses, the 6LoWPAN network can be threatened by every network layer attack, especially DoS attacks (Le, Loo, Lasebae, Aiash, & Luo, 2012).

As mentioned earlier, due to the specific architecture of Internet of Things, this network can be attacked from all the Internet and 6LoWPAN threats. In other words, IoT can be threatened by cyber (external) attacks and insider (internal) attacks, which are raised by the Internet and 6LoWPAN, respectively. Therefore, to obtain a security mechanism in multi-faceted detection, protecting IoT against external and internal threats is very important (Shahid, 2013). Generally, the main goal of this thesis is proposing an effective method for intrusion detection and identification of attacks (i.e., cyber-attacks and insider-attacks) that can be occurred in IoT. In fact, this thesis essentially follows two main objectives:

1) Cyber-attack identification: As can be seen in Figure 1-1, in IoT, the IP-based WSNs such as 6LoWPAN can connect to the Internet or other LANs[4] through 6BR. Therefore, the objects of these kinds of networks can be threatened by cyber-attacks of the Internet. For example, in R2L[5] attacks, an end-user on the Internet can illegally access to the private information of 6LoWPAN's objects. Hence, the proposed intrusion detection method should be able to detect some kind of attacks that can be threatened IoT by the Internet (or LANs).
2) Insider-attack identification: Another goal of this thesis is to provide the ability to detect internal attacks of IoT. Note this thesis focuses on detecting selective forwarding and sinkhole as two well-known DoS attacks in WSNs, which have a great malicious effect on the performance of 6LoWPAN.

Generally, the following items will be considered in the proposed approach:

- Proper efficiency: Because the most objects of 6LoWPAN are constrained (especially in terms of processor and memory), the proposed method should have a reasonable computational and memory overhead.
- The ability to detect new attacks: As mentioned earlier, in the IoT approach, because the heterogeneous objects can connect to the unreliable Internet, the possibility of anonymous

[2] Routing Protocol Low Power and Lossy Networks
[3] Routing Over Low power and Lossy networks
[4] Local Area Networks
[5] Remote to Local

attacks occurring in this kind of network is very high. Therefore, the proposed intrusion detection system should be able to detect new attacks.

- The performance of IDS in terms of DR[6] and FAR[7] is an important concern that should be considered in every kind of IDSs. Therefore, the proposed methods should have high DR and low FAR as much as possible.

1.3 The key Innovations

As can be seen in the thesis's title, this study focuses on employing proper machine learning techniques for detecting various threats that can occur in IoT. To the best knowledge of the author, this approach is one of the pioneer studies that tries to employ the idea of machine learning methods in IoT security for proposing a proper IDS that meets the IoT's specification. In other words, employing the machine learning methods for detecting intrusion in IoT, is surely a novel approach in IoT security. In sum, for presenting the various aspects of the thesis's innovations, Figure 1-2 depicts the general schema of the proposed model. The main contribution of this thesis which will be introduced in Chapter 3, includes the following sections:

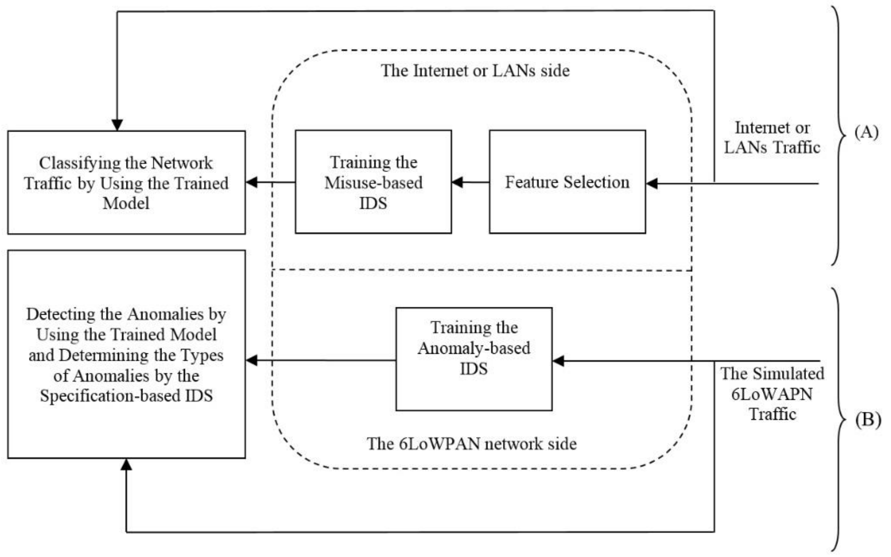

The Internet or LANs side
Classifying the Network Traffic by Using the Trained Model
Training the Misuse-based IDS
Feature Selection
Internet or LANs Traffic
(A)
Detecting the Anomalies by Using the Trained Model and Determining the Types of Anomalies by the Specification-based IDS
Training the Anomaly-based IDS
The Simulated 6LoWAPN Traffic
(B)
The 6LoWPAN network side

Figure 1-2: The general schema of the proposed IDS

As can be seen in Figure 1-2, for achieving the main goals of thesis, the proposed model consists of two components: (a) an IDS for detecting cyber-attacks raised by the Internet or LANs and (b) another IDS for detecting insider-attacks occurred by 6LoWPAN. According to the aforementioned components, the proposed IDS includes the following parts:

1) Feature selection: In this thesis, a novel feature selection method which is based on Binary Gravitational Search Algorithm (BGSA) and Mutual Information (MI) is presented for selecting the optimal feature set of the instances extracted from the Internet or LANs traffic.
2) An offline misuse-based IDS: For detecting and identifying intrusions raised by the Internet or LANs, a novel approach that is based on the supervised Optimum-path Forest (OPF) is introduced.

[6] Detection Rate
[7] False Alarm Rate

3) A real-time anomaly-based and specification-based IDS: To detect the anomalous behaviors of 6LoWPAN and as well as identifying the type of anomalies, this thesis proposes a real-time hybrid method based on anomaly-based and specification-based IDSs. Note the presented anomaly-based IDS is an anomaly detection engine based on the unsupervised OPF that is modified regarding the Map-Reduce architecture. Moreover, the proposed specification-based IDS is used to ensure whether the detected anomalies are attacks or not. Furthermore, if the detected anomalies are attacks, the types of them can be determined by the proposed specification-based IDS.

1.4 The Structure of Thesis

This thesis has been written in five chapters as follows:

Chapter 1: This chapter reviews the general issues of the topic in terms of the problem statement, the goal and research scope, as well as the aspects of innovation.

Chapter 2: The basic concepts and the related works are studied in this chapter. Indeed, the foundations of some of the technical principles (e.g., explaining the possible external and internal attacks that can occur in IoT, reviewing the RPL, introducing the IDSs and machine learning methods) employed in the thesis will be introduced. Moreover, at the last of this chapter, the related works are reviewed, briefly.

Chapter 3: In this chapter, the presented framework used for detecting and identifying the types of possible cyber-attacks and insider-attacks of IoT will be studied in detail. Generally, Chapter 3 includes two main sections: (1) the proposed offline misuse-based IDS and (2) the proposed real-time anomaly-based and specification-based IDS that are used for dealing with the cyber-attacks and insider-attacks of IoT, respectively. In the first section, a novel feature selection algorithm used in the preprocessing step is presented in details. Then, an offline misuse-based IDS which is proposed for detecting the possible external threats of IoT is introduced, completely. Next, the second part of this chapter presents a real-time hybrid of an anomaly-based and specification-based IDS used for detecting the possible internal threats of IoT.

Chapter 4: This chapter focuses on the evaluation of the proposed method. Therefore, at first, the standard metrics used to evaluate the proposed IDS are introduced, briefly. Then, the proposed feature selection method and misuse-based IDS are evaluated based on the presented metrics. Next, three IP-based WSNs that use the RPL as routing protocol are simulated for evaluating the efficiency of the proposed method in detecting sinkhole and selective forwarding attacks as two famous insider-attacks of IoT.

Chapter 5: This chapter reviews and summarizes the thesis. Moreover, the limitations of the research and future work will be reviewed, briefly.

Chapter 2: Background and Survey

Internet of Things is a novel approach in computer networks that has caused a revolution in telecommunications. Every identifiable physical object of IoT can send data to or receive data from the Internet or other physical objects. 6LoWPAN as the main effort to make the concept of real IoT is an IP-based WSNs that can connect the smart physical objects to the Internet through wireless communications. Notice 6LoWPAN uses RPL and IPv6 protocols for routing the packets in a low-power and lossy network. As mentioned in Chapter 1, security in IoT is one of the main challenges that cannot be obtained just by using encryption or access control. Nowadays, the intrusion detection systems are one of the main and effective solutions that can be used to overcome the security threats in computer networks, especially IoT.

Because this thesis specifically focuses on proposing an IDS, which uses a hybrid of machine learning methods for detecting and identifying the possible attacks of IoT, this chapter studies the possible external and internal threats of IoT, firstly. Then the intrusion detection systems and machine learning are reviewed, completely. Moreover, the application of machine learning algorithms in IDSs is also considered. On the other hand, sinkhole and selective forwarding attacks as two dangerous threats of IoT, and RPL (routing protocol of 6LoWPAN) are also reviewed, briefly.

2.1 The External Threats of IoT (Cyber-Attacks)

Generally, the majority of security researchers who work in the area of intrusion detection have used the KDD’99 as a benchmark dataset for evaluating their methods. In 1999, the MIT Lincoln Laboratory, which was supported by Defense Advanced Research Projects Agency and Air Force Research Laboratory, started to provide and deliver the KDD’99 dataset for evaluating the IDSs in computer networks (Ibrahim, Basheer, & Mahmod, 2013). Indeed, they simulated a fictitious military network including three machines that run various operating systems and services in seven weeks in order to obtain this dataset (Lahre, Dhar, Suresh, Kashyap, & Agrawal, 2013). They also used an additional three machines and a sniffer for spoofing various IP addresses and storing traffic in TCP dump format, respectively (Lahre et al., 2013). It should be realized, the KDD'99 was obtained by employing the data mining methods introduced in the KDD[8] process over the aforementioned simulated traffics.

Generally, the simulated attacks in KDD'99 dataset, which are used in this thesis as cyber-attacks of IoT can be classified into the following four categories (Lahre et al., 2013):

1) **DoS[9] attacks:** In these kinds of attacks, attackers try to busy services of a host on the Internet in order to prevent its accessibilities.
2) **Prob attacks:** In these kinds of attacks, perpetrators attempt to gather different information about a target host or a network. Indeed, their preferred information is some useful

[8] Knowledge Discovery in Databases
[9] Denial of Service

information that can be used to cope with the security mechanisms of a host or a network, and consequently, it leads to a host or network vulnerability in front of security threats.

3) **U2R[10] attacks:** These kinds of attacks include some attacks that can locally access the victim machines in order to gain higher account privileges.
4) **R2L attacks:** These kinds of attacks try to access the victim machines without having the required permissions (user accounts).

2.2 The Internal Threats of IoT (Insider-Attacks)

As mentioned in Chapter 1, the focus of the majority of insider-attacks in 6LoWPAN is threatening the network layer through routing attacks, especially selective forwarding and sinkhole attacks. Since the 6LoWPANs use RPL as a standard routing protocol, thus RPL and its functionality will be considered before reviewing the mentioned attacks.

2.2.1 The IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL)

The RPL, which is based on the construction of a Destination-Oriented Directed Acyclic Graph (DODAG), is an IP-based distance vector and hop-by-hop routing protocol that is designed by the ROLL workgroup (which is a workgroup in IETF) to overcome the routing problems in the LLNs. RPL enables one-to-one, one-to-many, and many-to-many communication traffic by supporting different operations such as the unidirectional traffic towards a DODAG root, bidirectional traffic between resource-constrained devices (i.e., 6LoWPAN nodes), and bidirectional traffic between resource-constrained devices and the DODAG root (Wallgren et al., 2013). According to the DODAG architecture, the nodes are organized in a hierarchical tree structure and routed at a single root, as the destination and called 6BR, to avoid creating any network loop. The 6BR, which connects 6LoWPAN to the Internet through the backbone, is the root of DODAG and is responsible for the management of nodes. Figure 2-1 shows a typical RPL which consists of different 6LoWPAN nodes (i.e., source, router, and root nodes) that are connected together based on the DODAG topology. Each node has an ID based on an IPv6 address, a special rank, a set of neighbors, and one (or more) parent(s). The rank of each node determines the relative position of that node with respect to the DODAG root. This rank is strictly increasing in the top-down direction from the DODAG root toward the leaf nodes, and in opposition decreasing in the bottom-up direction toward the DODAG root (Wallgren et al., 2013). The nodes in a DODAG use an objective function, represented by an Objective Code Point (OCP), based on some optimization criteria (such as link reliability, latency, hop-count, and node energy) to optimize the paths toward the DODAG root (Le et al., 2012). A single 6LoWPAN network may include multiple RPL instances that work concurrently with different optimization criteria, where each instance consists of one (or more) DODAG(s). Hence, a 6LoWPAN node can belong to more than one DODAG in an RPL instance (Winter et al., 2012). To prevent probable loops created on the network, the message transmission is based on the rank rule (that means the node ranks are strictly decreasing along with the upstream transmission and vice versa) (Wallgren et al., 2013).

[10] User to Remote

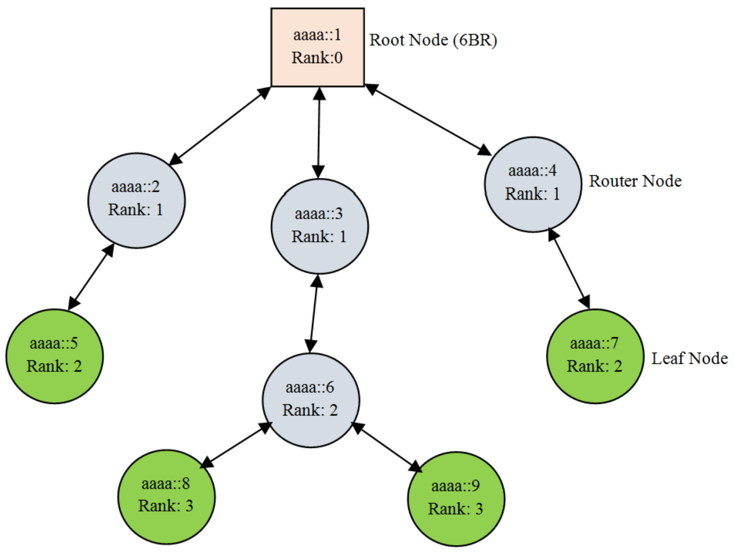


Figure 2-1: DODAG scheme including nodes with unique ranks and IPv6 addresses

2.2.1.1 Control Messages of RPL

To exchange routing graph information for RPL operations, such as constructing DODAG, the RPL defines three new ICMPv6[11] message types: a) DAG Information Object (DIO); b) DAG Information Solicitation (DIS); and c) Destination Advertisement Object (DAO). DIO messages carry information which is used for DODAG construction by allowing a node to determine parents and selecting the best one as the preferred parent. The DIS messages are used to solicit graph-related information (i.e., DODAG information object) from the neighbor nodes (Wallgren et al., 2013). The RPL supports downward traffic toward the leaf nodes by using DAO messages which advertise required information and also propagate destination information upwards along the DODAG (Winter et al., 2012).

2.2.1.2 DODAG Construction

In the process of DODAG construction, DIO messages (which contain important information such as rank, DAG-ID, and OCP) are periodically broadcasted by the DODAG root (Le et al., 2012). The nodes that receive DIO are considered as the neighbors of the DODAG root. They use DIO message information to join the DODAG and select the DODAG root as the parent. According to a specific objective function (such as min-hop), the neighbors assign their rank to 1 (the parent's rank that is 0; is incremented by 1) and start broadcasting their own DIO. When a node receives a DIO message, it calculates its rank from the OCP specified in a received DIO and forms a list of parents. Then, according to the OCP, the preferred parent is selected from the parent-list and broadcasts its own DIO. This procedure will be continued until the topology construction is completed (i.e., the best path to the DODAG root is identified for each node according to the objective function). To handle inconsistencies in the DODAG, RPL uses a trickle timer for determining the transmission rate of DIO messages. In a network with stable topology, the trickle timer interval is large, so the DIO messages will be rare. When the inconsistencies occur, the trickle timer will be reset and more DIO messages are sent from the nodes that cause inconsistencies (Wallgren et al., 2013).

[11] Internet Control Message Protocol version 6

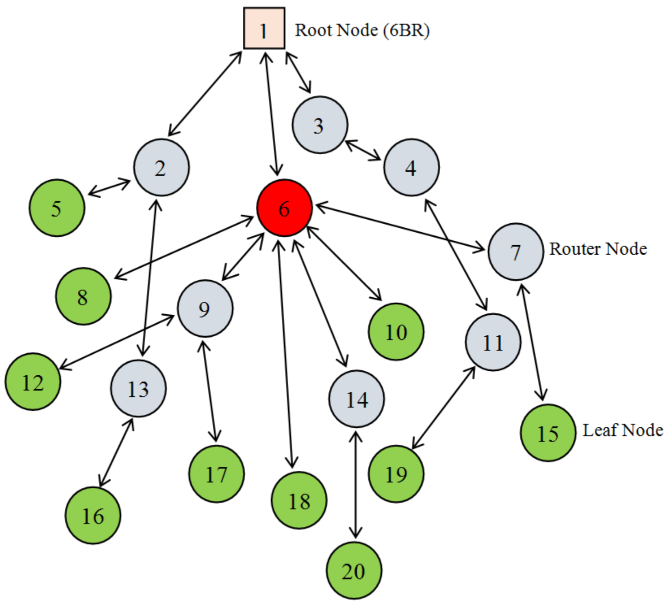


Figure 2-2: A screenshot of a sinkhole attack launched by node 6 in DODAG as a malicious node

2.2.2 Selective forwarding and Sinkhole Attacks

Among diverse attacks that can threat RPL, DoS as a devastating kind of attack has undesirable effects on 6LoWPAN accessibility. In computer networks, any event that details the services of machines (nodes) by disturbing the communications among them, is known as DoS (Kasinathan, Pastrone, Spirito, & Vinkovits, 2013). DoS is a popular kind of attacks, which can be created simply. But, because they may have various forms, the identification process of them is so hard (Rghioui et al., 2014). Selective forwarding and sinkhole attacks are well-knows routing attacks that try to disturb the routing process in WSN. In selective forwarding attacks, which primarily disrupt the routing path, malicious nodes selectively forward packets in order to remove some packets based on the importance of data or randomly (Shahid, 2013). For example, a malicious node forwards only routing messages and removes other packets for disrupting the network (Shahid, 2013). In sinkhole attacks, a malicious node represents itself to others as an optimal routing path for attracting nearby nodes to route traffic through it. As shown in Figure 2-2, in RPL, an intruder launches a sinkhole attack by propagating its rank as a better rank to make nodes down in the DODAG by selecting it as a preferred parent (Shahid, 2013). The sinkhole attacks may not necessarily threaten the network; however, they make serious problems when they couple with other attacks such as selective forwarding attack.

2.3 Intrusion Detection Systems

IDS is an effective tool or mechanism which gathers network traffic as input data for detecting intruders or malicious behaviors that are trying to threaten the network. Actually, IDS monitors the network traffic or system functionalities in order to detect the signs of illegal behaviors. To detect malicious behaviors, IDS filters the obtained data for determining the attack patterns. Generally, the main approach in IDSs is gathering and analyzing the data related to the network traffic or systems behaviors to identify threats and produce warnings (Le et al., 2012). As mentioned in Chapter 1, there are various metrics used for evaluating the intrusion detection systems; however, DR and FAR are two popular metrics, which are broadly used in evaluating and comparing different methods of intrusion detection. DR is the ratio between the number of correctly detected attacks and the total number of attacks that occurred in a network. While FAR is the ratio between the number of incorrectly detected attacks and the total number of normal connections in the obtained data. These metrics will be introduced in Chapter 4, accurately.

Today, the security scientists in computer networks classify IDSs into different categories based on the analysis methods, data sources, and system architectures.

2.3.1 IDS Classification Based on the Analysis Methods

Depending on the analysis methods, the computer security community has classified IDSs into three main categories: (a) misuse detection; (b) anomaly detection; and (c) specification-based systems. In the misuse detection systems, the predefined attack patterns are profiled in a signature database as a reference of intrusion patterns to match against system behavior or network traffic for detecting intrusions (S.-Y. Wu & Yen, 2009). The misuse detection techniques are simple to use; however, specific knowledge of each attack is required and consequently, unknown abnormalities are not detectable. On the other hand, all of the known attacks can be detected with low False Alarm Rate (FAR), and the storage costs grow with the number of attacks because a signature of each known attack should be stored (Kasinathan et al., 2013). Generally, misuse detection is not a suitable method for detecting intrusions in WSNs or 6LoWPANs. Because the system needs to have enough knowledge about all kinds of possible attacks in order to define the attack patterns. In addition, increasing the number of attacks leads to grow the size of storage, while the storage size of WSNs and 6LoWPANs are limited. Moreover, 6LowPANs need the ability to detect new attacks, while the misuse detection methods skip it (Le et al., 2012).

An anomaly detection system focuses on normal system behaviors or network traffic. In other words, in this method, the system concentrates on classifying the normal behavior of a network or host system. Actually, to anomaly detection, the obtained data should be analyzed for determining the patterns of anomalies. The anomaly detection systems build usually one the following models as a baseline for describing ordinary behavior: (a) statistical; (b) knowledge-based; and (c) machine-learning (Golmah, 2014). In the observed data, any deviation from this model can be considered as an anomaly. Generally, there are two main stages in anomaly detection: (1) the construction stage in which a profile of the normal traffic or system behaviors will be built by employing the statistical, knowledge-based, or machine-learning algorithms; (2) the detection stage in which the current traffic or system behaviors are compared with the predefined profile (Stakhanova, Basu, & Wong, 2010). The anomaly detection algorithms are useful for new intrusions; however, they suffer from a high rate of false positive (unlike, the misuse detection models).

The specification-based systems work in the same way, as well. However, user guidance is required to extract legitimate system behavior or network traffic for developing a model of normal behavior in these systems in addition to employing statistical, knowledge-based, or machine learning techniques (Sekar et al., 2002). Note applying this method can be more accurate than the anomaly detection method; however, it is prone to error because of relying on user expertise (Sekar et al., 2002).

2.3.2 IDS Classification Based on the Data Sources

According to the data sources, IDSs take one of the following approaches for recognizing attacks: (a) host-based; (b) network-based; and (c) hybrid (Zhang, Feng, & Qin, 2013). In the host-based approach, only the host events are considered for detecting attacks. Therefore, the data for host-based IDS is provided by different activities of hosts such as audit records of an operating system and system logs; however, in the network-based IDS, data is mainly collected from the network segments such as Internet packets (S.-Y. Wu & Yen, 2009). In the hybrid approach, data provided by the host events and the network segments are considered in developing an IDS.

2.3.3 IDS Classification Based on the System Architecture

The system architecture of IDS has a great effect on the performance of IDS in the WSNs such as 6LoWPAN. According to the system architecture, IDSs in the RPL-based WSNs are classified into four main categories (Zhang et al., 2013): (a) stand-alone; (b) distributed and cooperative; (c) distributed and hierarchical; and (d) mobile agent. Some nodes which have Low-power and Lossy Networks (LLNs) devices or high-performance devices called Monitor Nodes (MNs) are used for monitoring the events in WSNs (Zhang et al., 2013). In the stand-alone architecture, each MN performs intrusion detection based on its own collected data, independently. The MNs in the stand-alone IDS are classified into two schemes: (1) centralized and (2) distributed. In the centralized scheme, each node is considered as an MN, and in the distributed scheme, multiple MNs are deployed on a WSN to cover the network (Zhang et al., 2013). In the distributed and cooperative architecture, intrusion detection is accomplished by the cooperation of MNs such that each MN performs as an IDS agent and participates in intrusion detection (Zhang et al., 2013). In this approach, IDS is applied as a local-agent or a neighbor-agent to a two-level coordinate architecture, where a local-agent can alert intrusion independently by detecting a threat with sufficient evidence. However, when a local-agent detects intrusion with weak evidence, it starts a cooperative detection procedure in an interaction with the neighbor-agents for global detection (Zhang et al., 2013). This kind of architecture is suitable for small-scale and flat network infrastructures; however, in a large-scale network, the distributed and hierarchical architecture is adequate for detecting an intrusion (Zhang et al., 2013). According to this approach, the network is partitioned into some clusters with a sink node as a Cluster Head (CH). The IDS in a distributed and hierarchical architecture is composed of two levels. At the first level, CH-agents, which are responsible to monitor the covered nodes and make the global intrusion detection decisions, are embedded in the sink nodes. At the second level, the local-agents, which are designed based on the stand-alone IDS, are deployed in each cluster to report the detection results to the CH-agents (Zhang et al., 2013). The last-mentioned IDS architecture in WSNs was the mobile agent. The mobile agent, as a self-controlling program segment, is a specific executable code that traverses from a node to another one (Hamedheidari & Rafeh, 2013). In this agent migration, which means moving an agent from a node to another selected node, the computation is performed in addition to data transmission (Hamedheidari & Rafeh, 2013). The mobile agents are assigned to the selected nodes for performing a monitoring task and intrusion detection (Zhang et al., 2013). The selection of agents may be changed after a certain period of time or after the task is completed. By moving the processing function to the data instead of bringing the data to a central processor, the mobile agents can greatly reduce the communication cost in the links with low bandwidths.

