Research on Detecting Malicious Nodes in Wireless Sensor Networks

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2020

Boqi GAO

List of Publications

1. Journal Paper

1. B. Gao, T. Maekawa, D. Amagata, and T. Hara. Robust Malicious Node Detection with Ensemble Learning in MANETs. *IPSJ Journal*, 60(2):501–513, 2019.
2. B. Gao, T. Maekawa, D. Amagata, and T. Hara. Detecting Reinforcement Learning-based Grey Hole Attack in Mobile Wireless Sensor Networks. *IEICE Trans. on Communications*, E103-B(5):OO–OO, 2020.

2. International Conference Paper

1. B. Gao, T. Maekawa D. Amagata, and T. Hara. Environment-Adaptive IPMalicious Node Detection in MANETs with Ensemble Learning. In *Proc. of Int'l Conf. on Distributed Computing Systems (ICDCS)*, pages 556–566, 2018.

3. Domestic Conference Paper (with peer-review)

1. B. Gao, D. Amagata, T. Maekawa, and T. Hara. Detecting Energy Depriving Malicious Nodes by Unsupervised Learning in Energy Harvesting Cooperative Wireless Sensor Networks. In *Proc. of IPSJ DPS Workshop*, pages 97–104, 2019.

4. Domestic Conference Paper

1. B. Gao, T. Maekawa, D. Amagata, and T. Hara. Malicious Node Detection with Machine Learning in MANETs. In *Proc. of DEIM Forum*, online, 2017.
2. B. Gao, D. Amagata, T. Maekawa, and T. Hara. Detecting Malicious Nodes with Learning Ability in Mobile Wireless Sensor Networks. In *Proc. of DEIM Forum*, online, 2019.

Abstract

With the rapid advances in wireless communication and digital electronics, wireless sensor networks (WSNs) have demonstrated their feasibility for monitoring events and environments. For example, WSNs can be deployed to monitor air conditions, water flows, and temperatures. People can obtain useful information from the gathered data from WSNs. A WSN is comprised of solely of sensor nodes, and no special infrastructure is required. Due to such characteristics, malicious nodes can easily join a WSN. That is, WSNs are inherently vulnerable to malicious nodes. The malicious nodes execute inappropriate actions (*e.g.*, dropping packets and sending redundant packets) to destruct the network reliability and functionality, which triggers event losses. Even worse, such malicious nodes may lead to severe risks, particularly for real-time and safety-critical monitoring WSN applications, such as extreme weather monitoring, water quality monitoring, and forest fire alarming. Therefore, methods to detect malicious nodes in WSNs are strongly required.

Recently, with the development of machine learning technologies, many studies have proposed machine learning-based approaches to detect malicious nodes in WSNs. Generally, machine learning-based approaches train classifiers, which are based on observation of behaviors of nodes, to classify the category of nodes. However, existing methods ignore three main challenges. (i) Since general machine learning methods rely on training data, trained classifiers do not work well in test environments that are different from training environments. (ii) From the perspective of malicious nodes, malicious nodes can also learn from the detection methods to avoid being detected. Therefore, a fixed classifier cannot detect malicious nodes with learning ability. (iii) Energy harvesting cooperative (EHC) schemes are studied a lot recently. Nodes are allowed to share energy in EHC-WSNs, and some malicious nodes utilize this scheme to deprive energy from other nodes. Since the residual energy storage is private data, which could be fabricated by malicious nodes, regular machine learning-based methods do not work well because learning from fabricated data is meaningless.

In this thesis, we focus on detecting malicious nodes in WSNs with machine learning-based methods, and tackle the above-mentioned challenges. This thesis consists of five chapters. We introduce the research background and issues for detecting malicious nodes in WSNs in Chapter 1. In Chapter 2, we address the problem of detecting malicious nodes in unknown environments. In Chapter 3, we address the problem of detecting malicious nodes with learning ability. Then, in Chapter 4, we focus on detecting energy depriving malicious nodes. Finally, in Chapter 5, we summarize this thesis and discuss our future work.

In particular, in Chapter 2, we propose an ensemble learning method to detect malicious nodes in unknown network environments. We first prepare weak malicious node detectors trained in diverse environments, and then construct a strong ensemble malicious node detector, which is tailored to a given test environment, by fusing weak detectors whose performances are estimated to be high in the test environment. We investigate the performance of our method and confirm that our method significantly outperforms the state-of-the-art methods in terms of detection accuracy and false detection rate.

In Chapter 3, we construct a framework where the malicious and normal nodes can learn by competition. In our framework, malicious nodes learn to avoid being detected and normal nodes learn to detect malicious nodes with high accuracy. We design reinforcement learning-based malicious nodes, and define a novel observation space and sparse reward function for the reinforcement learning. A malicious node thus can utilize this method to learn from existing detection methods. We also design an adaptive learning method to detect these smart malicious nodes. We construct a robust classifier, which is frequently updated, to detect these smart malicious nodes. Extensive experiments show that, in contrast to existing attack models, the developed malicious nodes can degrade network performance without being detected. We also conduct simulation experiments to verify that our proposed method can detect malicious node with learning ability with high detection rate, in comparison with the existing methods.

In Chapter 4, we propose an unsupervised learning-based method for detecting energy depriving malicious nodes in an energy harvesting cooperative wireless sensor network (EHC-WSN). In EHC-WSNs, nodes wirelessly transfer a portion of their energy to their neighboring nodes if their neighboring nodes lack energy. An energy depriving malicious node may forge that it has little energy, thus it can deprive energy from its neighboring nodes. Detecting energy depriving malicious nodes is not a trivial task because the real energy storage is a private data of each node. For detecting such malicious nodes, we utilize an unsupervised approach because it is impossible to prepare labeled data in a WSN in the real world. In our method, each node first observes energy of its neighboring nodes, then it utilizes this information to obtain data points for clustering. The results through simulation experiments confirm the advantages of our proposed method over the comparison methods in terms of detection accuracy and false detection rate.

In Chapter 5, we conclude this thesis and discuss about our future work. Our proposed methods can detect malicious nodes in WNSs with a high detection rate. In particular, our proposed methods can detect different categories of malicious nodes even in unknown network environments. Even high-level malicious nodes that can learn cannot avoid being detected by our proposed methods. Our proposed methods can also detect malicious nodes in EHC-WSNs, which are a category of state-of-the-art WSNs. Therefore, our achievements contribute to increase the security of WSN services and utilization.

Contents

Chapter 1

Introduction

The rapid development of wireless communication technology in this era has led to many spectacular innovations [101]. Among them, wireless sensor networks (WSNs) [3] are one of the most highlighted innovations, where a number of small electronic devices (in this thesis, we call them nodes) perform data sensing and enable us to obtain the latest information from a physical world in real-time, and such information can be utilized to monitor events and understand current conditions.

WSNs have been increasingly adopted across the world due to reliable and practical reasons. The most important one of them is that the implementation of wireless sensor networks is easy (compared with wired devices that run on long and weighted cables). After the installation of wireless sensor nodes, they will form a WSN in a self-organizing manner.

Conventionally, the generated data from sensor nodes is aggregated to a centralized server called sink. End-users can analyze data from the sink to figure out what is happening in the monitored area. Figure 1.1 shows a forest fire monitoring WSN. In this figure, when a forest fire occurs, a nearby node generates data of high temperature from its temperature sensor. Then, the data is forwarded to the sink by multi-hop mechanism (*i.e.*, a node in the middle position of the other two nodes helps to transmit the data packet), and the end-user will find that monitored temperature is abnormal. The end-user, therefore, considers that a forest fire may occur.

Besides the forest fire monitoring, WSNs have many crucial monitoring applications as follows.

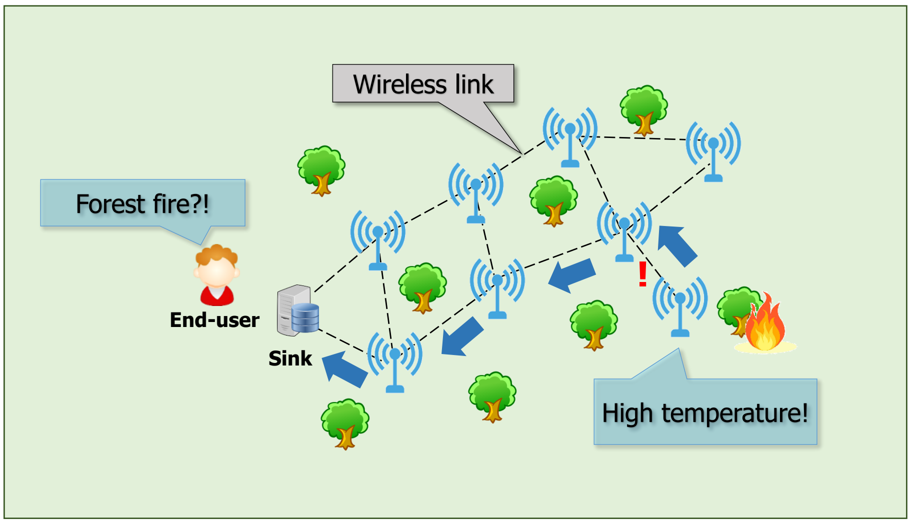


Figure 1.1: An example of a wireless sensor network for forest fire monitoring. Data of high temperature is forwarded to the sink by multi-hop mechanism.

1. Air condition monitoring: This monitors high values of elements (*e.g.*, carbon dioxide) and dangerous toxic pollutants (*e.g.*, PM 2.5) in air in diverse areas.

2. Weather monitoring: This monitors the speed of winds, extremely high temperatures and moisture of an area where is easy to suffer from bad weathers (*e.g.*, extremely heavy rain).

3. Underwater monitoring: This monitors the flow of a river, rhythm of a tide, and the quality of water to prevent damage to people's health and lives.

If these monitoring systems are effectively deployed, life-saving countermeasures can be appropriately imposed to prevent catastrophic situations. However, everything does not always go on as we expect. WSNs may suffer from attackers, who implement malicious nodes into them, due to the self-organizing inherence of WSNs, which is called the intrusion of malicious nodes. The intrusion of malicious nodes carries massive damage to WSNs in different ways. Figure 1.2 shows an example of malicious node intrusion, which influences the packet transmission. Similar to the example mentioned above, when a forest fire occurs, a node in a forest fire monitoring WSN, which is nearby

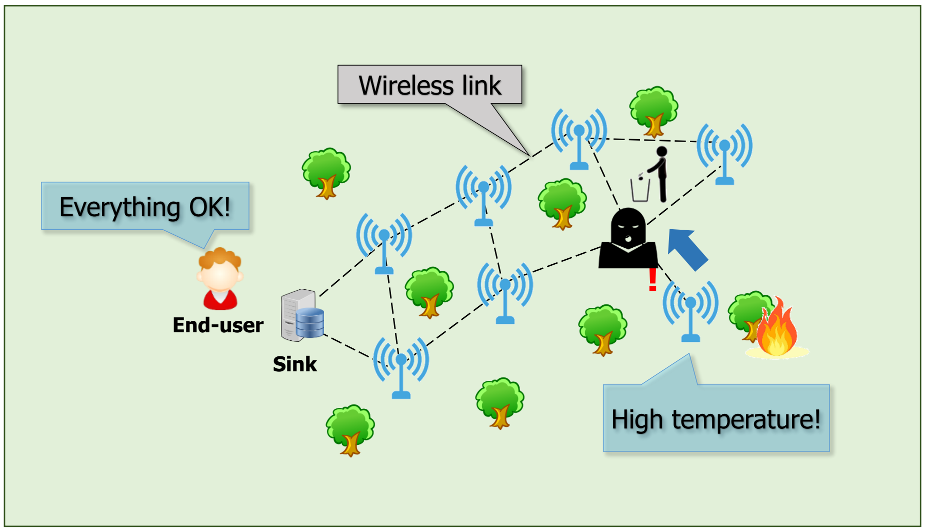


Figure 1.2: A wireless sensor network with malicious node intrusion. The malicious node drops the received data packets. Therefore, the data packets cannot be forwarded to the sink.

the fire, transmits data of high temperate to the sink. However, when a malicious node intrudes into the network, the malicious node does not transmit the received message but simply drops it. The data thus cannot be forwarded to the sink. The end-user will not notice that there is a forest fire happening. This example shows a situation where a packet dropping attack [110] happens.

With the development of energy harvesting cooperative (EHC) technologies, energy harvesting cooperative WSNs (EHC-WSNs) have been created to extend the lifetime of WSNs[34]. An EHC-WSN is a WSN where nodes can harvest energy from ambient environments and transfer energy to other neighboring nodes with low energy storage. Besides the above-mentioned packet dropping attack model, the malicious nodes can also hold other attack models that influence EHC-WSNs. Malicious nodes can pretend to have little energy storage to deprive energy from neighboring nodes. The neighboring nodes, thus, become no function when they have no energy. The EHC-WSNs then loss their monitoring function because nodes can no longer monitor events.

The intrusion of malicious nodes may cause severe event losses. Event would lead to catastrophic situations, such as miss observing a sudden huge torrent in an underwater

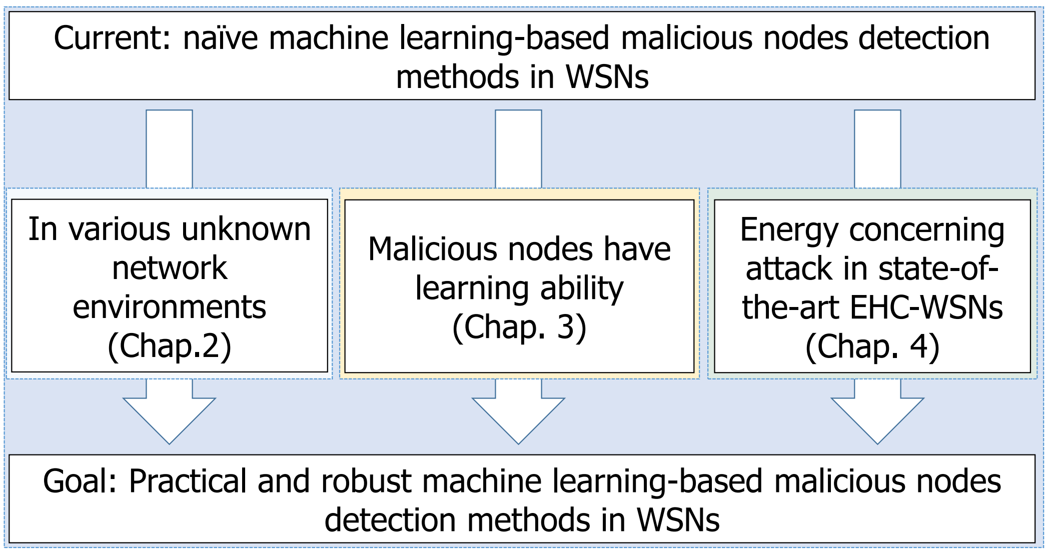

Current: naïve machine learning-based malicious nodes detection methods in WSNs
In various unknown network environments (Chap.2)
Malicious nodes have learning ability (Chap. 3)
Energy concerning attack in state-of-the-art EHC-WSNs (Chap. 4)
Goal: Practical and robust machine learning-based malicious nodes detection methods in WSNs

Figure 1.3: In this thesis, we tackle three challenging issues to obtain practical and robust machine learning-based malicious nodes detection methods in WSNs.

monitoring WSNs, and miss observing toxic pollutants in an air condition monitoring WSN. These event losses could cause disastrous consequences, which threaten people's lives. Therefore, detecting malicious nodes in WSNs is a crucial task. However, it should be noted that there is still a gap between the security of WSNs and detecting malicious nodes in WSNs. That is, detecting malicious nodes in WSNs definitely contribute to the security of WSNs, but the effect of detecting malicious nodes needs further investigation.

1.1 Research Issues

For utilization of WSNs, we need to provide secure protocols. A large number of works [18, 35, 41, 88, 90] have focused on enhancing the security level for wireless sensor networks, for example, building encryption frameworks and designing handcrafted detecting rules. However, faced with the fact that malicious nodes may overcome the encryption system and intrude into the network, only an encryption framework is not enough to protect the security of WSNs. Moreover, handcrafted rules are not capable of robustly detecting malicious nodes under different kinds of attack models. Therefore, machine learning-based methods [2, 73] have been proposed to detect malicious nodes.

Machine learning-based methods build classifiers based on the observed behaviors of malicious nodes (*e.g.*, the numbers of messages received and sent, and the number of neighboring nodes), and each normal node is equipped with the classifier for detecting malicious nodes. With the machine learning-based methods, malicious node detection rules can be automatically built, even in environments where multiple attacks exist. However, some challenging issues still exist when we utilize machine learning-based techniques to detect malicious nodes in WSNs. As shown in Figure 1.3, naïve machine learning-based methods cannot solve following issues. i) Detecting malicious nodes in unknown network environments. ii) Detecting malicious nodes with learning ability. iii) Detecting energy depriving malicious nodes in state-of-the-art EHC-WSNs. By tackling these challenging issues, we can obtain more practical and robust machine learning-based malicious nodes detection methods in WSNs. In this section, we describe the details of three research issues for accurately detecting malicious nodes in WSNs.

1.1.1 Unknown network environments

WSNs can be implemented in different places for various purposes as above-mentioned. Different WSNs have different network parameters (*e.g.*, network topologies, packet transmission frequencies, and number of nodes). Since the trained classifier detects malicious nodes based on the training data, a classifier trained in a specific environment may not work well in other testing environments with different network parameters. For example, assume that we have prepared a classifier A to detect packet dropping malicious nodes in a WSN where packet frequency is low. Since nodes communicate using wireless links where the network bandwidth is limited, the increase of packet frequency results in packet losses. If we simply apply classifier A to another WSN where packet frequency is high, because the packet losses in this network are naturally high, the classifier A may classify normal nodes as malicious nodes due to the high packet losses of normal nodes. That is, simply applying any trained classifiers to unknown network environments is impractical.

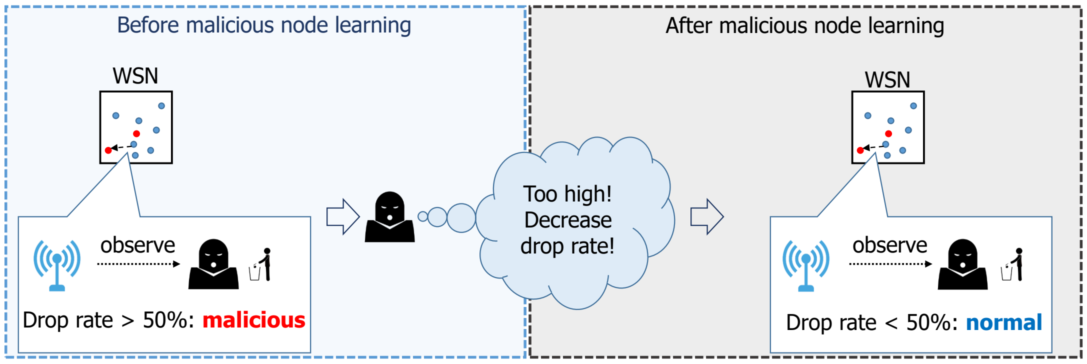

Before malicious node learning
WSN
observe
Drop rate > 50%: malicious
Too high! Decrease drop rate!
After malicious node learning
WSN
observe
Drop rate < 50%: normal

Figure 1.4: A malicious node with learning ability learns from the detection method to avoid being detected

It is also impractical to obtain a machine learning-based method which is trained in a simulated training environment whose network parameters are exactly the same as those of the real test environment. This is because the network parameters of the real test environments are unknown in advance. Therefore, a robust malicious node detection method that works well in diverse environments is required.

1.1.2 Malicious nodes with learning ability

Machine learning-based methods have been proposed to detect malicious nodes in WSNs. However, the development of technology also pushes the dark side going forward. From the perspective of malicious nodes, it is natural to consider that malicious nodes also have learning ability. That is, malicious nodes can learn from detection methods to try to avoid being detected.

Figure 1.4 shows an example of a malicious node with learning ability. In this example, a trained classifier is utilized to detect malicious nodes in a WSN. Each normal node observes the behaviors of neighboring nodes and feeds them into the classifier to decide the category of neighboring nodes. Assume that one of the rules of this classifier is that if a node drops 50 percent of packets, this node is decided as a malicious node. The malicious node, which is equipped with a machine learning-based method (*e.g.*, a reinforcement learning-based method), is capable of learning from the built classifier. After the learning procedure, the malicious node figures out that if the packets dropping

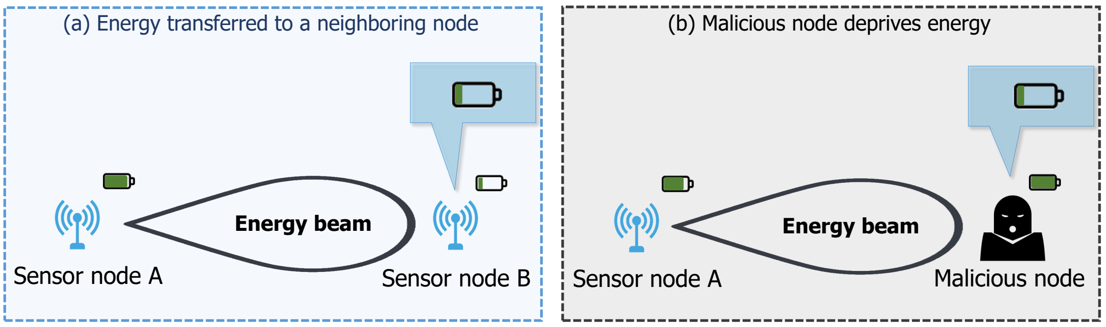

(a) Energy transferred to a neighboring node
Energy beam
Sensor node A
Sensor node B
(b) Malicious node deprives energy
Energy beam
Sensor node A
Malicious node

Figure 1.5: Energy transferring with and without malicious node intrusion

rate of it is higher than 50 percent, it will be detected. As a result, the malicious node can decrease its packet dropping rate to lower than 50 percent to try to avoid being detected.

Therefore, a method to detect such kind of smart malicious nodes is required. This detection method should be robust enough to detect malicious nodes, even if these malicious nodes can learn this method.

1.1.3 Energy depriving attack in energy harvesting cooperative WSN

Wireless sensor nodes are small and lightweight electronic devices. For the purpose of easy implementation and transportation, battery capacities of wireless sensor nodes are often limited [74]. Therefore, the lifetime of a wireless sensor network is strictly limited. Many studies [12, 63] have proposed methods for extending the lifetime of WSNs. For example, duty cycle scheduling methods are a category of techniques that let nodes sleep (*i.e.*, stop working) and wake up periodically to save energy.

Recently, with the development of energy harvesting cooperative technology, energy harvesting cooperative WSNs (EHC-WSNs) are created [34]. An EHC-WSN is a WSN where nodes can harvest energy from ambient environments (*e.g.*, harvesting from solar energy [108] and vibration [15]) and transfer energy to other neighboring nodes. EHC-WSNs relieve the bottleneck of energy limitation in WSNs by extending the lifetime of nodes. However, little study concerned about the security of EHC-WSN.

Note that the ambient environments are always changing. For example, in a solar energy harvesting system, a node harvests much less energy than others if it is covered by shadow. Therefore, in EHC-WSNs, a node transfers energy to a neighboring node if this neighboring node has low energy storage (Figure 1.5 (a)). However, this setting may suffer from malicious nodes if the malicious nodes pretend to have little energy storages (Figure 1.5 (b)). Malicious nodes thus can deprive the energy of normal nodes.

Detecting energy depriving malicious nodes is not a trivial task because the energy storage of each node is private data. Even if energy transferring schemes of EHC-WSNs require nodes to report their energy storages, malicious nodes can still fabricate their energy storages to pretend to have little energy storages. Therefore, a method to detect energy depriving malicious nodes is required.

1.2 Research Contents

In this thesis, we propose malicious node detection methods in wireless sensor networks. Our methods, which utilize various machine learning-based techniques, address and solve the research issues stated in Chapter 1.1, to accomplish our goal shown in Figure 1.3 that we create practical and robust machine learning-based malicious nodes detection methods in WSNs. The outlines of the proposed methods are as follows.

- Detecting malicious node by ensemble learning

  In Chapter 2, we address the issue of utilizing machine learning-based methods to build classifiers to detect malicious nodes in WSNs where the test environments are unknown. To address this problem, we design an environment-adaptive malicious node detector, as well as robust classification features, for WSNs where multiple attacks exist. In our method, we first prepare weak malicious node detectors trained in different network environments. Then, we design a weak detector evaluator, which estimates the detection performance of a trained weak detector when the detector is applied in a test environment where training data is unavailable. Therefore, utilizing the weak detector evaluator, our method can automatically find detectors that would work well in the test environment, and constructs a strong ensemble malicious node detector tailored to the test environment, based on the found weak detectors. In addition, to robustly detect various attacks in diverse environments, we extract inherent features of these attacks for building a weak detector. In these ways, we can detect malicious nodes in an unknown test environment.

- Detecting malicious node with learning ability

  In Chapter 3, we address the problem that malicious nodes can learn from the detecting method to avoid being detected, and to cause stronger harmful effects (*e.g.*, decrease the packet transmission rate). To address this problem, we construct a framework where the malicious and normal nodes can learn by competition. In our framework, malicious nodes learn to avoid being detected and to cause stronger harmful effects, and normal nodes learn to detect malicious nodes with high accuracy. To enable the malicious nodes to learn from the detection method by themselves, we utilize reinforcement learning-based techniques. In our method, malicious nodes obtain a positive reward when they degrade the performance of the WSN. Therefore, to obtain more rewards, malicious nodes will try to carry stronger harmful attacks. However, if these nodes behave too maliciously, they may be easily detected by countermeasures of the WSN. We thus assume that the malicious nodes receive a large negative reward if they are detected within a time threshold. We also prepare a design of the state space to enable the malicious nodes to recognize their current situations. After learning in a large number of episodes, the malicious nodes, which employ a reinforcement learning-based method, eventually learn how to perform as maliciously as possible without being detected. In addition, we design a method that robustly detects the above attack model. We extract inherent features for the detection method to detect malicious nodes, and the detection method is updated simultaneously within the episodes. Over a certain period, the detection method is adaptively updated to determine the optimal rule of attack detection.

- Detecting energy depriving malicious node in EHC-WSNs

  As stated in Section 1.1.3, EHC-WSNs have been proposed to extend the lifetime of wireless sensor networks. In an EHC-WSN, a sensor node `a` with plenty of energy can wirelessly transfer its energy to a neighboring node `b` with little energy to extend the life-time of node `b`. However, EHC-WSNs suffer from the energy depriving attack, where malicious nodes pretend to have little energy storages to deprive energy from normal nodes.

In Chapter 4, we address the problem of detecting energy depriving nodes. We propose a method that detects the energy depriving node by unsupervised learning. In our method, each normal node, playing the role as an observing node, first obtains data about behaviors of neighboring nodes. Then, the observing nodes create data points from the observed data and utilize the data points to form clusters. After the clustering, observing nodes utilize the clustering results to identify malicious nodes.

1.3 Organization of Thesis

This thesis consists of five chapters, and the remainder of the thesis is organized as follows.

In Chapter 2, we introduce and discuss about the problem of detecting malicious nodes in WSNs where a given environment is unknown in Section 2.1. Section 2.3 describes the assumptions of our work including the network environment and routing protocol employed, and attacks which are assumed to be executed by malicious nodes. Our proposed method is described in Section 2.4. In this section, we first describe the overview of our method. Then, we introduce our designed features, and describe how we ensemble weak detectors to a strong robust detector. The experiment setup and the simulation results to exhibit the performance of our proposed method are described and discussed in Section 2.5. In Section 2.2, we review the prior works related to the topic of this chapter. Finally, we conclude our work in this chapter in Section 2.6. The study in this chapter is based on our works published in [29], [30], and [31].

In Chapter 3, we introduce and discuss about the problem that the malicious nodes can learn from the detecting method to avoid being detected in Section 3.1. Section 3.2 reviews the related work of this topic. We describe the assumptions of our work in this chapter in Section 3.3. In Section 3.4, we describe our proposed methods. Our proposed methods in this chapter consist of the design of malicious nodes, and a method that detects the malicious nodes with learning ability. We compare our proposed method with existing methods by conducting experiments, and we conduct network parameter analysis as well. In addition, we compare the performances of malicious nodes with and without learning ability. The details are discussed in Section 3.5. Finally, this chapter is summarized in Section 3.6. The study in this chapter is based on our works published in [28] and [32].

In Chapter 4, we focus on detecting energy depriving malicious nodes in EHC-WSNs. We give an introduction of EHC-WSNs and threats to ECH-WSNs in Section 4.1. In Section 4.2, we review related works of this topic in the chapter. The assumptions, which include the network model, schemes of energy harvesting and energy cooperation, and attack model, are described in Section 4.3. Section 4.4 shows our proposed method. We conduct experiments to examine the performance of our proposed method, and the setting and results of experiments are described in Section 4.5. Finally, our work in this chapter is concluded in Section 4.6. The study in this chapter is based on our work published in [27].

Finally, in Chapter 5, we summarize this thesis and discuss about future works.

Chapter 2

Detecting Malicious Nodes in Wireless Sensor Networks by Ensemble Learning

2.1 Introduction

Recent advances in micro-electro-mechanical systems (MEMS) technology, wireless communications, and digital electronics have enabled the development of low-cost, low-power, multifunctional sensor nodes that are small in size and communicate untethered in short distances [93]. These tiny sensor nodes, which consist of sensing, data processing, and communicating components, leverage the idea of wireless sensor networks (WSNs) based on collaborative effort of a large number of nodes. A WSN is typically self-organized, *i.e.*, it consists only of (mobile) wireless nodes, and each node can directly communicate with other nodes, if they are within the communication range. Since WSNs do not require any infrastructures, they can provide supports in a wide variety of situations, for example, in fire monitoring and underwater monitoring systems [23, 49, 85]. However, due to the self-organized characteristic, malicious nodes can easily join a WSN. That is, WSNs are inherently vulnerable to malicious nodes [26, 95, 113], and the malicious nodes disrupt communications in WSNs, causing packet transmission failures [37]; all of which presents security challenges to WSN environments.

2.1.1 Motivation

Numerous studies have proposed rule-based methods for avoiding attacks [76, 82] and/or detecting malicious nodes [11, 55, 59, 60, 98, 107]. However, they assume unrealistic situations, specifically, where only a single kind of attack is executed in a given WSN. We can intuitively consider that (i) malicious nodes change their attack models to escape a malicious detection method for a specific attack and (ii) different malicious nodes execute different attacks. The existing methods cannot deal with such cases, resulting in less practical effectiveness.

The above rule-based methods cannot deal well with multiple attacks situations, because preparing a handcrafted robust detection rule for these attacks is impractical. Therefore, machine learning-based approaches have recently been receiving attention [2, 21, 70]. Such approaches build a classifier based on the observed behavior of malicious nodes (*e.g.*, the numbers of messages received and sent), and each normal node is equipped with the classifier for detecting malicious nodes. One advantage of this approach is that it can automatically build a robust malicious detection rule, even in environments where multiple attacks exist, if we prepare training data obtained from similar multiple attacks environments. Note that, since the trained classifier detects malicious nodes based on the training data, a classifier trained in a specific environment may not work well in other testing environments with different network parameters (such as the number of nodes and network size), which is an inherent problem in WSNs environments. However, the existing machine learning-based methods [2, 21, 70] do not address this issue (*i.e.*, difference in training and testing environments). When we deploy a machine learning-based malicious detection method for WSNs in a real test environment, using a trained model in simulated environments is the most practical way because collecting labeled data in real training environments is impractical. In such a case, it is also impractical to prepare simulated training environments whose network parameters are exactly the same as those of the real test environments. This is because the network parameters of the real test environments are unknown in advance. Therefore, a robust malicious node detection method that works well in diverse environments is required.

2.1.2 Contribution

To address this problem, we design an environment-adaptive malicious node detector, as well as robust classification features, for WSN where multiple attacks exist. In our method, we first prepare weak malicious node detectors trained in different environments. Then, our method automatically finds detectors that may work well in a test environment, and constructs a strong ensemble malicious node detector tailored to the test environment, based on the found weak detectors. Our contributions of this chapter are summarized below.

- We tackle the problem of malicious node detection in WSNs where multiple attack models exist. To our knowledge, this is the first work that constructs a robust malicious node detector based on ensemble learning. There is no other study that tests the performance of a detection model trained in environments that are different from a testing environment.

- We propose a method for constructing a robust malicious node detector tailored to a test environment by combining weak detectors trained in diverse environments that are estimated to work well in the test environment.

- We conduct empirical study to investigate the performance of our method and confirm that our method outperforms the state-of-the-art.

The organization of this chapter is as follows. Section 2.3 introduces the assumption in this chapter. Our proposed method is described in Section 2.4, and experimental results are illustrated in Section 2.5. We review related works in Section 2.2, and this chapter is concluded in Section 2.6.

2.2 Related work

Since WSN security is a challenging issue, there are many existing studies.

As a famous attack in WSNs is packet dropping one (*e.g.*, black hole and gray hole attacks), many works have developed techniques for avoiding the attack with sophisticated multipath approaches [102], analyzing the impact of the attack [95], and preventing the attack [86]. Other attacks have also been addressed. In [1], sybil attacks are detected by using RSS (received signal strength), while [22] has proposed a method that detects wormhole attack with topology information. In flooding attack scenario, [90] has proposed a trust estimation method based on DSR (dynamic source routing protocol) and [45] has considered how to prolong network lifetime, since flooding attack consumes node energies. These works assume that there is a single attack model in the network, and the proposed methods work only in their respective assumed environments. Therefore, it is not trivial to extend the methods to deal with the case where multiple attack models exist in a WSN and nodes can arbitrarily change attack models.

Recently, machine learning approaches have received attention [2, 70, 89], since they require no complex system parameters (*e.g.*, thresholds for identifying malicious nodes). In addition, machine learning approach has potential to detect malicious nodes in environments where multiple attack models exist. (Unfortunately, [2, 21, 89] miss this advantage and consider only a single attack.) [70] is a state-of-the-art method but is not designed to cope with a situation where training and test environments are different. Hence, this method does not accurately classify nodes.

2.3 Assumption

2.3.1 Network model

The environment is assumed to be a wireless sensor network (WSN) consisting of n wireless nodes with unique identifiers. Without loss of generality, we assume these nodes can move. These nodes move freely and can directly communicate with other nodes if they are within the communication range. We assume that all nodes have the same communication range[1], and if a given node is within the communication range of other nodes, it is a neighboring node of them.

[1]As an exception, some malicious nodes may use different channels, which is described in Section 2.3.2.

As a routing protocol, AODV [80], which is a standard routing protocol in WSNs, is employed. That is, when a node `s` wants to send a data packet to another node `d`, `s` broadcasts a route request (RReq) to create a packet transmission route, and this message is transmitted by some intermediate nodes. When node `d` receives the RReq, it sends a route reply (RRep) toward `s`, then `s` sends the data packet and it is transmitted through the route. To maintain the route, route error (Rerr) and hello messages are also utilized (see [80] for detail).

2.3.2 Attack model

This section introduces attacks which are assumed to be executed by malicious nodes.

Black hole attack [68, 82]. Malicious nodes act like a cosmic black hole. Specifically, the malicious nodes send fake RReps to all RReqs and pretend to have a route to given destination nodes. After receiving data packets, the malicious nodes drop them.

Gray hole attack [107]. This attack is similar to black hole attack. The difference is that malicious nodes drop data packets stochastically.

Sybil attack [1]. Malicious nodes, which execute this attack, pretend to be normal nodes. In this situation, nodes near the malicious nodes consider that the neighboring nodes are normal, so packets are consequently dragged to the malicious nodes.

Routing packet dropping attack [60]. In this attack routing packets (*e.g.*, RReq, RRep, and Rerr) are randomly dropped.

Rushing attack [41]. In this attack, when a malicious node receives a RReq from its neighboring node, it broadcasts the RReq quickly, so that it can be an intermediate node of the packet transmission route.

Wormhole attack [25]. A malicious node executing this attack stores all received messages and data packets, and send them to another *distant* malicious node on their own channel (*e.g.*, a wired link or an out-of-band hidden channel). The distant malicious node receiving the messages broadcasts them locally, creating an illusion that the neighboring nodes of these two malicious nodes are very close to each other.

Jelly fish delay attack [55]. In jelly fish delay attack, malicious nodes hold received messages for a while before transmission. Malicious nodes use this attack to disturb the

Table 2.1: Classification of attacks based on their objectives

Class	Attack
CLF	Routing packet dropping, sybil attack, jelly fish delay attack
DDP	Black hole attack, gray hole attack
DRA	Sybil attack, rushing attack, wormhole attack, black hole attack
CRT	RReq flooding attack, hello flooding attack

activeness of the routing tables of their neighboring nodes.

Flooding attack [110]. The objective of this attack is energy consumption. A typical approach of this attack is to broadcast a RReq without a destination. Since all nodes transmit this message, redundant traffic incurs.

It is important to note that the above attack models have been extensively studied, and are worth being considered at the same time, to improve WSN security.

To efficiently detect multiple attacks, we classify attack types into the four groups below, according to the objectives of the attacks. The objective-based attack types are classified as follows: (i) causing link failure (CLF), (ii) dropping data packets (DDP), (iii) dragging routes to attackers (DRA), and (iv) causing redundant traffic (CRT).

Our attack classification is summarized in Table 2.1. Note that some attacks, such as black hole and sybil attacks, are categorized into multiple classes. For example, in a black hole attack, malicious nodes send RReps and drop data packets, and this attack model has two objectives: packet dropping and route dragging.

2.3.3 Malicious node detection

Each normal node in a given WSN holds a classifier to detect malicious nodes. Specifically, each normal node judges whether or not the neighboring nodes are malicious by using our method, which employs observed behaviors of the neighboring nodes by the normal node and the classifiers. The detection operation can be executed at any time, as long as the information on the observed behaviors is enough for accurate detection. It is clear that less information makes machine learning approaches not function,

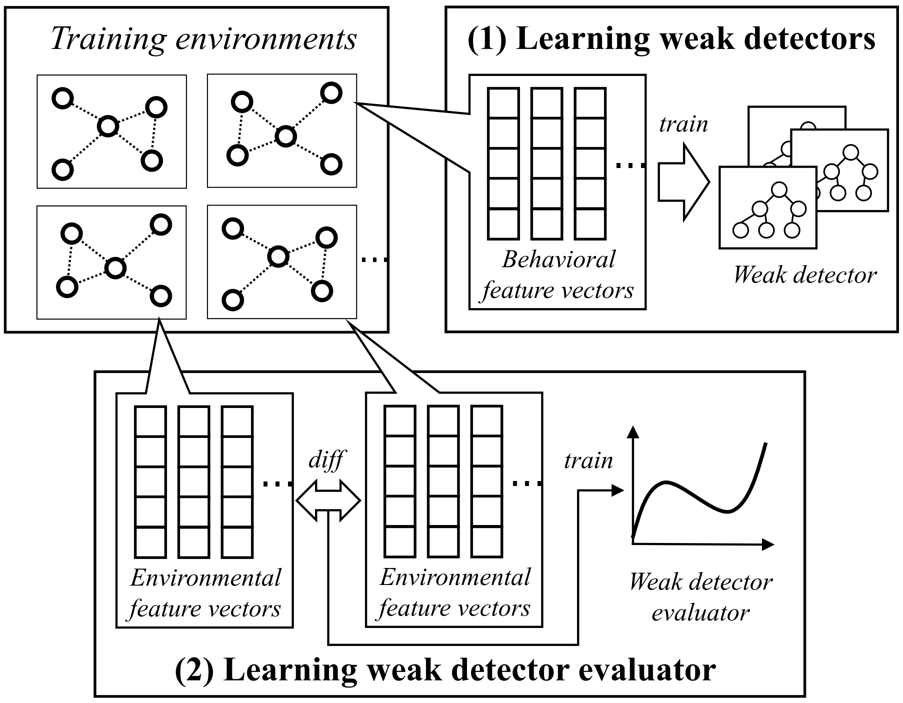

Training environments
(1) Learning weak detectors
train
Behavioral feature vectors
Weak detector
diff
train
Environmental feature vectors
Environmental feature vectors
Weak detector evaluator
(2) Learning weak detector evaluator

Figure 2.1: Overview of the two steps of training phase in the proposed method.

so frequent execution of malicious nodes detection is hard to be assumed. In our experiments, after normal nodes observe the behavior of their neighboring nodes over a certain period (*e.g.*, 50 or more seconds), they judge whether or not the neighboring nodes are malicious. Therefore, the malicious node detection procedure does not incur high computation costs of the normal nodes.

2.4 Proposed method

2.4.1 Overview

Figures 2.1 and 2.2 present an overview of the proposed method. The method, which is based on machine learning techniques, consists of a training phase and test phase. In the training phase, we simulate various environments with different environmental parameters, *e.g.*, node speed and node density, and train a *weak malicious node detector* (*weak detector* for short) for each training environment using data obtained from the

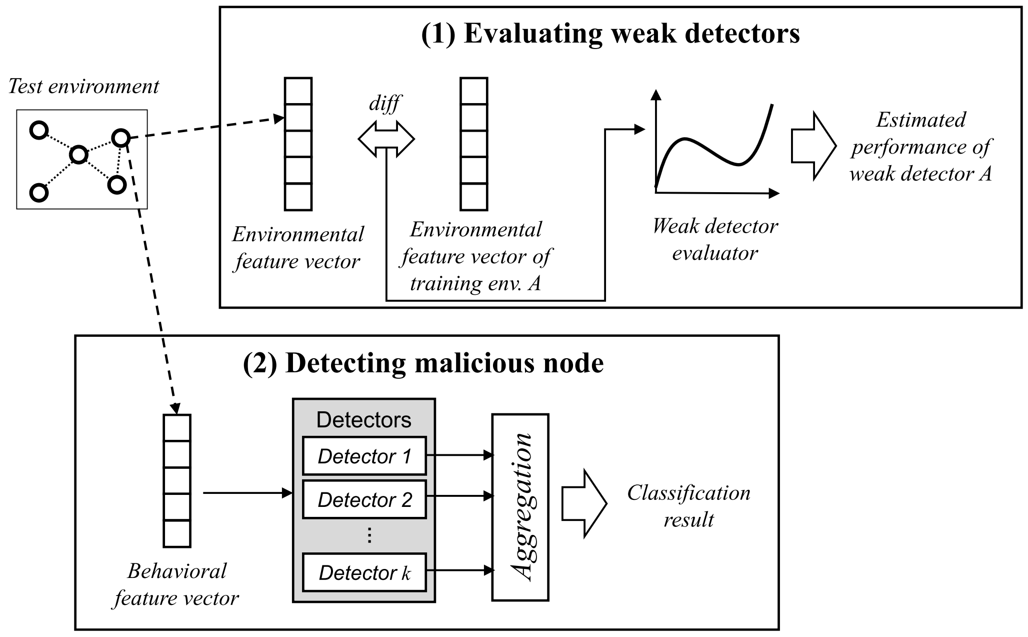

(1) Evaluating weak detectors
Test environment
diff
Environmental feature vector
Environmental feature vector of training env. A
Weak detector evaluator
Estimated performance of weak detector A
(2) Detecting malicious node
Detectors
Detector 1
Detector 2
Detector k
Aggregation
Behavioral feature vector
Classification result

Figure 2.2: Overview of the two steps of test phase in the proposed method.

all normal nodes in the environment (Figure 2.1 (1)). Since each normal node observes the behavior of its neighboring nodes (*e.g.*, packet transmission frequency) and judges whether or not each of the neighboring nodes is malicious, the normal node extracts features from the observed behavior of the neighboring node and feeds them into a weak detector. These features are hereinafter called *behavioral features*. It is important to note that the behavioral features are obtained from messages sent by neighbor nodes. In other words, each normal node can obtain the behavioral features by overhearing messages, so our method incurs no additional communication costs.

In addition, we train a *weak detector evaluator*, which estimates the detection performance of a trained weak detector when the detector is run in a test environment where training data is unavailable (Figure 2.1 (2)). Note that the weak detector is trained in a training environment different from the test environment. Since we do not have ground truth labels in the test environment and thus cannot directly compute the performance of the weak detector in the test environment, we assume that a normal node in the test environment estimates the performance of the weak detector using its observable information. To achieve the performance estimation in an unseen test environment, we again use supervised machine learning techniques. We first select pairs of simulated training environments and calculate training data for the performance estimation. Assume that we select environments A and B. We then compute features that well describe environment A from observable information by nodes in the environment. We also compute features describing environment B. After that, differences in the two environments are calculated from the features of the two environments. In addition, we run the weak detector, which is trained in environment A, in environment B to compute the detection performance. By using the calculated environmental differences associated with the performance of the weak detector, we train an estimator that estimates the performance of a weak detector when it is run in an unseen test environment.

Our focus on environmental differences is derived from the supposition that a weak detector trained in an environment with one or more parameters completely different from those of the test environment may not work well in the latter environment. For example, it is obvious that a weak detector trained in an environment where the node density is high will not work well in sparse environments, since, if the node density differs, the message reception ratio may also differ, resulting in different distributions of behavioral features in the two environments. The inputs of the weak detector evaluator are features defining the difference between the test and training environments, which is computed based on features describing characteristics of each environment, which are hereinafter called *environmental features*. Note that these environmental features are computed based on information observable by normal nodes. We assume that such features are representative information concerning a given environment of interest (*e.g.*, information related to node speed), and affect the performance of a weak detector trained in a different environment.

In the test phase, a normal node estimates the performance of each weak detector in a test environment, using environmental features from both the test environment and the training environment in which the weak detector was trained (Figure 2.2 (1)). The normal node then constructs an ensemble strong detector, tailored to the test environment, by combining weak detectors whose performances are estimated to be high in the test environment. The normal node uses the ensemble detector to detect neighboring malicious nodes (Figure 2.2 (2)).

2.4.2 Features

Here we describes the behavioral features, which are used in malicious node detection, and the environmental features, which are used in the performance estimation of weak detectors.

Behavioral features

A number of attack models have been considered in WSN environments, and many studies classify such attacks based on layers or routing protocols [6]. However, to efficiently detect multiple attacks, we classify attack types into the four groups below, according to the objectives of the attacks; and then design features to be extracted, by focusing on the behavior of malicious nodes attempting to achieve these objectives (see Section 2.3.2). Note that the features are extracted from the behavior of a node of interest observed over a specified period.

We assume that each node observes the behavior of its neighboring node. Before we describe the behavioral features, we present the information related to the neighboring node observed by each node (Table 2.2), which is used to compute the behavioral features. Remember that in AODV, messages are transmitted by broadcasting, and thus observing nodes can overhear the messages.

Table 2.3 shows the 18 behavioral features used in our method, all designed based on the classification of attacks. Many existing machine learning-based malicious detection studies design classification features based on specific *numbers* (*e.g.*, the number of hello messages); however, here they are designed based on *ratios* calculated from the information observed by nodes, because such ratios are more robust against differences in network parameters than mere numbers. For example, it is clear that features based solely on numbers can be easily influenced by the number of neighboring nodes.

We now explain the relationship between our designed behavioral features in Table 2.3 and the classifications in Table 2.1. RRep sent ratio (RepSenRatio), RRep ignored ratio (RepIgnRatio), Rerr sent ratio (RerSenRatio), and Rerr received ratio (RerRecRatio) are designed for CLF. A link failure is caused by network topology change and by ignoring routing messages such as RReps. If routing messages are ignored by malicious nodes, normal nodes cannot properly update their routing tables, and thereby

Table 2.2: Information observed by each node

Information	Definition
NReqRec	# RReq overheard from one neighbor
NRepRec	# RRep overheard from one neighbor
NRepSen	# RRep sent to one neighbor
NRerRec	# Rerr overheard from one neighbor
NRerSen	# Rerr sent to one neighbor
NHelRec	# hello messages overheard from one neighbor
NDatRec	# data packets overheard from one neighbor
NDatSen	# data packets sent to one neighbor
TReqRec	Total # RReq overheard by observing node
TReqSen	Total # RReq sent by observing node
TRepRec	Total # RRep overheard by observing node
TRepSen	Total # RRep sent by observing node
TRerRec	Total # Rerr overheard by observing node
TRerSen	Total # Rerr sent by observing node
THelRec	Total # hello messages overheard by observing node
THelSen	Total # hello messages sent by observing node
TDatRec	Total # data packets overheard by observing node
TDatSen	Total # data packets sent by observing node

cannot transmit messages according to the latest network topology. To deal with this problem, the ratios related to the received and sent routing messages are designed to determine whether or not neighboring nodes drop routing messages.

Data received ratio (DatRecRatio), Data ignored ratio (DatIgnRatio), RRep useless ratio (RepUslRatio), RRep useful ratio (RepUsfRatio), and RReq useful ratio (ReqUsfRatio) are designed for DDP. We can know whether or not neighboring nodes transmit data packets properly, from the ratios calculated based on the number of overheard data packets. For example, if the DatIgnRatio of one neighboring node is much lower than those of other neighboring nodes, it is reasonable to infer that this neighboring node has ignored some data packets.

Table 2.3: Behavioral features

Behavioral features	Definition
RepSenRatio	NRepRec / TRepRec
RepRecRatio	NRepRec / TReqRec
RepIgnRatio	NRepRec / NRepSen
ReqRecRatio	NReqRec / TReqRec
DatSenRatio	NDatSen / TDatSen
DatRecRatio	NDatRec / TDatRec
DatIgnRatio	NDatRec / TDatSen
RerRecRatio	NRerRec / TRerRec
HelRecRatio	NHelRec / THelRec
AllPckRatio	(NDatSen+NRepSen) / (NDatRec+NRepRec)
RepUslRatio	NDatRec / NRepSen
RepReqRatio	NRepRec / TReqSen
RerSenRatio	NRerSen / TRerSen
HelCheckRatio	NHelRec / THelSen
ReqIgnRatio	NReqRec / TReqSen
RepUsfRatio	NRepSen / TDatRec
ReqUsfRatio	NReqRec / TDatRec
HelUsfRatio	THelSen / TDatRec

Data packet sent ratio (DatSenRatio), RRep checked by RReq ratio (RepReqRatio), RRep received ratio (RepRecRatio), all routing packets ratio (AllPckRatio), and RReq ignore ratio (ReqIgnRatio) are designed for DRA, since, if a malicious node wants to drag routes to itself, it will send more RReps and receive many data packets.

RReq received ratio (ReqRecRatio), Hello check ratio (HelCheckRatio), Hello received ratio (HelRecRatio), and Hello useful ratio (HelUsfRatio) are designed for CRT, because malicious nodes executing flooding attacks, send extra hello messages and/or RReqs to generate redundant traffic, and these features are thus useful for identifying such malicious nodes.

Table 2.4: Environmental features

Env. features	Definition
AvgReqRec	Avg. # RReq overheard by observing node
AvgReqSen	Avg. # RReq sent by observing node
AvgRepRec	Avg. # RRep overheard by observing node
AvgRepSen	Avg. # RRep sent by observing node
AvgDatRec	Avg. # data packets overheard by observing node
AvgDatSen	Avg. # data packets sent by observing node
AvgRerRec	Avg. # Rerr overheard by observing node
AvgRerSen	Avg. # Rerr sent by observing node
AvgHelRec	Avg. # hello messages overheard by observing node
AvgHelSen	Avg. # hello messages sent by observing node
AvgNeiMet	Avg. # neighboring nodes

Environmental features

We extract features that well describe the characteristics of the network (environment) in which a given node is present. Based on these features, we then find a weak detector that performs well in a test environment. To capture the general characteristics of a given network, we extract the features listed in Table 2.4 from information about sent and received packets observed by a normal node over a specified period. Put simply, each normal node counts the number of a particular kind of sent/received packet within a given time window. Let x_i be the number of such packets that a given node observes within the time window. When ω is the duration of the window in seconds, the specific environmental feature is simply calculated as $\frac{\sum x_i}{\omega}$.

2.4.3 Estimating the performance of the malicious node detector

After training a weak detector in each training environment, we train a weak detector evaluator, which estimates the performance of a given weak detector in an environment different from that in which it was trained.

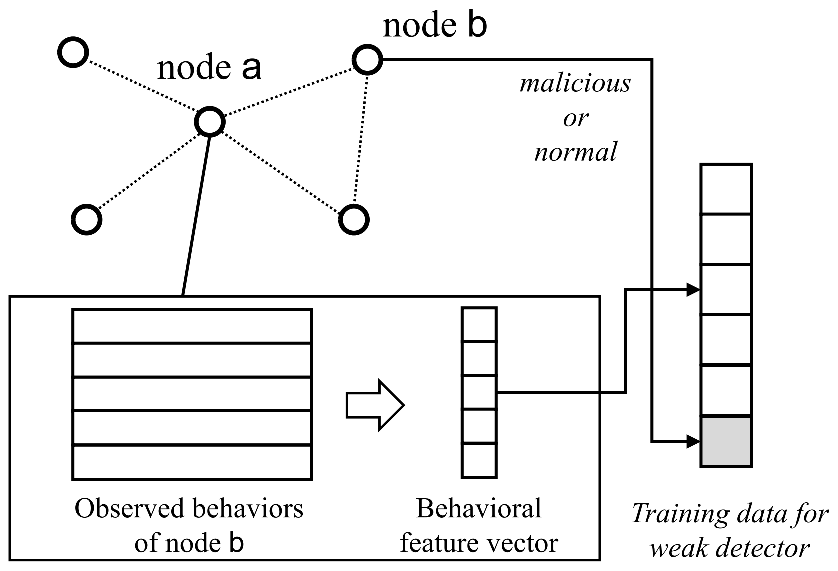


Figure 2.3: Preparing training data for the weak detector.