2.4 Machine Learning and its Applications in IDS

With the fast growth of computer networks causing attacks to become more sophisticated, detecting intrusions based on the usual method is impossible. Machine learning (MI) as the most effective soft computing approach is widely used in intrusion detection systems due to its proper results. Actually, different studies show that using machine learning algorithms in the area of intrusion detection can conclude high DR and low FAR while reducing computation and communication overheads (Zamani & Movahedi, 2013).

Roughly speaking, machine learning studies on some algorithms that are able to construct a learning model on the given input datasets to provide the ability of prediction. The input dataset consists of training and testing datasets used to train the learning model and evaluate the trained model, respectively. There are three major types of learning algorithm named supervised, unsupervised and semi-supervised learning. In mathematical terms, the supervised learning attempts to find a proper function by employing the labeled data samples. Actually, this decision function can map the unlabeled data samples to the corresponding class. However, because of the lack of labeled data samples, the unsupervised learning partitions data samples into different clusters based on their similarities. It is noted that supervised and unsupervised learning algorithms are usually used in misuse-based and anomaly-based intrusion detection systems. Semi-supervised learning is a hybrid method in machine learning that usually uses small labeled data samples with a great volume of the unlabeled data samples for learning.

The most prominent approaches of ML-based intrusion detection systems can be divided into artificial intelligence (AI) and computational intelligence (CI) (Zamani & Movahedi, 2013). AI-based techniques support symbolic knowledge representation, which points out a classic realm of AI such as statistical models, while CI-based techniques can handle the numeric representation of information (Zamani & Movahedi, 2013). Actually, these kinds of methods refer to nature-based techniques such as evolutionary computation, fuzzy logic, artificial neural networks, and artificial immune systems. Both of AI- and CI-based approaches can offer suitable intrusion detection models. However, CI-based techniques fit the requirement of efficient IDSs due to their proper characteristics such as high computational speed, fault tolerance, adaptation, and resilience in front of noisy information (S. X. Wu & Banzhaf, 2010).

In ML-based intrusion detection systems, at first, a training dataset includes labeled or unlabeled data samples will be created through the collected data relating to the system behaviors or network traffic. Then the target learning algorithm trains the learning model (training phase). After the training phase, the obtained model can classify or cluster new data samples. Note in classification approaches, some classes which are known as attacks are clear while in clustering approaches some clusters that include fewer samples are assumed as an anomaly since the number of attack samples is usually fewer than the normal samples. Indeed, the anomaly detection engine will label a sample as anomalous if it belongs to the pointed small clusters.

2.5 Related Work

Since the 1970s to the present, there has been a lot of research on intrusion detection systems. In 1980, James Anderson (Anderson, 1980) introduced the concept of intrusion detection for the first time. In his paper, he presented some methods for the security surveillance of computer systems. From 1984 to 1986 the studies of Denning and Neumann (Denning & Neumann, 1985) in the security area of computer systems concluded an IDS based on expert systems called IDES[12], which could detect intrusion in real-time. In this project, the misuse-based and anomaly-based approaches were used in a hybrid way.

Generally, the studies on intrusion detection systems can be reviewed from different aspects such as the methods used in IDSs, the benchmark datasets used in IDSs, and the type of networks. The traditional IDSs (e.g., SNORT and Bro) were based on misuse detection methods, although these kinds of techniques cannot identify new attacks. Over the last two decades, anomaly detection methods were the main approach of the security communities due to the great capabilities of these kinds of methods in detecting diverse threats. For example, Anuar et al. (Anuar, Sallehudin, Gani, & Zakaria, 2008) presented a hybrid statistical strategy based on the data mining and decision tree classification in order to analyze the normal and attack traffic, statistically. As another example, Mabu et al. (Mabu, Chen, Lu, Shimada, & Hirasawa, 2011) proposed a fuzzy class-association-rule mining method for detecting the intrusions. Their method, which was based on genetic network programming, employed directed graph structures instead of a string genetic algorithm in order to extract many important class-association rules.

As mentioned earlier, from the aspect of analysis methods, machine learning algorithms are the most efficient techniques that are broadly engaged in various kinds of IDSs. Kim et al. (Kim, Lee, & Kim, 2014) proposed a hybrid intrusion detection method that integrated hierarchically a misuse detection model based on the C4.5 decision tree algorithm and an anomaly detection model based on multiple one-class Support Vector Machines (SVMs). The experimental results showed improvement in the IDS performance in terms of unknown-attacks detection and speed of detection. Wang et al. (G. Wang, Hao, Ma, & Huang, 2010) proposed an approach based on artificial neural networks (ANNs) and fuzzy clustering, called FC-ANN, for achieving high DR and low false positive rate. The fuzzy clustering technique was used for generating different training subsets. These subsets were employed for training the ANNs. The result of simulation experiments on the KDD'99 dataset showed superior performance as compared to the back-propagation neural network (BPNN) and other well-known methods such as decision tree and naïve Bayes. Sheikhan and Khalili (Sheikhan & Khalili, 2010) proposed a rule extraction module based on a dynamic cell structure (DCS) neural network and a modified version of the LERX algorithm. Their experimental results have shown the superiority of the presented method as compared to the multi-layer perceptron (MLP) in identifying the hard-detectable attacks.

As discussed previously, the majority of researchers in the security communities who work in the area of intrusion detection used the KDD’99 as a benchmark dataset for evaluating their methods. Because the KDD’99 dataset has some problems like having redundant and duplicate records that will cause negative effects on the evaluation result when being used as an evaluation dataset, Tavallaee et al. (Tavallaee, Bagheri, Lu, & Ghorbani, 2009) introduced a new dataset named NSL-KDD dataset that is a revised version of KDD’99 consisting of selected records of the complete original KDD’99 dataset. It is noted that some researchers such as Raza et al. (Raza, Wallgren, & Voigt, 2013) used the simulated traffic rather than employing the offline benchmark datasets for evaluating their proposed IDSs.

[12] Intrusion Detection Expert System

There are diverse studies on intrusion detection that focus on the type of networks. For instance, Zhijie and Ruchuang (Zhijie & Ruchuang, 2012) presented a new anomaly detection method based on a Markov model in order to predict the traffic of large-scale wireless networks. Indeed, they proposed their technique for detecting some kind of attacks such as selective forwarding attacks which make more influence on packet traffic. The experimental results of their method had higher DR, lower FAR, and lower computation and communication overheads in comparison with other techniques.

During the last years, there are few studies that focus on presenting a dedicate IDS for the Internet of Things. For example, Liu et al. (C. Liu, Yang, Chen, Zhang, & Zeng, 2011) proposed an IDS based on the Artificial Immune System (AIS) in order to identify intrusions in IoT. In this method, they defined IDS and IoT environment in their AIS, firstly. Then the self-adapting and self-learning mechanisms were simulated. They succeeded to show the proper performance of the proposed method in detecting intrusions of IoT. Amin et al. (Amin, jig Yoon, Siddiqui, & Hong, 2009) proposed an IDS for IP based Ubiquitous Sensor Network (IP-USN). Similar to 6LoWPAN, IP-USN is another effort to make the concept of real IoT. In their research, they studied three possible attacks (i.e., attacks from the Internet hosts, within sensor networks, and on the Internet clients) which may happen in IP-USN. In their method, the proposed IDS has resided on the IP-USN gateway. Their presented method included two main components called Internet Packet Analyzer (IPA) and USN Packet Analyzer (UNA). Moreover, IPA consisted of two modules for anomaly detection and pattern classification. The anomaly detection module was in charge of identifying anomalous traffic. The UNA component employed a slave node (surveillance nodes) to generate and send warning messages toward the master node. The master node was responsible for making decisions and enforcing policy within its domain. So the kind of node could detect intrusions in the network. Indeed, they took a bottom-up approach that began from reviewing the traditional attacks of WSNs and continued toward the possible attack scenario in IP-USN for designing a dedicated IDS. Kasinathan et al. (Kasinathan et al., 2013) introduced a Denial of Service (DoS) attacks detection architecture for 6LoWPAN. The proposed IDS was immune against DoS attacks. In addition, it was applicable and scalable in the real IoT applications. They integrated an IDS into a framework that was developed in the European Business-Based Internet of Things and Services project named Ebbits. The proposed method used an IDS probe to listen to the 6LoWPAN network traffic. Their simulation results showed the capability of the proposed architecture in detecting DoS attacks. Moreover, because the proposed IDS was hosted on a powerful Linux host in the Ebbits network, it could overcome the concerns of resource constraints in traditional low-power devices. Le et al. (Le, Loo, Luo, & Lasebae, 2011) worked on security aspects of RPL by introducing a specification-based IDS for detecting a new type of threat called topology attack. This type of attack, which was originally applied to RPL, changes the node operation for breaking the optimized network topology. The experimental results showed effective detection of RPL topology attacks with reasonable overhead.

Perhaps the most important research that has already been done in designing an appropriative intrusion detection system for IoT is the SVELTE project. Indeed, in this project, Raza et al. (Raza et al., 2013) proposed a novel real-time IDS for IoT called SVELTE. They implemented the proposed model in the Contiki operating system and targeted only routing attacks such as sinkhole and selective forwarding. Their proposed system consisted of three main centralized modules which were located in 6BR. The first module called 6Mapper was responsible to gather network data and construct a DODAG of the network in 6BR. The second module analyzed mapped data in order to detect intrusions. This module could identify attacks through (1) detecting inconsistency in obtained

DODAG, (2) checking the availability of the network nodes, and (3) validating the routing graph. The third module was a distributed mini-firewall which was designed to filter malicious traffic before it reaches the resource-constrained devices. They showed through simulated scenarios that SVELTE has a small overhead to deploy on the constrained nodes and can detect most of the malicious nodes that launch sinkhole and/or selective forwarding attacks.

Chapter 3: The Proposed Intrusion Detection System

Since there is not a considerable study in the IoT security that uses the machine learning approaches to detect intrusions, the main purpose of the proposed model is to use the machine learning algorithms in order to present an ML-based intrusion detection technique for identifying security anomalies in the Internet of Things. This chapter introduces the proposed model by describing the different applied approaches. As mentioned in Chapter 1, the proposed model considers both cyber-attacks and insider-attacks, which may be launched by the Internet and 6LoWPAN, respectively. Therefore, as shown in Figure 1-2, the proposed model consists of two main components to confront with the Internet and 6LoWPAN threats.

3.1 Cyber-Attack Identification

Generally, resource-constrained nodes of 6LoWPANs can be attacked by diverse cyber-attacks on the Internet (or LANs) because the 6LoWPAN can connect to the Internet (or LANs) through 6BR. In the proposed model, the detection of cyber-attacks will be done offline by employing a novel misuse-based intrusion detection engine. As shown in Figure 1-2, the proposed component comprises of two following sequential phases:

3.1.1 Preprocessing

Intrusion detection systems usually face with the high-dimensional large-scale data including irrelevant or redundant features. These kinds of features lead to reduce the speed of the detection process and increase resource consumption. For example, in ML-based intrusion detection techniques, every feature of the training data may not be relevant to the detection task, and in the worst case, irrelevant features may introduce noise and redundancy into the design of classifiers. So, choosing a good subset of features will be critical to improving the performance of classifiers (Pei, Goodman, & Punch, 1998). In other words, to reduce the computational complexity and increase the accuracy of these methods, feature selection is needed in a preprocessing stage, because the number of features which is used in an IDS is usually large and feature selection algorithms can identify the most relevant ones. Thus, feature selection methods are used to determine the best minimal set of features that does not contain redundant features. Feature selection is useful to avoid the over-fitting problem, improving the performance of classification models, reducing the training and testing time, improving stability against noise, and reducing the measurement and storage requirements (Kumar & Kumar, 2012). Because the proposed method uses a high-dimensional NSL-KDD dataset as the simulated traffic of the Internet side, this thesis provides a novel hybrid method based on Binary Gravitational Search Algorithm (BGSA) and Mutual Information (MI) to reduce the dataset's dimension. Before introducing the proposed method, some of the fundamental concepts used in the proposed technique will be reviewed.

3.1.1.1 Feature Selection Problem

Suppose that $T = (D, F, C)$ is a dataset with *n* instances, *m* dimensions and *k* labels or target classes where $D = \{o_1, o_2, \dots o_n\}$, $F = \{f_1, f_2, \dots, f_m\}$ and $C = \{c_1, c_2, \dots, c_k\}$ are the sets of instances, features, and labels, respectively. The goal of machine learning is to provide a model $h : F \rightarrow C$ that maps the input feature space *F* onto the class space *C*, and it would be better if the model *h* contained those input features that contribute much more information to the distribution of the classes (H. Liu, Wu, & Zhang, 2014). Feature subset selection problem refers to finding an optimal feature subset like F' with d member of the whole feature space where $F' \subseteq F$ and $d < m$, and it can lead to the best possible accuracy in the classification or more clearly in optimizing a criterion function. The number of suitable features is not predictable, so all feature subsets should be evaluated to find an optimal feature subset. Let the number of features in the dataset be *n*, so the number of the feature subsets will be $2^n - 1$. When *n* is large, it is impossible to evaluate all feature subsets, therefore finding an optimum feature subset is usually NP-hard (Deisy, Baskar, Ramraj, Koori, & Jeevanandam, 2010).

3.1.1.2 Feature Selection Techniques

The feature selection method has two main steps: (a) the generation procedure that finds an optimal feature subset, (b) an evaluation procedure to measure the optimality of the generating subset of features by using an evaluation criterion (Deisy et al., 2010). Depending on feature subset evaluation, the feature selection methods can be classified into the filter, wrapper, and embedded methods (Kumar & Kumar, 2012). The filter methods use general characteristics of the training data to evaluate features and operate independent of learning algorithms, so these methods are appropriate for processing high-dimensional data (Hoque, Bhattacharyya, & Kalita, 2014). Wrapper methods use learning algorithms to evaluate the feature subset selection, and this means that they use classification feedbacks for feature selection. These methods have improved the accuracy of classification with a high computational-complexity (Hoque et al., 2014). Therefore, they are intractable for large-scale problems. In the embedded approach, feature selection is a part of the training phase in machine learning, so the feature selection is specific to the applied learning algorithms (Hoque et al., 2014). Another approach to feature selection is based on hybrid methods. This approach is a combination of filter and wrapper-based methods. The filter method selects the candidate features which are refined by the wrapper part (Hoque et al., 2014).

3.1.1.3 Binary Gravitational Search Algorithm

Finding an optimum feature subset depends on the search strategy. The search strategy is classified into exhaustive, heuristic and random searches and is combined with several types of measures to form different algorithms (Ruiz, Riquelme, & Aguilar-Ruiz, 2005). The time complexity is exponential in terms of data dimensionality for exhaustive search and quadratic for heuristic search. Hence, heuristic search algorithms have been widely used for feature selection. BGSA is an efficient population-based heuristic search that can be used in various optimization problems. According to the Newtonian gravitational law, each particle attracts other particles with a "gravitational force" which causes a movement of all objects globally toward the objects with heavier masses. This force is directly proportional to masses of particles and inversely proportional to the square of the distance between them (Rashedi, Nezamabadi-Pour, & Saryazdi, 2009):

$$F = G\frac{M_1 M_2}{R^2} \tag{3-1}$$

where M_1 and M_2 are the masses of particles, and R is the distance between them. Based on Eq. (3-1), the gravitational force between large particles which are closer will be large. G is the gravitational constant and its actual value depends on the age of the universe (Rashedi et al., 2009):

$$G(t) = G_0\left(1-\frac{t}{t_{max}}\right) \tag{3-2}$$

where t_{max} is the total number of iterations. G_0 is the value of the gravitational constant at the beginning. Computer science uses the Newtonian gravitational law in optimization problems such as some random candidate solutions called agents that are created for an optimization problem as an initial population. Then, each agent moves other agents based on Newtonian gravitational law. So the problem space is searched to find the best possible solution. According to GSA, at the beginning of the algorithm, a population of *n* agents is created. The position of each agent (particle), which is a solution, is defined as follows:

$$X_i = \left(x_i^1, \ldots, x_i^d, \ldots, x_i^m\right); \quad i = 1, 2, \ldots, n \tag{3-3}$$

where x_i^d is the position of the ith mass in the *d*-th dimension and *m* is the number of dimensions. Then by computing and using the fitness value of each particle, their masses are calculated as follows (for a maximization problem):

$$q_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \quad , \quad M_i(t) = \frac{q_i(t)}{\sum_{j=1}^{n} q_j(t)} \tag{3-4}$$

where $fit_i(t)$ presents the fitness value of the *i*-th agent in the *t*-th iteration (specific time). $best(t)$ and $worst(t)$ show the best and worst values in *t*-th iteration and are defined as follows:

$$best(t) = \max_{j\in\{1,\ldots,n\}} fit_j(t) \ , \quad worst(t) = \min_{j\in\{1,\ldots,n\}} fit_j(t) \tag{3-5}$$

In the next step, the force acting from the *j*-th agent to the *i*-th agent in the *t*-th iteration is calculated as follows:

$$F_{ij}^d(t) = G(t)\frac{M_i(t)\times M_j(t)}{R_{ij}+\varepsilon}\left(x_j^d(t) - x_i^d(t)\right) \tag{3-6}$$

where M_i and M_j are the mass of agents *i* and *j*, respectively and ε is a small constant. R_{ij} is the hamming distance between two agents *i* and *j*. The total force acting from other agents to the *i*-th agent in dimension *d* is calculated by summing the randomly weighted forces that are extracted from the *d*-th dimension of other agents:

$$F_i^d(t) = \sum_{j\in Kbest,\, j\neq i} \gamma_j F_{ij}^d(t) \tag{3-7}$$

where γ_j is a random value between 0 and 1. Note that only *k* best agents which have the highest fitness value are considered to attract the others (Sheikhan, 2014) because this limitation will improve the performance of GSA by controlling exploration and exploitation to avoid trapping in the local optimum solution. In the beginning, a set of *k* best comprises of all the agents and decreases linearly to one, over time. In the next step the acceleration of the *i*-th agent in the dimension *d* is computed based on the law of motion as follows:

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} \tag{3-8}$$

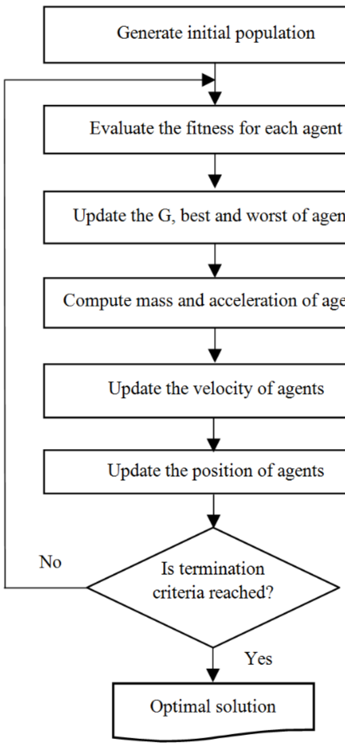

Generate initial population
Evaluate the fitness for each agent
Update the G, best and worst of agents
Compute mass and acceleration of agents
Update the velocity of agents
Update the position of agents
Is termination criteria reached?
No
Yes
Optimal solution

Figure 3-1: Flowchart of BGSA

Then the velocity of an agent in the dimension d at the *t*-th iteration is computed by current velocity and its acceleration as follows:

$$v_i^d(t+1) = \alpha_i v_i^d(t) + a_i^d(t) \tag{3-9}$$

where α_i is a random value between 0 and 1. In the BGSA, the position of an agent in each dimension can take only 1 or 0 value (means that does the equivalent subset include the corresponding feature or not?) The position updating of agents are calculated according to the mass velocity probability as follows:

$$x_i^d(t+1) = \begin{cases} compelent\left(x_i^d(t)\right); & \gamma_i < S\left(v_i^d(t)\right) = \left|tanh\left(v_i^d(t)\right)\right| \\ x_i^d(t); & otherwise \end{cases} \tag{3-10}$$

To reach a good convergence rate, Rashedi et al. (Rashedi, Nezamabadi-Pour, & Saryazdi, 2010) limited the velocity by $|v_i^d(t)| < v_{max} = 6$. The above steps will continue until the allowed iteration number is reached or an acceptable solution is found. The flowchart of BGSA is shown in Figure 3-1.

3.1.1.4 Mutual Information

Generally, features can be classified into three categories: relevant, redundant, and noisy features (Bonev, 2010). Relevant features are the features that have information about the target classes. It means that they can classify instances by themselves or in a subset with other features. As shown in Figure 3-2, they include two different types: strong relevance and weak relevance. A feature f_i is strongly relevant if its removal degrades the performance of the classifier. It is weakly relevant if it is not strongly relevant and if F is a subset of features, the performance of the classifier on $F \cup \{f_i\}$ will be better when just F is used (Bonev, 2010).

Because the redundant features can provide the same information about the target classes like selected features, therefore they can be removed from features space. Some features do not include information about the target classes and also they are not redundant, which are called noisy features and they should be removed, because they have a negative effect on classification accuracy.

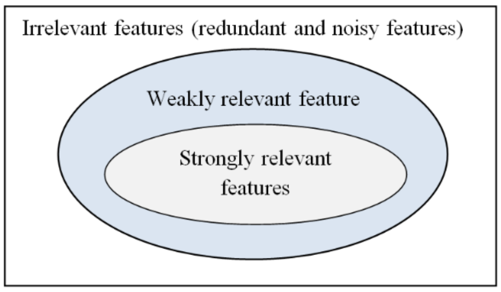

Irrelevant features (redundant and noisy features)
Weakly relevant feature
Strongly relevant features

Figure 3-2: The illustration of feature types as a feature space

In most intrusion detection datasets, some of the features are practically either redundant or irrelevant (noisy) to the classification problem and they cause a low DR. Generally, the feature selection methods find the optimal features by measuring the relation amount of features to the target feature. A variety of evaluation criteria have been introduced for filter-based methods which can be classified into five groups (Kumar & Kumar, 2012): distance, information (or uncertainty), dependency, consistency, and classifier error rate. The information metric is a non-linear correlation metric and is a good measurement to quantify the uncertainty of the feature (Deisy et al., 2010). One of the popular and effective methods for improving feature selection is based on using mutual information. MI is an information theory metric that is used to measure the relevance of features and it shows how much information is shared between features. Suppose X and Y are two discrete random variables that take on values from sets of A and B, respectively. The mutual information of these two variables is defined as:

$$I(X;Y) = \sum_{x \in A} \sum_{y \in B} p(x,y)\ log \frac{p(x,y)}{p(x)p(y)}; \quad p(x) = \Pr(X = x),\ p(y) = \Pr(Y = y) \tag{3-11}$$

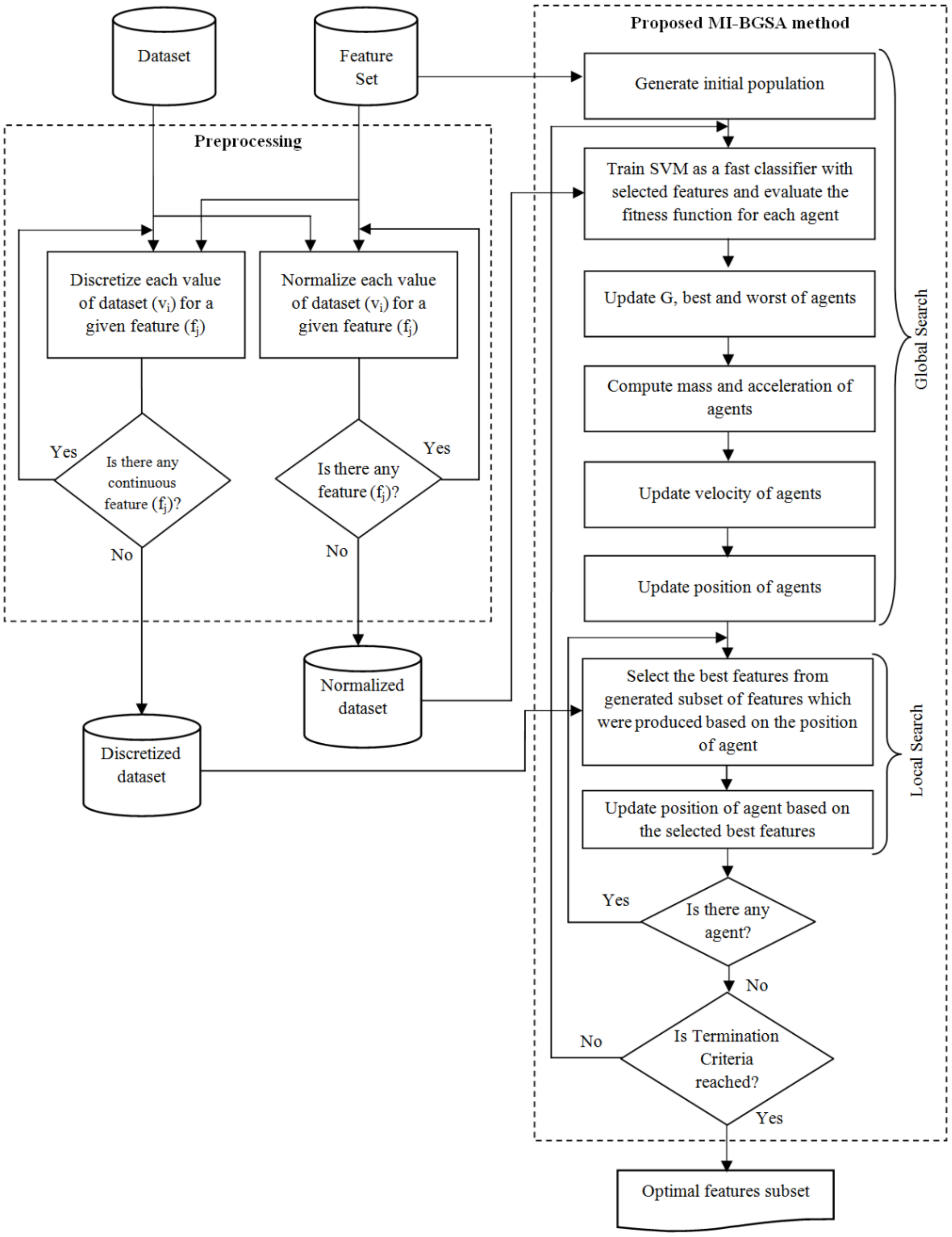

Dataset
Feature Set
Proposed MI-BGSA method
Generate initial population
Preprocessing
Train SVM as a fast classifier with selected features and evaluate the fitness function for each agent
Discretize each value of dataset (v_i) for a given feature (f_j)
Normalize each value of dataset (v_i) for a given feature (f_j)
Update G, best and worst of agents
Compute mass and acceleration of agents
Global Search
Yes
Is there any continuous feature (f_j)?
Is there any feature (f_j)?
Yes
Update velocity of agents
No
No
Update position of agents
Normalized dataset
Select the best features from generated subset of features which were produced based on the position of agent
Discretized dataset
Local Search
Update position of agent based on the selected best features
Yes
Is there any agent?
No
No
Is Termination Criteria reached?
Yes
Optimal features subset

Figure 3-3: The proposed hybrid framework for feature selection

where $p(x)$ and $p(y)$ are the marginal probability distribution functions for X and Y and $p(x,y)$ is the joint probability distribution function of X and Y. If X and Y are closely related to each other, the mutual information between them will be very high and vice versa. The main idea for using MI in feature selection is that the features should be highly correlated with the target class, without having any correlation with each other (Kumar & Kumar, 2012). According to this idea, Batti (Battiti, 1994) proposed a method called information-based feature selection (MIFS) where its evaluation function was defined as follows:

$$J(f_i) = I(f_i;C) - \beta \sum_{f_s \in S} I(f_i;f_s) \tag{3-12}$$

where f_i is the feature which we want it to be selected, C is the target class, and S is the candidate selected feature set. β is a parameter that is determined empirically and varies between 0 and 1 (Kumar & Kumar, 2012). The aim of this method is to maximize the relevance between input features and target class and to minimize the redundancy between selected features (Amiri, Yousefi, Lucas, Shakery, & Yazdani, 2011). In Eq. (3-12), the first term gives the feature relevance between the new input feature and output feature, and the second term is like a penalty for the first term and measures the correlation between the new feature and selected features. When the number of selected inputs increases, the effect of the first term would decrease. In order to avoid that, Amiri et al. proposed a modified MIFS (MMIFS) by modifying the MIFS evaluation function as follows (Amiri et al., 2011):

$$J(f_i) = I(f_i;C) - \frac{\beta}{|S|} \sum_{f_s \in S} I(f_i;f_s) \tag{3-13}$$

where $|S|$ is the number of selected features. Determining an appropriate value assignment for β is a critical task because it has a great effect on the appropriate feature selection in the MIFS algorithm.