Weak malicious node detector for each environment

We prepare labeled training data for each simulated training environment, and train a weak detector for that environment using this training data. Assume that node a observes the behavior of node b (Figure 2.3). Node a extracts behavioral features from the observed behavior of node b, and constructs a behavioral feature vector concatenating the extracted features. In addition, as it is known, in the *training* environment, whether or not node b is malicious, we associate this information with the behavioral feature vector as a label. By applying the above procedures to each pair of neighboring nodes in the training environment, we can obtain a set of labeled behavioral feature vectors, which are used to train a weak detector in this training environment. Since a weak detector classifies a neighboring node of interest into a malicious or normal class, we employ the random forest [10] in [109], which is the state-of-the-art discriminative classifier, as the weak detector. Note that we randomly undersampled the majority class so that the proportion of training instances from each class was equal.

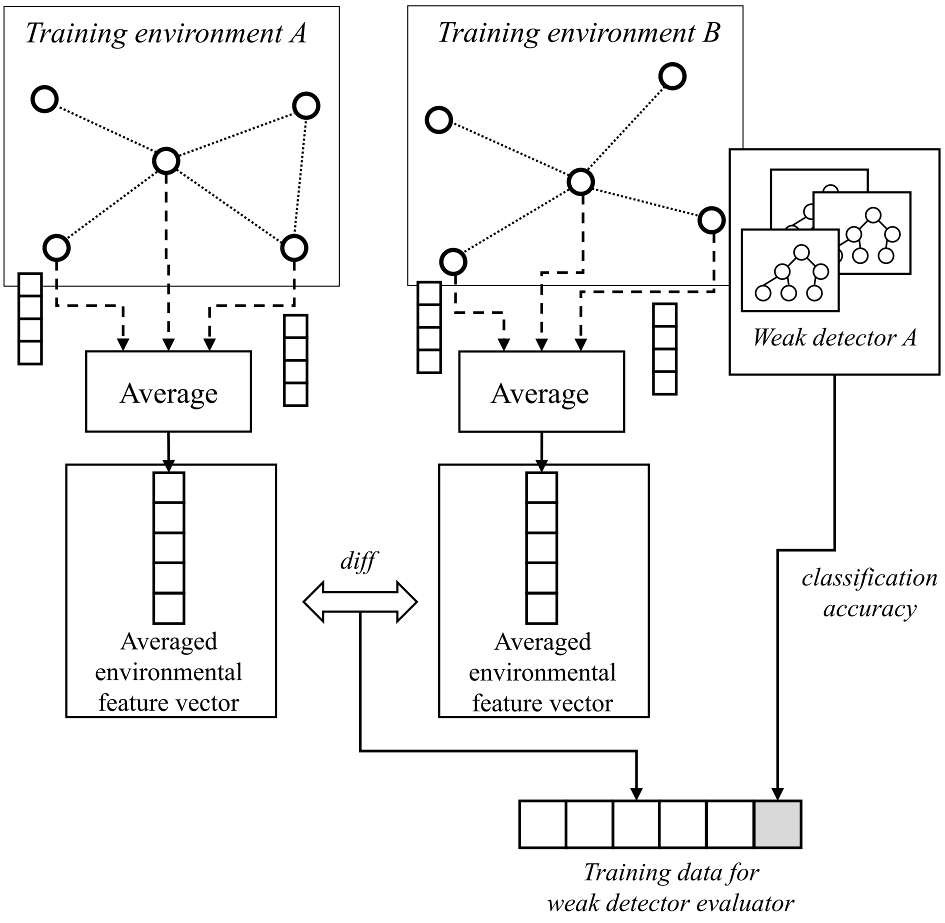

Training environment A
Training environment B
Average
Average
Weak detector A
Averaged environmental feature vector
Averaged environmental feature vector
diff
classification accuracy
Training data for weak detector evaluator

Figure 2.4: Preparing training data for weak detector evaluator.

Weak detector evaluator

We then train a weak detector evaluator using the weak detectors of the training environments and environmental features extracted from these training environments.

Here we consider the detection performance of a weak detector trained in environment A, when this detector is executed in training environment B (Figure 2.4). To estimate its performance, we utilize the respective environmental features of these two environments. We first construct an environmental feature vector concatenating environmental features calculated based on the observations of each normal node in the two training environments. Therefore, the number of constructed environmental feature vectors for each training environment is identical to the total number of normal nodes in the training environment. We then compute the averaged environmental feature vector for each environment by employing the environmental feature vectors of all the normal nodes in the environment. (Because the training environments are simulated environments, we can aggregate environmental features calculated by the nodes in the environments.) We compare the environmental feature vectors of these two environments to compute features that represent the difference between these environments. Finally, we employ the computed features as explanatory variables to estimate the projected performance of the environment-A weak detector in environment B. As we consider that the performance of a weak detector is affected by the difference between the training and testing environments, we learn the relationship between the detector's performance in the test environment and explanatory variables representing the environmental difference.

Let $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ be averaged environmental feature vectors obtained in training environments A and B, respectively. We first compute the difference between $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ as follows.

$$d(\boldsymbol{x}_A, \boldsymbol{x}_B) = \begin{pmatrix} |x_{A,1} - x_{B,1}| \\ |x_{A,2} - x_{B,2}| \\ \vdots \\ |x_{A,e} - x_{B,e}| \end{pmatrix},$$

where $x_{A,i}$ is the ith element of $\boldsymbol{x}_A$ and e is the number of environmental features, and then employ this computed difference in the form of explanatory variables. In the case of the independent variable in this regression task (*i.e.*, estimating the detector's performance), we employ the classification accuracy (average F-measure) of an environment-A weak detector in environment B. That is, the weak detector actually classifies behavioral feature vectors obtained in environment B, and then its classification accuracy is used as the independent variable. Using the above procedures, we can compute training data for estimating the classification accuracy of the environment-A weak detector in environment B from explanatory variables representing the difference between these environments. We compute the training data for each pair of training environments in the respective environments, and train the weak detector estimator on the computed training data. The performance of the environment-A weak detector is estimated as:

$$w_A = f\big(d(\boldsymbol{x}_A, \boldsymbol{x}_B)\big),$$

where $f(\cdot)$ is a regression model that calculates the performance w_A using the computed differences. In this study, we employ the SVM regressor [92] for the regression model.

2.4.4 Ensemble malicious node detector

Here we describe procedures for constructing an ensemble malicious node detector tailored to normal node $\mathtt{a}$ in a given test environment using environmental features computed based on observed send and transmission packet information for a certain period. We first estimate the performance of a weak detector of each training environment using the environmental features obtained by node $\mathtt{a}$; and then construct a strong ensemble detector tailored to node $\mathtt{a}$, by combining the top-k weak detectors with respect to estimated performance. Because the memory resource of mobile devices is limited, we select only top-k useful detectors. The ensemble detector aggregates the detection results of the weak detectors based on their estimated performance, as follows.

$$Pr_{agg}(y = mal|\boldsymbol{f}_b) = \frac{1}{W}\sum_{i=1}^{k} w_i Pr_i(y = mal|\boldsymbol{f}_b),$$

where $Pr_i(y = mal|\boldsymbol{f}_b)$ is the probability output by the ith weak detector for which node $\mathtt{b}$ is malicious, and $\boldsymbol{f}_b$ is a behavioral feature vector constructed based on the observed behavior of node $\mathtt{b}$. In addition, w_i is the projected performance of the ith weak detector, which is estimated by the weak detector evaluator, and $W = \sum_{i=1}^{k} w_i$. Based on the aggregated result $Pr_{agg}(y = mal|\boldsymbol{f}_b)$, the detection result for node $\mathtt{b}$ is determined as follows:

$$\hat{y} = \begin{cases} malicious & (Pr_{agg}(y = mal|\boldsymbol{f}_b) > 0.5) \\ normal & (otherwise) \end{cases}$$

2.4.5 Detecting malicious nodes

We assume that node $\mathtt{a}$ in a given test environment constructs an ensemble detector by fusing weak detectors of selected training environments in advance. To find efficient weak detectors, node a first computes an environmental feature vector based on information about send and receive packets for a specific time period, which are observed by node a. Assume that node a estimates the performance of a weak detector of training environment A. Node a computes the difference between the computed environmental feature vector and an averaged environmental feature vector computed by employing environmental feature vectors of all nodes in training environment A. Based on the computed difference values, we estimate the performance of the weak detector of training environment A using the weak detector evaluator. Node a constructs an ensemble detector combining the top-k weak detectors, and then detects neighboring malicious nodes using the ensemble detector. Assume that, in a given test environment, node a is judging whether or not neighboring node b is malicious. Node a observes the behavior of node b over a specified period, and computes a behavioral feature vector based on this observed behavior. The computed vector is then fed into the ensemble detector and, the detection result for node b is obtained.

2.5 Experiment

This section summarizes our experiments evaluating the performance of the proposed method.

2.5.1 Setting

We used Qualnet 7.4 network simulator[2]. Each node transmitted messages and data packets, whose payload sizes were 256 bytes, using an IEEE 802.11b device. The communication range of each node was adjusted to roughly 100 meters, and the network bandwidth was 11Mbps. As with existing works [1, 26, 49, 91], we used the random way point model [9], with a maximum movement speed of v_{max} and pause time of 0. (The velocity of each node was randomly chosen from $(0, v_{max}]$.) When there were n nodes in a network, there were $n \cdot m$ ($m \in [0.1, 0.4]$) malicious nodes in the network. We randomly chose a pair of source node and destination node every f seconds. If the source node has an active route to the destination node, the source node sends a data

[2]http://web.scalable-networks.com/qualnet-network-simulator-software

Table 2.5: Parameter configuration

Parameter	Values
n	50, 60, 70, 80, 90, 100
m	0.1, 0.2, 0.3, 0.4
v_{max} [m/sec]	1.0, 2.0, 3.0, 4.0
Network size [m^2]	500 $\times$ 500, 600 $\times$ 600, 700 $\times$ 700, 800 $\times$ 800, 900 $\times$ 900, 1000 $\times$ 1000
f [sec]	1.0, 2.0, 3.0, 4.0
t [sec]	50, 100, 200, 300, 400, 600, 800, 1,000

packet to the destination node directly. Otherwise the source node broadcasts an RReq to find a route to the destination node. The network parameters are described in Table 2.5, and the simulation time was t [sec] (the default time was 300 [sec]).

Note that attack models were categorized into two patterns: passive and proactive. Passive attacks included black hole attacks, gray hole attacks, sybil attacks, routing packet dropping attacks, rushing attacks, wormhole attacks, and jelly fish attacks; and proactive attacks included RReq flooding attacks and hello flooding attacks. When malicious nodes received messages, they chose an attack from among the passive attacks uniformly at random, and executed it. Malicious nodes executed RReq or hello flooding attacks for 30 seconds, and then halted the attacks for 30 seconds after execution. (Thus, after the halt, they recommenced one of the two attacks.) During the attacks, they broadcasted an RReq or hello packet every 0.5 seconds.

Evaluation methods. We simulated 2,304 ($6 \times 4 \times 4 \times 6 \times 4$) environments with different parameters (see Table 2.5), randomly selecting 90% of these environments as training environments, and using the rest as test environments. To investigate the effectiveness of the proposed method, we prepared the following methods.

- *MITR13* [70]: This is a state-of-the-art technique utilizing MultiLayer Perceptron for classification proposed by Mitrokotsa *et. al.* The method employs the following features: TReqRec, TReqSen, TRepRec, TRepSen, TRerRec, TRerSen, TDatRec, TDatSen[3], number of neighboring nodes (NeiNum), ratio of routing table update with respect to entries (PCR), and ratio of routing table update with respect to hop counts (PCH). As in [70], we tuned the MITR13 parameters based on cross-validation [53].

- *DC*: This method employs a single malicious node detector based on a discriminative classifier (*e.g.*, SVM or Naive Bayes), which is trained on labeled training data obtained in all the training environments. The method employs our designed behavioral features.

- *Proposed*: This is the proposed method. The parameter of k is set as 10.

- *Proposed w/o EV*: This method constructs a malicious node detector based on ensemble learning by fusing randomly-selected k weak detectors of training environments. Therefore, this method does not estimate the projected performances of weak detectors using environmental features (*i.e.*, the weights of weak detectors are identical). We prepare this method to investigate the effectiveness of the weak detector evaluator.

All data obtained during the simulation time were used to compute behavioral and environmental features. (As MITR13, DC, and Proposed w/o EV do not consider environmental features, these were not obtained here.) We assumed that nodes executed a classification method at the end of the simulation (only normal nodes executed the method), in which each node classified all the nodes whose behavioral features had been observed by the node.

Criteria. As mentioned earlier, our method incurs no additional communication costs. Also, the computational costs of constructing an ensemble detector and testing a neighboring node (*i.e.*, malicious node judgement) are short[4]. We therefore focus on the following criteria to measure the performance of the above methods.

[3]MITR13 does not convert these listed features to ratios, which differ from our method.

[4]In our experiments, which were conducted on a mini PC with 2.3 GHz Intel Core i3 processor, the average computation costs of constructing an ensemble detector and testing are respectively 2.47 [sec] and 46.66 [msec] (the language is C#).

- Accuracy: This is represented by $\frac{T_{nor\rightarrow nor,mal\rightarrow mal}}{T}$, where $T_{nor\rightarrow nor,mal\rightarrow mal}$ and T are respectively the set of correctly classified **instances** and the set of all **instances**.

- Detection rate: This is represented by $\frac{T_{mal\rightarrow mal}}{T_{mal}}$, where $T_{mal\rightarrow mal}$ and T_{mal} are respectively the set of correctly classified **instances** describing malicious nodes and the set of all **instances** describing malicious nodes.

- Mis-Detection rate: This is represented by $\frac{T_{nor\rightarrow mal}}{T_{nor}}$, where $T_{nor\rightarrow mal}$ and T_{nor} are respectively the set of wrongly classified **instances** describing normal nodes and the set of all **instances** describing normal nodes.

2.5.2 Result

Selection of classifier in DC. Figures 2.5–2.7 show the performances of the varieties of DC. We tested random forest, Naive Bayes, decision tree, and SVM as classifiers[5], and the random forest, which is known to be state-of-the-art and robust, achieves the best performance. Hereafter, DC denotes the random forest and is compared with our method.

Comparison with MITR13. Figures 2.8–2.10 show the performances of the methods, and the performance of MITR13, which is a state-of-the-art malicious detection method, is much worse than those of the other methods. In particular, MITR13 is about 40% worse than that of Proposed with regard to the three criteria. This is because features used in MITR13 are not designed to be applied to a situation where training and test environments are different. By using all the training data, we computed information gain[6] of behavioral features and the features used in MITR13. They are respectively described in Tables 2.6 and 2.7. As shown in these tables, the info. gain of features used in MITR13 is quite low, showing its poor detection performance in the diverse environments. On the other hand, the info. gain of behavioral features is high. We bold

[5]We employed Weka data mining toolkit [109] to run these classifiers. We used default parameters of these classifiers.

[6]The information gain is used to find distinguishing features of instances. The information gain of a feature increases the better the feature classifies the instances.

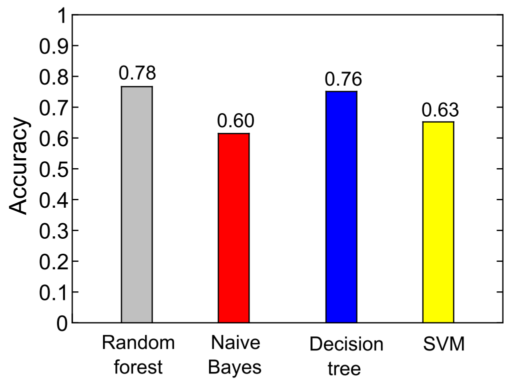


Figure 2.5: Accuracies of SVM, Naive Bayes, decision tree, and random forest

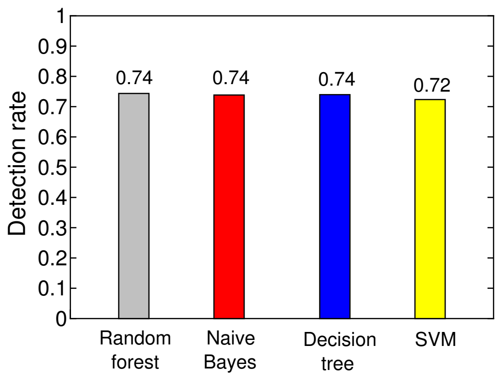


Figure 2.6: Detection rates of SVM, Naive Bayes, decision tree, and random forest

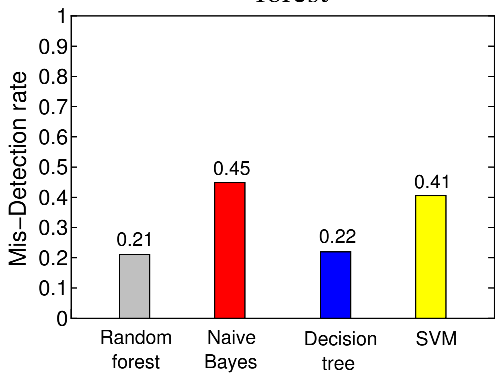


Figure 2.7: Mis-Detection rates of SVM, Naive Bayes, decision tree, and random forest

top-5 info. gain in Table 2.6, and the set of the corresponding features is useful to detect all attack classes illustrated in Table 2.1 (see Section 2.4.2). This result shows that our designed features are effective and can be employed for detecting famous attacks in WSNs.

Comparison with DC. Figures 2.8–2.10 show the performances of our method (Proposed), EL, DC, and MITR13. We can see from Figures 2.8–2.10 that the performance of DC (random forest) is about 10% worse than that of Proposed with regard to the

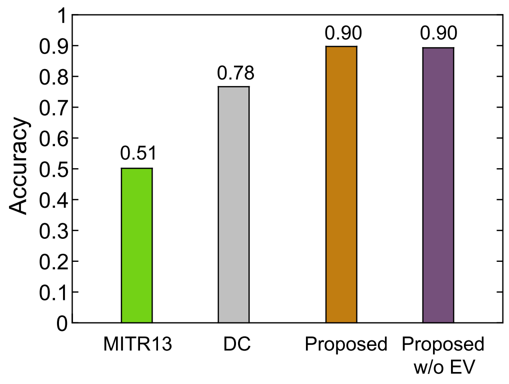


Figure 2.8: Accuracies of four methods

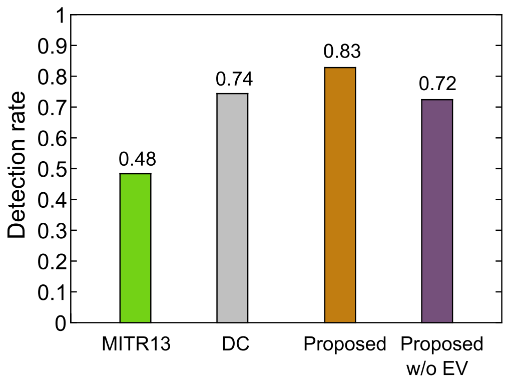


Figure 2.9: Detection rates of four methods

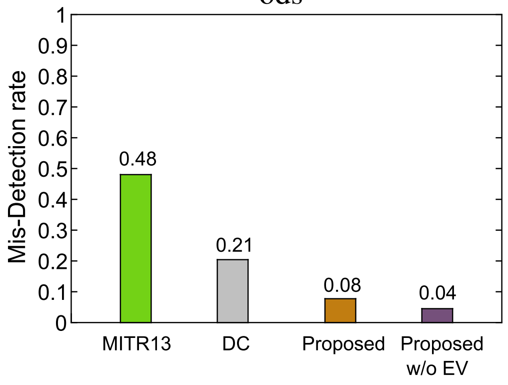


Figure 2.10: Mis-Detection rates of four methods

three criteria. Because DC builds a classifier using training data from all the training environments, the trained model can fit a majority of training environments. In contrast, Proposed prepares a weak detector that is tailored to each training environment. Figure 2.11 depicts the histogram, whose bin size is 0.02, of the accuracies (average F-measure) for all the weak detectors when each of the weak classifiers was applied to each of the test environments. As shown in the figure, we can find that many weak classifiers do not work well in the test environments. Proposed combines weak detectors that are estimated to work well in test environments, and significantly outperforms DC. The average F-measure of weak detectors selected by Proposed was 0.89, and we can say that Proposed successfully selects weak detectors that work well in the test

Table 2.6: Info. gain of behavioral features

Behavioral features	Info. gain
RepSenRatio	0.021
RepRecRatio	0.021
RepIgnRatio	**0.141**
ReqRecRatio	**0.092**
DatSenRatio	0.003
DatRecRatio	0.023
DatIgnRatio	0.024
RerRecRatio	0.004
HelRecRatio	0.028
AllPckRatio	0.015
RepUslRatio	0.023
RepReqRatio	0.018
RerSenRatio	0.021
HelCheckRatio	**0.190**
ReqIgnRatio	**0.179**
RepUsfRatio	0.014
ReqUsfRatio	**0.103**
HelUsfRatio	0.015

Table 2.7: Info. gain of features used in MITR13

Behavioral features	Info. gain
TReqRec	6.4×10^{-5}
TReqSen	5.4×10^{-5}
TRepRec	3.7×10^{-5}
TRepSen	3.1×10^{-5}
TRerRec	3.3×10^{-5}
TRerSen	3.2×10^{-5}
TDatRec	5.4×10^{-5}
TDatSen	5.4×10^{-5}
NeiNum	3.0×10^{-4}
PCR	3.3×10^{-4}
PCH	0

environments, resulting in high accuracy.

Weak detector evaluator. Figures 2.8–2.10 show the performance of Proposed w/o EV, which does not employ the weak detector evaluator. The detection rate of Proposed w/o EV is somewhat worse than that of DC and is also about 10% worse than that of Proposed because randomly selected conservative weak detectors are used in some test environments whose network parameters are very different from those of the training environments, resulting in slight improvement in the mis-detection rate. Here we

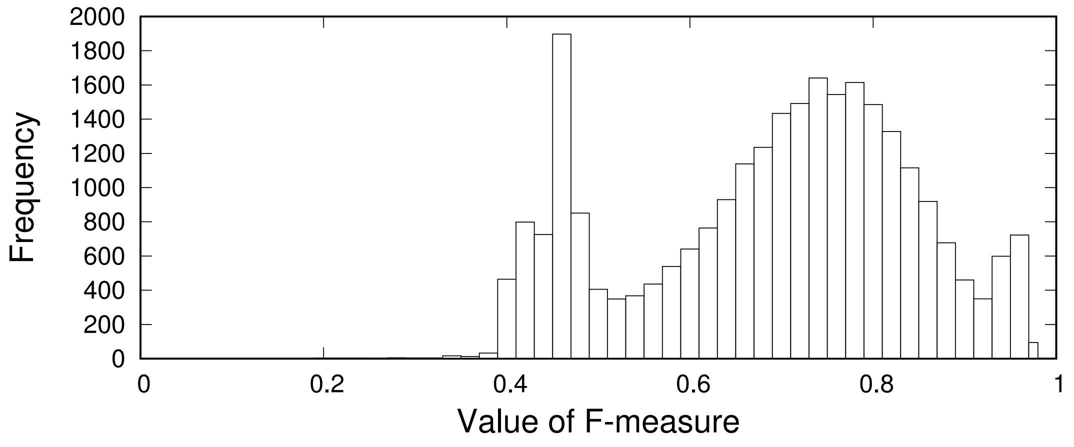


Figure 2.11: Histogram of average F-measure for all the weak detectors used for building the weak detector evaluator

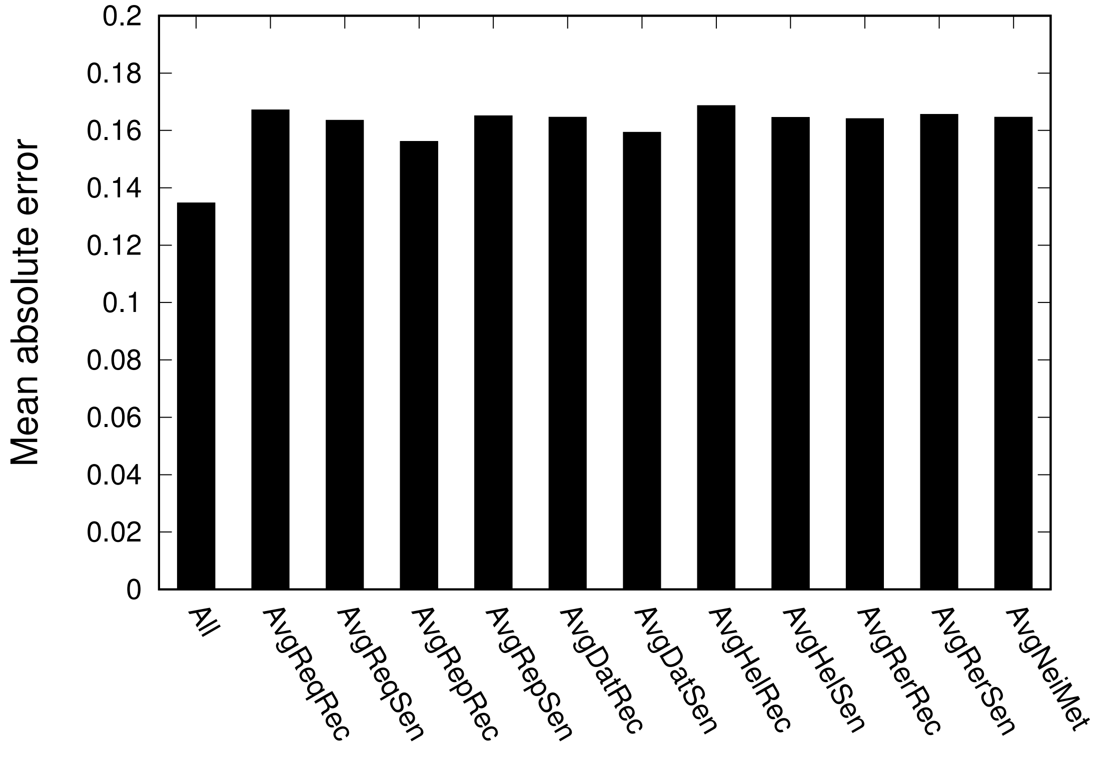


Figure 2.12: Mean absolute error of estimation

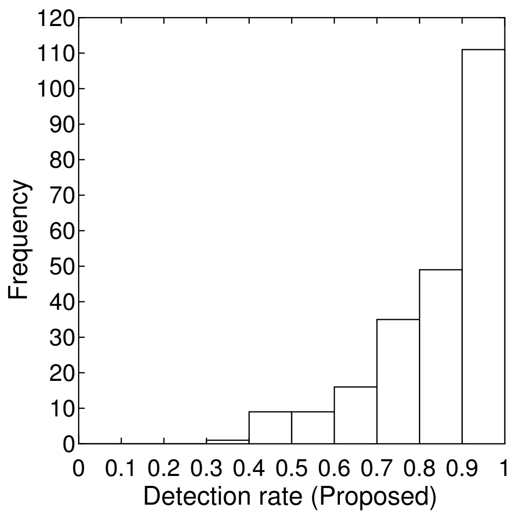


Figure 2.13: Distribution of detection rates of Proposed in each environment

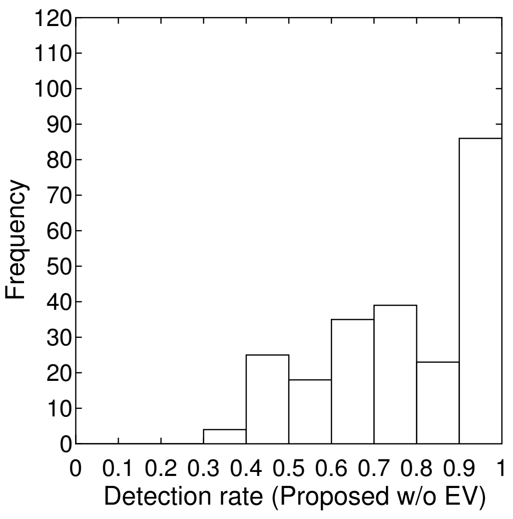


Figure 2.14: Distribution of detection rates of Proposed w/o EV

evaluate the performance of the weak detector evaluator, which estimates projected F-measures of weak detectors in test environments. Figure 2.12 shows the mean absolute error (MAE) of the estimates when all the eleven environmental features for estimation were used. We can see from the result of *All* that the weak detector evaluator can estimate the F-measures with a very small error. Figure 2.12 also shows the mean absolute error when a single environmental feature is used for estimation. The MAE of the case where a single each environmental feature is used is higher than that of the case where all environmental features are used but the difference is slight. Since each environmental feature is correlated, each MAE is similar to each other. Therefore, even a single environmental feature provides high estimation accuracy (although using all features enhance the performance).

Figures 2.13 and 2.14 show the distributions of the detection rates derived from the detection rate in each test environment (Proposed and Proposed w/o EV, respectively). As can be seen in the results, the number of test environments with low detection rates (*e.g.*, 0.4, 0.5, and 0.6) in Figure 2.14 is much larger than that in Figure 2.13. These results indicate that randomly selected weak detectors did not work well in test environments whose network parameters are very different from those of a majority of

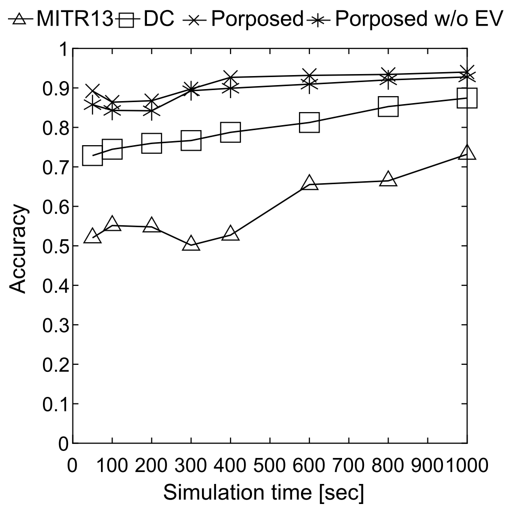


Figure 2.15: Accuracy vs. simulation time

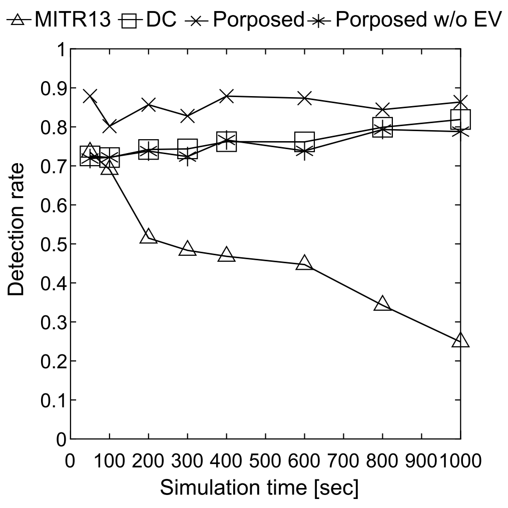


Figure 2.16: Detection rate vs. simulation time

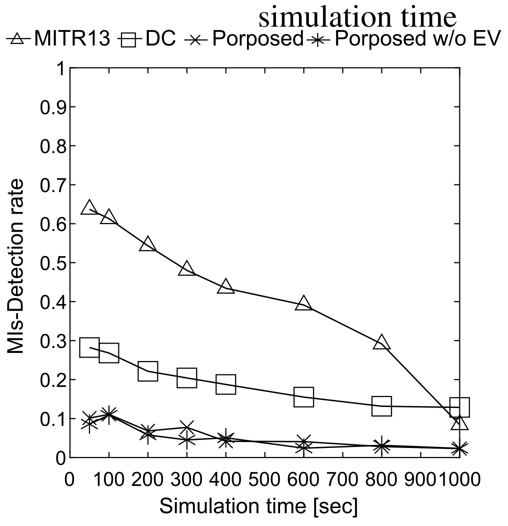


Figure 2.17: Mis-Detection rate vs. simulation time

training environments. (In our experiment, test environments with large f sometimes had poor detection rates.) In contrast, Proposed, which selects weak detectors to be used by employing the weak detector evaluator, succeeded in reducing the number of test environments with low detection rates. As above, Proposed is effective for test environments whose high-impact network parameters are different from those of a majority of training environments.

Impact of simulation time. Finally, we varied t to investigate the impact of simulation time, and the performance of each method is shown in Figures 2.15–2.17. We can see that Proposed keeps outperforming the other methods and has the highest accuracy (as well as the highest detection rate and lowest mis-detection rate). This result also implies that our designed features are effective for malicious node detection and can grasp the difference between malicious and normal nodes, even when t is small (*e.g.*, 50 [sec]). We observe that MITR13 cannot detect malicious nodes well when observing time is long. Recall that MITR13 mainly uses numbers of messages as features. As observing time becomes longer, the difference between malicious and normal nodes becomes smaller, resulting in poor performance.

2.6 Conclusion

In this chapter, we presented a robust malicious node detection method for WSNs. We constructed a robust malicious detector by efficiently fusing weak malicious detectors trained in diverse environments. The experiment revealed that the proposed method significantly outperformed state-of-the-art malicious detection methods.

As a part of our future work, we plan to investigate the performance of a malicious node detector trained in various simulated environments when the detector is run in real WSN environments.

Chapter 3

Detecting Reinforcement Learning-based Malicious Nodes in Wireless Sensor Networks

3.1 Introduction

Wireless sensor networks (WSNs) are facing threats from malicious nodes that disturb packet transmissions, leading to poor WSN performance. Existing studies have proposed a variety of routing attack models [18]. All of which mainly aim at disturbing packet transmissions and causing redundant traffic. However, the existing attacks simply follows pre-defined routines, which means that the malicious nodes cannot modify their attack patterns, and thus are easily detected through related countermeasures [1, 90]. Besides, the assumption that malicious nodes have no learning ability is extremely unrealistic, given the rapid development of machine learning. Therefore, in this chapter, we assume that malicious nodes have learning ability, and can learn from the countermeasures to avoid being detected. In particular, we focus on designing a grey hole attack with learning ability, because the grey hole attack is a representative attack model, which is able to destroy the routing procedure and data transmissions of a sensor network (we do not consider multiple attack models because it is difficult to enable malicious nodes to learn with different attack models). A malicious node with learning ability which executes the grey hole attack can adjust its behaviors based on learning. The above-mentioned attack countermeasures cannot deal well with learning-based attack models, because a static rule is ineffective in responding to an attack model in which nodes can change their behaviors.

Contribution. To design a grey hole attack model with learning ability, we utilize reinforcement learning to enable malicious nodes to learn by themselves in a WSN. In addition, we design a method that robustly detects the above attack model. We extract inherent features for the detection method to detect malicious nodes, and the detection method is updated simultaneously within the episodes. The following are the principal contributions of the study in this chapter:

- We design a reinforcement learning-based grey hole attack model for WSNs. To our knowledge, this is the first study that applies reinforcement learning algorithms to build a smart grey hole attack model for WSNs.
- We propose a method for discrete-time adaptive updating of the countermeasure, to detect the novel attack model.
- We conduct extensive experiments to investigate the performance of the proposed attack model and corresponding countermeasure in terms of accuracy, detection rate, mis-detection rate, and detection time. The results reveal that malicious nodes in the attack model are hardly detected by existing countermeasures, and that the proposed countermeasure outperforms the state-of-the-art alternatives.

Organization. Here we describe the organization of this chapter. In Section 3.2, we review related works. Section 3.3 introduces the assumptions of the study in this chapter. The proposed method is described in Section 3.4. The experimental results are summarized in Section 3.5. Section 3.6 presents our conclusion.

3.2 Related work

Reinforcement learning. Reinforcement learning (RL) has become an increasingly popular focus of research, due to its effectiveness in various tasks. As a model-free reinforcement learning technique, the Q-learning algorithm can derive the optimal strategy, if all the feasible actions are repeatedly sampled over all the states in a Markovian decision process [84]. Mnih *et al.* [71] first successfully approximated the Q function with a deep convolutional neural network, and enabled an agent to beat a human expert in several Atari games. Moreover, in [54], Kulkarni *et al.* firstly proposed the idea of hierarchical deep reinforcement learning to separate a task into two levels. The upper level makes decision, and the lower level executes actions. Vezhnevets *et al.* [105] further developed the hierarchical method, and their method was capable of doing more complicate tasks. Note that, the hierarchical deep reinforcement learning method is not suitable for our problem because there is no obvious hierarchical structure in our system.

Later on, RL algorithms have also been utilized in many WSN-related applications, such as the scheduling of energy harvesting nodes [7] and detecting the rough edge of a VANET [66]. In the domain of WSN security, Li *et al.* [61] used game theory to protect packet transmissions against smart attacks. Xiao *et al.* [111] utilized RL against smart jamming. Aiming to build a secure routing method, Liu *et al.* [65] utilized Q-learning to enable a node to determine the credibility of a specific neighboring node. These studies, however, all assume that malicious nodes follow existing patterns, and the proposed methods are only effective for pre-defined attack models. Malicious nodes with smart attack can learn from the a specific detection method. Therefore, the above-mentioned methods are not capable of detecting malicious nodes with smart attack.

Grey hole attack model and countermeasures in WSNs. Security in WSNs is a challenging issue, and many existing studies have focused on it, proposing a variety of attack models [50, 97]. The grey hole attack is particularly notable among the proposed attack models. In this model, malicious nodes can selectively (randomly) forward packets or reply requests, to drag routes to themselves. The ordinary grey hole attack is a smart version of the famous black hole attack [88], and is hard for ordinary reputation-based detection methods to counter [87]. Some studies have developed techniques for avoiding the attack, using multipath approaches [102], analyzing the impact of the attack [95], and preventing the attack [86]. Many studies utilized machine learning method to detect malicious nodes [78, 116]. They designed features and building classifiers to detect malicious nodes. However, the existing grey hole attack model and corresponding countermeasures do not assume that the malicious nodes are equipped with learning ability.

3.3 Assumption

In this section, we describe the assumption of our study. Our assumed network model is the same as the network model in Chapter 2 (see Section 2.3.2).

3.3.1 Mobile nodes

In this study, we assume that the WSN contains two categories of nodes, normal and malicious, which perform the grey hole attack; and that each node category has its own servers (normal and malicious, respectively) with machine learning functions. In addition, we assume that malicious nodes work together to destroy the regular routing procedure and disturb the proper data transmissions in the WSN. Furthermore, we assume that the malicious nodes can use reinforcement learning to learn not to be detected while executing malicious behaviors. The normal nodes, meanwhile, update their detection method to track the latest behavioral pattern of the malicious nodes. The normal nodes also work together to detect malicious nodes, using a specific data collecting method (*e.g.*, flooding through the network).

3.3.2 Deep Q-learning for the malicious server

We assume that malicious nodes employ deep Q-learning[71] as the reinforcement learning algorithm. Below we present a brief summary of deep Q-learning.

Reinforcement learning deals with learning an optimal policy π^* for an agent interacting in an unknown environment. At each time step t, an agent observes the current state s_t of the environment, decides on an action a_t according to a policy π, and observes a reward signal r_t. The goal of the agent is to find a policy that maximizes the expected sum of discounted rewards R_t

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'},$$

where T is the time at which the episode terminates, and $\gamma \in$ [0,1] is a discount factor that determines the importance of future rewards. The Q-function of a given policy π is defined as the expected return from executing an action a in a state s:

$$Q^{\pi}(s, a) = \mathbb{E}[R_t | s_t = s, a_t = a]$$

Finally, the updating rule for the Q-values for an action selection policy π is as follows:

$$Q^{\pi}(s_t, a_t) = (1 - \alpha)Q^{\pi}(s_t, a_t) + \alpha[R_t(s_t, a_t) + \gamma \underset{a}{\max} Q^{\pi}(s_{t+1}, a)]$$

where

$$0<\alpha<1$$

is the learning rate.

In the deep Q-learning method, a function approximation is used to estimate the action-value function Q. In particular, deep Q-learning uses a neural network parametrized by θ for approximation:

$$Q(s, a; \theta) \approx Q(s, a)$$

To find the optimal θ, we apply gradient descent by following loss function [71]:

$$L_t(\theta_t) = \mathbb{E}_{s,a,r,s_{t+1}} \left[(y_t - Q_{\theta_t}(s, a))^2 \right],$$

where t is the current time step and $y_t = r + \gamma \max_{a_{t+1}} Q_{\theta_t}(s_{t+1}, a_{t+1})$.

It has been shown that, with sufficiently large number of learning episodes, the deep Q-learning algorithm converges and returns the optimal policy π^* [33].

Instead of performing the Q-learning updates in an online learning fashion, a common practice is to use experience replay to break the correlation between successive samples [4]. At each time step, an agent experience (s_t, a_t, r_t, s_{t+1}) is stored in a replay memory, and the Q-learning updates are performed on batches of experiences randomly sampled from the memory. In the training of the deep Q-learning agent, we assume that malicious nodes share the same malicious server, so note that the s_t and s_{t+1} in one experience tuple (s_t, a_t, r_t, s_{t+1}) are the state and next state of a particular malicious node.

At every training step, the next action is generated using an ϵ-greedy strategy: with a probability ϵ, the next action is selected randomly, and with a probability $1 - \epsilon$, the next action is the best one obtained by the deep Q-learning algorithm. In the proposed method, we start with ϵ = 1 and progressively decay ϵ.

3.4 Proposed method

3.4.1 Overview

Here, we provide an overview of the proposed method. The method, which is based on machine learning techniques, consists of following two phases.

- A training phase based on simulations (by a network simulator). Therefore, all the nodes can connect to their respective server without additional packet transmissions.

- A test phase that utilizes the malicious and normal servers built in the training phase.

Figure 3.1 illustrates the training phase of the method. In this phase, we simulate various environments with different environmental parameters (*e.g.*, initial node positions), and train the deep Q-learning agent with reinforcement learning algorithms in an on-line manner, using data obtained from the malicious nodes in the simulated environments. At the same time, the normal server is also trained using the data collected from the normal nodes. It is important to recall that, in the training phase, all the malicious and normal nodes respectively share their own server. In the training phase, only one normal server and one malicious server are trained, because all the data can be collected to build a strong detection algorithm and a strong attack model. Therefore, only one detection algorithm and one model for malicious nodes are formed.

The reason why the servers are trained on different network environments is because training on a single network environment is not generalized [30] (*i.e.*, in a single network environment, the normal server may overfit the behavioral pattern of the malicious nodes).

In the test phase, each normal/malicious node is equipped with a fixed normal/malicious server obtained in the training phase. In the test phase, the categories of neighboring nodes are unknown, which is different from the training phase. Normal nodes observe behaviors of neighboring nodes thus to collect data. Then, a normal node use the trained normal server to detect malicious nodes, whose action is decided by their malicious server. In this procedure, a normal node does not transmit any observed data to any

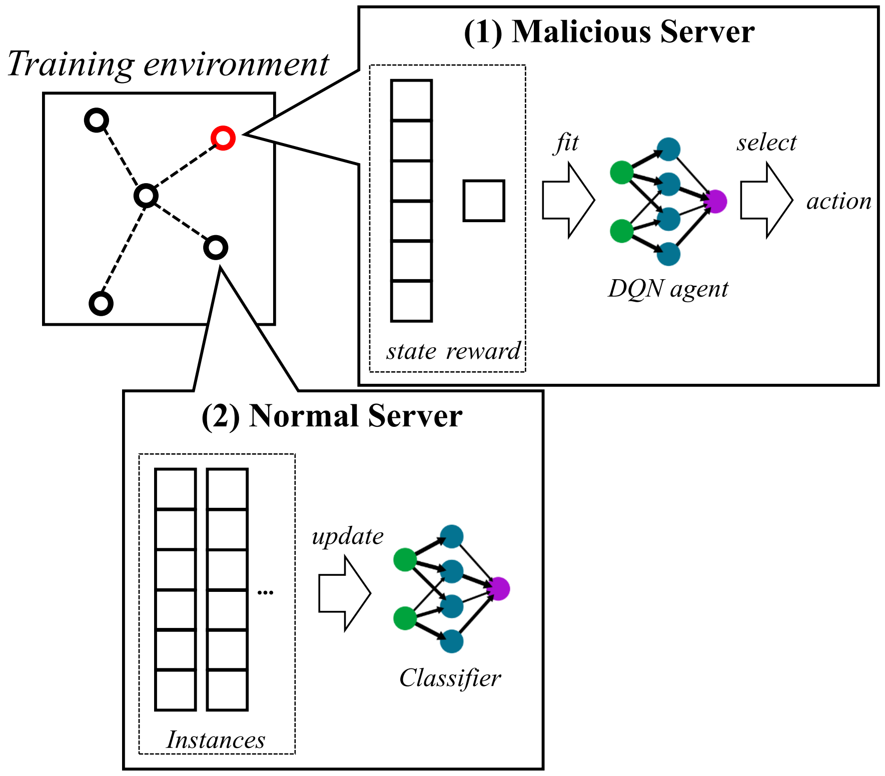


Figure 3.1: Overview of the training phase of the proposed method. (1) Malicious server monitors the WSN, utilizes our method to obtain state and reward, and then updates the deep Q-learning agent and selects an action. (2) Normal server collects instances from normal nodes and then updates the classifier.

neighboring node. Therefore, a malicious node cannot connect to a normal node to steal observed data. Note that the servers are never updated in the test phase, because, (i) for the normal server, the node category of neighboring nodes is unknown, and (ii) for the malicious server, the reward is unknown. Therefore, in the test phase, both malicious and normal nodes do not need to collect data from other nodes.

In addition, we assume that both malicious and normal nodes observe the behavior of their neighboring nodes. Before we describe the data collected for the normal and malicious servers, we illustrate the information related to the neighboring node observed by each node in Table 2.2 (see Section 2.4.2 in Chapter 2).

3.4.2 Smart grey hole attack model

Here we describe the proposed smart grey hole attack, a novel attack model based on reinforcement learning. Malicious nodes in this attack model aim at disturbing the routing and data transmission of a WSN, without being detected.

Ordinary grey hole attack model

Before we illustrate our smart grey hole attack model, we first detail the ordinary grey hole attack model. This model mainly consists of two components: i) When a grey hole malicious node receives an RReq, it may perform normally (follow the routing protocol of the WSN), or send back an RRep with one hop count, which means that the malicious node states that it is near the destination. ii) When the malicious node receives a data packet, it may also perform normally, or just drop it. The probability that malicious nodes perform normally or not is decided by a packet dropping ratio (*e.g.*, 50%).

However, this static pre-defined packet dropping ratio makes the grey hole malicious nodes easy to be detected by some machine learning-based detection methods [70], because these methods can learn from the behaviors of malicious nodes and predict their packet dropping ratio. Therefore, in this study, we employ reinforcement learning to enable malicious nodes to learn an appropriate behavior.

Overview of smart grey hole attack model

In the training phase of this model, we run a large number of episodes of simulations. In each of which, we employ different network parameters. During each training episode, the malicious server monitors the WSN and employs the proposed method to obtain a reward and a state after any malicious node receives a packet. The malicious server then updates itself, and selects an action for this malicious node. Each episode ends when all the malicious nodes in the WSN have been detected, or the simulation time exceeds a threshold value.

In the test phase, each malicious node is equipped with a trained malicious server obtained in the training phase. The malicious server of each node is never further updated; however, it still utilizes the proposed method to obtain the states and select actions for its malicious node.

State and reward engineering

As described in Section 3.3.2, reinforcement learning utilizes a series of tuples (s_t, a_t, r_t, s_{t+1}) for updating the policy of an agent. However, in contrast to existing simple game-like learning tasks (*e.g.*, the Cart-pole game from OpenAI Gym[1]), which have obvious state and reward declarations, and Atari game missions, which can utilize convolutional neural networks to conveniently extract features from game picture frames, the malicious server in the proposed system does not have an existing method to obtain states and rewards from monitoring data. Thus, we design a method to obtain these pieces of information.

State engineering. When a malicious node observes a $state$, it selects an $action$ according to the decision of the malicious server. To enable the malicious nodes to recognize their situation, we design the states (as vectors to input into neural networks) by extracting the short-term and long-term information. In particular, we extract the packet category ID and neighboring node ID as the short-term information. The packet category ID shows the category of this received packet (*e.g.*, RReq is 1 and RRep is 2), and node ID represents the ID of the node who sends this packet. Note that the packet category and the node IDs are not numeric data, but categorical data. For instance, compared with node ID 3, node ID 2 and node ID 8 have just the same differences. However, if we simply utilize these node IDs (2, 3, 8) as input of the neural networks, the neural networks of the malicious server will treat 2 as more similar to 3 than 8 [19]. As a result, we employ one-hot encoding [20] to encode the node ID and packet category ID, in order to appropriately process the categorical data.

Figure 3.2 illustrates our one-hot encoding for node ID. In a WSN of n nodes, the node ID k of a received packet is transformed into a vector of length n, whose the $k\text{-}th$ $dimension$ is 1, and the rest dimension of the vector is 0. The one-hot encoding of the packet category ID is the same as that of the node ID. In this way, a malicious node can learn which neighboring node has sent the packet, and learns the behaviors of its neighboring nodes based on the short-term information.

However, the malicious nodes cannot clearly recognize their situations based solely on these two vectors, because the received packet category ID and neighboring node ID

[1]https://gym.openai.com/

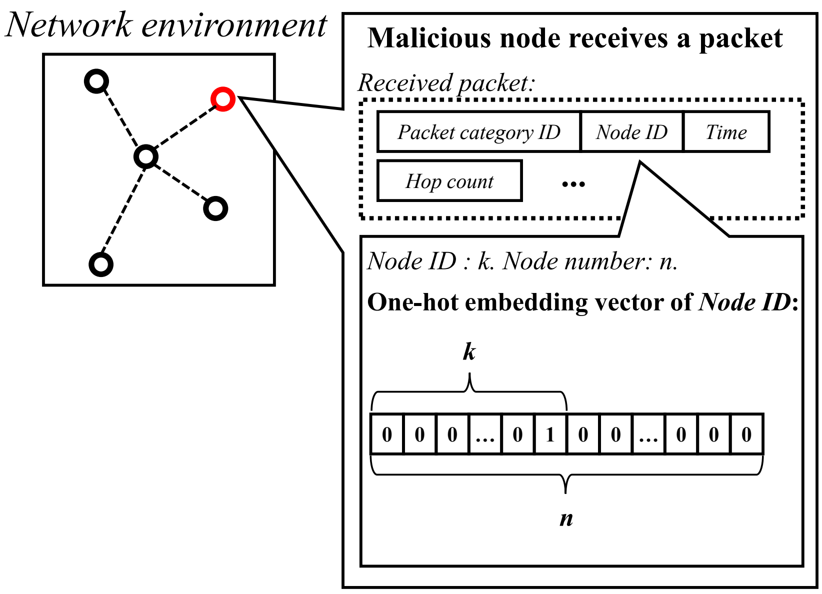


Figure 3.2: One-hot encoding of node ID

may always be similar during the training simulation, and can only represent the short-term situation. Therefore, we also extract ratios that can clearly describe the malicious nodes' long-term situations. Table 3.1 presents a summary of the long-term information of malicious nodes. In this table, we calculate the connection between the sent and received numbers of each category of packet, and the total sent and received numbers. These ratios can not only represent the node density (ActNeiRatio) near a malicious node, but illustrate the packets sent and received frequency of a particular category of packet. For example, if a malicious node sends too many RReps, its RepTotSenRatio is extremely high. In such a situation, it will soon be detected by some machine learning-based detecton method. Consequently, this malicious node will try to send fewer RReps not to be detected. A further advantage of the states is that the values of these processed ratios are all between [0, 1], which means our state design does not need normalization, and can rapidly converge.

Finally, we concatenate the one-hot vector of packet category ID, one-hot vector of neighboring node ID, and extracted ratios together as our state vector. Therefore, our state vector can represent both the long-term and short-term trends of the WSN for the

Table 3.1: Long-term information of malicious nodes

Long-term information	Definition
ActNeiRatio	1/ActNei
ReqTotSenRatio	TReqSen / TPckSen
RepTotSenRatio	TRepSen / TPckSen
RerTotSenRatio	TRerSen / TPckSen
HelTotSenRatio	THelSen / TPckSen
DatTotSenRatio	TDatSen / TPckSen
ReqTotRecRatio	TReqRec / TPckRec
RepTotRecRatio	TRepRec / TPckRec
RerTotRecRatio	TRerRec / TPckRec
HelTotRecRatio	THelRec / TPckRec
DatTotRecRatio	TDatRec / TPckRec
ReqRecTotSenRatio	TReqRec / TPckSen
RepRecTotSenRatio	TRepRec / TPckSen
RerRecTotSenRatio	TRerRec / TPckSen
HelRecTotSenRatio	THelRec / TPckSen
DatRecTotSenRatio	TDatRec / TPckSen
ReqSenTotRecRatio	TReqSen / TPckRec
RepSenTotRecRatio	TRepSen / TPckRec
RerSenTotRecRatio	TRerSen / TPckRec
HelSenTotRecRatio	THelSen / TPckRec
DatSenTotRecRatio	TDatSen / TPckRec

malicious nodes.