3.1.1.4 The Proposed Feature Selection Method

In this section, the proposed hybrid filter-wrapper feature selection method is introduced. This method is based on BGSA as a population-based heuristic search. The proposed method uses mutual information to compute the feature-feature and feature-class mutual information for reaching a maximum relevance between selected features and the target class and a minimum redundancy between selected features. This hybrid of a wrapper and filter types of feature subset selection algorithm was used for improving the performance of the intrusion detection system. Figure 3-3 depicts the proposed hybrid framework.
According to Figure 3-3, the feature selection process was performed in the following steps:

Step 1 (Initialization): Some of the feature subsets were generated randomly as the initial population of agents (*n*). Each agent was comprised of a binary string with a length of *m* (the number of features in the dataset), where the value of each bit showed the presence of a corresponding feature in an agent.

Step 2 (Evaluation): Each agent was evaluated by the fitness function in this step. It means that for each agent (feature subset), the SVM model (as a binary classifier) was formed by training instances in the dataset that included just features in the corresponding feature subset, and it was evaluated by testing instances. In this step, the best position of the agent (best feature subset) between the current position of agents and the best position up to the previous iterations was identified based on a fitness function. The fitness function was a two-objective function based on maximizing DR and minimizing FPR. DR is the ratio between the number of correctly detected attacks and the total number of attacks. FPR is the ratio between the number of incorrectly detected attacks and the total number of normal connections in the testing set. It was assumed that the DR and the FPR had the same importance; therefore, a fitness function was introduced based on minimizing the Euclidean distance between each agent point with coordinates of (FPR_i, DR_i) and the perfect classification point with coordinates of (0%,100%) in a receiver operating characteristic (ROC) curve. Figure 3-4 shows a simple example of the Euclidean distance between an agent point and the perfect classification point in the ROC curve. FPR_i and DR_i are the false positive rate and the detection rate of the *i*-th agent. The proposed fitness function is defined as follows:

$$Fitness = \min\left(\{d(a_i, p) = \sqrt{(p_1 - a_{i1})^2 + (p_2 - a_{i2})^2} \mid 1 \le i \le n\right) \tag{3-14}$$

where a_i and p are the *i*-th agent point and the perfect classification point, respectively, and n is the number of agents.

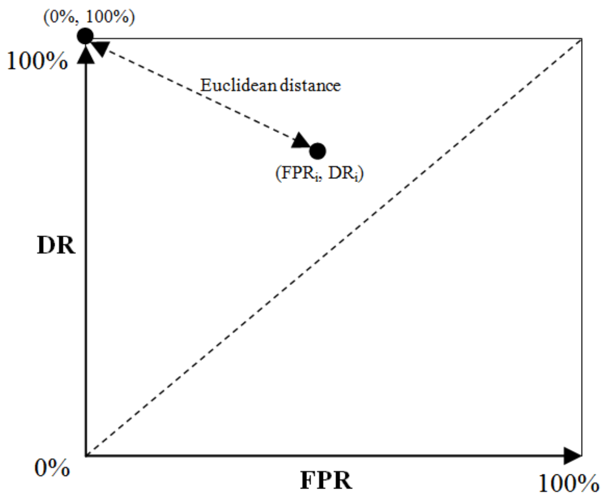


Figure 3-4: A simple ROC curve with one agent

Step 3 (Updating): The position of agents was updated based on Newtonian gravitational laws. To do so, *G*, the best and the worst values of the population were updated in the beginning using Eqs. (3-2) and (3-5), respectively. Then, the mass (*M*) and the acceleration (*a*) of agents were calculated using Eqs. (3-4) and (3-8), respectively. After that, the velocity and the position were updated using Eqs. (3-9) and (3-10), respectively. Then, if the maximum number of iterations was reached (the termination condition), the best position of the agent was returned as an optimal feature subset.

Because the position of each agent was updated only based on Eq. (3-10), this means that the select/deselect decision about a special feature is independent of other feature selection decisions, which can lead to an increase in the redundancy or statistical dependencies among the selected features. So, mutual information was used in the position updating phase in order to find an optimum feature subset for improving the performance of the feature selection method. To use the mutual information in the third step, as can be seen in Figure. 3-3, the proposed method was included two search strategies of optimization: (a) the global search, as the outer optimization layer, deals with the problem of choosing the optimal feature subset based on the BGSA in a wrapper manner, (b) the local search, as the inner optimization layer, prunes the feature subset generated by the wrapper model. Feature subset pruning was based on the MMIFS method. The following pseudo-code describes the proposed algorithm:

Algorithm 1. MI-BGSA feature selection method

Input:

- SM is the search mode with permissible values of 1 and 0, where 1 means that the algorithm will search for the best pre-specified features and 0 means that the algorithm will search for an optimum set of features out of all available features.
- *k* is the pre-determined feature set size.

Output: Selected features *F′*.

Initialization:

- Initialize *n* (number of agents), *m* (number of features), and t_{max} (maximum number of iterations).
- Set parameters G_0, v_{max}, and *β*.
- Initialize $t = 0$ and the position of the agents $X_i(t)$ for $i = 1,2,..,n$ randomly with a binary value.
- Set δ_{MI} by a pre-defined threshold computed by the proposed fitness function.
- Initialize the fitness of the agents $Fitness(t) = [0]_{n\times 1}$.
- Initialize the velocity of the agents $V_i(t) = [0]_{1\times m}$ for $i = 1,2,..,n$.
- Set $Fbest = 0$ as the best fitness and $Lbest = [0]_{1\times m}$ as the best position.

Steps:

```
(1)  while $t < t_{max}$
(2)      Calculate $Fitness(t)$ via evaluating the position of agents by using the proposed MI-BGSA
         fitness function.
(3)      Calculate $best(t) = \max(Fitness(t))$, $worst(t) = \min(Fitness(t))$, and $G(t) = G_0(1 - \frac{t}{t_{\max}})$.
(4)      if $Fbest < best(t)$
(5)          Set $Fbest = best(t)$ and $Lbest = X_{\underset{(row\_index)}{\arg\max\{Fitness(t)\}}}(t)$.
(6)      end
(7)      Calculate $q_i(t)$ and $M_i(t)$ for $i = 1,2,..,n$ using Eq. (3-4).
(8)      Calculate $F_{ij}^d(t)$ for $i, j = 1,2,..,n$ and $d = 1,2,..,m$ using Eq. (3-6).
(9)      Calculate $F_i^d(t)$ for $i = 1,2,..,n$ and $d = 1,2,..,m$ using Eq. (3-7).
(10)     Calculate $a_i^d(t)$ for $i = 1,2,..,n$ and $d = 1,2,..,m$ using Eq. (3-8).
(11)     Calculate $v_i^d(t+1)$ for $i = 1,2,..,n$ and $d = 1,2,..,m$ using Eq. (3-9).
(12)     for each agent $i = 1,2,..,n$
(13)         Update $x_i^d(t+1)$ using Eq. (3-10) for $i = 1,2,..,n$ and $d = 1,2,..,m$.
(14)     end for
(15)     for each agent $i = 1,2,..,n$
(16)         Initialization: Set $F = \{\}$ as the initial set of input features which to be filled by the selected
             features that found in the outer layer, $S = \{\}$ as the selected feature, and y = 'class-output'.
(17)         for each feature $f_d$ from the original feature set ($d = 1,2,..,m$)
(18)             if $x_i^d(t+1) = 1$
(19)                 $F = F \cup \{f_d\}$,
(20)             end
(21)         end for
(22)         for each feature $f_d \in F$
(23)             compute $I(f_d; y)$
(24)         end for
(25)         Selection of the first feature: Find the feature $f_d$ which maximizes $I(f_d; y)$, set
             $F = F - \{f_d\}$, and
             $S = \{f_d\}$.
(26)         Set count = 1 where SM is 1
(27)         while $F \neq \emptyset$ where SM is 0 or $count \leq k$ where SM is 1
(28)             for each pair of features $(f_d, f_s)$, where $(f_d \in F), (f_s \in S)$
```

```
(29)                compute $I(f_d;f_s)$
(30)            end for
(31)            for each feature $f_d \in F$
(32)                compute $J(f_d) = I(f_d;y) - \frac{\beta}{|S|}\sum_{f_s \in S} I(f_d;f_s)$
(33)            end for
(34)            Choose the feature $f_d$ as the one that maximizes $J(f_d)$ and set $Max = \max(J(f_d))$
(35)            if $Max \geq \delta_{MI}$ where SM is 0 or $count \leq k$ where SM is 1
(36)                Set $F = F - \{f_d\}$, $S = S \cup \{f_d\}$
(37)                Set $count = count + 1$ where SM is 1
(38)            else
(39)                Set $F = F - \{f_d\}$ and $x_i^{f_d}(t+1) = 0$ (the candidate feature $f_d$ is skipped in the i-th
                    agent)
(40)            end if
(41)        end while
(42)    end for
(43) end while
(44) Set $F' = \{$if $Lbest^d = 1$ then return $d \mid d = 1,2,..,m\}$ as the final features subset
(45) return $F'$
```

Lines 2-14 in Algorithm 1 show the global search that leads to selecting an optimal feature subset. Then, as shown in lines 15-42, the proposed method works as the MMIFS method for pruning the optimal feature subset which can lead to select the best features from the optimal feature set. The MMIFS is a greedy forward feature selection algorithm, in which each single input feature is added to the selected feature set based on feature-feature and feature-class mutual information. The objective of this method is to maximize the relevance between the selected features and the target class and to minimize the redundancy among the selected features. As shown in Algorithm 1, the MMIFS procedure (lines 15-42) selects one feature at each time. This selection is based on maximizing the MMIFS evaluation function. It means that a feature (e.g., f_d) will be selected in each time which can lead to maximizing the information with the target class and minimizing the information with the selected features. As seen in Algorithm 1, the proposed method works in two search modes: (a) selection of the optimal feature set size, (b) selection of the pre-determined feature set size. When the proposed method works in the second search mode, the MMIFS procedure can be performed until the pre-determined feature set size is selected. In the first search mode, a threshold (i.e., δ_{MI}) should be defined. In this mode, we assume that when the evaluation of candidate feature f_d (that is a feature which can lead to maximization of MMIFS evaluation function) is greater than or equal to the threshold; then, f_d will be selected. In this paper, we used the genetic algorithm (GA) with the aim of finding an appropriate threshold. GA is an adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics which uses random search to solve optimization problems. Each population in the proposed GA represents a possible solution to find an appropriate threshold. In GA, each population is evaluated by a fitness function. The fitness function used in GA was based on the MMIFS method. The following pseudo-code describes the proposed GA-based fitness function:

Algorithm 2. Proposed genetic algorithm fitness function

```
Input: A binary string with size of N as a chromosome.
Output: Fitness value R.
Initialization:
```

```
• Set $F = \{f_1 \ldots, f_m\}$ as the full feature set of the dataset, $S = \{\}$ as the selected feature and
  $y$ = 'target class'.
Steps:
(1)  Set $cnt = count$ number of 1 value in input binary string;
(2)  Set $\delta_{MI} = cnt \times 0.01$;
(3)  for each feature $f_i \in F$
(4)      Compute $I(f_i; y)$;
(5)  end for
(6)  Find the feature $f_i$ which maximizes $I(f_i; y)$; set $F = F - \{\}$, $S = \{f_i\}$;
(7)  while $F \neq \emptyset$
(8)      for each pair of features $(f_i, f_s)$, where $(f_i \in F)$, $(f_s \in S)$
(9)          Compute $I(f_i; f_s)$;
(10)     end for
(11)     for each feature $f_i \in F$
(12)         Compute $J(\mathrm{f}_i) = I(f_i; y) - \left(\frac{\beta}{|S|}\right) \sum_{f_s \in S} I(f_i; f_s)$;
(13)     end for
(14)     Choose the feature $f_i$ as the one that maximizes $J(\mathrm{f}_i)$ and set $Max = \max(j(f_i))$;
(15)     if Max $\geq \delta_{MI}$ Max $\geq \delta_{MI}$
(16)         Set $F = F - \{f_i\}$, $S = S \cup \{f_i\}$;
(17)     else
(18)         Set $F = F - \{f_i\}$;
(19)     end if
(20) end while
(21) Set R by evaluating the S by using the proposed MI-BGSA fitness function;
(22) return R;
```

Assuming $0 \leq J(f_i) \leq 1$, so the threshold should be between 0 and 1. Therefore, each population in GA which represents a possible solution is coded with a binary string vector as a chromosome with a size of 100. The value of each bit is 0.01; it means the threshold will be accurate up to 2 decimal points. As seen in Algorithm 2, the generated feature subset in the MMIFS method will be evaluated using the proposed MI-BGSA fitness function which is originally used in the proposed MI-BGSA feature selection method (for evaluating each agent). The following pseudo-code describes the proposed MI-BGSA fitness function:

Algorithm 3. MI-BGSA fitness function

```
Inputs: $Tr = D_1(F, C)$ and $Te = D_2(F, C)$ as the training and testing datasets where F and C are the feature
        set and the target class, respectively.
Output: fitness value R
Steps:
(1) Train SVM classifier using Tr
(2) Test SVM classifier using Te
(3) Compute DR and FPR using the test result of the previous step
(4) Set R by computing the fitness value using Eq. (3-14)
(5) return R
```

3.1.2 The Presented Approach

As shown in Figure 1-1, in order to detect cyber-attacks, which may be launched by the Internet (or LANs), this thesis presents a novel misuse-based algorithm by introducing a new version of OPF called Modified OPF (MOPF). Optimum-path forest (Joao P Papa, Falcao, & Suzuki, 2009) is a graph-based machine learning method that reduces a classification problem into the problem of partitioning the vertices of a graph. This following characteristics leading to OPF became an efficient machine learning algorithm (Amorim & de Carvalho, 2012):

- OPF is a simple and fast classifier.
- OPF is a parameter-independent that originally supports multi-class problems.
- OPF can handle partial overlapping among the classes due it does not make any assumption about the shape of classes.
- OPF can be used in both classification and clustering problems.

Generally, the proposed model consists of three main modules: (a) partitioning; (b) pruning; and (c) detecting. One of the main challenges in the OPF is the size of the training set. In the first module, the K-Means clustering algorithm was used as an unsupervised learning method for addressing the problem of scalability in large datasets. Moreover, this partitioning can enhance the detection accuracy of low-frequent attacks, such as R2L and U2R attacks, in the NSL-KDD dataset. The K-Means clustering algorithm partitions the original heterogeneous training dataset into *k* homogeneous clusters which will be used as the training and evaluation datasets of *k* extended models of OPF named Advanced OPF (AOPF) in the detecting module. With the aim of speeding up the OPF, the centrality and the prestige concepts in social network analysis were used in the second module for pruning the training set of OPF by identifying and selecting the most informative samples. In our work, the proposed pruning module can identify informative samples, so the size of training and evaluation sets can be reduced for speeding up the OPF. Then, AOPFs were trained in the detecting module by using different training and evaluation sets which were projected in the first module. Before introducing the proposed method, some of the fundamental concepts used in the presented misuse-based IDS will be reviewed.

3.1.2.1 Supervised Optimum-path Forest Algorithm

In 2009, Papa et al. (Joao P Papa et al., 2009) introduced a supervised machine learning algorithm based on the graph theory, called optimum-path forest. They reduced a pattern recognition problem into an optimal graph partitioning in a given feature space. In the OPF, each sample is represented by a feature vector and is shown as a node in a complete weighted graph. The weighted arcs, which are defined by adjacency relations between samples, link all pairs of nodes in this graph. The arcs are undirected and the distance between two feature vectors is considered as the weight of arc in the complete graph. Each path connects a pair of nodes. A path is composed of a sequence of distinct samples and a connectivity function, such as the maximum arc-weight along the path, which assigns a cost to the path (Amorim & de Carvalho, 2012). Generally, the OPF-based classification is composed of two distinct phases: a) training and b) classification. In the training phase, some key samples from the training set, called prototypes, should be identified for each class in the classification problem. Then, the complete graph will be partitioned into optimum-path trees (OPTs) by a competitive process between prototypes (as the roots of the OPTs) which introduces optimum paths to the remaining nodes of the graph. The OPF will be composed of a union of OPTs. The nodes of OPT will be strongly connected to their prototypes as compared to other prototypes in the OPF. Hence, each sample that belongs to an OPT has the same class as its prototypes. In the classification phase, all of the arcs that connect an unlabeled sample to all samples in the OPF are considered to classify that unlabeled sample. By evaluating the paths from the nodes of OPF to an unlabeled sample, a node can be found which offers an optimum path to that unlabeled sample. Hence, the class of an unlabeled sample will be same as the root's class of the best node. In addition to the training and the classification phases, another phase (called learning) is usually used for improving the accuracy of OPF. The learning phase is performed by using the classification errors on the evaluation set. Suppose $Z = Z_1 \cup Z_2 \cup Z_3$ as a labeled dataset in which Z_1 , Z_2 , and Z_3 are the training, the evaluation, and the test datasets, respectively. It is noted that Z_2 is used for improving the accuracy of learning model on Z_3 by teaching the classifier using randomly selected samples of the same class from non-prototype samples in Z_1 and replacing them with the misclassified samples of Z_2 (Amorim & de Carvalho, 2012). The details of OPF procedures are described in the following subsections.

A. Training Phase

Identification of the optimum set of prototypes, called $S^* \subset Z_1$ is one of the main processes in the training phase. Suppose $G = (Z_1, A)$ is a complete weighted graph with the specifications mentioned above. The samples in the training dataset are represented by the nodes of *G*, and each pair of samples is defined by its arc in $A = Z_1 \times Z_1$. To find the prototypes from Z_1 , an MST from G should be computed. The closest nodes of MST which have different labels in Z_1 are the prototypes of OPF (Joao P Papa et al., 2009). The optimum set of prototypes (S^*) is a set of prototypes that minimizes the classification error of the OPF training algorithm in Z_1 (Joao P Papa et al., 2009).

An OPT should be found in the training phase for each $s \in Z_1$ that rooted a special prototype $p \in S$, where *S* is the set of prototypes. It means that the OPF classifier assigns one optimum-path from *S* to each sample $s \in Z_1$. Notably, a sequence of distinct samples such as $\pi_t = \langle s_1, s_2, \ldots, s_k, t \rangle$ with terminus *t* is a path, and a path with one sample like $\pi_t = \langle t \rangle$ is called trivial(Joao P Papa et al., 2009). A path-value function f_{max} assigns a path cost to each path π_t. The f_{max} is determined by Eq. (3-15) (Joao P Papa et al., 2009):

$$f_{max}(s) = \begin{cases} 0 & if\ s \in S \\ +\infty & otherwise \end{cases} \quad , f_{max}(\pi_s . s,t) = \max\{f_{max}(\pi_s),\ d(s,t)\} \tag{3-15}$$

where s is a training sample (i.e., ($s \in Z_1$)) and *S* represents the prototype set. Moreover, the distance between samples *s* and *t* is denoted by $d(s,t)$. The maximum distance between two adjacent samples along $\pi_s . \langle s,t \rangle$ is computed by $f_{max}(\pi_s . \langle s,t \rangle)$ where $\pi_s . \langle s,t \rangle$ denotes the concatenation of π_s and $\langle s,t \rangle$ (as a path and an arc, respectively). Partitioning the OPF for computing OPTs is performed by minimization of f_{max} which assigns an optimum path $P^*(t)$ from the set of prototypes ($S \subset Z_1$) to every sample $t \in Z_1$ whose minimum cost $C(t)$ is calculated as follows (Joao P Papa et al., 2009):

$$C(t) = \min_{\forall \pi_t \in (Z_1, A)} \{f_{max}(\pi_t)\} \tag{3-16}$$

The minimization of f_{max} is performed in the training phase of OPF. To see more details about the OPF training algorithm, refer to (Joao P Papa et al., 2009).

B. Classification and Learning Phases

To classify each unlabeled sample such as *t*, we assume *t* as a part of the training set. Let all of the nodes and their arcs connect corresponding samples in the training set to *t*. The purpose of the classification phase is to find an optimum path $P^*(t)$ from S^* (as optimum path prototypes) to t, and then labeling t with the class of its root which is shown by $L(t)=\lambda(R(t))$, in which $R(t)$ denotes the root of t and $\lambda(R(t))$ is a function that assigns the correct label to $R(t)$ (Joao P Papa et al., 2009). The optimum path can be found incrementally by evaluating the optimum cost as follows (Joao P Papa et al., 2009):

$$C(t)=\min\{\max\{C(s),\ d(s,t)\}\}\ ; \quad \forall s \in Z_1 \tag{3-17}$$

where $C(s)$ is the minimum cost of *s*. Suppose $s^* \in Z_1$ is the best node that satisfies Eq. (3-17). It is noted that $L(s^*)=\lambda(R(t))$, so the classifier assigns $L(s^*)$ to *t* as the class of *t* and an error is occurred when $L(s^*)\neq\lambda(t)$ (Joao P Papa et al., 2009).

One of the main challenges in building an efficient classifier is the quality of training samples in a classification problem. Hence, it would be interesting to select the most informative samples for Z_1 which can improve the accuracy of the classifier on Z_3. Papa et al. (Joao P Papa et al., 2009) presented a general learning algorithm for building an efficient classifier that used an evaluation set Z_2 to enhance the composition of samples in the training set Z_1 without changing its size. They used Z_2 for evaluating the classifier which was projected on Z_1. Then, they replaced misclassified samples of Z_2 with the samples of the same class from non-prototype samples in Z_1 which were randomly selected. This process was repeated for *T* iterations by using the new sets of Z_1 and Z_2. More details about the OPF classification and learning algorithms can be found in (Pereira, Nakamura, Costa, & Papa, 2012) and (JoãO P Papa, FalcãO, De Albuquerque, & Tavares, 2012).