Reward shaping. The reward design in reinforcement learning is important because it describes how the agent behaves. Therefore, we explicitly design the reward for the malicious server.

Malicious nodes in our smart grey hole attack model aim to: i) drop data packets and ii) remain undetected. The result of dropping data packets can be represented by how much the transmission rate is reduced, and the result of attempting to remain undetected can be represented as the length of time that the malicious nodes remain undetected. As a result, the following rewards for shaping the reward function of the malicious server are considered.

- Positive reward for lowering the transmission rate.
- Positive reward for not being detected before a pre-defined time threshold.
- Negative reward for being detected.

Next, we need to consider the fact that setting the reward magnitudes and frequencies is difficult, and they often depend on the application. For instance, in the Atari's Pong game, the rewards are bounded by -1 and $+1$, while in Atari's Mr. Pac-Man eating a single ghost can yield a reward of up to +1600. To overcome this and for learning rapidly, we here employ bounded reward [72]. If the malicious nodes are all detected before the pre-defined threshold, the reward is -1. Otherwise, the reward is $+1$. With regard to lowering the transmission rate, the above reward design cannot show a decrease in the transmission rate, so we design the reward as $1 - transmission\ rate$, to reduce imbalance. We then add the positive and negative reward as a final reward.

Note that in our setting, we utilize the sparse reward function. That is, the malicious server receives no non-zero reward until an episode is done. As aforementioned, an episode is done when i) all the malicious nodes have been detected, or ii) the simulation time passes the threshold value. We finally summarize our reward:

$$reward = \begin{cases} 0 & (not\ end) \\ 1 + 1 - transmissionrate & (end\ \& \\ \ not\ detected) & \\ -1 + 1 - transmissionrate & (end\ \& \\ \ detected). & \end{cases}$$

3.4.3 Countermeasure

This section describes our countermeasure for the smart grey hole attack model.

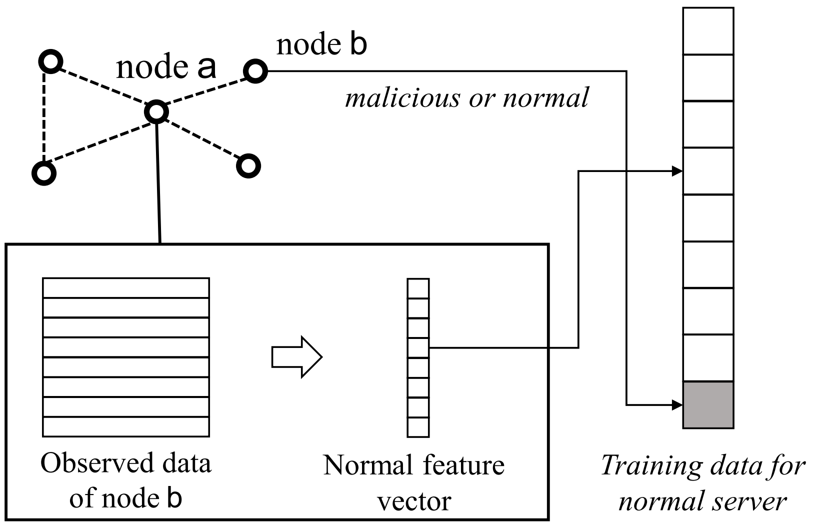


Figure 3.3: Preparing training data for the normal server

Overview of countermeasure

Now we aim to build a robust classifier to detect smart grey hole malicious nodes, and a neural network classifier in the normal server is built to classify the neighboring nodes of each normal node. The classifier is actually trained in the training phase at the same time as the malicious server is updated, meaning that through the large number of episodes in this phase, the classifier is also trained, and a trained classifier is obtained at the end of the training phase.

In the test phase, we install this trained normal server on each normal node, and normal nodes utilize majority votes to decide the category of their neighboring nodes.

Constructing classifier in normal server

For the normal classifier's features, we extract inherent features [30] to represent the malicious behaviors (Table 2.3, see Section 2.3 in Chapter 2). The normal server obtains these features from observed data, and these features enable the normal classifier to identify malicious nodes in a WSN. For instance, assume that node a observes its neighboring node `b`, the DatRecRatio can help node `a` to measure the numbers of data packets received from node `b`, because if node `b` is a malicious node with grey hole attack, the data packets received from it is comparably low. We also design the ActNeiRatio, which is calculated as 1/ActNei, to enable the normal nodes to determine the number of surrounding active neighboring nodes.

Figures 3.3 shows the procedure of preparing training data for normal server. Similar to the above-mentioned example, node `a` extracts normal features from the observed data of node `b`, and constructs a noraml feature vector concatenating the extracted features. In addition, as it is known, in the training phase, whether or not node `b` is malicious, we associate this information with the normal feature vector as a label. By applying the above procedures to each pair of neighboring node, we can obtain a set of labeled normal feature vectors, which are used to train the normal server in the training phase. Since the normal server classifies a neighboring node of interest into a malicious or normal class, we utilize SMOTE [16] to over-sample the minority class, so that the proportion of training instances from each class is equal. Therefore the model can have better performance because of the balanced training data.

Frequent adaptation

Since the malicious nodes employ reinforcement learning to adjust their behavioral pattern, the normal nodes must frequently and simultaneously need to update their server in the training phase. That is, the classifier is trained at the same time as the deep Q-learning agent of the malicious server is updated. We thus ensure that normal nodes frequently connect to the normal server to update the classifier and detect malicious nodes. Before each adaptation, the normal server collects training data (the features and categories of neighboring nodes) from all the normal nodes, and then uses this data to update the normal classifier.

Pre-training of normal classifier

In the proposed method, we train a neural network classifier and utilize it to detect malicious nodes. However, if we randomly initialize the neural network classifier, it will have low accuracy and may slowly converge. Therefore, for the purpose of fast adaption and accelerating training speed, before the start of training phase, we pre-train the classifier with data obtained in different network environments. In particular, we run simulations involving normal and malicious nodes with ordinary grey hole attack models in various network environments with different parameters, to obtain data by our proposed method. We then utilize this data to construct a pre-trained classifier, and conduct the training phase on this pre-trained classifier instead of a randomly initialized classifier.

Detecting malicious nodes

We assume that node `a` in a given test environment needs to be decided its category by its neighboring nodes. We assume that each neighboring node holds a trained classifier built by our proposed detection method in advance. Assume that, in a given test environment, node `a` judges whether or not node `b` is malicious. Node `a` observes the behavior of node `b` over a period of time and computes a normal feature vector. The computed vector is then fed into the classifier, and the detection category for node `b` by node `a` is obtained.

We utilize majority voting to decide final category of a node. For instance, if node `b` has three neighboring normal nodes, and two of them classify it as malicious node, then node `b` is identified as a malicious node. The majority voting result for node `b` is determined as follows:

$$\hat{y} = \begin{cases} normal & (normal\ vote\ ratio > 0.5) \\ malicious & (otherwise) \end{cases}$$

where normal vote ratio represents the percentage of the neighboring nodes of node `b` voting it as a normal node. We set 0.5 as a detection threshold. In the training procedure of our proposed method, all normal nodes share a same normal server. As a result, in the testing, when the category of a node is decided, there is no difference of the weights of voting from its neighboring nodes. Note that when the normal vote ratio of a node is 0.5, this node is identified as malicious nodes because detecting malicious nodes is considered as more important than mis-detecting normal nodes.

After a node `m` is decided as a malicious node by its neighboring normal nodes, these neighboring nodes will ignore `m` from the sensor network by i) dropping all the packets received from m, and ii) sending no packet to m. Thus, m can no longer harm the wireless sensor network.

3.5 Experiments

3.5.1 Setting

We used the Qualnet 7.4 network simulator. Each node transmitted messages and data packets with a payload sizes of 256 bytes, using an IEEE 802.11b device. The communication range of each node was adjusted to roughly 100 meters, and the network bandwidth was 11Mbps. Similarly to previous studies [26], [49], we used the random way point model [9], with a maximum movement speed of 4.0[m/sec] and pause time of 0. (The velocity of each node was randomly chosen from (0, 4.0)) When there were n nodes in a network, there were $n \cdot m$ ($m \in [0.1, 0.4]$) malicious nodes in the network. The network parameters are described in TABLE 3.2. We conducted parameter analysis, and the default value of each parameter is also shown in TABLE 3.2. When we investigated the influence of a particular network parameter, the other parameters were fixed by the default values.

We randomly chose a pair of source and destination nodes every 0.2 seconds. If the source node had an active route to the destination node, the source node sent a data packet to the destination node directly. Otherwise the source node broadcasts an RReq to find a route to the destination node.

We ran simulations of 1500 episodes in the training phase, and trained both malicious and normal servers simultaneously with random initial node positions and random node IDs. We also selected m from 0.1, 0.2, and 0.4 for each episode in the training phase, and in the test phase, m was 0.3.

Figure 3.4 is the flow chart of the whole experiment procedure.

Competitors. To investigate the effectiveness of the proposed countermeasure (denoted by P-detection), we prepared the following methods.

- *MITR13* [70]: This is the same as MITR13 in Chapter 2 (see Section 2.5.1).
- *Rule-based* [77]: This is a technique combining rule based method with SVM.

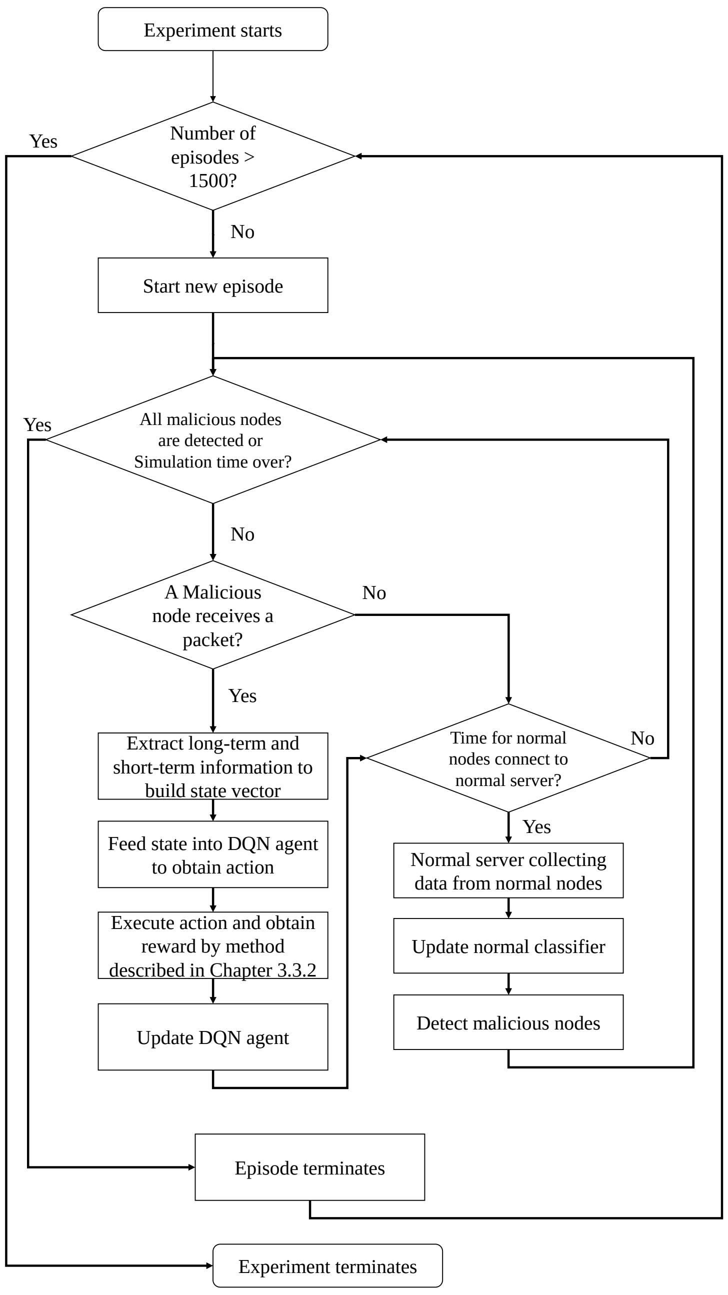

Experiment starts
Number of episodes > 1500?
Yes
No
Start new episode
All malicious nodes are detected or Simulation time over?
Yes
No
A Malicious node receives a packet?
No
Yes
Extract long-term and short-term information to build state vector
Feed state into DQN agent to obtain action
Execute action and obtain reward by method described in Chapter 3.3.2
Update DQN agent
Time for normal nodes connect to normal server?
No
Yes
Normal server collecting data from normal nodes
Update normal classifier
Detect malicious nodes
Episode terminates
Experiment terminates

Figure 3.4: Flow chart of our experiment

Table 3.2: Parameter configuration

Parameter	Values
n	20 (default), 40, 60
m	0.1, 0.2 (default), 0.3, 0.4
Network size [m] $\times$ [m]	200 $\times$ 200, 300 $\times$ 300 (default), 400 $\times$ 400
normal node interval contact seconds [sec]	0.4 (default), 1.2, 2
Threshold time for giving a positive reward [sec]	5, 10 (default), 15, 20
packet dropping ratio	0.25, 0.5(default), 0.75

This method designs a rule for deciding a categories of neighboring nodes. This method employs the following features: packet delivery ratio (PDER), packet modification rate (PMOR), and packet misroute rate (PMISR).

Note that we also update MITR13 and SVM (decided by rule) in the same way as the proposed countermeasure.

To investigate the effectiveness of our proposed grey hole attack model (P-attack), we compared P-attack with ordinary grey hole attack model (O-attack).

Criteria. We focused on the following criteria to measure the performance of MITR13 and P-detection.

- Accuracy, Detection rate, and Mis-detection rate: These criteria are the same as those in Chapter 2.
- Detection time: This indicates how much time normal nodes spend to detect all the malicious nodes in the WSN.

3.5.2 Experimental results and analysis

Influence of minimum epsilon. We present the influence of minimum epsilon in the average total reward of the last 50 episodes in Table 3.3. From this table, we can see

Table 3.3: Influence of minimum epsilon

Minimum epsilon	0	0.01	0.1	0.5
Avg. reward of last 50 episodes	1.39	1.41	1.28	0.94

Table 3.4: Decrease in transmission rate

	Avg. first 50 episodes	Avg. last 50 episodes
Trans. rate	71.7	64.7

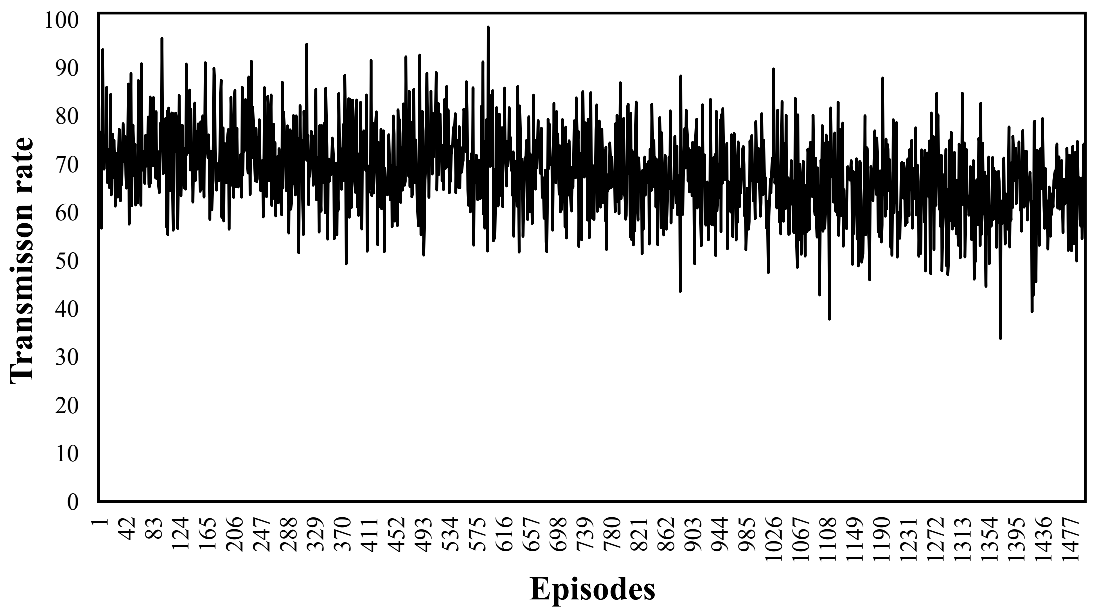


Figure 3.5: Transmisson rate in training phase

that the minimum epsilon value of 0.01 yields the highest average total reward. With the minimum epsilon 0.1 and 0.5, more random actions are chosen. Those random actions are detected by the normal nodes. The method with minimum epsilon being 0.01 also outperforms that with being 0. Minimum epsilon of 0 has no exploration rate at all in the last few episodes, which is easily detected by normal nodes. We consequently use 0.01 as our minimum epsilon in the whole experiment.

Decrease in transmission rate. To illustrate the performance of P-attack, we show the decrease in the transmission rate of the WSN in the training phase (Figure 3.5). We can see that, with the learning of the smart malicious nodes, the transmission rate of the WSN gradually decreases during the training phase. Table 3.4 shows the a comparison of the average transmission rates of the first and last 50 episodes. We can see that the transmission rate drops after the training phase. This shows that the proposed P-attack can learn to disturb the transmission of data packet in WSNs.

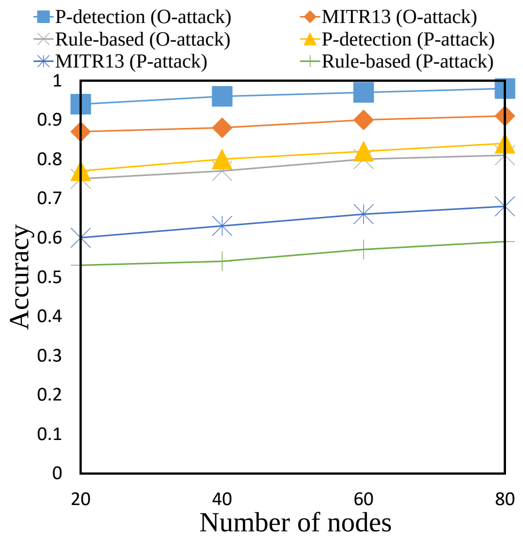
Key parameter analysis

Here, we investigate the performances of P-detection, MITR13, and Rule-based methods under different network parameters.

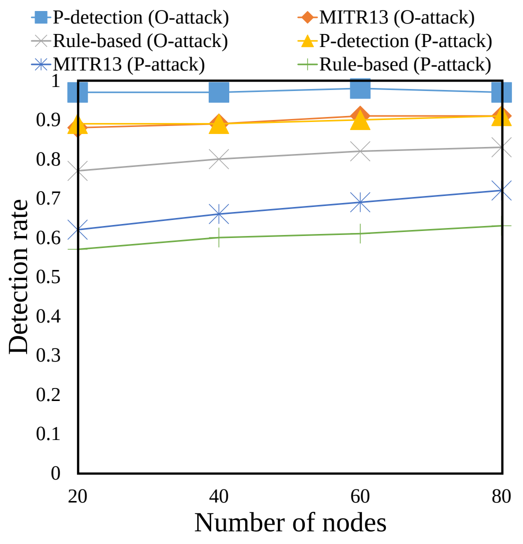
Influence of number of nodes. Figures 3.6(a)–3.6(c) show the performances of P-detection, MITR13, and Rule-based methods with different number of nodes. All these methods obtain better performances as the number of nodes becomes larger. This is because when the number of nodes becomes larger, normal nodes can observe more neighboring nodes. That is, normal nodes can obtain more data for training the classifier. Therefore, they can more clearly find the difference between neighboring malicious and normal nodes. From these figures, we can also see that the detection rate of P-attack is less than that of O-attack for each method, which demonstrates the strength of P-attack.

Influence of ratio of malicious nodes. Figures 3.7(a)–3.7(c) show the performances of P-detection, MITR13, and Rule-based methods with different ratio of malicious nodes. The performances of all the methods decrease when the ratio of malicious nodes become large. Grey hole attack nodes randomly drop packets, so when the number of malicious nodes becomes larger, it is more difficult to find the difference between packet dropping and packet loss.

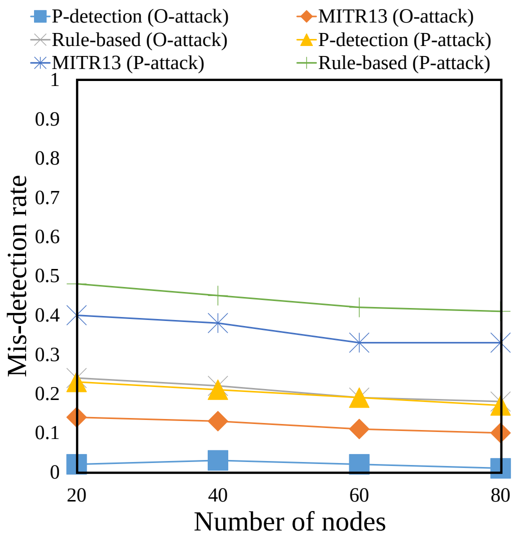
Influence of interval time of normal server update. Figures 3.8(a)–3.8(c) show the performances of P-detection, MITR13, and Rule-based methods with different interval time of normal server update. The performances of Rule-based do not vary in these situations because it is a pre-defined rule based method. There is no change for the results of P-detection and MITR13 for O-attacks because O-attack does not learn from the network, thus the interval time of server update do not have influence on these results. The performances of P-detection and MITR13 for P-attack degrade when the interval time becomes longer. This is because P-detection and MITR13 cannot identify the behaviors of smart malicious nodes if they do not learn frequently. Recall that, in



(a) Accuracy



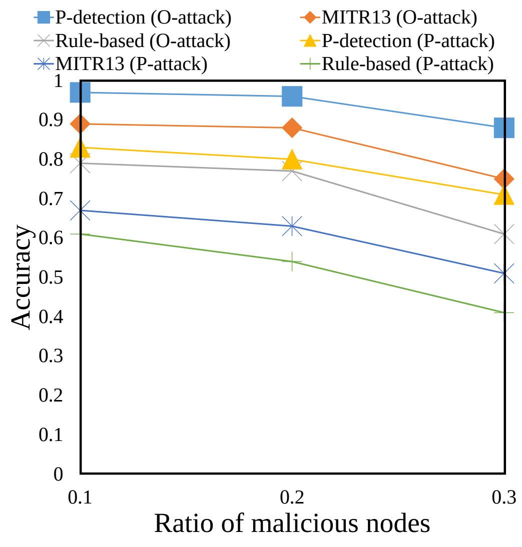
(b) Detection rate



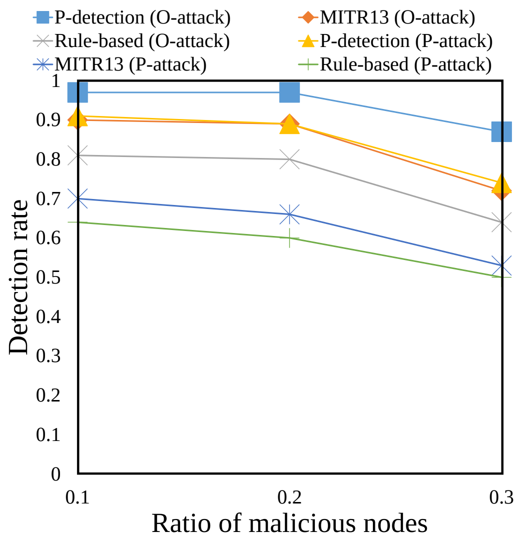
(c) Mis-detection rate

Figure 3.6: Influence of number of nodes on P-detection, MITR13, and Rule-based methods

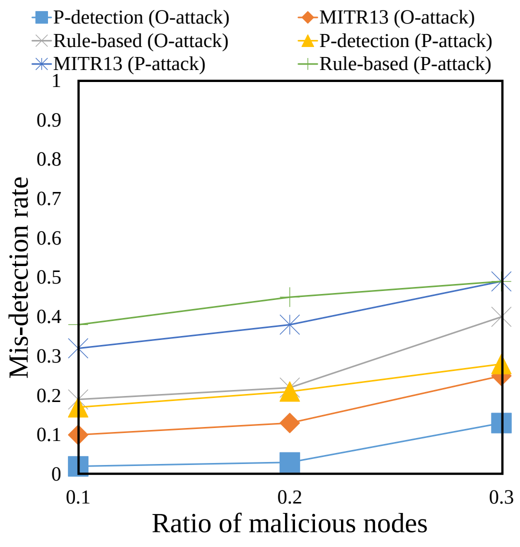
each update, the normal server can obtain data from all normal nodes. Therefore, when interval time becomes longer, the normal server has smaller amount of data for training, and the performance becomes worse.



(a) Accuracy



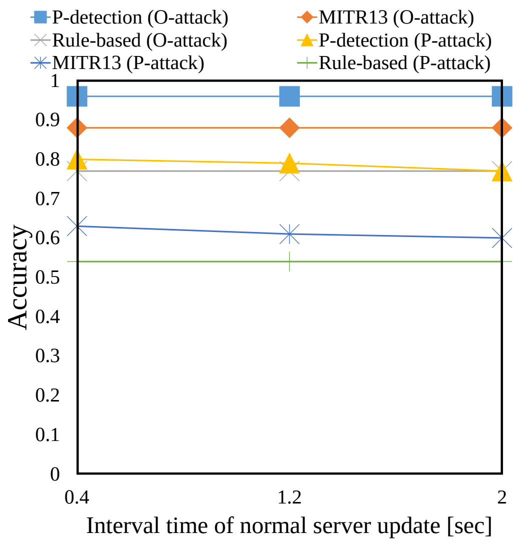
(b) Detection rate



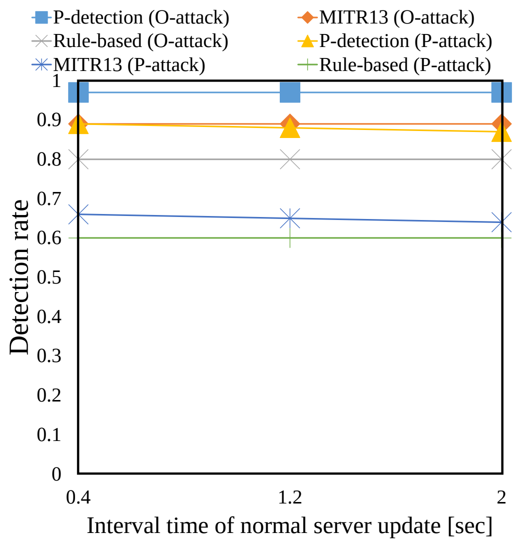
(c) Mis-detection rate

Figure 3.7: Influence of ratio of malicious nodes on P-detection, MITR13, and Rule-based methods

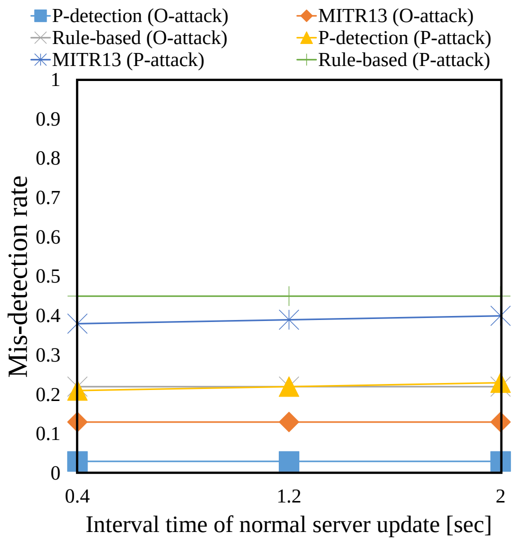
Influence of threshold for giving a positive reward. Figures 3.9(a)–3.9(c) show the performances of P-detection, MITR13, and Rule-based methods with different time threshold for giving a positive reward of the P-attack. When the time threshold becomes



(a) Accuracy



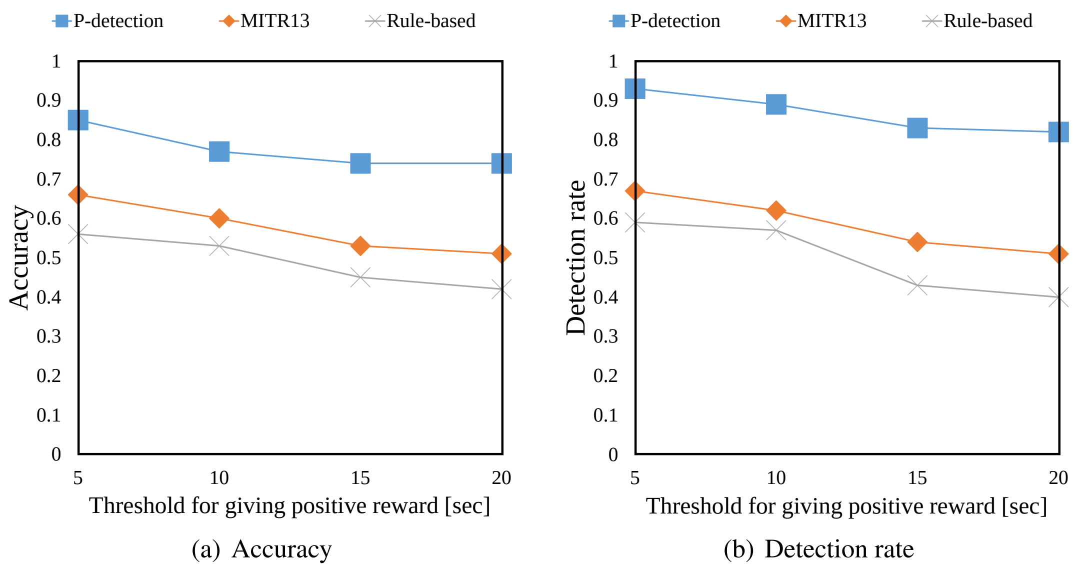
(b) Detection rate



(c) Mis-detection rate

Figure 3.8: Influence of interval time of normal server update on P-detection, MITR13, and Rule-based methods

larger, the detection rates and accuracies of P-detection, MITR13, and Rule-based methods decrease because malicious nodes aim to have longer life time, so they need to perform more normally (*i.e.*, to perform like normal node to avoid being detected), thus the



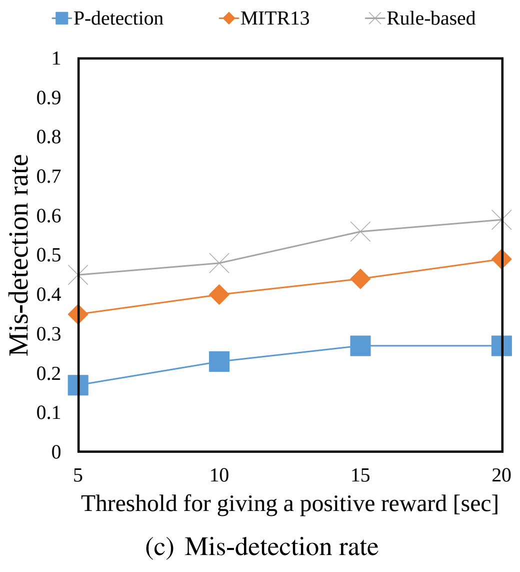
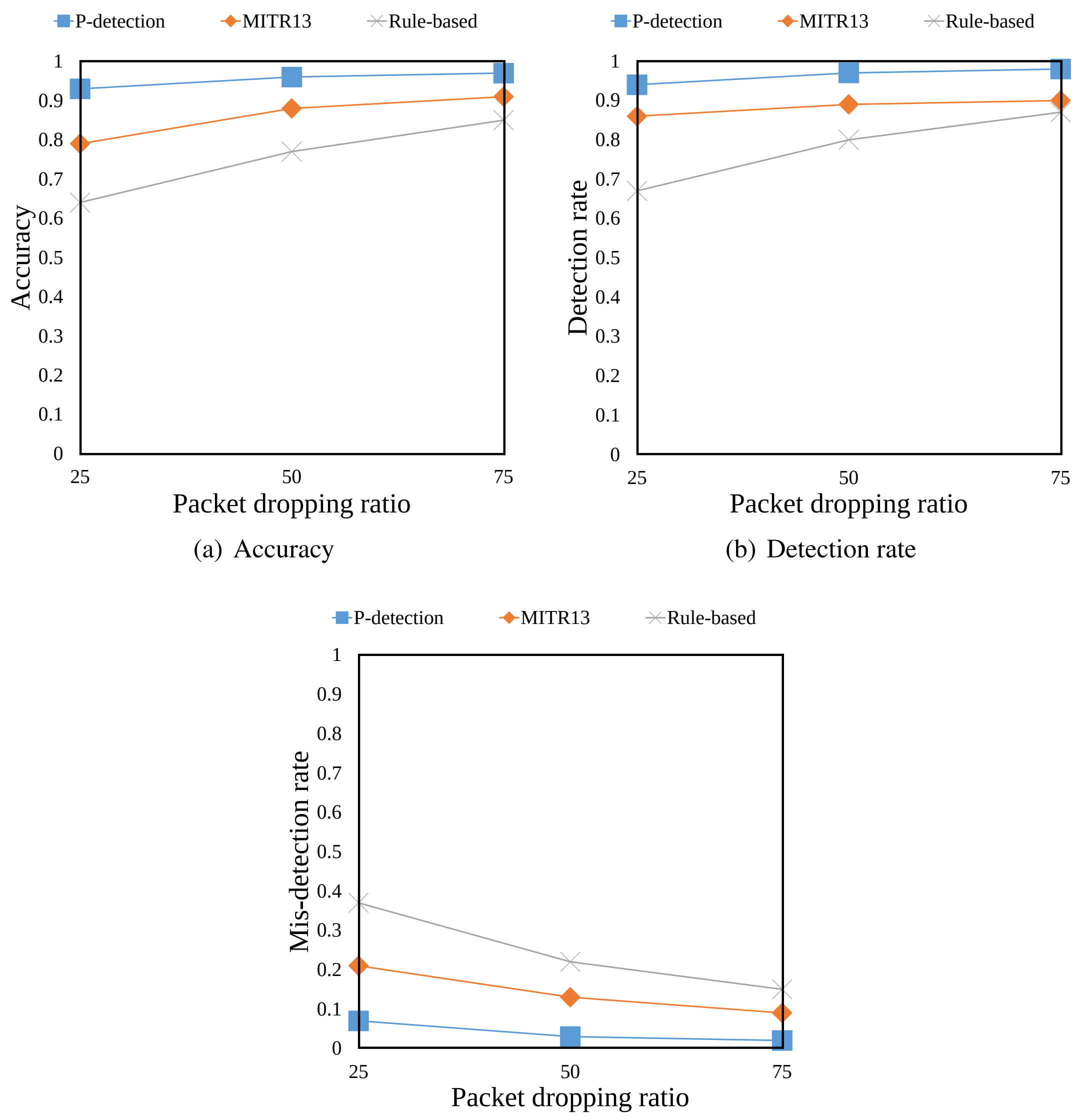


Figure 3.9: Influence of time threshold for giving a positive reward on P-detection, MITR13, and Rule-based methods

detection rates decrease. The detection rate of P-detection is still larger than 0.8 when the time threshold is 20 second, outperforming MITR13 and the Rule-based method.

Influence of packet dropping ratio. Figures 3.10(a)–3.10(c) show the performances of



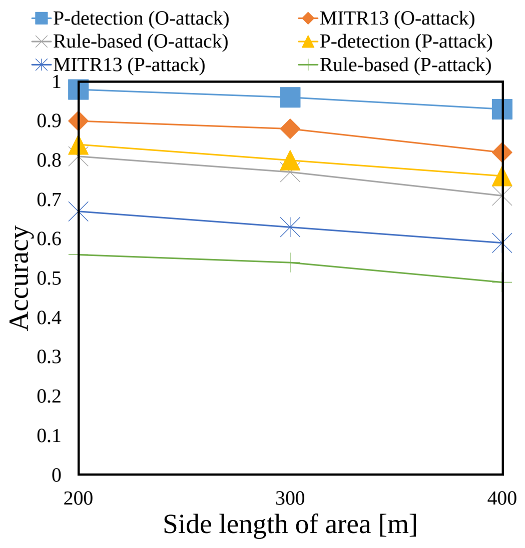
(a) Accuracy

(b) Detection rate

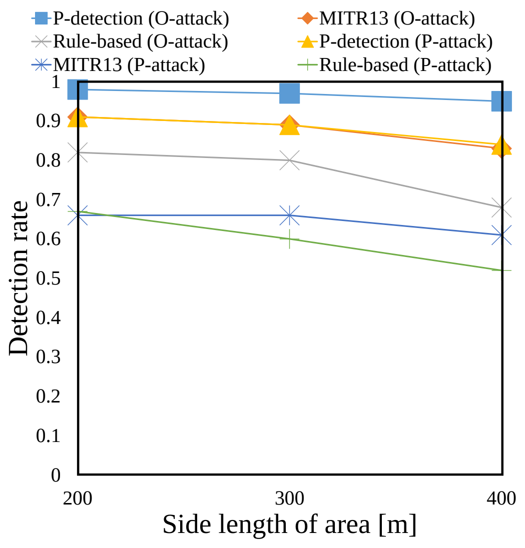
(c) Mis-detection rate

Figure 3.10: Influence of packet dropping ratio for O-attack on P-detection, MITR13, and Rule-based methods

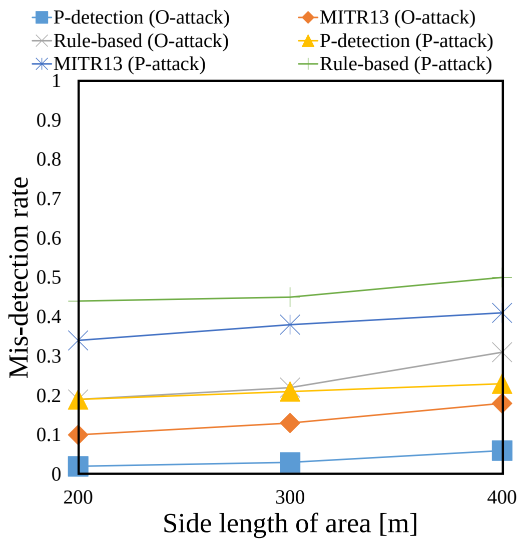
P-detection, MITR13, and Rule-based methods with different packet dropping ratios for O-attack. When packet dropping ratio increases, the accuracies and detection rates of P-detection, MITR13, and Rule-based increase. This is because when malicious nodes



(a) Accuracy



(b) Detection rate



(c) Mis-detection rate

Figure 3.11: Influence of area size of network on P-detection, MITR13, and Rule-based methods

drop more packets, their behaviors will be more different from normal nodes. Therefore, malicious nodes have more possibility to be detected.

Influence of area size of network. Figures 3.11(a)–3.11(c) show the performances of

P-detection, MITR13, and Rule-based methods with different area sizes. From these figures, we can see that when area size becomes larger, the accuracies and detection rates for all three method drop. This is because i) when area size becomes larger, the packet loss happens more frequently, thus normal nodes miss more packets, which leads to worse performance of the classifier (because malicious nodes also drop packets), and ii) when node density is lower, normal server obtains smaller amount of training data from normal nodes, which influences the accuracy.

3.6 Conclusion

In this chapter, we have proposed a smart grey hole attack model, along with an effective countermeasure, for WSNs. We constructed a reinforcement learning-based attack model, with malicious nodes detected using our adaptive server. The experimental results revealed that the model's malicious nodes could learn from the state-of-art countermeasures and thereby extend their lifetime. Our countermeasure, as well, outperformed the state-of-art alternatives and detected the malicious nodes rapidly. Furthermore, faced with a different attack, our method can also be utilized to explore the state.

As part of our future work, we will investigate a particularly powerful all-round attack which not only utilizes the grey hole attack function, but can harm an entire mobile WSN in various ways. Designing an all-round attack is challenging because the state and action spaces are huge. Hence, the neural network of malicious node is difficult to converge.

Chapter 4

Detecting Energy Depriving Malicious Nodes in Energy Harvesting Cooperative Wireless Sensor Networks by Unsupervised Learning

4.1 Introduction

The recent breakthrough in energy harvesting cooperation (EHC) technology [69] relieves the bottleneck of energy limitation in wireless sensor networks (WSNs). Therefore, the concept of energy harvesting cooperative wireless sensor networks (EHC-WSNs) has come up and attracts attentions increasingly [34]. An EHC-WSN is a WSN where nodes can harvest energy from ambient environments (*e.g.*, harvesting from solar energy [108] and vibration [15]) and transfer energy to other nodes.

Much effort has been devoted to extending the lifetime of EHC-WSNs. Most studies focus on optimizing the energy transferring [34] and designing energy-aware routing protocols [5] to enable a WSN to have a longer lifetime. Meanwhile, few works focused on security issues of EHC-WSNs. For example, in the energy cooperative architecture, a node can obtain energy from its neighboring nodes if it lacks energy. A malicious node hence can claim that it lacks energy, even if it has enough energy. In this case, it can deprive energy of its neighboring nodes, and they lose their energy. This kind of energy depriving attack may destruct the network reliability and functionality, which triggers event losses. Even worse, such an attack may lead to severe risks, particularly for real-time and safety-critical applications, such as extreme weather monitoring [47], water quality monitoring [23], and forest fire alarming [85]. Therefore, a valuable insight should be offered into attacks and security issues in EHC-WSNs. In this chapter, we focus on detecting energy depriving malicious nodes in EHC-WSNs. This is the first study that focuses on detecting energy depriving malicious nodes.

4.1.1 Motivation

Numerous studies have figured out various attack models for malicious node and proposed classifier, rule, and encryption-based methods for detecting malicious nodes in a WSN environment [67, 79, 83]. However, these techniques cannot provide security with EHC-WSNs because they do not consider malicious nodes that harm energy harvesting and energy cooperation.

When developing a countermeasure for the energy depriving attack, we face two challenges. (i) The information about energy storage of a node is private data, and it cannot be known by other nodes. This is an inherent problem because a malicious node can easily claim that it has little energy without any risk. Some energy-aware routing protocols demand nodes to report their current status of the energy storage periodically or add the status of energy storage to header [44, 104]. However, malicious nodes can still ignore these settings and pretend to have little energy. We may be able to design a rule-based method that decides nodes that always claim to have low energy as malicious nodes. However, this is impractical because of the second challenge that (ii) the energy harvesting efficiency of each node in EHC-WSN is different. For example, in forest fire alarming EHC-WSN where nodes harvest solar energy, the movements of the sun and clouds will result in shadows over some nodes. Therefore, the harvesting efficiencies would be low. A rule-based method that simply decides nodes with low energy as malicious nodes thereby cannot deal well with this problem. Moreover, any classifier-based methods are not suitable for this situation because we cannot prepare enough training data. This is because the malicious nodes have different forged amount of energy storages, and different EHC-WSNs have different network environments. Therefore, it is impossible to prepare a robust classifier based on not enough training data. Consequently, a well-designed energy depriving nodes detection method in EHC-WSNs is required.

4.1.2 Contribution

In this chapter, we design a malicious node detection method based on unsupervised learning for EHC-WSNs, where energy depriving malicious nodes exist. Specifically, we make the following contributions.

- We tackle the problem of energy depriving nodes detection in EHC-WSNs online. To the best of our knowledge, we are the first to address this problem.
- We propose a deep neural network-based clustering method to detect energy depriving malicious nodes in EHC-WSNs.
- We propose a method to obtain data for the clustering, and propose inherent features to represent the energy depriving attack model.
- We conduct extensive experiments to investigate the performance of our method. Our experimental results demonstrate that our method outperforms the baseline method.

This chapter is organized in the following way. Section 4.2 gives a brief overview of related works. Section 4.3 introduces our assumption in this chapter. Section 4.4 presents our proposed method, and experimental results are illustrated in Section 4.5. Finally, this chapter is concluded in Section 4.6.

4.2 Related works

Energy harvesting and energy cooperation are promising methods for extending the lifetime of WSNs. In [81], Raghunathan *et al.* firstly designed a solar energy harvesting wireless system. They proposed that wireless nodes can harvest from solar energy to relieve energy constraints. Inspired by their work, many studies suggested that nodes in WSNs can harvest from various ambient environments [15, 100]. Later on, with the rapid development of wireless transfer technology, the lifetime of a WSN has been further extended. In [42], Huang *et al.* firstly enabled the wireless power transfer (WPT) in cellular networks. They designed the architecture, model, and deployment for WPT in cellular networks. As an application of wireless energy transfer in WSNs, Shi *et al.* [94] studied a scenario where nodes in a WSN charge their energy from a mobile charging vehicle wirelessly. Their experimental results proved that the lifetime of a WSN could be extensively extended by wireless energy transfer. Then, Gurakan *et al.* [34] proposed a method that combines the energy harvesting and wireless energy transfer to create wireless energy harvesting cooperation (EHC) systems (EHC-WSNs). The work of Minasian *et al.* [69] further improved EHC systems by optimizing energy allocation. These studies proved that EHC-WSNs are promising; however, no work addressed security issues of EHC-WSNs.

For WSNs, a large number of studies have investigated various security issues. For a single kind of attack detection, Li *et al.* [58] developed a method that detects jamming attacks by enabling normal nodes to respond correctly to the jammers. For replica node attacks, Ho *et al.* [38] utilized a sequential analysis to identify abnormal nodes in a WSN. To address the security issues of EHC networks, Kang *et al.* [48] proposed 4 kinds of attack models in EHC networks, and they proposed a naïve approach to mitigate the effect of malicious nodes. A few years later, considering security issues in wireless rechargeable sensor networks, Lin *et al.* [62] proposed a novel attack model, which is called the cooperative denial of charging attack, to demonstrate that security in wireless rechargeable sensor networks needs to be further emphasized. However, they do not concern about the energy depriving attack.

Besides the countermeasure for a single kind of attack, many studies also design secure routing protocols for WSNs. In [64], Liu *et al.* firstly proposed an application-friendly method to detect insider attackers in WSNs by monitoring many aspects of sensor networking behaviors. Hu *et al.* [40] presented an attack-tolerant time-synchronization for secure data aggregation in WSNs. However, these studies consider only a WSN that is unaware of energy issues. When energy issues are concerned, their methods become inapplicable.

4.3 Assumption

In this section, we describe the assumption of our study. We assume an energy harvesting cooperative wireless sensor network (EHC-WSN) consisting of W_n wireless nodes with unique identifiers. Similarly to existing study of EHC-WSNs [34], we assume the nodes are static.

As a routing protocol, AODV, which is a standard routing protocol in WSNs, is employed. Note that nodes have to add their current status of energy storage in the headers of all the packets they send for energy cooperation.

4.3.1 Energy harvesting and energy cooperation

We assume that all nodes are capable of harvesting energy from ambient environments. Due to the random nature of ambient sources (*e.g.*, shadows over solar energy harvesting panel), we assume that each node harvests energy with different efficiency, which is the same as existing studies [17, 24]. All nodes have the same maximum energy storages, and cannot harvest more energy if the current energy storages reach the maximum energy storages.

Following the groundbreaking work of Zhang *et al.* [115], which has proved that the energy can be simultaneously transfered with wireless informationwe, we assume that nodes transfer information and energy simultaneously when they send packets. The amount of transferred energy is based on the water-filling algorithm [34]. That is, when a node $\mathtt{s}$ transfers energy to a node $\mathtt{d}$, it aims to balance the energy storage of them. We also assume that when energy is transferred between nodes, a particular amount of the energy is lost because of the power loss [75]. Let E_s and E_d denote the status of energy storage of node $\mathtt{s}$ and node $\mathtt{d}$, respectively. Assume that E_s is larger than E_d, and let E_{tr} and E_{re} denote the energy transferred from node $\mathtt{s}$ and received by node $\mathtt{d}$, respectively. Let λ denote the energy transferring efficiency:

$$E_{re} = \lambda \cdot E_{tr}$$

Consequently, in order to keep the balance of status of energy storage after energy transferring, E_{tr} is calculated as:

$$E_{tr} = \frac{E_s - E_d}{1 + \lambda}.$$

4.3.2 Attack model

In the energy depriving attack in an EHC-WSN, a malicious node pretends to have less energy level than its real energy level before it sends a packet by adding noises to its forged amount of energy. In particular, malicious nodes remove small and different values from their real energy storages for avoid being detected. Recall that the status of energy storage should be included in the header.

4.4 Proposed method

In this section, we describe our proposed method for detecting energy depriving nodes in EHC-WSNs. We utilize a clustering method because malicious nodes may have different forged energy levels, and a classifier-based detection method may not work well. Moreover, the energy consumptions of nodes near and far away from the sink are different (it is clear that a node near the sink consumes more energy because of the frequent packet transmission). Hence, it is difficult to prepare a classifier-based solution to recognize such differences. On the other hand, in a clustering-based approach, a node only focuses on data from neighboring nodes, and it does not suffer from the differences of positions. Therefore, we utilize a clustering method to detect malicious neighboring nodes.

A large number of studies, *e.g.*, [13, 43, 112], have demonstrated that, compared with other clustering methods, deep neural network-based clustering methods have better performance due to the theoretical function approximation properties [39] and their feature learning capabilities [8]. Therefore, we utilize a deep neural network-based clustering method. Note that, in general, the task of clustering is to divide a set of data points into some clusters. In our method, each normal node, playing the role as an observing node, first prepares data points for clustering by observing its neighboring nodes. Then, observing nodes utilize the data points to form clusters. Two clusters, which are normal and malicious clusters are formed. After the clustering, observing nodes utilize the clustering results to decide malicious nodes. For a neighboring node `a`, if the data points from node `a` in the malicious cluster are more than data points from node `a` in the normal cluster, node `a` is decided as a malicious node.

4.4.1 Preparation of data points for clustering

In this section, we describe how normal nodes prepare data points for clustering. We assume that a node clusters its neighboring nodes at time slot T, and we assume the node holds the observed data from its neighboring nodes. In our method, we split the observed data to create more data points for clustering because large amount of data can elevate the performance of clustering the method.

As mentioned, nodes have to add their current status of energy storage in the headers. Each normal node thus can observe the status of energy storage of its neighboring nodes by overhearing their packets. At each time slot, normal nodes create features from observed energy. These features are hereinafter called *energy features.* At each time slot, an energy feature vector of each neighboring node is created. Therefore, at time slot T, a normal node has T energy feature vectors for each of its neighboring nodes.

Figure 4.1 illustrates the energy feature vector set extraction procedure. In this figure, node a extracts the energy feature vector set of its neighboring nodes. Hereinafter, we use the term *original feature vector set* to denote the set of T vectors of a neighboring node. It is important to note that the energy feature vector set of each neighboring node is time-series data because it is obtained along with time.

Recall that our approach is to cluster the neighboring nodes of each normal node. We have already obtained the original feature vector set of each neighboring node by the above procedure. However, because we utilize a deep neural network-based clustering method, it is impractical to treat an original feature vector set as a data point for clustering. The reason is that the number of energy feature vector sets is usually small (the number of energy feature vector sets is equal to the number of neighboring nodes). It is clear that less information makes machine learning approaches not functional and easy to overfit [14]. We hence need more data points to enable the clustering method functional.

To deal with this problem, we propose a method that creates more data points for clustering. Instead of using an original feature vector set as a data point for clustering, we use subsets of it. Let k denote the size of a subset. Our method extracts vectors between $(sk + 1)$-th vector to $(s + 1)k$-th vector from each vector set to form the s-th subsets. In each subset, the vectors are still time-series data. Our method thus

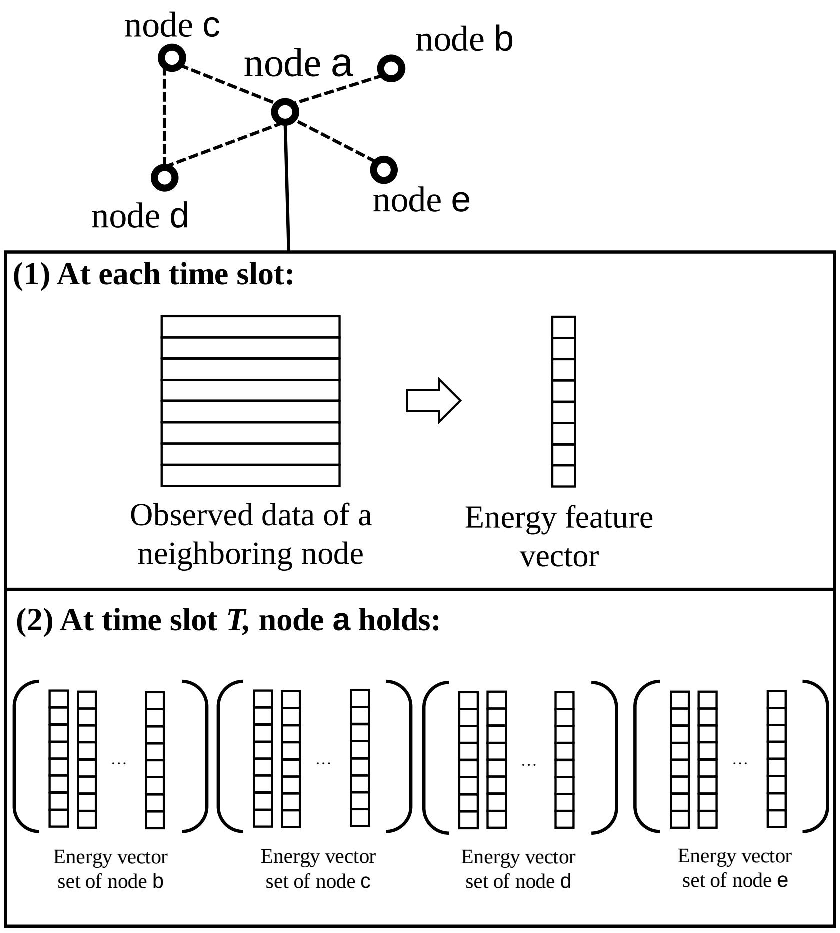

node c
node a
node b
node d
node e
(1) At each time slot:
Observed data of a neighboring node
Energy feature vector
(2) At time slot T, node a holds:
Energy vector set of node b
Energy vector set of node c
Energy vector set of node d
Energy vector set of node e

Figure 4.1: An example of energy feature vector set creation. (1) At each time slot, node a creates an energy feature vector of each neighboring node. (2) At time T, for each neighboring node, node a holds an energy vector set which contains T energy feature vectors.

maintains the properties of time-series in each subset. Figure 4.2 shows an example of our method. In this example, assume that a normal node has already observed an original feature vector set with size 10. We can create 2 subsets with size 5 from the original feature vector set with size 10. Then, we treat these subsets as data points for clustering. That is, a subset is a data point for clustering.