3.1.2.2 K-Means Clustering Algorithm

Clustering is based on an unsupervised learning algorithm that can cluster data samples into disjoint partitions based on their similarities. The K-Means is a popular and crisp clustering technique (with non-overlapping partitions) that is a numeric, non-deterministic, and iterative method (Sangita & Dhanamma, 2011). Suppose $D=\{x_i \mid i=1,2,\ldots,n\}$ is a set of n samples. The K-Means algorithm partitions these samples into *k* clusters $C=\{C_1, C_2,\ldots,C_k\}$ such that the following conditions be satisfied (Bakshi, Jagadev, Dehuri, & Wang, 2014):

a) $C_i \neq \varnothing : \forall 1 \le i \le k$
b) $C_i \cap C_j = \varnothing : \forall 1 \le i, j \le k, i \neq j$
c) $\bigcup_{i=1}^{k} C_i = D$

The main goal of K-Means clustering algorithm is to minimize the sum of dissimilarity of all samples in a cluster from the centroid. In other words, if a criterion is defined as follows:

$$g_n(C) = \underset{C}{\arg\min} \sum_{i=1}^{k} \sum_{x_j \in C_i} \left| x_j - \mu_i \right|^2 \tag{3-18}$$

then we want to partition *n* samples into *k* clusters such that the above criterion within each cluster becomes minimum. In Eq. (3-18), μ_i is the centroid of all samples which belong to the cluster C_i. Notably, $g_n(C)$ is the K-Means objective function in which $\sum_{i=1}^{k} \sum_{x_j \in C_i} \left| x_j - \mu_i \right|^2$ (as an error function) shows the quality of the clustering. The steps of K-Means clustering algorithm are as follows (Sangita & Dhanamma, 2011):

Step 1: Generate *k* random centroids from the dataset.
Step 2: For each sample in the dataset, calculate its Euclidean distance from the centroids.
Step 3: Assign each sample to a cluster that is the nearest to the centroid of the cluster based on the calculated Euclidean distance (computed in Step 2).
Step 4: Recalculate the new cluster centroids by computing the mean of attribute values of the samples in each cluster.
Step 5: Repeat Steps 2 to 4 until the stopping criterion is met.

One of the key parameters in K-Means clustering algorithm is the value of *k* which has a great effect on the performance of the algorithm. The Davies-Bouldin (DB) index is used as the evaluation metric of K-Means clustering algorithm in this study (Davies & Bouldin, 1979):

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{j=1,\dots,k} \left(R_{ij} \right); \quad R_{ij} = \frac{S_i + S_j}{d_{ij}}, j \neq i \tag{3-19}$$

where S_i and S_j are the average distances of all elements in the ith and the *j*-th clusters from the centroid of each cluster, respectively. d_{ij} denotes the distance between centroids of the *i*-th and the *j*-th clusters. As seen in Eq. (3-19), the DB index is a function of the ratio of the sum of within-cluster scatter to between-cluster separation. The within-cluster scatter should be minimized and the between-cluster separation should be maximized (Ray & Turi, 1999). R_{ij} in Eq. (3-19) shows the similarity measure of clusters. Moreover, R_{ij} is based on the dispersion measure of a cluster (S_i) and the cluster dissimilarity measure (d_{ij}) (Davies & Bouldin, 1979). The minimum value of DB index shows a better quality of the clusters and consequently better performance of K-Means clustering algorithm. To see more details about this metric, refer to (Davies & Bouldin, 1979).

In the proposed model, K-Means clustering algorithm was used for partitioning the original training set to *k* clusters that were used as the training sets of k OPFs. This approach was used to addressing the scalability problem.

3.1.2.3 Centrality and Prestige Concepts in Social Network Analysis

The social network is a social structure made up of a set of nodes as the social actors who are connected by one or more specific types of interdependency. The social actors and their interactions or relationships can be represented by a graph. Hence, the graph theory is an essential approach which is widely used in the social network analysis. Identification of influential actors is one of the primary applications of the graph theory in the social network analysis for finding the most important ones. Centrality and prestige (Wasserman & Faust, 1994) are two fundamental measures that are widely used for this purpose. By using the centrality measure in social network analysis, some actors who are extensively involved in relationships with other actors can be found (Musiał, Kazienko, & Bródka, 2009). A variety of metrics such as the degree centrality, the closeness centrality, and the betweenness centrality are used for measuring the centrality (Rusinowska, Berghammer, De Swart, & Grabisch, 2011).

The prestige is a property that is derived from the patterns of social ties of a particular social network (Rusinowska et al., 2011). In the social network analysis, the prestige measure focuses on in-links. In other words, the opinion of other actors (which are expressed by their arcs) determines the importance of an actor (instead of the opinion of a special actor) (Varlamis, Eirinaki, & Louta, 2013). Different measures such as the node degree-based prestige, the proximity prestige, the rank prestige, and the node position are used for measuring the prestige (Musiał et al., 2009). The prestige measure differentiates in-links and out-links, so it is a more refined measure of prominence of an actor than the centrality in the social network analysis (Varlamis et al., 2013). In this study, we have solely focused on the betweenness centrality (BC) and the proximity prestige (PR).

The BC measure indicates the importance of an actor in terms of connecting other actors (Rusinowska et al., 2011). In other words, BC is based on an actor who makes to tie two other actors in the social network, provided that this connection path is the shortest path between those two actors. For each pair of actors, this metric takes into account how many times a special actor can interrupt the shortest distance between two actors. In a directed graph, the BC is computed as follows (Wasserman & Faust, 1994):

$$BC(x_k) = \sum_{x_i \neq x_j \neq x_k ,\ x_i, x_j, x_k \in M} \frac{b_{x_i x_j}(x_k)}{b_{x_i x_j}} \tag{3-20}$$

where M is the node set, $b_{x_i x_j}(x_k)$ is the number of the shortest paths between node x_i and node x_j that contains node x_k, and $b_{x_i x_j}$ is the number of the shortest paths between node x_i and node x_j.

The PR measure shows how much all other actors within a social network are closed to a special actor (Musiał et al., 2009). This metric takes into account the influence domain of an actor, which is a set of actors who are directly (or indirectly) connected to it (Perez, Chiclana, & Ahmadi, 2011). In a directed graph, PR is computed as follows (Musiał et al., 2009):

$$PR(x_i) = \frac{\frac{I_i}{n-1}}{\frac{\sum_{j=1}^{I_i} d(y_j, x_i)}{I_i}} = \frac{I_i^2}{(n-1)\sum_{j=1}^{I_i} d(y_j, x_i)} \tag{3-21}$$

where n is the number of nodes (as actors in the directed graph) and I_i specifies the number of all actors who are in the influence domain of actor x_i. Moreover, y_j is an actor in the influence domain of actor x_i. The distance between two nodes y_j and x_i is shown by $d(y_j, x_i)$.

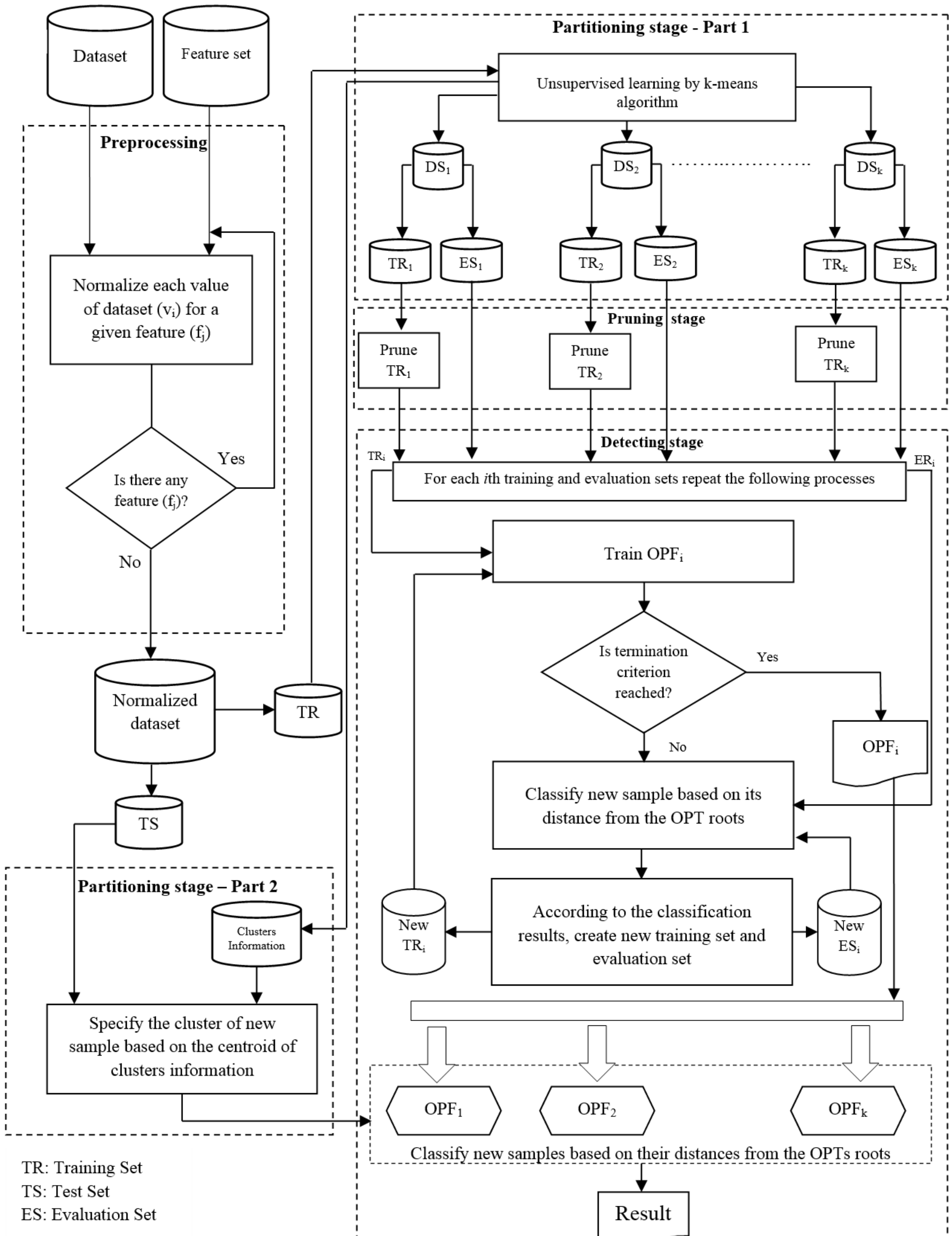

Dataset
Feature set
Preprocessing
Normalize each value of dataset (v_i) for a given feature (f_j)
Is there any feature (f_j)?
Yes
No
Normalized dataset
TR
TS
Partitioning stage – Part 2
Clusters Information
Specify the cluster of new sample based on the centroid of clusters information
Partitioning stage - Part 1
Unsupervised learning by k-means algorithm
DS_1
DS_2
DS_k
TR_1
ES_1
TR_2
ES_2
TR_k
ES_k
Pruning stage
Prune TR_1
Prune TR_2
Prune TR_k
Detecting stage
TR_i
ER_i
For each ith training and evaluation sets repeat the following processes
Train OPF_i
Is termination criterion reached?
Yes
No
OPF_i
Classify new sample based on its distance from the OPT roots
According to the classification results, create new training set and evaluation set
New TR_i
New ES_i
OPF_1
OPF_2
OPF_k
Classify new samples based on their distances from the OPTs roots
Result
TR: Training Set
TS: Test Set
ES: Evaluation Set

Figure 3-5: The framework of proposed misuse-based IDS.

3.1.2.4 The Proposed Offline Misuse-based IDS

In this section, the proposed IDS model is introduced. This model is based on a modified OPF. As mentioned earlier, there are three main modules in the proposed model: (a) partitioning; (b) pruning; and (c) detecting. In the partitioning module, the K-Means clustering algorithm is used as an unsupervised learning algorithm for grouping all samples from original training and evaluation sets into smaller groups that will be used in the detecting module as the training sets of OPF classifiers. Before using the generated training and evaluation sets in the OPFs, the most informative samples are selected in the pruning module. In the pruning module, the BC and PR metrics, which were introduced in the social network analysis, are used for identifying the influential samples and selecting the most informative ones from the training and evaluation sets. Notably, for each generated training or evaluation set, the pruning module is employed as a preprocessing stage for the OPF classifiers. The detecting module is used for anomaly detection and classification of attacks. Different OPFs are projected in this module based on each of the pruned training and evaluation sets. In the classification phase of OPF, all of the arcs that link a new sample to the training samples are considered. In this way, the distance between the new sample and the root (prototype) of each training sample in the OPTs becomes important. The proposed IDS framework is depicted in Figure 3-5. According to Figure 3-5, these steps are followed in the proposed IDS framework:

Step 1 (Preprocessing): One of the main challenges of the datasets used in machine learning is data imbalance. Therefore, it is essential to normalize the value of each feature to avoid data imbalance. As mentioned previously, in this study the NSL-KDD is employed for simulating the Internet (or LANs) traffic. The features in the NSL-KDD dataset have three different types: (a) continuous; (b) discrete; and (c) symbolic. These features have significantly different ranges and various resolutions; therefore, most of the classifiers are not able to process data in this format (Perez et al., 2011). In this study, all the features of the dataset were normalized before the partitioning phase. In a preprocessing step, the discrete features are normalized by transforming them into a suitable range (W. Wang, Zhang, Gombault, & Knapskog, 2009). Moreover, the symbolic feature should be replaced by numeric codes. Note, the value of numeric features was scaled by the statistical normalization as given below (W. Wang et al., 2009):

$$x_i = \frac{v_i - \mu}{\sigma};\ i = 1, 2, \ldots, N \tag{3-22}$$

where *N* is the number of instances in the dataset and v_i is the value of the ith instance in the dataset for the given feature. μ and σ are the mean and the standard deviation, respectively.

Step 2 (Partitioning): In this step, the training set is portioned into *k* clusters by using K-Means clustering algorithm for overload reduction of the intrusion detection engine and also for addressing the problem of scalability in a large dataset. As seen in Figure 3-5 (Partitioning stage- Part 2), the engine should determine which preference group of the training samples is associated with a new sample (when a new sample is classified)? Then, the corresponding OPF will be selected for the classification process. As given in Eq. (3-23), the algorithm calculates the Euclidean distance between the new sample and the centroid of each cluster for finding the best cluster for a new sample. It is noted that the best cluster is selected based on the minimum Euclidean distance between its centroid and the new sample (as compared to the other centroids):

$$\underset{i}{\arg\min} \left\{ \sqrt{(x_1 - \mu_{1i})^2 + (x_2 - \mu_{2i})^2 + \ldots + (x_m - \mu_{mi})^2}\ ; 1 \le i \le k \right\} \tag{3-23}$$

where $X = (x_1, x_2, \ldots, x_m)$ and $\mu_i = (\mu_{1i}, \mu_{2i}, \ldots, \mu_{mi})$ are the new sample and the centroid of *i*-th cluster, respectively.

Step 3 (Pruning): As discussed earlier, each training set such as TR_i , should be pruned in this step based on the BC or the PR metrics (that were originally employed in the social network analysis) for finding the most informative samples. Before using one of these two metrics as a suitable metric for pruning the training set, the performance was compared based on the proposed pruning approach. According to the mentioned evaluation, one of the BC or the PR metrics was used as a suitable metric in the proposed pruning phase. The details of this comparison are discussed in Section 4.2 of Chapter 4. Unlike the centrality measures (e.g., BC) which are used originally for undirected graphs, the prestige measure is used for directed graphs. Therefore, a directed graph should be extracted from the complete weighted graph to employ PR in the proposed model. On the other hand, the original complete weighted graph (which is used in the OPF) can be used for computing the BC; however, the order of computation time of the BC in weighted graph is $O\left(nm + n^3 logn\right)$ (Bader, Kintali, Madduri, & Mihail, 2007) in which *n* is the number of nodes and *m* is the number of edges. In this study, the computation of the BC (or even the PR) is a time-consuming process due to the size of the input dataset. In order to deal with this problem, the solution of this thesis was inspired by the proposal of Kang et al. (Kang, Papadimitriou, Sun, & Tong, 2011) for large-scale networks. This proposal was grounded on a new graph called the directed line graph. In order to work with a graph consisting of a reasonable number of edges, the obtained extracted directed graph, which was used for computing the PR, was also employed for computing the BC.

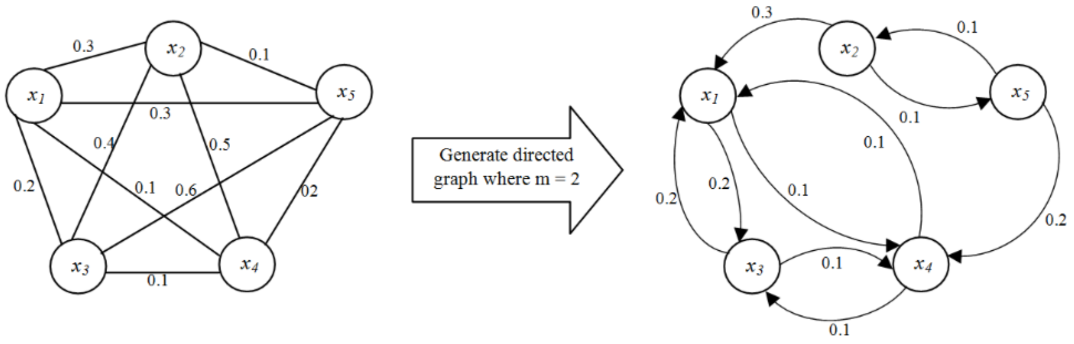


Figure 3-6: A simple example of generating a directed graph from a complete undirected graph

In sum, a directed graph was used in the proposed model for computing BC and PR metrics. As mentioned earlier, we should create a directed graph from the complete graph that was originally used in the OPF for employing these metrics in the proposed model. In the directed graph, the directed relations between each pair of nodes are based on the *m* best neighbors. To do it, as shown in Figure 3-6, for each node x_i in the complete undirected graph, an ordered list of m best neighbor nodes (as x_i neighbors) will be selected based on the weight of the arcs (i.e., the most similarity to x_i or minimum distance to x_i). By using these ordered lists, the direction of relations between the training samples can be identified for constructing the directed graph. When the directed graph was constructed, BC and PR were computed for each node in the directed graph (using Eqs. 3-20 and 3-21, respectively). For example, according to the generated directed graph in Figure 3-6, BC and PR of node x_4 are calculated as follows:

$$PR(x_4) = \frac{I_4^2}{(n-1)\sum_{j=1}^{I_4} d(y_j, x_4)} \overset{I_4=4,\, n=5}{\Rightarrow} PR(x_4) = \frac{16}{4\sum_{j=1}^{4} d(y_j, x_4)} \Rightarrow$$

$$PR(x_4) = \frac{16}{4\big(d(x_1,x_4)+d(x_2,x_4)+d(x_3,x_4)+d(x_5,x_4)\big)} = \frac{16}{4(0.1+0.3+0.1+0.2)} \approx 5.71$$

$$BC(x_4) = \sum_{x_i \neq x_j \neq x_4,\; x_i, x_j, x_4 \in M} \frac{b_{x_i x_j}(x_4)}{b_{x_i x_j}} = \frac{b_{x_1x_2}(x_4)}{b_{x_1x_2}} + \frac{b_{x_2x_1}(x_4)}{b_{x_2x_1}} + \frac{b_{x_1x_3}(x_4)}{b_{x_1x_3}} + \frac{b_{x_3x_1}(x_4)}{b_{x_3x_1}} +$$
$$\frac{b_{x_1x_5}(x_4)}{b_{x_1x_5}} + \frac{b_{x_5x_1}(x_4)}{b_{x_5x_1}} + \frac{b_{x_2x_3}(x_4)}{b_{x_2x_3}} + \frac{b_{x_3x_2}(x_4)}{b_{x_3x_2}} + \frac{b_{x_2x_5}(x_4)}{b_{x_2x_5}} + \frac{b_{x_5x_2}(x_4)}{b_{x_5x_2}} + \frac{b_{x_3x_5}(x_4)}{b_{x_3x_5}} + \frac{b_{x_5x_3}(x_4)}{b_{x_5x_3}}$$
$$= 0 + 0 + \frac{1}{2} + \frac{1}{2} + 0 + \frac{1}{1} + \frac{1}{1} + 0 + 0 + 0 + 0 + \frac{1}{1} = 4$$

Our simulation results showed that in the OPF (unlike social networks), some samples that have the worst BC or PR are selected as the informative samples. When the BCs (or PRs) of all samples in the training and evaluation subsets are computed, then γ percent of samples will be selected as the most informative samples according to the worst BCs (or PRs). In this study, the genetic algorithm (GA) is employed for finding an appropriate value for γ. The GA, which is based on the evolutionary ideas of genetics and natural selection, is an adaptive heuristic search algorithm for solving optimization problems. Any population in the proposed GA represents a possible solution of appropriate γ. In GA, a fitness function is used for evaluating the population. The fitness function, which is used in this study, is based on a proposed version of OPF (called advanced OPF (AOPF)) that will be explained in Step 4 of the proposed framework. The following pseudo-code describes the proposed algorithm for obtaining the fitness value (Algorithm 4):

Algorithm 4. The proposed algorithm for obtaining fitness value in finding the optimum value of γ

```
Input: A binary string with size of N as a chromosome (corresponding to $\gamma$). TR and TS are the training set and test set, respectively.
Output: Fitness value R.
Steps:
(1)  Set $\gamma$ as the count number of "1" values in the input binary string;
(2)  Set $S = \{\}$ as the betweenness centrality set (or the proximity prestige set)
(3)  Create a directed graph (DG) from the training samples;
(4)  for each node $n_i \in DG$ (which is corresponding to $s_i \in TR$)
(5)      Compute the BC (or PR) of $n_i$ based on DG and add the result to S;
(6)  end
(7)  Set $S_{optimal}$ by $\gamma$ percent of the worst values in S;
(8)  Set pruned TR (PTR) by the training samples corresponding to $S_{optimal}$;
(9)  Create an OPF model from PTR by the OPF training algorithm;
(10) Perform AOPF classification algorithm to classify the new samples in TS by using the OPF
     model;
(11) Compute DR and FAR based on the classified samples;
(12) Set R with the Euclidean distance between the points with coordinates of $(FAR, DR)$ and
     $(0\%, 100\%)$;
(13) return R;
```

In Algorithm 4, the *TR* and *TS* are two new subsets of the NSL-KDD dataset that are used as training and test sets in the current simulation. More details about these sets and the GA parameter settings will be discussed in Section 4.2 of Chapter 4. The GA fitness function is based on minimizing the Euclidean distance between the classification point such as *i* with coordinates of $(FARi, DRi)$ and the perfect classification point with coordinates of $(0\%, 100\%)$ in a receiver operating characteristic (ROC) curve. The best point is a point that its Euclidean distance to the perfect point is the least as compared to other points. The Euclidean distance between the perfect point in the ROC curve and the current point with the coordinate of (FAR, DR) is computed in Step 12 of Algorithm 4. It is clear that the appropriate value of γ should be between 0 and 100 ($0 < \gamma \leq 100$). Therefore, each population in GA, which represents a possible solution, is coded with a binary string vector as a chromosome with a size of 100.

Step 4 (Detecting): In this step, an OPF model will be projected for each pruned training (or evaluation) set. As seen in Figure 3-5, the learning algorithm which works based on the classification error of the OPF model on the evaluation set is used in a few iterations for improving the composition of samples in the training (or evaluation) set. To evaluate the performance of OPF models on the evaluation sets, the following accuracy metric is calculated in each iteration *t* (as mentioned in (Joao P Papa et al., 2009)):

$$L(t) = 1 - \frac{\sum_{i=1}^{c} E(i)}{2c} \tag{3-24}$$

where *c* is the number of the target classes and $E(i)$ is the partial sum error of class *i* that is defined as follows (Joao P Papa et al., 2009):

$$E(i) = \frac{FP(i)}{|Z_2| - |NZ_2(i)|} + \frac{FN(i)}{|NZ_2(i)|}; \qquad i = 1, 2, \ldots, c \tag{3-25}$$

where $|Z_2|$ is the size of Z_2 and $NZ_2(i)$ is the number of samples in Z_2 that belong to class *i*. $FP(i)$ and $FN(i)$ are the false positive and false negative rates of class *i* on Z_2, respectively. After projecting efficient OPF classifiers, the new samples in the test set can be classified.

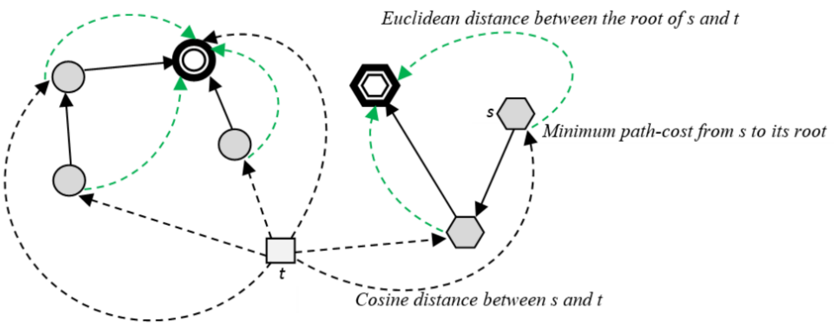

Euclidean distance between the root of s and t
s
Minimum path-cost from s to its root
t
Cosine distance between s and t

Figure 3-7: Classification of a new unlabeled instance by using the projected OPF based on its distance from the OPT roots

To classify a new unlabeled sample (in the learning and the classification phases of the OPF); all of the arcs that connect the unlabeled sample to the samples in the training set are considered (with the aim of finding an optimum path from prototypes to the unlabeled sample). However, in the proposed new version of OPF, called AOPF, the distance between the unlabeled sample and the root (prototype) of each sample in the training set is also considered as an important factor for improving the performance of a traditional OPF (as shown in Figure 3-7). We assume that the proposed approach may select more appropriate training samples as compared to the traditional approach. Hence, it can lead to finding the best optimum path from the prototypes to the unlabeled sample. As seen in Figure 3-7, the cosine distance (CD) was empirically chosen instead of the Euclidean distance (ED) for computing the weight of arcs in the complete weighted graph. The computation time of ED is less than that of CD for high-dimensional datasets (e.g., NSL-KDD dataset); however, our experimental results showed that when we used the CD (as a distance measure for computing the weight of arcs), the performance of OPF was improved. Feature selection methods can be used for reducing the dimension and consequently the computation time reduction of CD in high-dimensional datasets. Moreover, each feature in the dataset may not be relevant to the classification task (e.g. due to the noise); therefore, choosing relevant features (as a proper subset of all features) can improve the performance of classifiers such as OPF.