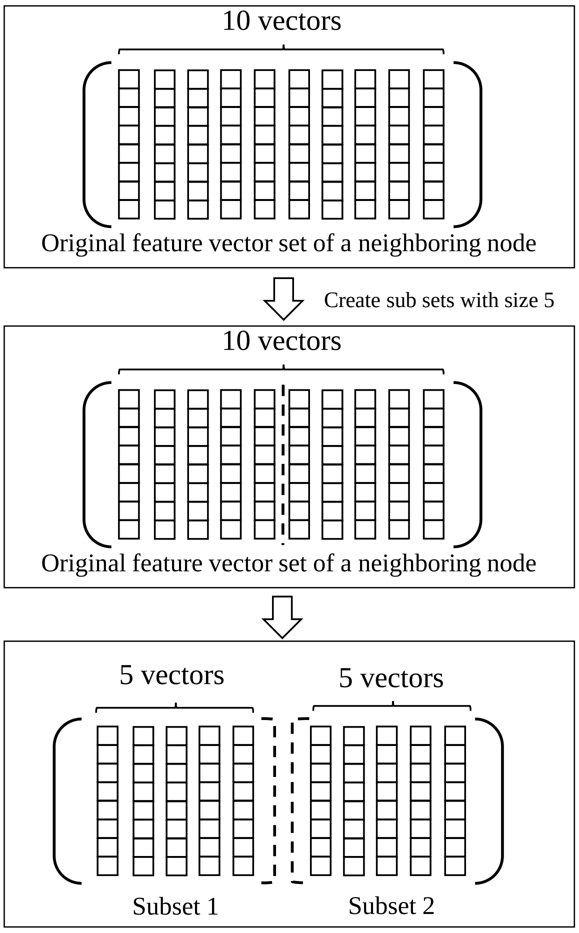

10 vectors
Original feature vector set of a neighboring node
Create sub sets with size 5
10 vectors
Original feature vector set of a neighboring node
5 vectors
5 vectors
Subset 1
Subset 2

Figure 4.2: An example of creating subsets. Two subsets with size 5 are created from the original feature vector set with size 10.

4.4.2 Clustering method

As mentioned above, we utilize a deep neural network-based clustering method to cluster the above-mentioned subsets. In particular, we re-organize the method proposed in [112], which is a standard deep neural network-based clustering method, called deep embedding clustering (DEC).

We first make a brief introduction of DEC. DEC is a method that simultaneously learns feature representations and cluster assignments using deep neural networks. DEC learns a mapping from the data space to a lower-dimensional feature space in which it iteratively optimizes a clustering objective. In general, the task of clustering is to cluster a set X of x points into y clusters. Instead of clustering directly in the *data space* X, DEC aims to transform the data with a non-linear mapping $f_\theta : X \rightarrow Y$, where θ is a learn-able parameter, which is parametrized by deep neural networks, and Y is the latent cluster space. Finally, DEC can learn the parameters of the deep neural networks, and the deep neural networks represent a clustering model of $X \rightarrow Y$ (see more details in [112]).

The original model of DEC only employs fully connected dense layers, which are not good at processing time-series data [114]. Recall that our subsets for clustering are time-series data. We thus re-organize the structure of deep neural networks of DEC by adding 1D-convolutional layers and pooling layers to process time-series data better, which is inspired by existing studies [56, 114]. To avoid overfitting, we discard some fully connected layers of the original DEC structure. Figure 4.3 shows our re-organization to original DEC.

Note that our clustering method can set the number of clusters as a hyper-parameter. We thus set the number of clusters as two because we assume two categories of nodes (normal and malicious nodes).

4.4.3 Energy features

Here we describe energy features, which are used in malicious node detection. We assume that each node observes the status of the energy storage of its neighboring node. Before we describe the energy features, we present the information about energy related to neighboring nodes observed by each node (Table 4.1). This is used to compute the energy features. To enable normal nodes to have better understandings of neighboring nodes, we design short-term and long-term features. Short-term features enable normal nodes to recognize the instant status of energy storage of themselves and their neighboring nodes, and long-term features let normal nodes understand the historical behaviors about the energy of their neighboring nodes. In particular, we directly employ the current status of energy storages of the observing node and the neighboring node as two short-term features. They are utilized as energy features because (i) they can directly represent the status of energy storage at that time slot, and (ii) they both have the max-

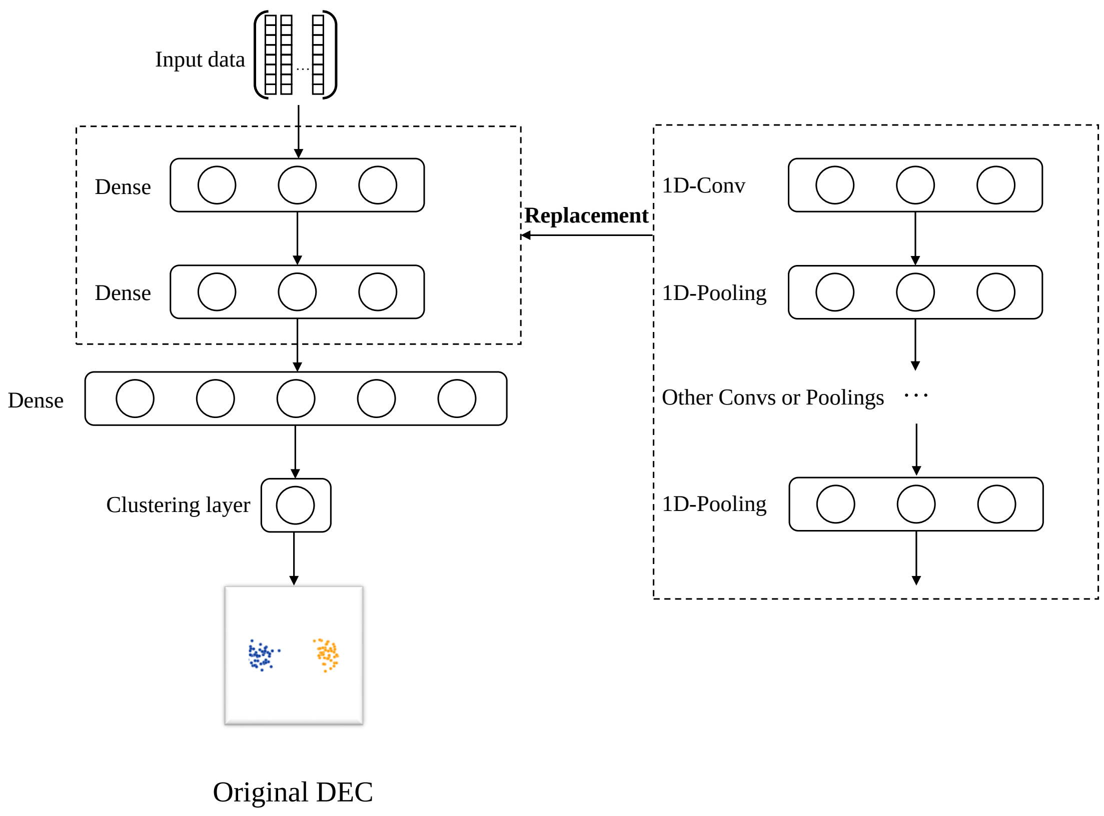


Figure 4.3: Re-organization of the deep neural network structure of DEC by replacing some dense layers with convolutional and pooling layers of dense layers to processing time-series data

imum and minimum values, which are proper as inputs of the deep neural networks, because they can be normalized smoothly.

Table 4.2 shows the long-term energy features used in our method, and they are designed based on the inherence of the energy depriving attack. These long-term energy features are all ratios calculated from the information observed by nodes. Such ratios are more robust against the difference in node density than numbers, because it is clear that the number of neighboring nodes can easily influence features based solely on numbers.

Here, we take TransferRatio (energy transferred ratio) and StorageRatio (energy storage ratio) as two examples to illustrate the reason why our energy features can describe the inherence of energy depriving attack. Assume that a normal node a observes

Table 4.1: Information about energy observed by each node

Information	Definition
NTra	Energy transferred to one neighbor
NRec	Energy received from one neighbor
TTra	Total energy transfered
TRec	Total energy received
SS	Current status of energy storage of observing node
SN	Current status of energy storage of one neighbor
TH	Total energy harvested

Table 4.2: Long-term energy features

Behavioral features	Definition
TransferRatio	NTra / TTra
ReceiveRatio	NRec / TRec
CompensateRatio	NTra / TRec
IncomeRatio	NRec / TTra
GiveRatio	NTra / TH
VoluntaryRatio	TTra / TH
StorageRatio	SS / SN
DeliveryRatio	TTra / TH

its neighboring node `b`. (i) TransferRatio: this feature of `b` can help node `a` to measure how much energy is transferred to node `b` compared with all its transferred energy. If node `b` is a malicious node, `a` transfers comparably more energy to node `b` than the other neighboring nodes because it claims to have lower energy. (ii) StorageRatio: this feature measures the ratio of the current status of energy storage of nodes `b` and `a`. If node `b` is a malicious node, this feature value would be comparably lower than the feature values of the other neighboring nodes, because node `b` forges its status of energy storage to a lower value. Therefore, `a` can utilize these two features to cluster `b` as a malicious node. Other long-term energy features are also designed based on the same intuition.

It is important to note that the energy features are obtained from messages sent by neighboring nodes. In other words, each normal node can obtain the energy features by overhearing messages, so our method incurs no additional communication cost.

4.4.4 Pre-training of model

In our proposed method, we utilize a deep learning-based clustering method. However, if we simply initialize the neural network model for clustering randomly or utilizing methods not fit the environment, this neural network model will converge slowly. Therefore, to elevate accuracy and accelerate training speed, we pre-train the neural network model with data obtained in different network environments. Assume that an EHC-WSN is deployed in a particular area. We thus can iterate simulations involving normal and malicious nodes with energy depriving attack model in this area with different node positions, to obtain data by our proposed method. We then utilize this data to construct a pre-trained model.

For example, assume that we plan to detect malicious nodes when the an EHC-WSN has been deployed in real world for 400 second.. We can conduct a simulation, whose simulation time is 400 second to obtain training data, and utilize the training data to construct a pre-trained model. Assume that the pre-trained model of 400 second is model MA. In real malicious node detection, we perform training of the clustering models from model MA when this EHC-WSN has been deployed for 400 second.

In the pre-training procedure, we utilize SMOTE [16] to over-sample the minority class, so that the proportion of training instances from each class is equal, therefore the pre-trained model can have better performance because of the balanced training data.

4.4.5 Malicious node detection

In this section, we describe our method for determining the categories of the two clusters and deciding malicious nodes. After two clusters are formed, an observing node needs to determine the categories of them, *i.e.*, which one contains the subsets of malicious nodes. Then, based on the result, an observing node decides malicious nodes.

Cluster of malicious subsets decision. We assume that a normal node a in a given EHC-WSN forms clusters of subsets of its neighboring nodes. After two clusters are formed, we calculate the average of feature SN (see Table 4.1) for both the clusters:

$$SN_{ave} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{k} SN}{N \cdot k},$$

where N denotes the number of subsets from a cluster.

Then, the cluster with lower SN_{ave} is decided as a cluster that contains subsets from malicious nodes. This is because of the inherence of the energy depriving malicious nodes, *i.e.*, they forge to have little energy.

Malicious node decision. After the categories of two clusters are obtained, node $\mathtt{a}$ decides the malicious neighboring nodes. Assume that node $\mathtt{a}$ holds n subsets of each neighboring node, and m subsets of a particular neighboring node are in the cluster with malicious subsets. Let $\hat{y}$ denote the result of malicious node decision determined by node $\mathtt{a}$ for this neighboring node. $\hat{y}$ is obtained as follows:

$$\hat{y} = \begin{cases} normal & (m < \frac{n}{2}) \\ malicious & (otherwise) \end{cases}$$

We set $\frac{n}{2}$ as a decision threshold because during the clustering procedure, all the subsets are obtained from the same node (node $\mathtt{a}$). As a result, in the malicious node decision procedure, there is no difference between the weights of each subset. When $m = \frac{n}{2}$, this neighboring node is identified as a malicious node because detecting malicious nodes is considered as more important than mis-detecting normal nodes. Note that our detection method detects malicious nodes online. Therefore, the malicious node detection procedure does not require labeled data from a particular network environment.

4.5 Experiment

This section summarizes our experiments that evaluate the performance of the proposed method.

4.5.1 Setting

We used the Qualnet 7.4 network simulator, and we set our experiments similarly to previous studies [17, 34, 62, 75].

We randomly deployed 500 nodes in a $100m \times 100m$ square field, and 100 nodes were malicious nodes. Each node transmitted messages and data packets with a payload sizes of 256 bytes, using an IEEE 802.11b device. The communication range of each node was adjusted to roughly 3 meters, and the network bandwidth was 11Mbps. The maximum energy of each node was $100mJ$. We decided a time slot as 4 seconds, and a node harvested energy per time slot randomly from $0.01mJ$ to $1mJ$ [17]. The energy consumption of transmitting and receiving a packet were $0.1mJ$ and $0.08mJ$, respectively. We considered a random destination node scenario [51] because the development of edge computing enables each sensor node to process data [93]. We randomly chose a pair of source node and destination node every 1 seconds. If the source node has an active route to the destination node, the source node sends a data packet to the destination node directly. Otherwise, the source node broadcasts an RReq to find a route to the destination node. When a node forwards a data packet through a neighboring node, if the status of energy storage of this neighboring node is less than this node, this node transfers energy to this neighboring node simultaneously (see Section 4.3.1). The simulation time was 4000 second, which means a simulation consisted of 1000 time slots, and the subset size k is 10. Similarly to [75], we set λ as 0.29.

Malicious nodes. In our assumption, malicious nodes forged to have little energy (Section 4.3.2). However, if malicious nodes always forged its status of energy storage to an extremely low value (*e.g.*, 0), it is clear that these malicious nodes can be detected easily. We thus added noises to their forged amount of energies. In particular, let E_{real} and E_{forged} denote the real status of energy storage of a malicious node and the forged status of energy storage, respectively. To obtain E_{forged}, this malicious node deducts E_{real} by a Gaussian white noise [57] denoted by E_{gwn} as:

$$E_{forged} = E_{real} - E_{gwn}.$$

In this way, malicious nodes remove small and different values from their real energy storages for avoid being detected. Then, the malicious node will add E_{forged} into the header when it sends a packet. The mean of E_{gwn} is 0.1, which is the same as the energy consumption of transmitting a packet, and the standard deviation is 0.05.

Evaluation methods. We evaluated the following methods.

- *K-means*: This method clusters the original energy vector set into two clusters by K-means [36], which is a standard method for clustering. After two clusters are formed, each node decides the cluster with lower average SN as a cluster that contains malicious energy vector set. Then, the neighboring nodes holds the original energy vector set of the malicious cluster are decided as malicious nodes. We prepare this method to investigate the effectiveness of creating subsets.

- *WSK-means*: This method clusters subsets into two clusters by K-means. We prepare this method to investigate the effectiveness of our deep neural network-based clustering method.
- *WSDEC*: This method clusters subsets into clusters by the original DEC [112]. Therefore, this method does not have convolutional and pooling layers in the deep neural network model. We prepare this method to investigate the effectiveness of our method to handle time-series data.
- *WSCNNDEC*: This method clusters subsets into clusters by the re-organized DEC. Therefore, this method only utilizes the model initialization method proposed in [112]. This method does not pre-train the neural network model. We prepare this method to investigate the effectiveness of pre-training model.
- *Proposed*: This is the proposed method in this chapter.
- *Proposed-one*: This method clusters subsets into clusters by the re-organized DEC with pre-training of model. However, in the method, only one pre-trained model, which is pre-trained at 4000 second, is prepared. We prepare this method to investigate the effectiveness of preparing multiple pre-trained models with exactly same detection time.

All data obtained during the simulation time were used to compute energy features. We assumed that nodes executed a neighboring node detection procedure every 100 time slots.

Implementation. We implemented our deep neural network set on Keras 2.2.4 [1] with TensorFlow [2] as backend. Determining hyper-parameters by cross-validation on a validation set is not an option in unsupervised clustering because we do not have labeled

[1]https://keras.io/

[2]https://www.tensorflow.org/

data. Thus we use commonly used parameters for deep neural networks. In particular, inspired by [103], we set network dimensions of $WSDEC$ to d-25-25-100-10, where d is the original data-space dimension determined by the subset size k and the number of time slots. All layers are densely (fully) connected. For our proposed method, we set network dimensions to d-Conv1D(25, 10)-Conv1D(25, 10)-MaxPooling1D(4)-Conv1D(50, 10)-Conv1D(50,10)-GlobalAveragePooling()-100-10, where Conv1D denotes a 1-dimension convolutional layer. The number of clusters are set as 2, because we have two categories of nodes. In the training procedure, we utilize fine-tuning [52] technique. That is, we only train the last two dense layers of the pre-trained neural network model (see Section 4.4.4), and the other layers are fixed. This is because we have small amount of training data, and fine-tuning helps to elevate performance in such a situation [99].

Criteria. As mentioned earlier, our method incurs no additional communication costs. We therefore focus on the following criteria to measure the performance of the above methods.

- Accuracy: This is represented by $\frac{N_{nor\to nor,mal\to mal}}{N}$, where $N_{nor\to nor,mal\to mal}$ and N are respectively the set of correctly decided neighboring nodes of all normal nodes and the set of all neighboring nodes of all normal nodes.

- Detection rate: This is represented by $\frac{N_{mal\to mal}}{N_{mal}}$, where $N_{mal\to mal}$ and N_{mal} are respectively the set of correctly decided malicious neighboring nodes of all normal nodes and the set of all malicious neighboring nodes of all normal nodes.

- Mis-Detection rate: This is represented by $\frac{N_{nor\to mal}}{N_{nor}}$, where $N_{nor\to mal}$ and N_{nor} are respectively the set of wrongly decided normal neighboring nodes of all normal nodes and the set of all normal neighboring nodes of all normal nodes.

4.5.2 Result

Comparison with K-means and WSK-means. Figures 4.4–4.6 show the performances of our method (Proposed), K-means, and WSK-means. In particular, K-means is averagely 45% worse than that of Proposed with regard to the three criteria. This is because (i) K-means uses the original vector set, (ii) compared with K-means, deep neural

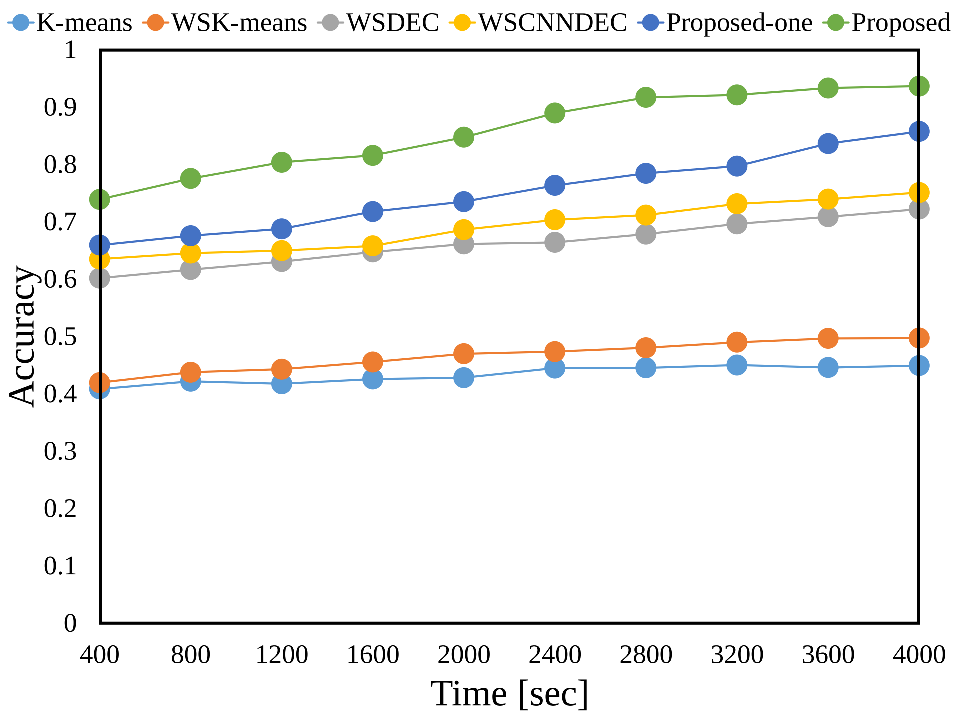


Figure 4.4: Accuracies of evaluation methods

network-based clustering method has better feature learning capabilities, and (iii) our data points are not balanced (the data points from normal nodes is much more than those of malicious nodes), while K-means is not good to handle unbalanced data [46]. Compared with K-means, WSK-means is about averagely 5% better with regard to the three criteria. This is because WSK-means clusters subsets, and more data points provide a better performance.

Comparison with WSDEC and WSCNNDEC. Figures 4.4–4.6 show the performances of Proposed, WSDEC, and WSCNNDEC. We can see that Proposed is averagely 20% better than that of WSDEC with regard to the three criteria. This is because Proposed utilizes pre-trained models, while WSDEC does not. Note that, compared with WSDEC, WSCNNDEC obtains better results. This is because subsets are time-series data, and WSCNNDEC utilizes convolutional layers to process the time-series data more appropriately.

Comparison with Proposed-one. Figures 4.4–4.6 show the performances of Proposed

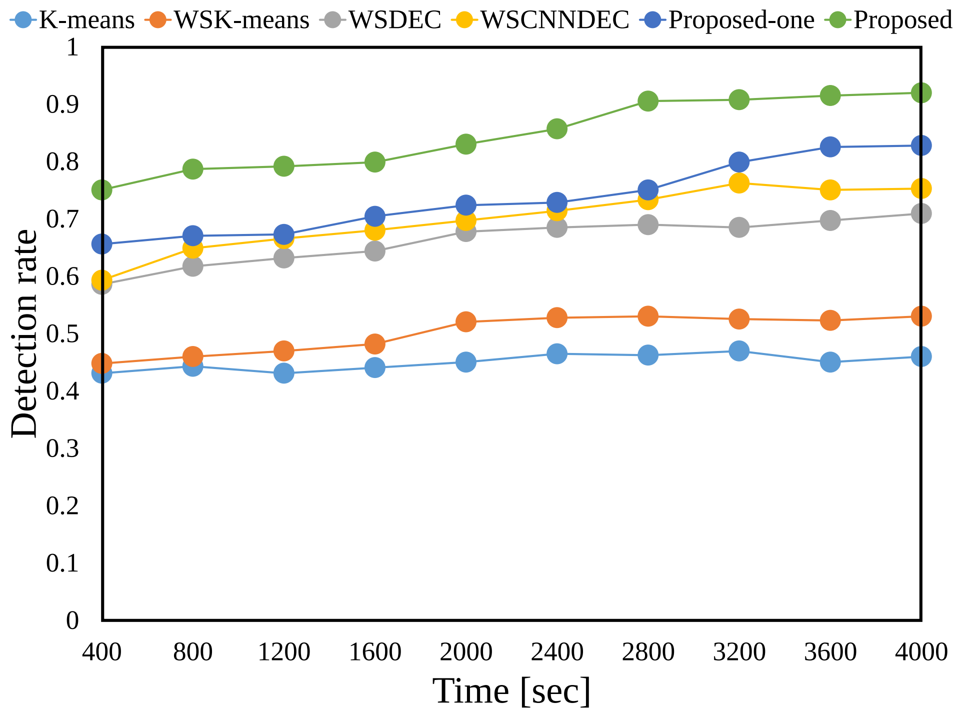


Figure 4.5: Detection rates of evaluation methods

and Proposed-one. Compared with Proposed-one, Proposed has better performance with regard to the three criteria. This is because we prepare pre-trained models at different times in Proposed, and we train models based on the pre-trained models obtained at the same times as detection times. Proposed-one only utilizes one pre-trained model trained at 4000 second. Therefore, only one pre-trained model cannot perform well in all the situations with different detection times.

Influence of simulation time. From Figures 4.4–4.6, we can see that the performances of all the methods become better as time spends. This is because longer time can generate larger amount of data points for clustering. It is clear that clustering methods can work better with more data points.