The CD is calculated using Eq. (3-26). As seen in Eq. (3-26), the cosine similarity is subtracted from 1. Unlike the ED, the cosine similarity measures similarity instead of dissimilarity. The cosine similarity is originally used for finding the similarity of two feature vectors such as $X_s = (x_{s1}, x_{s2}, \ldots, x_{sn})$ and $X_t = (x_{t1}, x_{t2}, \ldots, x_{tn})$:

$$CD = 1 - \frac{X_s \, . \, X_t}{X_s X_t} = 1 - \frac{\sum_{j=1}^{n} x_{sj} \, x_{tj}}{\sqrt{\sum_{j=1}^{n} (x_{sj})^2} \sqrt{\sum_{j=1}^{n} (x_{tj})^2}}) \tag{3-26}$$

In the proposed classification phase of the AOPF algorithm, the ED is used for measuring the distance between a new unlabeled sample and the root of samples in the OPF. As mentioned earlier, an optimum path $P^*(t)$ from S^* to t (as an unlabeled sample) should be found for classifying t, where t is a part of the OPF. Therefore, the optimum path can be found incrementally by evaluating the optimum cost using Eq. (3-17). In the proposed AOPF, the ED between the root of each sample in the training set and the unlabeled sample (i.e., $ED(R(s),t)$) is also considered in the cost function (Eq. 3-27):

$$C(t) = \min\left\{\max\left\{C(s),\ CD(s,t) + \delta \times ED(R(s),t)\right\}\right\} \ ; \quad \forall s \in Z_1 \tag{3-27}$$

where δ is used as a weight which shows the impact of $ED(R(s),t)$. By using Eq. (3-27), the classification algorithm of the OPF is changed as given in the following pseudocode (Algorithm 5):

Algorithm 5. AOPF classification algorithm

```
Input: Classifier [$P_1, C_1, L_1, Z_1^{'}$] where $P_1$, $C_1$, $L_1$ and $Z_1^{'}$ are the optimum-path forest, cost map, label map and ordered list, respectively, the test set $Z_3$ (or the evaluation set $Z_2$), and the pair of $(v, d)$ for feature vector and distance computations
Output: Label $L_2$ and predecessor $P_2$ maps defined for $Z_3$.
Auxiliary: Cost variables tmp and mincost.
 (1)  for each $t \in Z_3$
 (2)      Set $i = 1$ and $mincost = \max\{C_1(k_i), CD(k_i, t) + \delta.ED(R(k_i), t)\}$;
```

```
(3)     Set $L_2(t)=L_1(k_i)$ and $P_2(t)=k_i$ ;
(4)     while $i<|Z_1^{'}|$ and $mincost>C_1(k_{i+1})$
(5)         Compute $tmp=\max\{C_1(k_{i+1}),CD(k_{i+1},t)+\delta.ED(R(k_{i+1}),t)\}$;
(6)         if $tmp<mincost$
(7)             Set $mincost = tmp$;
(8)             Set $L_2(t)=L(k_{i+1})$ and $P_2(t)=k_{i+1}$ ;
(9)         end
(10)        Set $i=i+1$ ;
(11)    end
(12) end
(13) return $[L_2,P_2]$;
```

In Algorithm 5, $Z_1^{'}$ consists of all training nodes in non-decreasing order of the optimum path cost. C_1 and L_1 are the optimum path cost and the class label of the corresponding sample in $Z_1^{'}$, respectively. Moreover, L_2 and P_2 represent the parent (predecessor) and the label of each classified sample in tthe est set, respectively. Algorithm 5 is akin to the original classification algorithm (JoãO P Papa et al., 2012); however, Steps 2 and 5 are altered. As seen in Step 5 of Algorithm 5, a training sample (e.g., k_{i+1}) whose root (e.g., $R_{k_{i+1}}$) is distant from the unlabeled sample *t* (based on $ED(R(k_i),t)$), will be ignored by Step 6, because the long-distance represents the low similarity between *t* and $R_{k_{i+1}}$. It means that the *k_{i+1}*'s OPT (in which the $R_{k_{i+1}}$ is the OPT's root) may not be an appropriate OPT for t. In other words, the class of k_{i+1} may not be appropriate for *t*. As mentioned earlier, this approach can select more appropriate training samples as compared to the traditional approach in finding the best optimum-path from the prototypes to the unlabeled sample.

In this study, the GA is used for finding an optimum value for δ. The following pseudo-code describes the proposed algorithm for obtaining the fitness value (Algorithm 6):

Algorithm 6. The proposed algorithm for obtaining fitness value in finding the optimum value of δ

```
Input: A binary string with size of N as a chromosome (corresponding to $\delta$). TR and TS are the training set and test set, respectively.
Output: Fitness value R.
Steps:
    (1) Set $cnt$ as the count number of 1 values in the input binary string;
    (2) Set $\delta = cnt \times 0.001$;
    (3) Create an OPF model from the training set by the OPF training algorithm;
    (4) Perform AOPF classification algorithm to classify new samples in the test set;
    (5) Compute DR and FAR based on the classified samples;
    (6) Set R with the Euclidean distance between the points with coordinates of $(FAR, DR)$ and $(0\%, 100\%)$;
    (7) return R;
```

We have experimentally found that the right amount of δ should be between 0 and 1 ($0 \le \delta \le 1$). Therefore, each population in GA which represents a possible solution of appropriate δ is coded with a binary string vector as a chromosome with a size of 1000. The value of each bit is 0.001; it means the threshold will be accurate up to three decimal places

3.2 Insider-Attack Identification

As mentioned in Chapter 1, every layer of 6LoWPAN stack can be threatened by various attacks. However, because the routing attacks can take control of the information flow of networks, most of attacks focus on disturbing the network layer functionalities through the routing attacks, especially selective forwarding and sinkhole attacks. In order to detect these kinds of attacks and determine their types, this thesis presents a novel real-time hybrid intrusion detection framework that was based on anomaly-based and specification-based intrusion detection. Actually, in order to detect anomalies and identify the malicious nodes, the specification-based IDS agents located in 6LoWPAN nodes send their analytical results to the border router of 6LowPAN (6BR). Then anomaly-based IDS agent located in 6BR detects anomalies and identifies malicious nodes. Note the anomaly detection engine, which is based on MapReduce architecture, uses unsupervised OPF called Optimum-path Foreset Clustering (OPFC) for identifying anomalies. Moreover, the proposed IDS will make its final decision about detected anomalies and their types through a voting mechanism. Before introducing the proposed method, some of the fundamental concepts used in the presented real-time IDS will be reviewed.

3.2.1 Unsupervised Optimum-Path Forest

In 2009, Rocha et al. (Rocha, Cappabianco, & Falcão, 2009) introduced an unsupervised machine-learning algorithm for data clustering based on the graph theory called Optimum-Path Forest Clustering (OPFC). They reduce the clustering problem, as a pattern recognition problem, into optimal graph partitioning in a given feature space. In the OPFC, each sample in the dataset (represented by a feature vector) is shown as a node in the k-nearest neighbor graph (G_{k-nn}) that is connected with its *k* best neighbors in given feature space (Costa et al., 2015). In OPFC, the arcs are weighted by the distance between each pair of nodes and the nodes are weighted by the probability density function (pdf) of each node that is based on the distance between the samples and their k-nearest neighbors (Costa et al., 2015). When G_{k-nn} is created, the OPFC algorithm will find one sample (node) at each maximum pdf as a root of a dome or cluster which includes dense samples in the feature space. Then, an Optimum-Path Tree (OPT) will be created from each root to every node in the influence zone (cluster) such that each OPT node will be strongly connected to its root as compared to other obtained roots in the G_{k-nn} (Rocha et al., 2009). The OPF will be composed by a union of the OPTs.

Suppose Z as a clustering dataset such that each $s \in Z$ is shown by $\vec{v}(s)$ in the given feature space. The $G_{k-nn} = (Z, A_k)$ is defined such that the arcs in A_k connect k-nearest neighbors in the feature space (Costa et al., 2015). In G_{k-nn}, each arc $(s,t) \in A_k$ is shown by $d(s,t) = \vec{v}(t) - \vec{v}(s)$ which denotes the Euclidean distance between the corresponding feature vectors of *s* and *t*. As mentioned earlier, each $s \in Z$ is weighted by a probability density function defined as follows (Costa et al., 2015):

$$p(s) = \frac{1}{\sqrt{2\pi\sigma^2}\,|A_k(s)|} \sum_{\forall t \in A_k(s)} exp\left(\frac{-d^2(s,t)}{2\sigma^2}\right) \tag{3-28}$$

where $|A_k(s)| = k$, $\sigma = \frac{d_f}{3}$, and d_f is the maximum arc weight in G_{k-nn} . It is noted that $A_k(s)$ is the neighbor set of $s \in Z$. Since the arcs in A_k are asymmetric, hence the symmetric arcs should be added to the plateaus of pdf as given in Eq. (3-29) to guarantee a single root per maximum (cluster) (Rocha et al., 2009):

$$if\ t \in A_k(s), s \notin A_k(t)\, and\ p(s) = p(t)\, then\, A_k(t) = A_k(t) \cup \{s\} \tag{3-29}$$

In OPFC, a path π_t that includes a sequence of distinct adjacent nodes starts from the root of $t\,(R(t))$ and is terminated with *t*. Note that a path with one sample like $\pi_t = \langle t \rangle$ called trivial and $\pi_s . \langle s,t \rangle$ represents the concatenation of π_s and $\langle s,t \rangle$ denoting a path and an arc, respectively (Costa et al., 2015). In the OPFC algorithm, a connectivity function $f\,(\pi_t)$ assigns a path cost to each path π_t. For other paths, such as τ_t, if $f\,(\pi_t) \geq f\,(\tau_t)$ then π_t will be an optimum path (Costa et al., 2015). Among all possible paths from a root in the maxima of the pdf, the OPFC will assign a path as the optimum-path $P^*(t)$ to the $t \in Z$ such that the minimum density value along the path is maximum (Costa et al., 2015):

$$v(t) = \underset{\forall \pi_t \in (Z, A_k)}{max} \{f_{min}(\pi_t)\} \tag{3-30}$$

where $f_{min}(\pi_t)$ is defined as follows:

$$f_{min}(t) = \begin{cases} p(t) & if\ t \in R \\ p(t) - \delta & otherwise \end{cases} ; \quad f_{min}(\pi_s . \langle s,t \rangle) = min\{f_{min}(\pi_s), p(t)\} \tag{3-31}$$

where $\delta = min_{\forall (s,t) \in A_k | p(t) \neq p(s)} |p(t) - p(s)|$ (Notably, larger values of δ will lead to the maxima reduction) and R is the set of OPF's root which is found on-the-fly, with one element per each maximum of the pdf (Costa et al., 2015). The algorithm tries to maximize the connectivity map $v(t)$ *by* computing an OPF, which assigns the predecessor $p(t)$ to each sample $t \notin R$ or mark nil when $t \in R$. The following pseudo-code describes the OPFC algorithm (Costa et al., 2015):

Algorithm 7. OPFC implementation

Input: Graph (Z, A_k) and distance function *d*.
Output: Optimum-path forest *P*, connectivity map *V* and label map *L*.
Auxiliary: Priority queue *Q*, density map *p*, variables *tmp* and $l \leftarrow 1$

```
(1)  for each $s \in Z$
(2)      Compute $p(s)$ by Eq. (3-28);
(3)      Set $P(s) \leftarrow nil$ , $V(s) \leftarrow p(s) - \delta$ , and insert s in Q;
(4)      while Q is not empty, do
(5)          Remove from Q a sample such that $V(s)$ is maxim;
(6)          for each $t \in A_k(s)$ such that $V(t) < V(s)$
(7)              Compute $tmp \leftarrow \min\{V(s), p(t)\}$ ;
(8)              if $tmp > V(t)$ then
(9)                  Set $L(t) \leftarrow L(s), P(t) \leftarrow s, V(t) \leftarrow tmp$ ;
(10)                 Update position of t in Q;
```

```
(11)                end
(12)            end
(13)        end
(14) end
(15) return a classifier $[P,V,L]$;
```

An OPFC model classifies a new sample to a special cluster (that is created in the OPFC algorithm), by finding a root which provides the optimum path to the new sample. Therefore, in order to classify a new sample $t \in Z' \setminus Z$ according to the neighbors of *t* ($s \in A_k(t) \subset Z'$), the algorithm computes the pdf of *t* by using Eq. (3-28). Then, the optimum path can be found incrementally by evaluating the optimum cost as follows (Rocha et al., 2009):

$$v(t) = \max_{\forall (s,t) \in A_k} \{min\{v(s), p(t)\}\} \tag{3-32}$$

Suppose $s^* \in Z$ is the best node that satisfies Eq. (3-32), so the classifier selects the cluster of s^* as the class of *t*.

One of the main challenges in the OPFC algorithm is determining the optimal value for *k* (k-nearest neighbors). Generally, *k* can fall in $[\mathrm{k}_{\min}, \mathrm{k}_{\min}]$ ($1 \leq \mathrm{k}_{\min} < \mathrm{k}_{\max} \leq |\mathrm{Z}|$) (Rocha et al., 2009). Rocha et al. (Rocha et al., 2009) used the graph-cut metric for finding the proper value for *k*. suppose $\frac{1}{\mathrm{d}(\mathrm{s},\mathrm{t})}$ is the weight of edges in G_{k-nn} . According to Algorithm 7 for each $k \in [1, (|N|-1)]$ the value of graph-cut can compute as follows:

$$C(k) = \sum_{i=1}^{c} \frac{W_i'}{W_i + W_i'} \tag{3-33}$$

$$W_i = \sum_{(s,t) \in A_k \,|L(s)=L(t)=i} \frac{1}{d(s,t)} \tag{3-34}$$

$$W_i' = \sum_{(s,t) \in A_k \,|L(s)=i\,,\, L(t) \neq i} \frac{1}{d(s,t)} \tag{3-35}$$

The best value for graph-cut can be obtained by minimizing $C(k)$ such that W_i' employes all arc weights between cluster *i* and other clusters, and W_i employes all arc weights within a cluster. To see more details about the OPFC algorithm, refer to (Costa et al., 2015).

3-2-2 MapReduce Approach

Today, one of the main challenges of well-known corporations, such as Google or Yahoo, is the maintenance and analysis of big data for extracting useful knowledge. MapReduce approach (Dean & Ghemawat, 2008) is an efficient solution for big data problems. This approach employs algorithms that have parallelism capabilities in a parallel space. The MapReduce, which was first presented by Google, is a parallel programming model that is inspired by a functional programming language such as Lisp. This approach hides the details and complexity of parallel computation, data distribution, and fault tolerance (Dean & Ghemawat, 2008). In this approach, a big dataset is split into smaller datasets and stored on different machines. Then, these machines process the smaller datasets in parallel and finally, the results will be integrated. In fact, this algorithm reduces the intermediate space to the final solution space.

The MapReduce approach includes two main phases: (a) Map and (b) Reduce. As shown in Figure 3-8, in the Map phase, input data is split into smaller segments named chunk. Then, they are delivered to some machines called mappers who are responsible for the mapping operation (Aridhi, Lacomme, Ren, & Vincent, 2015). Then, each mapper converts the content of the chunk to a sequence of key-value pairs and consequently for each pair, a list of key-value pairs is generated by calling the user-defined "map" function ($map(k_1,v_1) \rightarrow list(k_2,v_2)$) (Aridhi et al., 2015). In the Reduce phase, the MapReduce framework performs sorting based on the keys and collects each key-value pair with the same key and sends them to the reducer node. In fact, a group consists of key-value pairs, which have the same key, will be produced in the Reduce phase for each generated key in the Map phase. Then, the user-defined "reduce" function accepts the mediate keys with a set of values representing the dimension of keys and merges the values by converting them to a smaller value (i.e., reducing the dimension) ($reduce(k_2, list(v_2)) \rightarrow list(v_3)$) (Dean & Ghemawat, 2008).

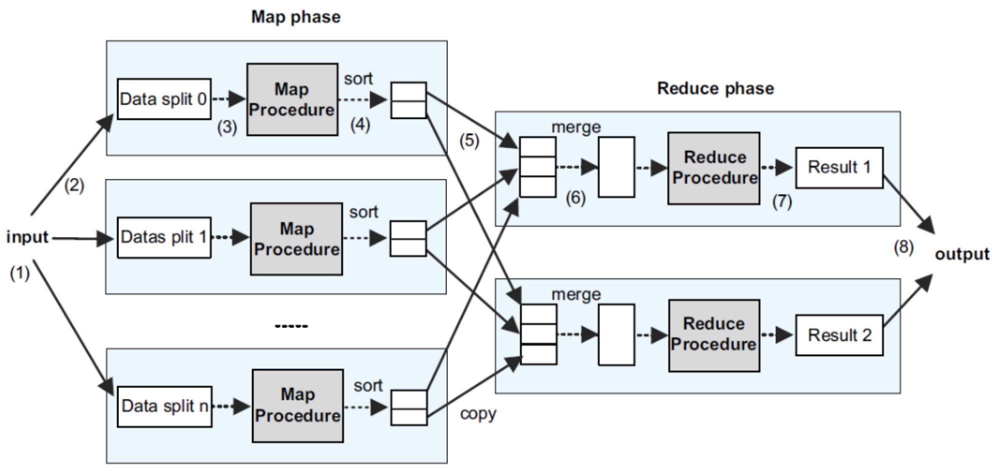


Figure 3-8: A block diagram of MapReduce operation

3-2-3 The Proposed Real-Time Anomaly-Based and Specification-Based IDS

As mentioned earlier, this study concentrates on detecting selective forwarding and sinkhole attacks as well-known insider-attacks in IoT which can get the control of data flow in 6LoWPAN. In this study, the intrusion detection task in the 6LoWPAN is presented as the following problem:

Suppose a 6LoWPAN network, which includes m+n homogeneous sensor nodes ($S = L \cup R \mid L = \{l_1, \ldots, l_m\},\ R = \{r_1, \ldots, r_n\}$) and one power root node as the 6BR. Assume that only m leaf nodes are responsible to generate data (e.g., by sensing the ambient temperature) as the source nodes and sending them (in packets' form) through n router nodes to the root node. The routing packets are based on the RPL protocol by using the DODAG graph. We assume that the RPL protocol supports solely unidirectional traffic from the leaves (sources) to the root. In this network, the attacks are caused by malicious nodes ($s_a \in R$) that perform as valid nodes. The goal of the proposed model is detecting outlier behaviors (as attacks) and identifying the malicious nodes that cause these behaviors.

Figure 3-9 shows the block diagram of the proposed framework. As seen, the proposed model is a hybrid method based on centralized and distributed intrusion detection models. This model consists of an Anomaly Agent-based IDS (AA-IDS) and some Specification Agent-based IDSs (SA-IDSs). In the proposed model, each router node monitors the input/output traffic and identifies the potential malicious nodes by using the SA-IDS, independently. Then, they send analysis results to the 6BR by embedding them into packets to be forwarded. In the 6BR, the AA-IDS projects some clustering models (based on the OPFC algorithm) for clustering the collected data and detecting anomalies. The task is performed based on features of each source node that were extracted from the incoming traffic. Then, the algorithm makes a final decision based on the local results of SA-IDS agents (which are hosted in the router nodes) and the global analysis results of the AA-IDS agent. In the proposed model, it is assumed that the attack traffic is much less than the normal traffic (Eik Loo, Yong Ng, Leckie, & Palaniswami, 2006). In the proposed framework, the intrusion detection and the identification of the malicious nodes are performed in the following three stages:

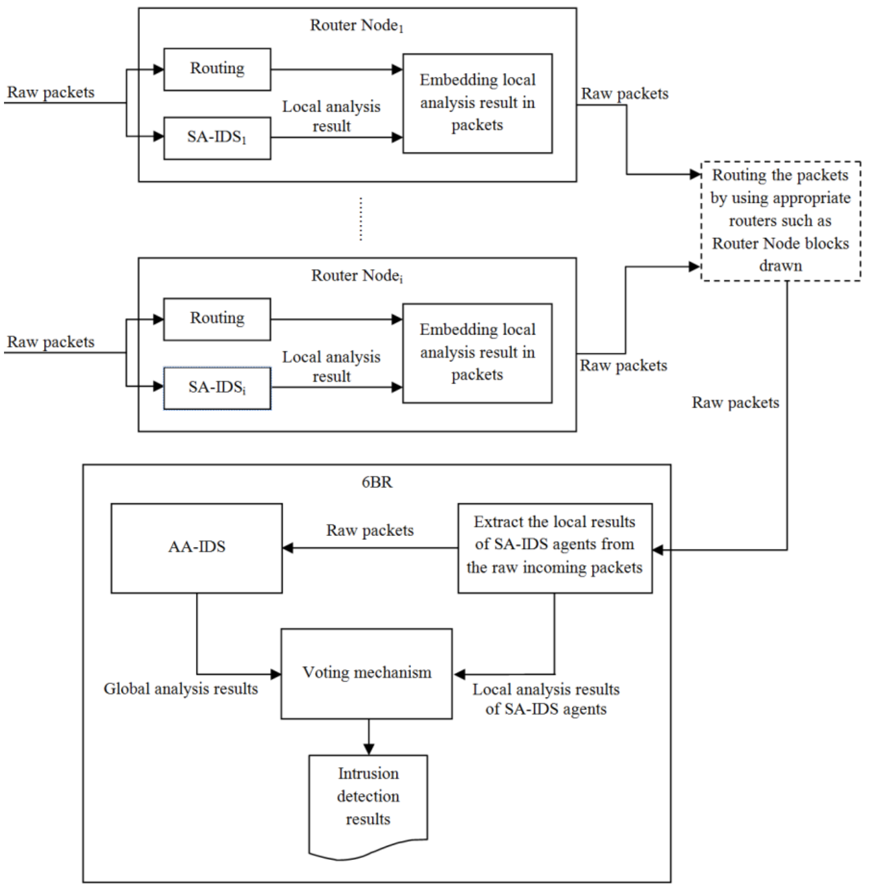

Router Node$_1$
Raw packets
Routing
SA-IDS$_1$
Local analysis result
Embedding local analysis result in packets
Raw packets
Routing the packets by using appropriate routers such as Router Node blocks drawn
Router Node$_i$
Raw packets
Routing
SA-IDS$_i$
Local analysis result
Embedding local analysis result in packets
Raw packets
Raw packets
6BR
AA-IDS
Raw packets
Extract the local results of SA-IDS agents from the raw incoming packets
Global analysis results
Voting mechanism
Local analysis results of SA-IDS agents
Intrusion detection results

Figure 3-9: Block diagram of the proposed intrusion detection framework

Stage 1 (identifying malicious nodes): the goal of this stage is identifying the suspicious nodes that may cause sinkhole and selective forwarding attacks in the 6LoWPAN. Identifying the suspicious nodes is based on a stand-alone architecture and is performed by using some light SA-IDSs that are located in the router nodes. Table 1 reviews the operation of the selective forwarding and sinkhole attacks in a WSN (Yan, Wang, Wang, & Liu, 2010). As seen in Table 3-1, the selective forwarding attack influences packet transmission, while the sinkhole attack has an adverse effect on the routing of packets.

Table 3-1: The comparison of two attacks in a WSN

Attack name	Behavior
Selective forwarding	Data forwarding misbehavior
Sinkhole	Route updating misbehavior

In the proposed model, the SA-IDS works as detailed below to detect these attacks:

1) Identifying suspicious nodes launching sinkhole attack: According to the sinkhole action which was mentioned in Section 2.2.2, Figure. 3-10 shows the influence of malicious node *S* on node A in the DODAG.

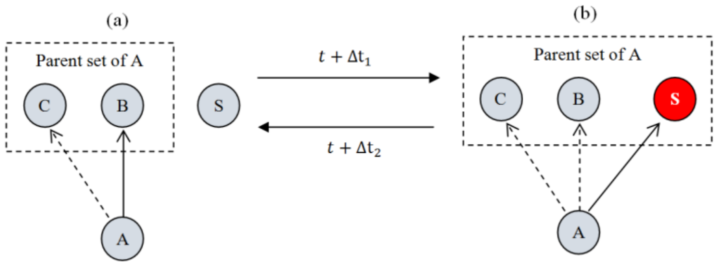


Figure 3-10: Sinkhole attack action in the DODAG

As seen in Figure 3-10, when node *S* (as a sinkhole attacker) wants to launch a sinkhole attack, it attracts node A to route the traffic through node *S*. Hence, node *A* adds node *S* to the parent set and selects it as a preferred parent for routing its packets. After a short time, node *S* resumes again its normal behavior. Therefore, node A removes it from the parent set and selects node *B* again as the preferred parent. Based on this action, the SA-IDS which is located in the router node (such as *A*) computes the rate of change in preferred parent and also the rate of change in the parent set at each time-slot Δw (as the non-traffic-related features) for identifying suspicious node (based on the routing table). If these values are greater than a predefined threshold, then the SA-IDS will identify and introduce the suspicious node. The following pseudo-code describes how a suspicious node (i.e., the agent of sinkhole attack) is identified by its lower node in the SA-IDS agent (Algorithm 1):

```
Algorithm 8. Detecting sinkhole suspicious node
Input: ParentSetThreshold and PreferredParentThreshold which represent the threshold for detecting anomaly behavior in
the parent set and the preferred parent, respectively. $\Delta w$ and $t_0$ represent the length of the time window and the
current time, respectively.
Output: SID represents the ID of a suspicious node.
Auxiliary: $cnt_1$ and $cnt_2$ represent the count of change in length of the parent set and the preferred parent,
respectively. CurrentLength and CurrentPreferredParnet represent the current length of the parent set and the ID of the
current preferred parent, respectively. PHList represents the preferred parent history list (with the $\langle PID, timeSpan \rangle$
item list structure where timeSpan shows the time interval that the node with PID node ID is a parent). LastChangeTime
represents the time of the last change in the preferred parent.
Initialization:
    • Set $t = t_0$ , $cnt_1 = 0$ , $cnt_2 = 0$ , $SID = nil$ and $LastChangeTime = t_0$;
    • Set $CurrentLength = Length(ParentList)$ and $CurrentPreferredParnet = PreferredParent$ ;
```

```
Steps:
(1)  while $t < t_0 + \Delta w$ , do
(2)      if $CurrentLength \neq Length(ParentList)$ then
(3)          Set $cnt_1 = cnt_1 + 1$;
(4)          Set $CurrentLength = Length(ParentList)$;
(5)      end
(6)      if $CurrentPreferredParnet \neq PreferredParent$ then
(7)          Set $cnt_2 = cnt_2 + 1$;
(8)          Set $timeSpan = t - LastChangeTime$ ;
(9)          Add $\langle CurrentPreferredParnet, timeSpan \rangle$ to PHList;
(10)         Set $CurrentPreferredParnet = PreferredParent$ ;
(11)         Set $LastChangeTime = t$ ;
(12)     end
(13)     Set $t = t + \Delta t$ ;
(14) end
(15) if $cnt_1 / \Delta w > ParentSetThreshold$ and $cnt_2 / \Delta w > PreferredParentThreshold$ then
(16)     Compute the total duration of each PID in PHList and add them as an ordered pair $(PID, timeSpan)$ to a
         tempList;
(17)     Sort tempList according to the timeSpan of node's ID and set SID as the node's ID with the minimum value;
(18) end
(19) return SID;
```

In the proposed model, we assume that the time interval that a malicious node is selected as the preferred parent is shorter than the corresponding time interval for other nodes (presenting the normal behavior). Therefore, as seen in lines 16 and 17 of Algorithm 8, if the non-traffic-related feature values are greater than the predefined thresholds, then the algorithm computes the total duration that each node was selected as a preferred parent of the host node (a node which the SA-IDS is located in it). Then, a node that corresponds to the minimum time interval will be introduced as a suspicious node.

2) Identifying suspicious nodes launching selective forwarding attack: According to the selective forwarding action which was mentioned in Section 2.2.2, Figure 3-11 shows the influence of malicious node S on node A in the DODAG.

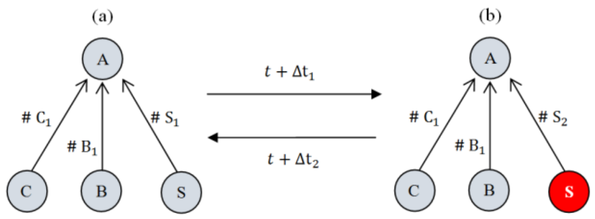


Figure 3-11: Selective forwarding attack action in the DODAG

When a malicious node wants to launch a selective forwarding attack, it selectively forwards packets to the root. So, a preferred parent node can identify the suspicious node by knowing the approximate number of packets received from each node (such as node S in Figure 3-11). In the proposed model, SA-IDS which is located in a router node (such as *A*) computes the packet receiving rate and the last packet received time at each time-slot Δw (as the traffic-related features) for each child node for identifying the suspicious node. For each child node such as *S*, if the packet receiving rate is smaller than a predefined threshold, and the last packet received timestamp is greater than a predefined threshold, then the SA-IDS will introduce node *S* as a suspicious node. The following pseudo-code describes how a suspicious node (i.e., the agent of selective forwarding attack) is identified by its upper node in the SA-IDS agent (Algorithm 9):

```
Algorithm 9. Detecting selective forwarding suspicious node
Input: PacketReceivingThreshold and TimeDelayThreshold represent the threshold for detecting anomalous behavior in
packet receiving rate and the last packet receiving timestamp, respectively. $\Delta w$ and $t_0$ denote the length of the time
window and the current time, respectively.
Output: SIDList represents the list of suspicious node IDs.
Auxiliary: packetItem with the format of the packet introduced in Figure 3-12. ChildList with the
$\langle ID, time, PacketReceiving \rangle$ item list structure which used the child's packet information where time shows the last
time that a packet is received and PacketReceiving represents the number of packets that have been received.
Initialization:
  • Set $t = t_0$ , $SIDList = nil$ and $childList = nil$ ;
Steps:
(1) while $t < t_0 + \Delta w$ do
(2)       Set $packetItem = getPacket()$;
(3)       Set $NodeID = packetItem.getRouterID()$;
(4)       Set $TimeStamp = packetItem.getRouterTimeStamp()$;
(5)       if ChildList has an item with a key equals to NodeID then
(6)           Set index = index of an item in ChildList that has a key equals to NodeID
(7)           Set $ChildList[index].TimeStamp = t - ChildList[index].TimeStamp$ ;
(8)           Set $ChildList[index].PacketReceiving = ChildList[index].PacketReceiving + 1$;
(9)       end
(10)      else
(11)          $ChildList.Add(\langle NodeID, t - t_0, 1 \rangle)$;
(12)      end
(13)      Set $t = t + \Delta t$ ;
(14) end
(15) for each item in ChildList do
(16)      if $item.PacketReceiving / \Delta w < PacketReceivingThreshold$ and
           $item.TimeStamp > TimeDelayThreshold$ then
(17)          $SIDList.Add(item.NodeID)$;
(18)      end
(19) end
(20) return $SIDList$ ;
```

In step 2 of Algorithm 9, $getPacket()$ represents a function that returns the current received packet information. Notably, we design and implement a WSN based on the RPL routing protocol in this study for simulating 6LoWPAN functionality (see Chapter 4). Generally, the structure of data packets in the simulations consists of two main parts (as shown in Figure 3-11): (a) data (fields) and (b) data access interface (functions).

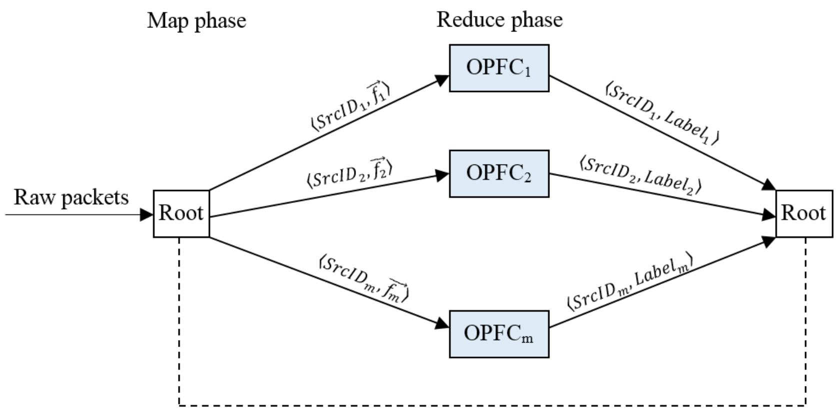
In Figure 3-12, *SrcID* and *SrcTimeStamp* represent the ID of the source node and the time of packet sending by the source node, respectively. *RouterID* and *RouterTimeStamp* represent the ID of the last router node (before the current node) and its forwarding packet time, respectively. *RouterID* and *RouterTimeStamp* are obtained by using $getRouterID()$ and $getRouterTimeStamp()$ functions, respectively. *HopCount* shows the number of hops taken by the packets and each router node increments it by $incHopCount(\ )$ function. One of the main fields used by the SA-IDS agent is SuspiciousList. In fact, when SA-IDS identifies a suspicious node, then SA-IDS adds an item with format $\langle SID, Type, TimeStamp \rangle$ (as the suspicious node information) to the *SuspiciousList* by using $addSuspiciousList(\ )$. The *SID*, *Type*, and *TimeStamp* represent the suspicious node's ID, the type of possible attack, and the identification time of suspicious nodes, respectively. We assume that the router node cannot access the *Data* and *SuspiciousList* with the aim of manipulating values).

Fields	**Functions**
- SequenceNumber	+ getSequenceNumber( )
- SrcID	+ getSrcID( )
- SrcTimeStamp	+ getSrcTimeStamp( )
- HopCount	+ incHopCount( )
- RouterID	+ getHopCount( )
- RouterTimeStamp	+ getRouteID( )
- SuspiciousList	+ getRouterTimeStamp( )
- ReceivingTimeStamp	+ addSuspiciousList( )
- Data	+ getReceivingTimeStamp( )

Figure 3-12: Structure of data packets in the simulated WSN

Stage 2 (anomaly detection in 6BR): in this stage, the AA-IDS creates a sample for each source node by extracting four traffic-related features from the raw received packet of the source node at each time-slot Δw : (a) packet receiving rate; (b) packet dropping rate; c) average latency; and (d) maximum hop-count. Then, the AA-IDS projects a clustering model based on an unsupervised OPF algorithm for each source node by using its generated samples. The algorithm selects a cluster (or clusters) including a few samples and then labels the samples as anomalous for each projected model.

By increasing the number of source nodes, the sequential projection of clustering models will be time-consuming that is not acceptable for a real-time model. The proposed anomaly detection method has the capability of parallelism because projecting and using clustering models are independent processes. In this study, we inspired the MapReduce approach to improving the speed of projecting models and anomaly detection. In fact, we proposed an anomaly detection method based on the MapReduce architecture. In other words, if an appropriate platform (hardware/software) is prepared, then the model can run in parallel on a distributed space based on the MapReduce architecture. Figure 3-13 shows the general architecture of the proposed anomaly detection model which is based on the MapReduce approach.



Map phase
Reduce phase
Raw packets
Root
$\langle SrcID_1, \vec{f_1} \rangle$
$\langle SrcID_2, \vec{f_2} \rangle$
$\langle SrcID_m, \vec{f_m} \rangle$
OPFC$_1$
OPFC$_2$
OPFC$_m$
$\langle SrcID_1, Label_1 \rangle$
$\langle SrcID_2, Label_2 \rangle$
$\langle SrcID_m, Label_m \rangle$
Root

Figure 3-13: General architecture of anomaly detection model based on the MapReduce approach

As seen in Figure 3-13, the root node (i.e., the 6BR) extracts mentioned traffic-related features from the receiving raw packets in each time-slot and creates a new sample for source nodes. Then, it sends the sample's information with key-value pair format to a node (i.e., the reducer) that is responsible to work with a special key. This format includes source ID as the key and feature vector as the value. Then, the reducer node projects a clustering model by using its samples which are received from the mapper node. As mentioned earlier, we assume that a cluster with fewer samples is an anomaly; hence, if the new sample belongs to this cluster, it is classified as anomalous and otherwise, it is classified as a normal sample. So, the reducer node returns a new key-value pair with $\langle SID, Label \rangle$ format (in response to the incoming key-value pair) to the root node. It is noted that the key and the value are source ID's sample and its label (i.e., anomalous/normal), respectively. The projecting of a clustering model in reducer continues until the number of samples for each source becomes equal to a threshold. Then, the reducer works as a classifier that means if the new received sample (from the mapper) belongs to the anomalous cluster(s), it will be introduced as an anomalous sample. In the step of extracting features for producing a new sample for each source node, such as A, the packet dropping rate is computed based on the following steps at each time-slot:

1) Sort the received packets from node *A* based on its SequenceNumber.
2) Calculate the sum of the distances between every two consecutive packets (based on *SequenceNumber*) and return the result as the packet dropping rate (for simplicity, we assume that each packet is sent only once).

Moreover, other features such as the maximum hop-count and the average latency are computed as follows:

$$\forall i \in L : MaxHopCount = Max\left(packet_j^i.getHopCount\left(\right)\right) \mid j \in P^i \tag{3-36}$$

$$\forall i \in L : Averagelatency = \frac{\sum_{j \in P^i} packet_j^i.getReceivingTimeStamp\left(\right) - packet_j^i.getSrcTimeStamp\left(\right)}{P^i} \tag{3-37}$$

where L is the set of source nodes in the network. P^i is the set of received packets from the *i*-th source in time-slot Δw and $\left\|P^i\right\|$ is the number of its members.

Stage 3 (anomaly detection decision based on a voting mechanism): in this stage, the proposed framework employs the first stage results to make a decision about abnormities detected in the second stage. The following pseudo-code describes the voting process at this stage (Algorithm 10).

Algorithm 10. A voting mechanism for making a decision about intrusion detection

```
Input: $AR$ and $SR$ as anomaly-based and specification-based detection results.
Output: $AttackList$ as the list of detected attacks.
Steps:
(1)  for each $\langle SrcID_i, Label_i \rangle \in AR$ do
(2)      if $Label_i = 1$ then
(3)          Set $index$ = index of $SR$ where $SrcID = SrcID_i$;
(4)          If $index \neq nil$ then
(5)              Set $AttackList = \langle SrcID_i, SR.SID, SR.Type \rangle$;
```

```
(6)         end
(7)     end
(8) end
(9) return $AttackList$ ;
```

So, Algorithm 10 returns a list which includes the information about source node ID, malicious node ID, and the type of attack, respectively (by this format: $\langle SrcID, AttckNodeID, AttckType \rangle$).

Chapter 4: Experimental Results

Over this chapter, the simulation results of the presented intrusion detection model are reported and discussed in separate parts. In the first part, the proposed cyber detection system, which consists of the preprocessing and offline misuse-based IDS phases is evaluated in details. Then, in the second part, the experimental results of employing the proposed real-time IDS in a simulated RPL-based WSN are evaluated, completely. However, before anything, the applied standard metrics that are used in evaluating the proposed methods will be reviewed.

4.1 Evaluation Criteria

There are diverse standard evaluation criteria, which are employed to evaluate the performance of the proposed methods. As can be seen in Eqs. 4-1, 4-2 and 4-3, the True Positive Rate (TPR) or DR, the False Positive Rate (FPR) or FAR and the Accuracy Rate (AR) are calculated as follows:

$$TPR = \frac{TP}{TP + FN} \tag{4-1}$$

$$FPR = \frac{FP}{TN + FP} \tag{4-2}$$

$$AR = \frac{(TP + TN)}{(TP + TN + FN + FP)} \tag{4-3}$$

where *TP* is the number of positive instances (attacks) that are classified correctly and *FN* is the number of positive instances (attacks) that are classified incorrectly. Furthermore, *FP* is the number of negative instances (normal) that are classified incorrectly and *TN* is the number of negative instances (normal) that are classified correctly. In addition to the aforementioned metrics, the Cost Per Example (CPE), which is widely used in evaluating the intrusion detection systems, is computed as follows:

$$CPE = \frac{1}{T}\sum_{i=1}^{m}\sum_{j=1}^{m} CM(i,j).C(i,j) \tag{4-4}$$

Table 4-1: Cost matrix for NSL-KDD dataset

Actual	**Predicted**				
	Normal	DoS	Probe	R2L	U2R
Normal	0	2	1	2	2
DoS	2	0	1	2	2
Probe	1	2	0	2	2
R2L	3	2	2	0	2
U2R	4	2	2	2	0

where *T* and *m* denote the total number of samples in the test set and the number of classes in the classification problem, respectively. CM is a square matrix, called confusion matrix, in which rows represent the actual classes, while the columns correspond to the predicted classes. Each cell, such as $CM(i,j)$ in the confusion matrix, represents the number of samples that originally belong to class *i*, but they are classified in class *j*. The cost penalties for this misclassification are represented by a cost matrix that is presented in Table 4-1 for the NSL-KDD dataset (Agarwal & Joshi, 2001).

4.2 The Experimental Results of the Proposed Feature Selection Algorithm

As mentioned previously, in order to select the optimal features of the NSL-KDD dataset, which is used for simulating the Internet (or LANs) traffic, this thesis presents a hybrid feature selection algorithm called MI-BGSA. In this section, an anomaly-based intrusion detection method that is based on SVM is used to evaluate the performance of the proposed algorithm. As shown in Table 4-2, the SVM was trained using 10,000 selected instances from NSL-KDD dataset in 100 iterations, and then, it was evaluated using another 5,000 selected instances of NSL-KDD dataset. Moreover, the applied dataset is normalized based on the normalization method that is mentioned in section 3.1.2.4.

Table 4-2: Size of the training and test datasets

dataset	**# Instances**	**# Normal Instances**	**# Anomaly Instances**
Training set	10,000	5,249	4,751
Test set	5,000	2,137	2,863

In this experiment, I compared the proposed model with the standard BGSA and the binary PSO (BPSO) (Nezamabadi-pour, Rostami-Shahrbabaki, & Maghfoori-Farsangi, 2008) feature selection methods. The parameters setting of the proposed MI-BGSA, BGSA, and BPSO feature selection methods is reported in Tables 3, 4 and 5, respectively.

Table 4-3: Parameters setting of MI-BGSA-based feature selection method in simulations

Parameters	**MI-BGSA**
N	5
t_{max}	30
G_0	100
v_{max}	6
β	0.5
δ_{MI}	0.12

Table 4-4: Parameters setting of BGSA-based feature selection method in simulations

Parameters	**BGSA**
N	5
t_{max}	100
G_0	100
v_{max}	6

Table 4-5: Parameters setting of BPSO-based feature selection method in simulations

Parameters	**BPSO**
N	5
t_{max}	100
v_{max}	6
c_1	2
c_2	2
w	0.2

As shown in Tables 4-3, 4-4 and 4-5, the feature selection methods assumed n = 5 (as an initial population) and $t_{\max} = 100$ (as the maximum number of iterations). The maximum velocity and the gravitational constant are set to $v_{\max} = 6$ and $G_0 = 100$, respectively. As seen in Table 4-3, we set $\beta = 0.5$ similar to (Amiri et al., 2011). The initial value of the inner-optimization module threshold in MI-BGSA is $\delta_{MI} = 0.12$, which is computed by the proposed GA-based method (Algorithm 2 in Chapter 3). In the BPSO algorithm w is the inertia weight or the momentum of particles that shows the effect of the previous velocity on the new velocity and c_1 and c_2 control the stochastic influence of the cognitive and social components of the overall velocity of a particle (Sheikhan, 2014). We assumed $c_1 = c_2 = 2$ and $w = 0.2$.

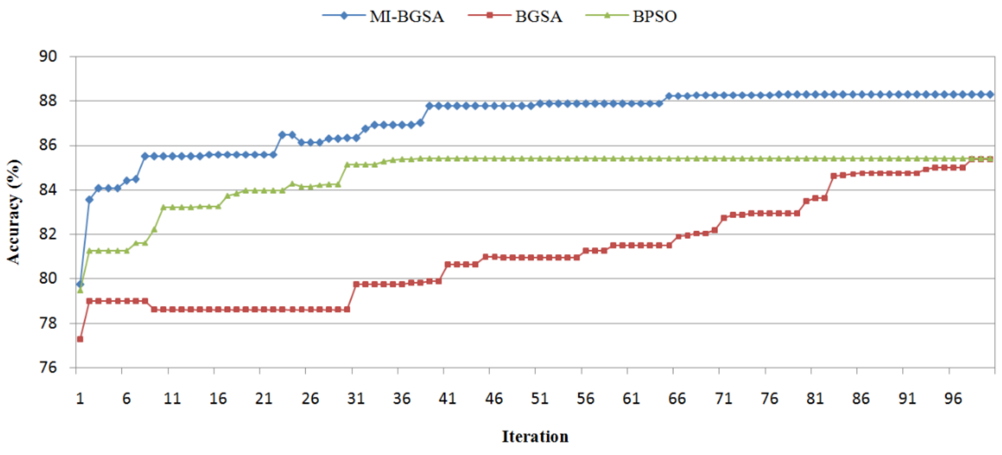


Figure 4-1: Average accuracy rate when using SVM classifier and MI-BGSA/BGSA/BPSO feature selection methods

In this thesis, the feature selection methods are implemented by MATLAB R2014a on a PC with an Intel(R) Core (TM) i5-4460, CPU 3.20 GHz and 8 GB RAM. All the feature selection methods were run three times and the average of AR, DR, and FPR are reported.

In this experiment, the optimum feature subset was considered. It means that the proposed method was found to be the best subset of features that maximizes the fitness function. The performance of the proposed model was compared to BGSA and BPSO, in terms of AR, DR, and FPR. We run the feature selection methods three times using SVM classifier. The average runtime behavior of AR, DR, and FPR in 100 iterations of different methods on NSL-KDD dataset are depicted in Figures 4-1, 4-2 and 4-3, respectively.

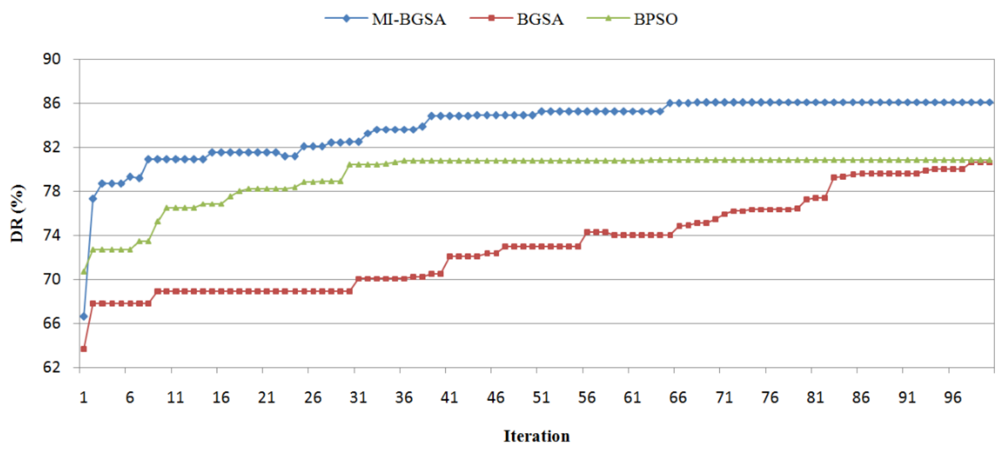


Figure 4-2: Average DR when using SVM classifier and MI-BGSA/BGSA/BPSO feature selection methods

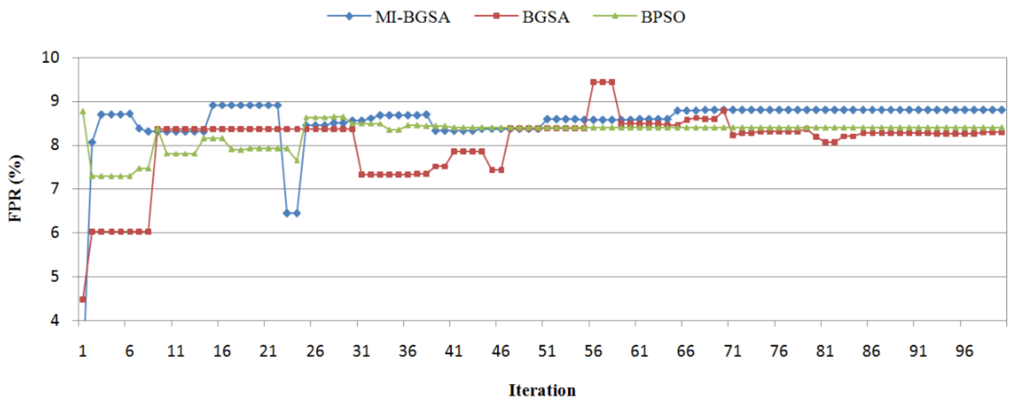


Figure 4-3: Average FPR when using SVM classifier and MI-BGSA/BGSA/BPSO feature selection methods

As shown in Figures 4-1 and 4-2, the proposed method performs better in terms of accuracy rate and detection rate. In other words, the proposed algorithm was not only concerned with reducing the number of features but also improved the performance of classification. Therefore, using the information theory in the BGSA feature selection method (with the aim of considering both feature–feature and feature–class mutual information) can improve the efficiency of standard BGSA. This search strategy, which was used as an inner-layer optimization, can increase the relevance between input features and the target class. In addition, this strategy can decrease the redundancy among the selected features, so it resulted in the improvement of the performance of standard BGSA in feature selection. The details of the selected features and performance metrics are reported in Table 4-6.

Table 4-6: The best experimental results of MI-BGSA, BGSA and BPSO feature selection methods using SVM classifier

FS Method	Accuracy (%)	DR (%)	FP (%)	Execution Time (s)	# Feature subset	Selected Features
MI-BGSA	88.362	86.308	8.887	727.465	5	{3, 4, 5, 6, 25}
BGSA	85.620	81.173	8.423	508.215	13	{1, 3, 5, 7, 8, 9, 18, 20, 22, 23, 26,27, 28}
BPSO	85.880	81.418	8.142	571.046	11	{3, 4, 5, 9, 10, 13, 18, 22, 25, 26, 41}

According to Table 4-6, the accuracy rate and DR were improved by 2.742% and 5.135% as compared to the standard BGSA, respectively. These rates were improved by 2.482% and 4.890% as compared to the BPSO feature selection method, respectively. It should be noted that the proposed feature selection algorithm selected only a small portion of the original features as compared to other feature selection methods (only 5 features among 41 features as compared to 13 and 11 selected features in BGSA and BPSO methods, respectively). The MI-BGSA attempts to determine the inter-variable relationship within a reduced subset which can predict output accurately. According to the experimental results reported in Table 4-6, the proposed feature selection algorithm can make dramatic reductions in the feature space and can consequently improve the classification performance.

In this study, the mentioned anomaly detection strategy is used for determining the best feature subset of every kind of attack in NSL-KDD. For example, for finding the optimal features in detecting DoS attacks, all samples that are labeled with DoS attack known as an anomaly, and the reminder samples are assumed as normal. Table 4-7 reports the optimal features for every kind of attack in NSL-KDD dataset. As can be seen in Table 4-7, the optimal subset of features of NSL-KDD dataset is the union of the obtained optimal features.

Table 4-7: The specification of the optimal feature of NSL-KDD obtained by using MI-BGSA feature selection

Attack Type	# Feature subset	Selected Features
DoS	8	{4, 5, 23, 25, 26, 29, 37, 38}
Prob	2	{23, 37}
U2R	18	{2, 3, 4, 5, 7, 8, 10, 11, 15, 16, 21, 24, 26, 27, 28, 30, 31, 40}
R2L	18	{2, 8, 10, 11, 12, 14, 15, 18, 19, 21, 22, 23, 26, 28, 29, 30, 34, 40}
Total	29	{2, 3, 4, 5, 7, 8, 10, 11, 12, 14, 15, 16, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 34, 37, 38, 40}

4.2 The Experimental Results of the Proposed Cyber-Attack Detection Technique

This thesis presents an offline misuse-based IDS for detecting the cyber-attacks of IoT named MOPF. As mentioned in section 3.2.2.4, the proposed method was composed of three modules (functional stages). These modules (i.e., partitioning, pruning, and detecting) were employed for improving the performance of a traditional OPF in IDSs. The partitioning module was used for addressing the scalability of large datasets, detecting low-frequent attacks, and speeding up the OPF. The pruning module was used for pruning the training and evaluation sets by identifying the most informative samples. Finally, the anomaly detection and the attack classification were performed by the proposed AOPF in the detecting module. Therefore, at first, the AOPF is compared with the traditional OPF in the experiments of this study. Then, to evaluate the performance of partitioning and pruning modules in the proposed model, the results of applying AOPF+P (partitioning), AOPF+Pr (pruning), and AOPF+P+Pr (called MOPF) are compared with the traditional OPF. To evaluate the performance of the proposed algorithm, OPF, AOPF, pruning, and partitioning modules were implemented in MATLAB R2014a on a PC with an Intel(R) Core (TM) i5-4460, CPU 3.20 GHz, and 8 GB RAM. Notably, the K-Means clustering algorithm (which was already implemented in MATLAB) was used in the partitioning module.

Table 4-8: Size of the training and test sets

Data type	Number of instances in the training set	Number of instances in the test set
Normal	3,490	1,526
DoS	2,489	1,028
Probe	684	284
R2L	228	109
U2R	109	53

In this study, 7,000 and 3,000 instances were randomly selected from the NSL-KDD as the training and test datasets, respectively. Table 4-8 lists the number of training and test instances. It is worthy to note that in all experiments, 4,000 initial training samples were used as the training set and the remaining 3,000 samples were used as the evaluation set. Notably, the distribution of data types in both training and test sets was almost the same as the NSL-KDD dataset. It means that the distribution of each data type (i.e., Normal, DoS, Probe, U2R, and R2L) and even the difficulty level of instances were considered in their random selection. Furthermore, in the applied dataset, just the obtained optimal feature subset of the preprocessing step is considered.

Table 4-9: Parameters setting of the proposed method in simulations

Parameter	Value	Phase/Stage
T	5	The learning phase of the OPF
k	4	Partitioning stage
γ	65%	Pruning stage
δ	0.696	Detecting stage

As discussed earlier, the proposed MOPF model was composed of several stages. The parameters setting in different stages are reported in Table 4-9. As seen in Table 4-9, the maximum iteration number of the learning algorithm in all versions of the OPF was set to 5 ($T = 5$). The reason is based on the experiments of Papa et al. (Joao P Papa et al., 2009) about the adequacy of repeating the learning procedure for a few iterations. The number of clusters in the K-Means clustering algorithm was set to 4. To select the best value of k in the K-Means clustering algorithm, this algorithm was iterated for different values of k and after each execution; the DB index was computed for each k. As shown in Figure 4-4, the value of DB index was minimized when $k = 4$; therefore, the best value of k was assumed as 4.

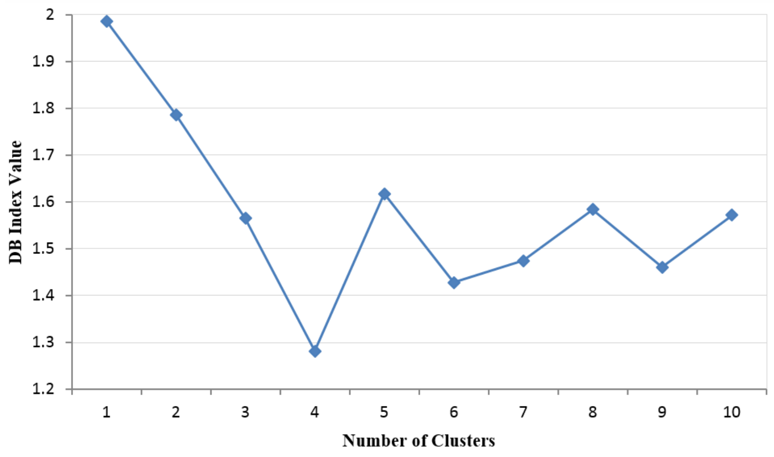


Figure 4-4: The DB index value versus the number of clusters

As reported in Table 4, the optimum values of γ and δ were 65% and 0.696, respectively. These settings were used as thresholds in the pruning and detecting modules, respectively. It is noted that these parameters were obtained by the GA, and Algorithms 4 and 6 were run for obtaining the fitness values. As mentioned earlier, two new subsets of NSL-KDD dataset were used as the training set and the test set in Algorithms 4 and 6. For this purpose, 1,500 and 750 instances were randomly selected from the NSL-KDD dataset as the new training set (TR) and test set (TS), respectively. The GA, which was already implemented in MATLAB, was used for finding the optimum values of γ and δ. The population size and the number of generations of both implementations of GA (which use Algorithms 4 and 6 as their fitness functions) were set to 5 and 10 in our simulations, respectively.

For computing the BC or the PR in the pruning stage, a directed graph should be constructed from the input samples. In constructing the directed graph, the m best neighbor nodes (as x_i neighbors) were selected based on the weight of the arcs in the complete weighted graph. For selecting the best value of m, we were inspired by the neighborhood selection problem in the recommendation systems and k-NN algorithm (which is used for the classification and regression). Generally, two approaches have been used in the recommendation systems for selecting the best neighbors of an active user: (a) selecting a fixed number of the best neighbors and (b) selecting the best neighbors based on a threshold on the similarity weights. Herlocker et al. (Herlocker, Konstan, & Riedl, 2002) demonstrated that the first approach which uses a fixed number of neighbors (i.e., 20 up to 60) performs better than the second approach. On the other hand, the appropriate value for m in k-NN algorithm is achieved by calculating the square root of the number of instances in the training set (as an empirical rule-of-thumb that was popularized by Duda et al. (Duda, Hart, & Stork, 2012)).

To evaluate the effectiveness of BC and PR measures in the pruning stage, they were compared in terms of Euclidean distance between the classification point with coordinates of (FAR, DR) and the perfect classification point in the ROC, when a lower distance shows better classification performance. The experimental results of these performance evaluations are shown in Figures 4-5 and 4-6.

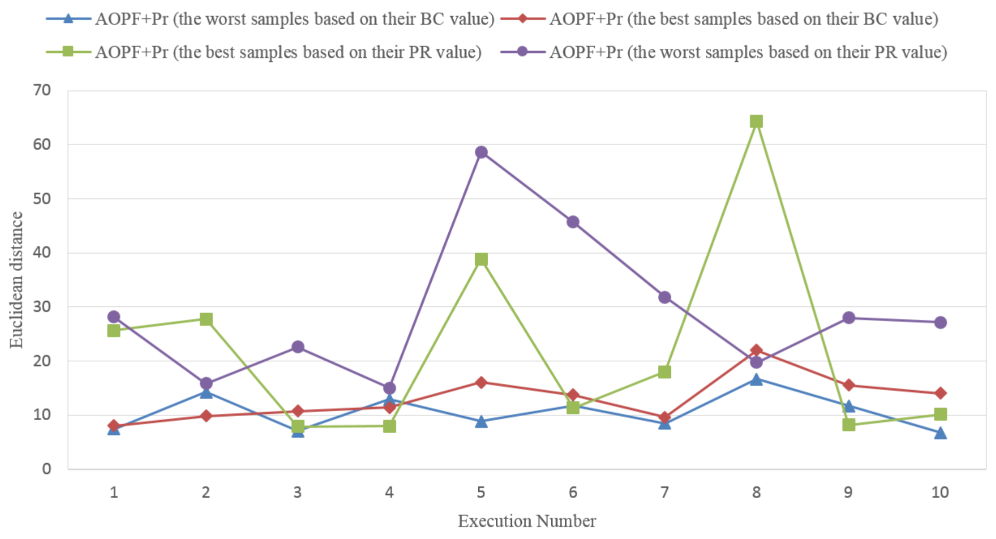


Figure 4-5: The comparison of AOPF performance when using different pruning strategies in intrusion detection application

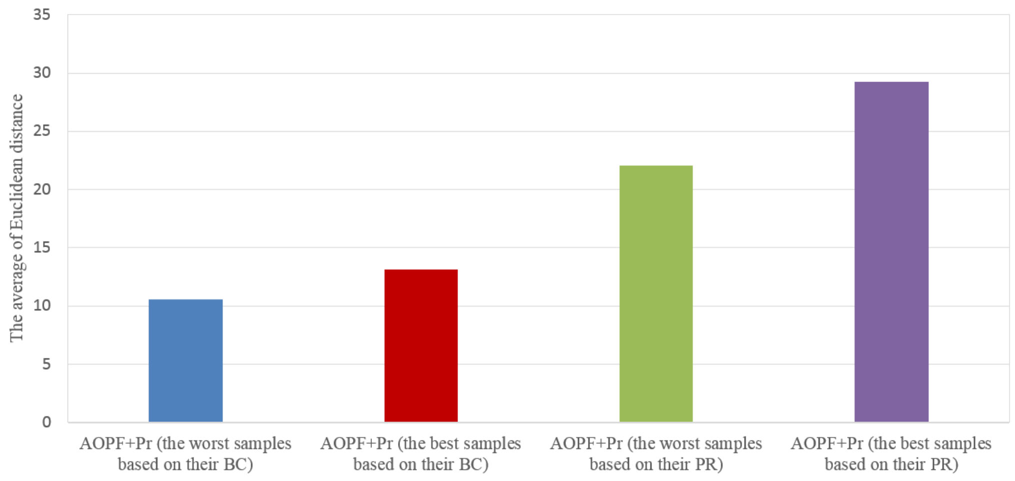


Figure 4-6: The average Euclidean distance of AOPF when using different pruning strategies

Figure 4-5 shows the performance comparison of four classifiers (based on the AOPF+Pr model) when using different pruning strategies in terms of Euclidean distance at 10 runs. It is noted that different datasets consisted of 1,000 samples were used in each execution as the training set and 500 samples as the test set (which were randomly selected from the original NSL-KDD dataset). At first, each pruning module computed the BC and the PR of each training sample. Secondly, 50% of the samples that had the worst BC values were selected as the training samples in the first simulated AOPF shown in Figure 4-5. Then, 50% of the samples that had the best BC values were selected as training samples in the second simulated AOPF shown in Figure 4-5. Similarly, this scenario was repeated for the samples with the worst and the best values of PR for the third and fourth simulated AOPFs shown in Figure 4-5, respectively. As seen in Figure 4-5, according to both BC and PR metrics, selecting some samples which had the worst BC (or PR) values led to better performance as compared to samples that had the best BC (or PR) values. For a more accurate comparison, the average Euclidean distance over 10 executions (for each type of pruning depicted in Figure 4-5) is shown in Figure 4-6. As seen in Figure 4-6, AOPF+Pr results in the best classification performance when the worst samples were used based on their BC value. Therefore, it is concluded that in the OPF machine learning, unlike social network analysis, some samples which have the worst BC (or PR) are the best informative samples for our classification problem. Furthermore, as seen in Figure 4-5, using the BC measure for identifying the most informative samples is better than using the PR measure. Hence, the BC measure was used as a suitable metric for finding the most informative samples in the pruning module of the proposed model.

Table 4-10: Accuracy rate and execution time of investigated OPF-based models

Model	Accuracy (%)	Training time (s)	Testing time (s)	Training + Testing time (s)
OPF	76.88	136.94	4.05	140.99
AOPF	90.53	139.94	511.89	651.83
AOPF+P	91.09	49.43	147.64	197.06
AOPF+Pr	90.27	57.60	269.81	327.41
MOPF	91.74	19.93	72.35	92.28

In this section, the performance of the proposed method is compared to the traditional OPF in terms of evaluation metrics introduced in section 4.2. As mentioned earlier, the proposed MOPF was composed of independent modules such as partitioning, pruning, and detecting. To evaluate the performance of the proposed model, four variants of modifications (i.e., AOPF, AOPF+P, AOPF+Pr, MOPF (AOPF+P+Pr)) are compared with the traditional OPF. The details of simulation results, when using the mentioned investigated models, are reported in Tables 4-10, 4-11, and 4-12 for evaluating the IDS classification performance.

The accuracy rate of the investigated proposed models and training and execution times are reported in Table 4-10. It is noted that the computation time of BC measure for each training sample is time-consuming, so for improving the speed of MOPF and AOPF+Pr (which use the value of each sample's BC in their pruning module), the BC calculation was performed offline. Moreover, the K-Means clustering, which was run in the partitioning modules of MOPF and AOPF+P, was also performed offline. The GA was run offline and its execution time is not included in Table 4-10, as well.

Table 4-11: Performance comparison of investigated models in terms of DR and FAR

Model	DR (%)	FAR (%)
OPF	90.30	4.19
AOPF	96.13	10.55
AOPF+P	93.83	1.96
AOPF+Pr	96.27	9.23
MOPF	96.20	1.44

As seen in Table 4-10, the accuracy rate is improved considerably when using the MOPF algorithm as compared to the traditional OPF. However, the accuracy improvement is achieved by using each of the proposed modules in the classification problem (such as AOPF+P or AOPF+Pr). The modification of optimum-path cost function in the classification phase of AOPF has led to high computational overload and consequently, high testing time. On the other hand, partitioning and pruning modules reduce the training time significantly.

This significant improvement is due to the computation time of MST. As mentioned earlier, one of the key operations in the training phase of the OPF algorithm is the identification of the optimum set of prototypes. This set is obtained using the closest nodes of MST which have different labels. Therefore, an MST should be computed based on the complete weighted graph (which was generated by using the training samples) for finding the prototypes from the training set. In this study, the prime algorithm (which was already implemented in MATLAB) was used to compute the MST in the training phase. The time complexity of this algorithm is $O(m.\log(n))$, where *n* and *m* are the number of nodes and edges, respectively. As noted earlier, the initial training set consisted of 7,000 samples in this study, where 3,000 samples were used as the evaluation set and the remaining 4,000 samples were used for training the OPF. Hence, the complete weighted graph, which was generated from these training samples, has 4,000 nodes and 7,998,000 edges. Indubitably, the computational time of MST from this graph will be high. By using the proposed partitioning module, the original training set was partitioned into four clusters. Subsequently, an OPF was generated for each cluster by means of a lighter graph (in terms of the number of nodes and edges) based on the cluster's training subset. This strategy reduces the computational time of MST and consequently reduces the training time of the OPF algorithm. Notably, this scenario is repeated for the pruning module.

Table 4-12: Confusion matrix and CPE of investigated models tested on NSL-KDD dataset

		Predicted					
Model	**Actual**	Normal	DoS	Probe	R2L	U2R	**CPE**
OPF	Normal	**1462**	23	11	26	4	0.1950
	DoS	29	**994**	4	1	0	
	Probe	33	23	**225**	2	1	
	R2L	69	4	1	**33**	2	
	U2R	12	19	3	3	**15**	
AOPF	Normal	**1365**	47	54	45	15	0.1473
	DoS	10	**1010**	8	0	0	
	Probe	20	7	**257**	0	4	
	R2L	22	0	0	**83**	4	
	U2R	5	0	1	4	**43**	
AOPF+P	Normal	**1496**	14	10	3	3	0.0983
	DoS	53	**961**	8	3	3	
	Probe	21	3	**258**	1	1	
	R2L	10	0	4	**93**	2	
	U2R	7	3	2	4	**37**	
AOPF+Pr	Normal	**1373**	46	54	38	15	0.1460
	DoS	6	**1014**	8	0	0	
	Probe	20	7	**257**	0	0	
	R2L	32	0	0	**74**	3	
	U2R	5	0	1	4	**43**	
MOPF	Normal	**1504**	5	7	5	5	0.0753
	DoS	20	**996**	7	4	1	
	Probe	27	7	**244**	2	4	
	R2L	7	4	0	**85**	13	
	U2R	2	1	2	5	**43**	

As shown in Table 4-10, the traditional OPF performs as the fastest model in terms of testing time; however, if the whole execution time (i.e., training and testing times) is considered, the MOPF will be faster than the traditional OPF (about 35% reduction in the execution time). This shows the superior computational efficiency of the proposed MOPF for the time of classifier training and testing (i.e., detection).

As seen in Table 4-11, the DR is improved when using AOPF and AOPF+Pr models as compared to other models. However, FAR of the mentioned models is high (and not acceptable) as compared to AOPF+P and MOPF models. The MOPF model consists of AOPF, partitioning, and pruning modules; therefore, we can conclude that the proposed MOPF can achieve the best performance among the investigated methods when considering both DR and FAR.

The confusion matrix and CPE of the five mentioned investigated models are reported in Table 4-12. Notably, the proposed MOPF model offers the best performance in terms of CPE as compared to the traditional OPF and other three investigated models.

One of the main advantages of the proposed MOPF method is its superior performance against low-frequent attacks such as U2R and R2L. In this way, the DR of different attack types, when using the proposed MOPF or the traditional OPF, is shown in Figure 4-7. As seen in Figure 4-7, the proposed MOPF model offers better DR than the traditional OPF for all attack types. In addition, the FAR of different attack types, when using the proposed MOPF or the traditional OPF, is shown in Figure 4-8. As seen in Figure 4-8, the proposed MOPF model offers better FAR for DoS, Probe, and U2R attacks than the traditional OPF. As seen in Figures 4-7 and 4-8, the FAR of R2L attack when using the traditional OPF is slightly better than the proposed MOPF model. However, the DR of U2R and R2L attacks and also the FAR of U2R attacks are considerably improved when using the proposed MOPF model.

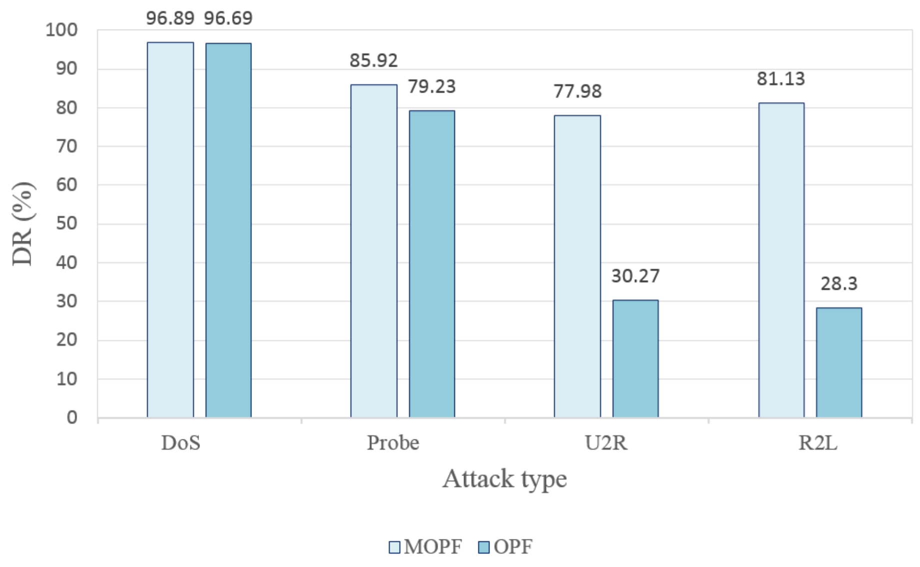


Figure 4-7: Performance comparison of the MOPF and the traditional OPF in terms of DR

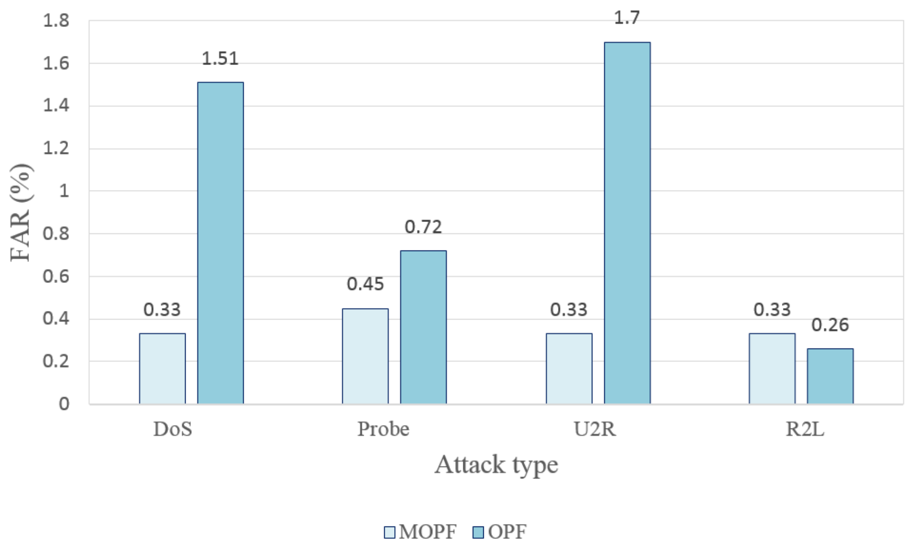


Figure 4-8: Performance comparison of the MOPF and the traditional OPF in terms of FAR

4.2 The Experimental Results of the Proposed Insider-Attack Detection Technique

In order to detect sinkhole and selective forwarding attacks as two kinds of dangerous insider-attacks of IoT, this thesis presents a hybrid real-time IDS based on some distributed agents called SA-IDS and a centralized agent named AA-IDS. Indeed, the proposed model is based on the agent programming (i.e., SA-IDS and AA-IDS agents are located at the router nodes and the 6BR, respectively); so, we developed a special WSN simulator in this study that is based on the RPL protocol using .Net Framework technology and C#.Net programming. So, a flexible evaluation platform was provided for developing the proposed intrusion detection framework, and also simulating the selective forwarding and sinkhole attacks. The assumptions in the developed simulator are given in Table 4-13.

Table 4-13: Assumptions in the developed simulator in this study

Parameter	Value
Network scale	100 m × 100 m
Routing protocol	RPL
Transmission range	10 m
Packet size	127 bytes
DIO size (Winter et al., 2012)	24 bytes
Node's energy	Infinite

It is noted that we did not study the energy overhead of the proposed framework in this study. Therefore, we assumed that the energy of all kinds of nodes was infinite, which means they were not constrained in terms of energy consumption. Moreover, we assumed that the network structure is static in all simulations. In other words, the sensor nodes were homogenous and distributed uniformly in the environment. In the 6BR implementation, which was based on the MapReduce architecture, the MATLAB server was used as the reducer node for projecting the clustering models with the aim of anomaly detection. We implemented an anomaly detection method that was based on the OPFC algorithm using MATLAB R2014a on a PC with an Intel(R) Core (TM) i5-4460, CPU 3.20 GHz, and 8 GB RAM. The anomaly detection was performed by the 6BR in which the corresponding reducer (i.e., MATLAB server) received a new sample of source *i* from the mapper (i.e., the 6BR). Then, a new clustering model was projected by using this new sample and other old samples. According to the clustering result, the label of the new sample was determined by the reducer.

One of the challenging issues in the OPFC algorithm is finding the appropriate value of parameter *k*. Generally, the value of k is selected in the interval $[\mathrm{k}_{\min}, \mathrm{k}_{\max}]$ where $1 \le k_{min} < k_{max} \le |Z|$ (Rocha et al., 2009). Rocha et al. (Rocha et al., 2009)used a graph-cut metric for finding the optimum value of *k*. The proposed anomaly detection should work in real-time; so, we experimentally assumed that $k = \left\lfloor \frac{\#\ samples}{3} \right\rfloor$ at the start of projecting the OPFC model in this study.

The specification-based agents use (i.e., *ParentSetThreshold* and *PreferredParentThreshold* in Algorithm 8, and *PacketReceivingThreshold* and *TimeDelayThreshold* in Algorithm 9) in detecting sinkhole and selective forwarding attacks. The nodes were different in the network (according to their positions, children, and parents); therefore, each node had its own thresholds that should be defined at runtime. We assumed that the malicious nodes begin to launch attacks after 2 minutes; hence, the thresholds of each node were specified in opening 2 minutes of the simulation based on the normal behavior. For example, to determine the ParentSetThreshold (which specifies the rate of change in parent set of a router node at each time-slot Δw), the algorithm computed the rate of change in router's parent set at each Δw in opening 2 minutes; then, the average of computed values was returned as the *ParentSetThreshold*.

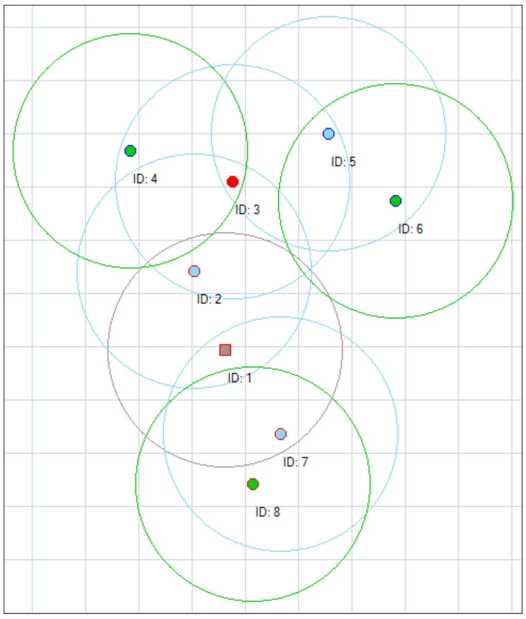


Figure 4-9: A simulation screenshot of the Selective forwarding attack launched by node 3

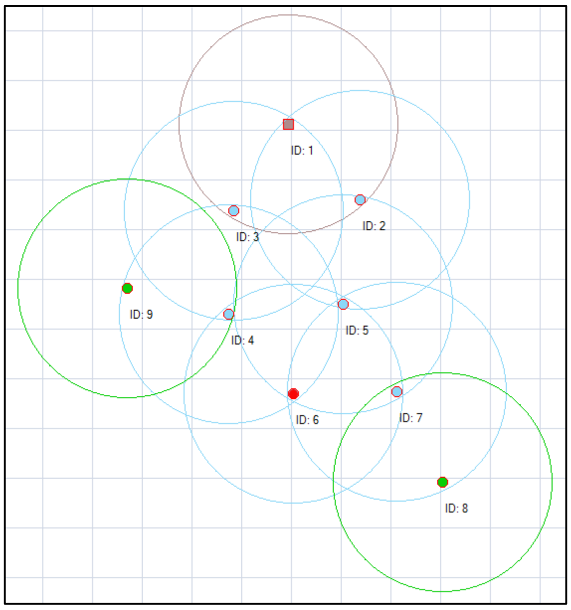


Figure 4-10: A simulation screenshot of the sinkhole attack launched by node 6

In this study, the performance of the proposed method was evaluated in terms of DR, FPR, and AR of detecting malicious nodes. This experiment was conducted for evaluating the performance of the proposed framework to deal with the selective forwarding attack, sinkhole attack, and joint occurrence of both attacks. Three simulations were performed in this experiment. The screenshots of these simulations are shown in Figures 4-9, 4-10, and 4-11, respectively. Through these simulations, the performance of the proposed framework was evaluated to deal with the selective forwarding attacks (Figure 4-9), the sinkhole attacks (Figure 4-10), and a joint occurrence of these attacks (Figure 4-11) in the 6LoWPAN. The assumptions in these simulations are given in Tables 4-14, 4-15, and 4-16, respectively.

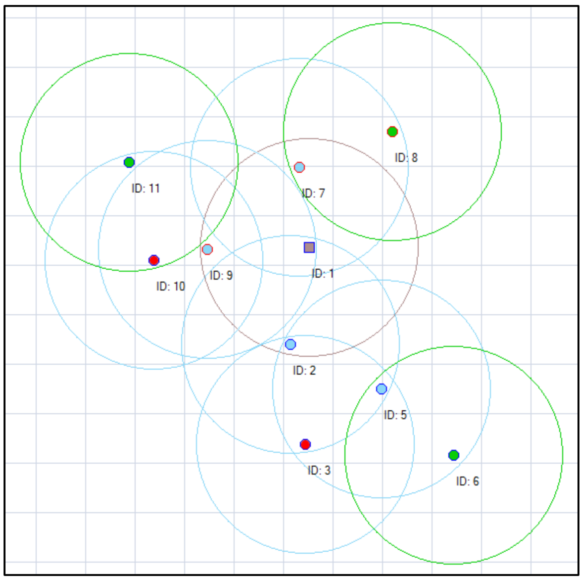


Figure 4-11: A simulation screenshot of the selective forwarding and sinkhole attacks launched simultaneously by node 10 and node 3, respectively

Table 4-14: Assumptions in the first simulation (shown in Figure 4-9)

Feature	Value
# Source	3 nodes with ids {4, 6, 8}
# Router	4 nodes with ids {2, 3, 5, 7}
Root id	1
Malicious node id	3 as selective forwarding agent
Δw	30 s
Simulation time	20 min

Table 4-15: Assumptions in the second simulation (shown in Figure 4-10)

Feature	Value
# Source	the nodes with ids {8, 9}
# Router	6 nodes with ids {2, 3, 4, 5, 6, 7}
Root id	1
Malicious node	6 as sinkhole agent
Δw	30 s
Simulation time	20 min

Table 4-16: Assumptions in the third simulation (shown in Figure 4-11)

Feature	Value
# Source	3 nodes with ids {6, 8, 11}
# Router	6 nodes with ids {2, 3, 5, 7, 9, 10}
Root id	1
Malicious node	3 and 10 as sinkhole and selective forwarding agents, respectively
Δw	30 s
Simulation time	30 min

The performance of the proposed model in these simulations is reported in Tables 4-17, 4-18, and 4-19, respectively. Notably, the AR metric is calculated as the division of the total number of malicious nodes (according to each sample) that are identified correctly to the total number of attack samples that are classified correctly.

Table 4-17: Performance of the proposed model in the first simulation (shown in Figure 4-9)

Method	DR (%)	FPR (%)	AR in detecting malicious nodes (%)
Anomaly-based detection	92.68	10.12	NA*
Hybrid (Anomaly-based+Specification-based) detection	85.36	1.26	91.43

* Not-Applicable

Table 4-18: Performance of the proposed model in the second simulation (shown in Figure 4-10)

Method	DR (%)	FPR (%)	AR in detecting malicious nodes (%)
Anomaly-based detection	100	5.97	NA*
Hybrid (Anomaly-based+Specification-based) detection	100	2.98	69.23

* Not-Applicable

Table 4-19: Performance of the proposed model in the third simulation (shown in Figure 4-11)

Method	DR (%)	FPR (%)	AR in detecting malicious nodes (%)
Anomaly-based detection	80.95	29.63	NA*
Hybrid (Anomaly-based+Specification-based) detection	76.19	5.92	87.50

* Not-Applicable

As seen in Tables 4-17 and 4-19, the DR of the proposed hybrid method is lower than the anomaly-based detection method (performed by the AA-IDS agent); however, the FPR is improved considerably when employing the hybrid model in which the specification-based detection is also performed by the SA-IDS agents besides the anomaly-based detection. Another advantage of using the specification-based detection in the proposed framework is the ability to detect malicious nodes as the cause of IoT's insider-attacks (as shown in the last column of Tables 4-17, 4-18, and 4-19).

Table 4-18 shows that DR of the anomaly-based detection and the hybrid detection methods in the second simulation is 100% (when the sinkhole attack is launched). However, as shown in Table 4-17, DR of the anomaly-based detection and the hybrid detection methods in the first simulation was 92.68% and 85.36%, respectively (when the selective forwarding attack was launched). This refers to the behavior of selective forwarding and sinkhole attacks. As mentioned earlier, the AA-IDS agent creates a new sample for each source node by extracting four traffic-related features from the raw received packet of the source node in each time-slot. These features are packet receiving rate, packet dropping rate, average latency, and maximum hop-count. The sinkhole attack has a considerable effect on the maximum hop-count feature, while the selective forwarding attack changes the packet dropping rate, and consequently the packet receiving rate. Packet dropping is one of the popular events in the networks (e.g., because of the congestion); so, distinguishing the valid packet dropping (due to the popular behavior of the networks) from the invalid packet dropping (because of the malicious behavior of the selective forwarding attack) is hard for the proposed anomaly detection model. As seen in Table 4-17, it caused the DR reduction of the proposed hybrid model in the first simulation as compared to the second simulation in which a sinkhole attack was launched (Table 4-18). Notably, the low AR in detecting malicious nodes in Table 4-18 is caused by the process of detecting the sinkhole attack in Algorithm 8. According to Algorithm 8, the detection of a suspicious node is based on the rate of change in preferred parent and also the rate of change in the parent set at each time-slot. In some cases, it may cause mistakes about a valid parent or an invalid parent. For example, the SA-IDS agent located in node 7 (Figure 4-10) may select node 5 (i.e., a valid parent) as a suspicious node instead of the actual malicious node (i.e., node 6).

Chapter 5: Conclusions

Over this chapter, the presented intrusion detection technique will skim, briefly. Furthermore, the limitations of the study and future work will be reviewed.

5.1 Research Summary

6LoWPAN as an IP-based WSN is one of the main efforts to make the real IoT in which the resource-constrained devices (e.g., sensors) can connect to the Internet through a border gateway called 6BR. This thesis proposes a novel ML-based intrusion detection technique in order to identify the possible attacks of 6LoWPAN as well as determine their types. Indeed, the insecure nature of 6LowPANs makes IoT becomes vulnerable in front of different kinds of attacks from the Internet and IP-based WSNs sides. Therefore, this thesis proposed a multi-faceted hybrid intrusion detection method in order to detect different kinds of attacks.

Generally, the proposed intrusion detection technique consists of two following components:

1) Cyber-attack identification that can also determine the kinds of detected attacks.
2) Insider-attack identification that is also able to determine the perpetrators of them (the malicious nodes).

The first component, which is employed for detecting cyber-attacks launched by the Internet side is an offline misused-based IDS based on a novel graph-based machine learning algorithm called supervised Optimum-Path Forest. Actually, the training and classification phases of OPF were modified based on the K-Means clustering algorithm and the social network analysis. Notably, in order to overcome the high-dimensional benchmark dataset used for evaluating the proposed intrusion detection engine, a novel population-based heuristic search is proposed based on Binary Gravitational Search Algorithm to reduce the dataset's dimension. Actually, the proposed method is a hybrid feature selection method used Mutual Information as a filter-based algorithm (local search) to identify the informative features determined by BGSA as a wrapper-based algorithm (global search). The experimental results of the applied models of this component show the superiority of them in comparison with other methods.

The second component is a novel real-time hybrid IDS that is proposed for detecting routing attacks (sinkhole and selective forwarding attacks) in 6LoWPAN as insider IoT's attacks. Actually, in this component, in order to detect anomalies and identify the malicious nodes as the cause of anomalies, the specification-based IDS agents located in 6LoWPAN nodes send their analytical results to the 6BR. Then the anomaly-based IDS agent located in 6BR detects anomalies and identifies the malicious nodes. Furthermore, the anomaly detection engine, which is based on MapReduce architecture, uses unsupervised OPF (OPFC) for identifying the anomalies. The final decision about detected anomalies will be made in the process of a voting mechanism.

5.2 Conclusion

As far as we know, since there are no a considerable study that uses ML-technique in order to detect intrusions of IoT, it seems the idea of using a learning algorithm in the security area of IoT is a valuable approach. In addition, the abilities of the proposed model that can detect both cyber-attacks and insider-attacks of IoT can distinguish this model from few existing approaches. By the way, the advantages of the presented technique can be summarized as follows:

1) The ability to detect cyber-attacks by determining their types
   - Because of the resource-limitation objects in IoT, the considerable reduction of the dataset's dimensions is useful for avoiding the over-fitting problem, improving the performance of classification models, reducing the training and testing time, improving stability against noise, and reducing the measurement and storage requirements.
   - The proposed cyber detection technique is a fast method due to the high speed of OPF algorithm.
   - The superior characteristics of OPF, the high performance of K-Means algorithm in partitioning the heterogeneous samples into homogeneous clusters, and the distinctive abilities of the pruning strategy in sample reduction have led to a considerable improvement in the proposed model in detecting low frequent attacks.

2) The ability to detect insider-attacks by determining their causes
   - This thesis tries to present a light specification-based IDS agent, because of the resource limitation of the host nodes in 6LoWPAN. For example, because the local analysis of the SA-IDS agents in router nodes is sent to 6BR through the usual traffic, there is no need for storing them in resource-constrained router nodes. This functionality can improve resource consumption, dramatically.
   - The proposed method sends the analysis results of local agents to the 6BR at regular intervals through normal data packets without using additional control messages. It is clear that this approach can lead to a significant reduction in the cost of communication.
   - Because the proposed anomaly detection engine is based on MapReduce, it can lead to accelerating the detection process due to the concurrent abilities.
   - The proposed model can determine the malicious nodes with low FAR.

5.2 Research Limitations and Future Work

Generally, the research limitations of this study are as follows:

1) Lack of an IoT-based dataset for using over intrusion detection system: The Existence an appropriate dataset is a key requirement for every machine learning algorithm. Unfortunately, this study faced with various challenges due to the fact that there is not standard dataset for using in ML-based IDS.
2) Lack of proper software/hardware infrastructure for experimental work: One of the main challenges of this project, especially in evaluating the proposed real-time IDS was the lack of proper infrastructure. For example, although the obtained centralized anomaly detection in 6loWPAN was based on MapReduce architecture, I could not evaluate the performance of the proposed model in a distributed platform since there was not a proper software/hardware infrastructure.

Furthermore, the main future work of this thesis are as follows:

1) Applying the proposed insider-attack detection technique in a distributed environment in order to evaluate the performance of the presented centralized anomaly detection, especially in terms of speed.
2) Developing the proposed RPL-based WSN in order to obtain the ability to evaluate the power consumption and communication cost.
3) Developing the proposed insider-attack detection technique in identifying other kinds of attacks besides to sinkhole and selective forwarding attacks.
4) Improving the proposed cyber-attack detection by applying real Internet traffic rather than using the NSL-KDD dataset.

References

Agarwal, R., & Joshi, M. V. (2001). *PNrule: a new framework for learning classifier models in data mining (a case-study in network intrusion detection).* Paper presented at the Proceedings of the 2001 SIAM International Conference on Data Mining.

Amin, S .O., jig Yoon, Y., Siddiqui, M. S., & Hong, C. S. (2009). *A novel intrusion detection framework for IP-based sensor networks.* Paper presented at the 2009 International Conference on Information Networking.

Amiri, F., Yousefi, M. R., Lucas, C., Shakery, A., & Yazdani, N. (2011). Mutual information-based feature selection for intrusion detection systems. *Journal of Network and Computer Applications, 34*(4), 1184-1199.

Amorim, W. P., & de Carvalho, M. H. (2012). *Supervised learning using local analysis in an optimal-path forest.* Paper presented at the 2012 25th SIBGRAPI Conference on Graphics, Patterns and Images.

Anderson, J. P. (1980). Computer security threat monitoring and surveillance, James P. *Anderson Co., Fort Washington, PA.*

Anuar, N. B., Sallehudin, H., Gani, A., & Zakaria, O. (2008). Identifying false alarm for network intrusion detection system using hybrid data mining and decision tree. *Malaysian journal of computer science, 21*(2), 101-115 .

Aridhi, S., Lacomme, P., Ren, L., & Vincent, B. (2015). A MapReduce-based approach for shortest path problem in large-scale networks. *Engineering Applications of Artificial Intelligence, 41*, 151-165 .

Bader, D. A., Kintali, S., Madduri, K., & Mihail, M. (2007). *Approximating betweenness centrality.* Paper presented at the International Workshop on Algorithms and Models for the Web-Graph.

Bakshi, S., Jagadev, A. K., Dehuri, S., & Wang, G. N. (2014). Enhancing scalability and accuracy of recommendation systems using unsupervised learning and particle swarm optimization. *Applied soft computing, 15*, 21-29.

Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on neural networks, 5*(4), 537-550.

Bonev, B. (2010). *Feature selection based on information theory*: Universidad de Alicant.

Costa, K. A., Pereira, L. A., Nakamura, R. Y., Pereira, C. R., Papa, J. P., & Falcão, A. X. (2015). A nature-inspired approach to speed up optimum-path forest clustering and its application to intrusion detection in computer networks. *Information sciences, 294*, 95-108.

Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*(2), 224-227.

Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM, 51*(1), 107-113.

Deisy, C., Baskar, S., Ramraj, N., Koori, J. S., & Jeevanandam, P. (2010). A novel information theoretic-interact algorithm (IT-IN) for feature selection using three machine learning algorithms. *Expert Systems with Applications, 37*(12), 7589-7597.

Denning, D., & Neumann, P. G. (1985). *Requirements and model for IDES-a real-time intrusion-detection expert system*: SRI International.

Duda, R. O., Hart, P. E., & Stork, D. G. (2012) .*Pattern classification*: John Wiley & Sons.

Eik Loo, C., Yong Ng, M., Leckie, C., & Palaniswami, M. (2006). Intrusion detection for routing attacks in sensor networks. *International Journal of Distributed Sensor Networks, 2*(4), 313-332.

Golmah, V. (2014). An efficient hybrid intrusion detection system based on C5. 0 and SVM. *International Journal of Database Theory and Application, 7*(2), 59-70.

Hamedheidari, S., & Rafeh, R. (2013). A novel agent-based approach to detect sinkhole attacks in wireless sensor networks. *computers & security, 37*, 1-14.

Herlocker, J., Konstan, J. A., & Riedl, J. (2002). An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information retrieval, 5*(4), 287-310.

Hoque, N., Bhattacharyya, D. K., & Kalita, J. K. (2014). MIFS-ND: A mutual information-based feature selection method. *Expert Systems with Applications, 41*(14), 6371-6385.

Ibrahim, L. M., Basheer, D. T., & Mahmod, M. S. (2013). A comparison study for intrusion database (Kdd99, Nsl-Kdd) based on self organization map (SOM) artificial neural network. *Journal of Engineering Science and Technology, 8*(1), 107-119.

Kang, U., Papadimitriou, S., Sun, J., & Tong, H. (2011). *Centralities in large networks: Algorithms and observations.* Paper presented at the Proceedings of the 2011 SIAM International Conference on Data Mining.

Kasinathan, P., Pastrone, C., Spirito, M. A., & Vinkovits, M. (2013). *Denial-of-Service detection in 6LoWPAN based Internet of Things.* Paper presented at the 2013 IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob).

Kim, G., Lee, S., & Kim, S. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications, 41*(4), 1690-1700.

Kumar, G., & Kumar, K. (2012). An information theoretic approach for feature selection. *Security and Communication Networks, 5*(2), 178-185.

Lahre, M. K., Dhar, M. T., Suresh, D., Kashyap, K., & Agrawal, P. (2013) .Analyze different approaches for IDS using KDD 99 data set. *International Journal on Recent and Innovation Trends in Computing and Communication, 1*(8), 645-651.

Le, A., Loo, J., Lasebae, A., Aiash, M., & Luo, Y. (2012). 6LoWPAN: a study on QoS security threats and countermeasures using intrusion detection system approach. *International Journal of Communication Systems, 25*(9), 1189-1212.

Le, A., Loo, J., Luo, Y., & Lasebae, A. (2011). *Specification-based IDS for securing RPL from topology attacks.* Paper presented at the 2011 IFIP Wireless Days (WD).

Liu, C., Yang, J., Chen, R., Zhang, Y., & Zeng, J. (2011). *Research on immunity-based intrusion detection technology for the internet of things.* Paper presented at the 2011 Seventh International Conference on Natural Computation.

Liu, H., Wu, X., & Zhang, S. (2014). A new supervised feature selection method for pattern classification. *Computational Intelligence, 30*(2), 342-361.

Mabu, S., Chen, C., Lu, N., Shimada, K., & Hirasawa, K. (2011). An intrusion-detection model based on fuzzy class-association-rule mining using genetic network programming. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 41*(1), 130-139.

Musiał, K., Kazienko, P., & Bródka, P. (2009). *User position measures in social networks.* Paper presented at the Proceedings of the 3rd Workshop on Social Network Mining and Analysis.

Nezamabadi-pour, H., Rostami-Shahrbabaki, M., & Maghfoori-Farsangi, M. (2008). Binary particle swarm optimization: challenges and new solutions. *CSI J Comput Sci Eng, 6*(1), 21-32.

Papa, J. P., FalcãO, A. X., De Albuquerque, V. H. C., & Tavares, J. M. R. (2012). Efficient supervised optimum-path forest classification for large datasets. *Pattern Recognition, 45*(1), 512-520.

Papa ,J. P., Falcao, A. X., & Suzuki, C. T. (2009). Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology, 19*(2), 120-131.

Pei, M., Goodman, E., & Punch, W. (1998). *Feature extraction using genetic algorithms.* Paper presented at the Proceedings of the 1st International Symposium on Intelligent Data Engineering and Learning, IDEAL.

Pereira, C. R., Nakamura, R. Y., Costa, K. A., & Papa, J. P. (2012). An Optimum-Path Forest framework for intrusion detection in computer networks. *Engineering Applications of Artificial Intelligence, 25*(6), 1226-1234.

Perez, L. G., Chiclana, F., & Ahmadi, S. (2011). *A social network representation for collaborative filtering recommender systems.* Paper presented at the 2011 11th International Conference on Intelligent Systems Design and Applications.

Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). GSA: a gravitational search algorithm. *Information sciences, 179*(13), 2232-2248.

Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2010). BGSA: binary gravitational search algorithm. *Natural Computing, 9*(3), 727-745.

Ray, S., & Turi, R. H. (1999). *Determination of number of clusters in k-means clustering and application in colour image segmentation.* Paper presented at the Proceedings of the 4th international conference on advances in pattern recognition and digital techniques.

Raza, S., Wallgren, L., & Voigt, T. (2013). SVELTE: Real-time intrusion detection in the Internet of Things. *Ad hoc networks, 11*(8), 2661-2674.

Rghioui, A., Khannous, A., & Bouhorma, M. (2014). Denial-of-Service attacks on 6LoWPAN-RPL networks: Threats and an intrusion detection system proposition. *Journal of Advanced Computer Science & Technology, 3*(2), 143.

Rocha, L. M., Cappabianco, F .A., & Falcão, A. X. (2009). Data clustering as an optimum‐path forest problem with applications in image analysis. *International Journal of Imaging Systems and Technology, 19*(2), 50-68.

Ruiz, R., Riquelme, J. C., & Aguilar-Ruiz, J. S. (2005). *Heuristic search over a ranking for feature selection.* Paper presented at the International Work-Conference on Artificial Neural Networks**.**

Rusinowska, A., Berghammer, R., De Swart, H., & Grabisch, M. (2011). *Social networks: prestige, centrality, and influence.* Paper presented at the International Conference on Relational and Algebraic Methods in Computer Science.

Sangita, O., & Dhanamma, J. (2011). *An improved k-means clustering approach for teaching evaluation.* Paper presented at the International Conference on Advances in Computing, Communication and Control.

Sekar, R., Gupta, A., Frullo, J., Shanbhag, T., Tiwari, A., Yang, H., & Zhou, S. (2002). *Specification-based anomaly detection: a new approach for detecting network intrusions.* Paper presented at the Proceedings of the 9th ACM conference on Computer and communications security.

Shahid, R. (2013). *Lightweight Security Solutions for the Internet of Things.* Doctoral Thesis. UMI Order Number: ID Code: 5548. Mälardalen University. ISSN 1651--4238.

Sheikhan, M. (2014). Generation of suprasegmental information for speech using a recurrent neural network and binary gravitational search algorithm for feature selection. *Applied intelligence, 40*(4), 772-790.

Sheikhan, M., & Khalili, A. (2010). Intrusion Detection Based on Rule Extraction from Dynamic Cell Structure Neural Networks. *Majlesi Journal of Electrical Engineering, 4*(4).

Stakhanova, N., Basu, S., & Wong, J. (2010). On the symbiosis of specification-based and anomaly-based detection. *computers & security, 29*(2), 253-268.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). *A detailed analysis of the KDD CUP 99 data set.* Paper presented at the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications.

Varlamis, I., Eirinaki, M., & Louta, M. (2013). Application of social network metrics to a trust-aware collaborative model for generating personalized user recommendations *The Influence of Technology on Social Network Analysis and Mining* (pp. 49-74): Springer.

Wallgren ,L., Raza, S., & Voigt, T. (2013). Routing Attacks and Countermeasures in the RPL-based Internet of Things. *International Journal of Distributed Sensor Networks, 9*(8), 794326.

Wang, G., Hao, J., Ma, J., & Huang, L. (2010). A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering. *Expert Systems with Applications, 37*(9), 6225-6232.

Wang, W., Zhang, X., Gombault, S., & Knapskog, S. J. (2009). *Attribute normalization in network intrusion detection.* Paper presented at the 10th International Symposium on Pervasive Systems, Algorithms, and Networks.

Wasserman, S., & Faust, K. (1994). *Social network analysis: Methods and applications* (Vol. 8): Cambridge university press.

Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., & Alexander, R. (2012). RPL: IPv6 routing protocol for low-power and lossy networks.

Wu, S.-Y., & Yen, E. (2009). Data mining-based intrusion detectors. *Expert Systems with Applications, 36*(3), 5605-5612.

Wu, S. X., & Banzhaf, W. (2010). The use of computational intelligence in intrusion detection systems: A review. *Applied soft computing, 10*(1), 1-35.

Yan, K., Wang, S., Wang, S., & Liu, C. (2010). *Hybrid intrusion detection system for enhancing the security of a cluster-based wireless sensor network.* Paper presented at the 3rd International Conference on Computer Science and Information Technology.

Zamani, M., & Movahedi, M. (2013). Machine learning techniques for intrusion detection. *arXiv preprint arXiv:1312.2177.*

Zhang, L., Feng, G., & Qin, S. (2013). Intrusion detection system for low-power and lossy networks. *Internet Draft (version 00).*

Zhijie, H., & Ruchuang, W. (2012). Intrusion detection for wireless sensor network based on traffic prediction model. *Physics Procedia, 25*, 2072-2080.