Information gain of energy features. Table 4.3 shows the information gain of energy features. The information gain is used to find distinguishing features of feature vectors. The more information gain of a feature increases, the better the feature classifies the vector. We obtained the information gain by adding ground truth label to each energy

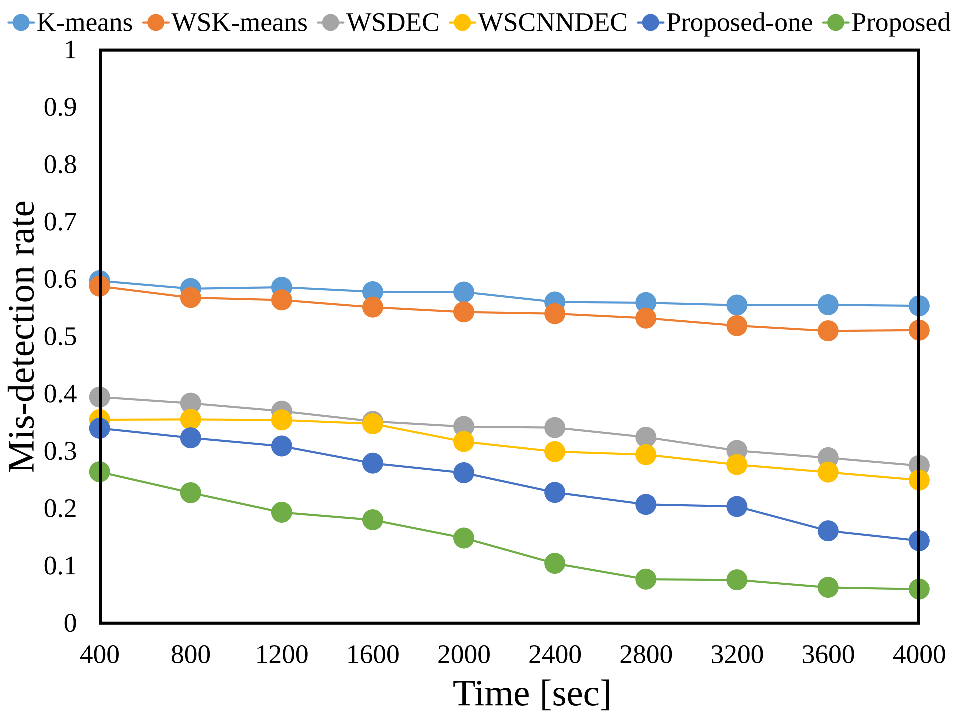


Figure 4.6: Mis-detection rates of evaluation methods

Table 4.3: Information gain of energy features

Energy features	Info. gain
SS	0.009
SN	0.012
TransferRatio	**0.134**
ReceiveRatio	**0.103**
IncomeRatio	0.026
GiveRatio	0.019
VoluntaryRatio	0.011
StorageRatio	0.023
DeliveryRatio	0.014
CompensateRatio	**0.091**

feature vector. For example, if a neighboring node is malicious, the labels of energy vectors observed from it are malicious. From this table, we can find that the information gain of our designed features are all over 0.009. Compared with other classifier based studies for detecting malicious nodes in WSNs [30, 70], our energy features have competitive or better information gain. We bold top-3 information gain. This result shows that our designed features are effective and can be employed for detecting energy depriving attack in EHC-WSNs.

Performance of proposed method when there is no malicious node existing. We conducted experiments to investigate the performance of the proposed method when there is no malicious node existing. We present the mis-detection rate of the proposed method at 4000 second (we do not present the detection rate because there is no malicious node, and the detection rate is meaningless). When there is no attacker in the network, the mis-detection rate is 11.7%. Our clustering-based method has some loss even when there is no malicious node existing. This is because our clustering-based method still forms two clusters of the data points when there is no malicious node. Therefore, some data points are forced to form a malicious cluster, which causes the increasing of mis-detection rate.

4.6 Conclusion

In this chapter, we presented an energy depriving malicious node detection method for EHC-WSNs. We designed inherent features of energy depriving attack. We proposed a method for obtaining more data points for clustering, as well as a deep neural network-based clustering method. The experiments revealed that the proposed method outperformed the comparison methods.

As a part of future work, we will focus on a light-weight malicious node detection method. A sensor nodes often have restricted computational resources. The computational units of the sensor node cannot handle a very large and complex detection model. Therefore, a light-weight energy depriving malicious node detection method with high detection accuracy is required. Furthermore, we need to decrease the mis-detection rate when there is no malicious node existing. Our proposed method now wrongly decides a part of neighboring nodes as malicious nodes when there is no malicious nodes existing in the network. We plan to tackle this problem in the future.

Chapter 5

Summary

5.1 Summary of Contributions

In this thesis, we have discussed about detecting malicious nodes in wireless sensor networks. In Chapter 1, we illustrated a number of crucial monitoring applications of WSNs to protect people from disasters, and to obtain important data from a wide field. However, these WSNs suffer from malicious nodes. We thus made clear research issues for detecting malicious nodes in WSNs in Chapter 1.

Although some studies utilized machine learning-based approaches to detecting malicious nodes in WSNs, they ignored the fact that machine learning models trained in a particular training network environment cannot work well in an unknown test network environment. Therefore, in Chapter 2, we have proposed an ensemble learning-based method to detect malicious nodes in unknown network environments. In our method, we first prepared weak malicious node detectors trained in different network environments, and then constructed a strong ensemble malicious node detector, which is tailored to a given test environment, by fusing weak detectors whose performances are estimated to be high in the test environment. To prepare a weak malicious node detector, we categorized attack models in WSNs by their purposes, and then extracted inherent features for the weak malicious node detector according to the purposes of these attack models. The experimental results have shown that our proposed method detects malicious nodes with the highest detection rate, in comparison with existing methods.

In Chapter 3, we have focused on detecting malicious nodes with learning ability in WSNs. Existing studies have proposed methods that detect pre-defined attacks. However, with the rapid development of machine-learning technology, it is natural to consider that malicious node can also employ machine learning-based methods to avoid being detected. Hence, we designed a smart grey hole attack where malicious nodes can smartly decide their behaviors by reinforcement learning. In particular, we designed actions, states, and rewards for the smart grey hole malicious nodes. To detect such malicious nodes, we extracted inherent features to build classifiers. Then, our classifier is frequently updated to learn the latest behavioral patterns of smart malicious nodes. The experimental results have shown that the smart malicious nodes are more difficultly to be detected than malicious nodes with a pre-defined attack model. Moreover, the experiments demonstrated that our proposed detection method can detect the smart malicious nodes more quickly and with a higher detection rate than existing detection methods.

In Chapter 4, we have tackled the problem of detecting energy depriving malicious nodes in EHC-WSNs. In an EHC-WSN, a node transmits its energy to its neighboring node with little energy storage to extend the life-time of this neighboring node. However, malicious nodes can pretend to have little energy to deprive energy of other nodes. Therefore, we proposed a method to detect energy depriving malicious nodes by unsupervised learning. In our proposed method, normal nodes first observe the behaviors of neighboring nodes and create data points. Then the normal nodes utilize our proposed clustering method to cluster these data points and decide the malicious nodes. The experimental results have shown that our proposed method outperforms the comparison methods in terms of accuracy, detection rate, and mis-detection rate.

In summary, detecting malicious nodes is a crucial issue when people are utilizing WSNs. Our proposed methods can detect malicious nodes in WSNs with a high detection rate. In particular, our proposed methods can detect different categories of malicious nodes even in unknown network environments. Even high-level malicious nodes that can learn cannot avoid being detected by our proposed methods. Our proposed methods can also detect malicious nodes in EHC-WSNs, which are a category of state-of-the-art WSNs. Therefore, our achievements contribute to increase the security of WSN services and utilization.

5.2 Future work

Through this thesis work, we found the following remaining issues.

5.2.1 Detecting malicious nodes with learning ability in unknown environments

In this thesis, we propose a method for detecting multiple attacks in unknown environments by ensemble learning (Chapter 2) and a method for detecting malicious nodes with learning ability (Chapter 3). However, it is not a trivial task to detect malicious nodes with learning ability in unknown environments. We cannot ensemble weak detectors to detect malicious nodes with learning ability, because in different training environments where we obtain the weak detectors, the malicious nodes have different behavioral patterns. That is, simply selecting weak detectors by a weak detector evaluator (which is obtained by environmental features, see Chapter 2) is not enough. We need to consider about different malicious node behavioral patterns as well. Therefore, we plan to extend our methods to detect malicious nodes with learning ability in unknown environments as a part of future work.

5.2.2 A light-weight malicious node detection method

Machine-learning based methods usually need large memory and computational resources. For example, a deep neural network model with 10000 parameters is about 40 KB. However, the memory of a sensor node is strictly restricted, usually about 64–128 KB [96]. Therefore, if the sensor nodes need to store more monitored data, the memory space for detection model needs to be smaller. Besides, a sensor node also only has restricted computational resources [106]. The computational units of the sensor node cannot handle a very large and complex detection model. Therefore, a light-weight malicious node detection method with high detection accuracy is required. We plan to extend our methods to light-weight methods as a part of future work.

5.2.3 Security of WSNs

In this thesis, we have proposed methods for detecting malicious nodes. It is clear that detecting the malicious nodes can contribute to the security of WSNs. However, we need to further investigate the influence of detecting malicious nodes on security of WSNs. For example, in our study in Chapter 2, our method detects malicious nodes with 90% of detection rate. We can consider that if a malicious node `m` is detected by neighboring normal nodes, we have many existing methods (*e.g.*, dropping the packets received from `m` and sending no packet to `m` to ignore `m` from the WSN) to deal with `m`. However, after the detected malicious nodes are ignored, we still do not know how the WSN performs (*e.g.*, the transmission rate and total traffic of a WSN) with the undetected malicious nodes. Therefore, we plan to investigate the connection between the security of WSN and detecting malicious nodes in WSN as a part of future work.

Acknowledgment

This thesis represents not only my research at Osaka University but also a part of my life that has a good opportunity to study and live in Japan these years. Completion of this thesis was possible because of the supports and efforts of a number of people.

First and foremost, I would like to express my sincere gratitude to my adviser, Professor Takahiro Hara, who always gives me countless opportunities from the first step in this laboratory. It has been an honor to be his PhD student. He has been doing his best cultivating me to grow as a good researcher from my Master through PhD. This thesis would not have finished without his valuable advice and his guidance.

I am grateful to my instructive professors, Professor Takuya Maekawa and Professor Daichi Amagata, for being attentive and caring about my study during these years. They always gave me helpful guidance and support throughout my study.

I would also like to acknowledge the committee members of my thesis, Professor Toru Fujiwara and Professor Shinji Shimojo at the Department of Multimedia Engineering of the Graduate School of Information Science and Technology of Osaka University. Their insightful and constructive comments considerably improved the quality of the thesis.

I would like to express my appreciation to Professor Makoto Onizuka and Professor Yasuyuki Matsushita at the Department of Multimedia Engineering of the Graduate School of Information Science and Technology of Osaka University for their innovative lectures and warm student life support.

I am thankful to Professor Shojiro Nishio, President of Osaka University who was my adviser during my research student years for giving warm support to my study and daily life, Professor Tomoki Yoshihisa, Professor Masumi Shirakawa for actively working for all students and always kindly giving the valuable guidance as well as many opportunities.

I would like to acknowledge my research team both past and present members, Dr. Yuka Komai, Dr. Kamalas Udomlamlert, Dr. Masahiro Yokoyama, Dr. Keisuke Goto, Mr. Takuji Tsuda, Mr. Yuki Nakayama, Mr. Syunya Nishio, Mr. Shunhei Hayashida, Mr. Yusuke Arai, Mr. Ryohei Nakatani, and Mr. Hayato Nakama. Also, it is my pleasure to work with all smart members in Hara laboratory. All of them have given me cooperative and active supports throughout my study.

Last but by no means least it gives me immense pleasure to offer my hearty thanks to my family. No words can express my appreciation their support and love in my life. My parents and my wife have provided me with the greatest concerns and cares, for which I am eternally grateful.

Finally, with all my best wishes to those wonderful people who ever gave me assistance, supports and encouragements throughout my research experience.

REFERENCE

[1] S. Abbas, M. Merabti, D. Llewellyn-Jones, and K. Kifayat. Lightweight sybil attack detection in manets. *IEEE Systems Journal*, 7(2):236–248, 2013.

[2] R. Akbani, T. Korkmaz, and G. Raju. Emltrust: an enhanced machine learning based reputation system for manets. *Ad Hoc Networks*, 10(3):435–457, 2012.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

[4] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Proc. of Annual Conf. on Neural Information Processing Systems (NIPS)*, pages 5048–5058, 2017.

[5] M. H. Anisi, G. Abdul-Salaam, M. Y. I. Idris, A. W. A. Wahab, and I. Ahmedy. Energy harvesting and battery power based routing in wireless sensor networks. *Wireless Networks*, 23(1):249–266, 2017.

[6] J. Arora, P. Singh, and S. Rani. Attacks in manets–a survey. *Int'l Journal of Multidisciplinary Management Studies (EXCEL)*, 3(10):151–157, 2013.

[7] H. Ayatollahi, C. Tapparello, and W. Heinzelman. Reinforcement learning in mimo wireless networks with energy harvesting. In *Proc. of Int'l Conf. on Communications (ICC)*, pages 1–6, 2017.

[8] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(8):1798–1828, 2013.

[9] C. Bettstetter, H. Hartenstein, and X. Pérez-Costa. Stochastic properties of the random waypoint mobility model. *Wireless Networks*, 10(5):555–567, 2004.

[10] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[11] S. Buchegger and J.-Y. Le Boudec. Performance analysis of the confidant protocol. In *Proc. of Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 226–236, 2002.

[12] M. Cardei and D.-Z. Du. Improving wireless sensor network lifetime through power aware organization. *Wireless networks*, 11(3):333–340, 2005.

[13] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *Proc. of European Conf. on Computer Vision (ECCV)*, pages 132–149, 2018.

[14] R. Caruana, S. Lawrence, and C. L. Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Proc. of Annual Conf. on Neural Information Processing Systems (NIPS)*, pages 402–408, 2001.

[15] S. Chamanian, S. Baghaee, H. Uluşan, Ö. Zorlu, E. Uysal-Biyikoglu, and H. Külah. Implementation of energy-neutral operation on vibration energy harvesting wsn. *IEEE Sensors Journal*, 2019.

[16] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research (JAIR)*, 16:321–357, 2002.

[17] Q. Chen, H. Gao, Z. Cai, L. Cheng, and J. Li. Energy-collision aware data aggregation scheduling for energy harvesting sensor networks. In *Proc. of Int'l Conf. on Computer Communications (INFOCOM)*, pages 117–125, 2018.

[18] X. Chen, K. Makki, K. Yen, and N. Pissinou. Sensor network security: A survey. *IEEE Communications Surveys and Tutorials*, 11(2):52–73, 2009.

[19] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proc. of Conf. on Empirical Methods in Natural Language Processing and Int'l Joint Conf. on Natural Language Processing (EMNLP–IJCNLP)*, pages 1724–1734, 2014.

[20] A. Coates and A. Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proc. of Int'l Conf. on Machine Learning (ICML)*, pages 921–928, 2011.

[21] H. Deng, Q.-A. Zeng, and D. P. Agrawal. Svm-based intrusion detection system for wireless ad hoc networks. In *Proc. of Vehicular Technology Conf. (VTC)*, volume 3, pages 2147–2151, 2003.

[22] D. Dong, M. Li, Y. Liu, X.-Y. Li, and X. Liao. Topological detection on wormholes in wireless ad hoc and sensor networks. *IEEE Trans. on Networking (TON)*, 19(6):1787–1796, 2011.

[23] W. Du, Z. Xing, M. Li, B. He, L. H. C. Chua, and H. Miao. Sensor placement and measurement of wind for water quality studies in urban reservoirs. *ACM Trans. on Sensor Networks (TOSN)*, 11(3):41:1–41:7, 2015.

[24] N. Elvin and A. Erturk. *Advances in energy harvesting methods*. Springer Science & Business Media, 2013.

[25] J. Eriksson, S. V. Krishnamurthy, and M. Faloutsos. Truelink: A practical countermeasure to the wormhole attack in wireless networks. In *Proc. of Int'l Conf. on Network Protocols*, pages 75–84, 2006.

[26] W. Galuba, P. Papadimitratos, M. Poturalski, K. Aberer, Z. Despotovic, and W. Kellerer. Castor: Scalable secure routing for ad hoc networks. In *Proc. of Int'l Conf. on Computer Communications (INFOCOM)*, pages 1–9, 2010.

[27] B. Gao, D. Amagata, T. Maekawa, and T. Hara. Detecting energy depriving malicious nodes by unsupervised learning in energy harvesting cooperative wireless sensor networks. In *Proc. of IPSJ DPS Workshop*, pages 97–104, 2019.

[28] B. Gao, D. Amagata, T. Maekawa, and T. Hara. Detecting malicious nodes with learning ability in mobile wireless sensor networks. In *Proc. of DEIM Forum, online*, 2019.

[29] B. Gao, T. Maekawa, D. Amagata, and T. Hara. Malicious node detection with machine learning in manets. In *Proc. of DEIM Forum, online*, 2017.

[30] B. Gao, T. Maekawa, D. Amagata, and T. Hara. Environment-adaptive malicious node detection in manets with ensemble learning. In *Proc. of Int'l Conf. on Distributed Computing Systems (ICDCS)*, pages 556–566, 2018.

[31] B. Gao, T. Maekawa, D. Amagata, and T. Hara. Robust malicious node detection with ensemble learning in manets. *IPSJ Journal*, 60(2):501–513, 2019.

[32] B. Gao, T. Maekawa, D. Amagata, and T. Hara. Detecting reinforcement learning-based grey hole attack in mobile wireless sensor networks. *IEICE Trans. on Communications*, 2020.

[33] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. Continuous deep q-learning with model-based acceleration. In *Proc. of Int'l Conf. on Machine Learning (ICML)*, pages 2829–2838, 2016.

[34] B. Gurakan, O. Ozel, J. Yang, and S. Ulukus. Energy cooperation in energy harvesting communications. *IEEE Trans. on Communications (TCOM)*, 61(12):4884–4898, 2013.

[35] M. A. Hamid, M. Rashid, and C. S. Hong. Routing security in sensor network: Hello flood attack and defense. *Proc. of Int'l Conf. on Networking, Embedded and Wireless Systems (ICNEWS)*, pages 2–4, 2006.

[36] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[37] E. Hernandez-Orallo, M. D. S. Olmos, J.-C. Cano, C. T. Calafate, and P. Manzoni. Cocowa: A collaborative contact-based watchdog for detecting selfish nodes. *IEEE Trans. on Mobile Computing (TMC)*, 14(6):1162–1175, 2015.

[38] J.-W. Ho, M. Wright, and S. K. Das. Fast detection of replica node attacks in mobile sensor networks using sequential analysis. In *Proc. of Int'l Conf. on Computer Communications (INFOCOM)*, pages 1773–1781, 2009.

[39] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[40] X. Hu, T. Park, and K. G. Shin. Attack-tolerant time-synchronization in wireless sensor networks. In *Proc. of Int'l Conf. on Computer Communications (INFOCOM)*, pages 41–45, 2008.

[41] Y.-C. Hu, A. Perrig, and D. B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Workshop on Wireless Security (WiSe)*, pages 30–40, 2003.

[42] K. Huang and V. K. Lau. Enabling wireless power transfer in cellular networks: Architecture, modeling and deployment. *IEEE Trans. on Wireless Communications (TWC)*, 13(2):902–912, 2014.

[43] P. Huang, Y. Huang, W. Wang, and L. Wang. Deep embedding network for clustering. In *Proc. of Int'l Conf. on Pattern Recognition (ICPR)*, pages 1532–1537, 2014.

[44] M. K. Jakobsen, J. Madsen, and M. R. Hansen. Dehar: A distributed energy harvesting aware routing algorithm for ad-hoc multi-hop wireless sensor networks. In *Proc. of Int'l Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–9, 2010.

[45] F.-C. Jiang, C.-H. Lin, and H.-W. Wu. Lifetime elongation of ad hoc networks under flooding attack using power-saving technique. *Ad Hoc Networks*, 21:84–96, 2014.

[46] M. Jianliang, S. Haikun, and B. Ling. The application on intrusion detection based on k-means cluster algorithm. In *Proc. of Int'l Forum on Information Technology and Applications (IFITA)*, volume 1, pages 150–152, 2009.

[47] E. Kanagaraj, L. Kamarudin, A. Zakaria, R. Gunasagaran, and A. Shakaff. Cloud-based remote environmental monitoring system with distributed wsn weather stations. In *IEEE SENSORS*, pages 1–4, 2015.

[48] J. Kang, R. Yu, S. Maharjan, Y. Zhang, X. Huang, S. Xie, H. Bogucka, and S. Gjessing. Toward secure energy harvesting cooperative networks. *IEEE Communications Magazine*, 53(8):114–121, 2015.

[49] B. Karaoglu and W. Heinzelman. Cooperative load balancing and dynamic channel allocation for cluster-based mobile ad hoc networks. *IEEE Trans. on Mobile Computing (TMC)*, 14(5):951–963, 2015.

[50] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *Proc. of Int'l Workshop on Sensor Network Protocols and Applications (SNPA)*, pages 113–127, 2003.

[51] M. I. Khan, W. N. Gansterer, and G. Haring. Static vs. mobile sink: The influence of basic parameters on energy efficiency in wireless sensor networks. *Computer communications*, 36(9):965–978, 2013.

[52] Y. Kim. Convolutional neural networks for sentence classification. In *Proc. of Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, 2014.

[53] R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc.of Int'l Joint Conf. on Artificial Intelligence (IJCAI)*, volume 14, pages 1137–1145, 1995.

[54] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Proc. of Annual Conf. on Neural Information Processing Systems (NIPS)*, pages 3675–3683, 2016.

[55] V. Laxmi, C. Lal, M. S. Gaur, and D. Mehta. Jellyfish attack: Analysis, detection and countermeasure in tcp-based manet. *Journal of Information Security and Applications (JISA)*, 22:99–112, 2015.

[56] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[57] O. Lepskii. On a problem of adaptive estimation in gaussian white noise. *Theory of Probability & Its Applications (TVP)*, 35(3):454–466, 1991.

[58] M. Li, I. Koutsopoulos, and R. Poovendran. Optimal jamming attacks and network defense policies in wireless sensor networks. In *Proc. of Int'l Conf. on Computer Communications (INFOCOM)*, pages 1307–1315, 2007.

[59] W. Li, A. Joshi, and T. Finin. Coping with node misbehaviors in ad hoc networks: A multi-dimensional trust management approach. In *Proc. of Int'l Conf. on Mobile Data Management (MDM)*, pages 85–94, 2010.

[60] X. Li, R. Lu, X. Liang, and X. Shen. Side channel monitoring: Packet drop attack detection in wireless ad hoc networks. In *Proc. of Int'l Conf. on Communications (ICC)*, pages 1–5, 2011.

[61] Y. Li, L. Xiao, H. Dai, and H. V. Poor. Game theoretic study of protecting mimo transmissions against smart attacks. In *Proc. of Int'l Conf. on Communications (ICC)*, pages 1–6, 2017.

[62] C. Lin, Z. Shang, W. Du, J. Ren, L. Wang, and G. Wu. Codoc: A novel attack for wireless rechargeable sensor networks through denial of charge. In *Proc. of Int'l Conf. on Computer Communications (INFOCOM)*, pages 856–864, 2019.

[63] P. Lin, C. Qiao, and X. Wang. Medium access control with a dynamic duty cycle for sensor networks. In *Proc. of IEEE Wireless Communications and Networking Conf. (CAT)*, volume 3, pages 1534–1539, 2004.

[64] F. Liu, X. Cheng, and D. Chen. Insider attacker detection in wireless sensor networks. In *Proc. of Int'l Conf. on Computer Communications (INFOCOM)*, pages 1937–1945, 2007.

[65] G. Liu, X. Wang, X. Li, J. Hao, and Z. Feng. Esrq: An efficient secure routing method in wireless sensor networks based on q-learning. In *Proc. of Int'l Conf.*

on Trust, Security and Privacy in Computing and Communications (TrustCom), pages 149–155, 2018.

[66] X. Lu, X. Wan, L. Xiao, Y. Tang, and W. Zhuang. Learning-based rogue edge detection in vanets with ambient radio signals. In *Proc. of Int'l Conf. on Communications (ICC)*, pages 1–6, 2018.

[67] D. Mansouri, L. Mokdad, J. Ben-Othman, and M. Ioualalen. Detecting dos attacks in wsn based on clustering technique. In *Proc. of Wireless Communications and Networking Conf. (WCNC)*, pages 2214–2219, 2013.

[68] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proc. of Int'l Conf. on Mobile Computing and Networking (MobiCom)*, pages 255–265, 2000.

[69] A. Minasian, S. ShahbazPanahi, and R. S. Adve. Energy harvesting cooperative communication systems. *IEEE Trans. on Wireless Communications (TWC)*, 13(11):6118–6131, 2014.

[70] A. Mitrokotsa and C. Dimitrakakis. Intrusion detection in manet using classification algorithms: The effects of cost and model selection. *Ad Hoc Networks*, 11(1):226–237, 2013.

[71] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[72] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[73] A. Nadeem and M. P. Howarth. An intrusion detection & adaptive response mechanism for manets. *Ad Hoc Networks*, 13:368–380, 2014.

[74] H. A. Nguyen, A. Förster, D. Puccinelli, and S. Giordano. Sensor node lifetime: An experimental study. In *Proc. of Workshops of Int'l Conf. on Pervasive Computing and Communications (PerCom)*, pages 202–207, 2011.

[75] C. Park, S. Lee, G.-H. Cho, and C. T. Rim. Innovative 5-m-off-distance inductive power transfer systems with optimally shaped dipole coils. *IEEE Trans. on Power Electronics (TPE)*, 30(2):817–827, 2014.

[76] H. P. Patel and M. Chaudhari. A time space cryptography hashing solution for prevention jellyfish reordering attack in wireless adhoc networks. In *Proc. of Int'l Conf. on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–6, 2013.

[77] M. Patel and S. Sharma. Detection of malicious attack in manet a behavioral approach. In *Proc. of Int'l Advance Computing Conf. (IACC)*, pages 388–393, 2013.

[78] N. J. Patel and R. H. Jhaveri. Detecting packet dropping nodes using machine learning techniques in mobile ad-hoc network: A survey. In *Proc. of Int'l Conf. on Signal Processing And Communication Engineering Systems (SPACES)*, pages 468–472, 2015.

[79] A.-S. K. Pathan. *Security of self-organizing networks: MANET, WSN, WMN, VANET*. CRC press, 2016.

[80] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *Proc.of Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 90–100, 1999.

[81] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava. Design considerations for solar energy harvesting wireless embedded systems. In *Proc. of Int'l Conf. on Information Processing in Sensor Networks (IPSN)*, page 64, 2005.

[82] S. Ramaswamy, H. Fu, M. Sreekantaradhya, J. Dixon, and K. E. Nygard. Prevention of cooperative black hole attack in wireless ad hoc networks. In *Proc. of Int'l Conf. on Wireless Networks (ICWN)*, pages 570–575, 2003.

[83] D. R. Raymond and S. F. Midkiff. Denial-of-service in wireless sensor networks: Attacks and defenses. *Proc. of Int'l Conf. on Pervasive Computing and Communications (PerCom)*, pages 74–81, 2008.

[84] E. Rodrigues Gomes and R. Kowalczyk. Dynamic analysis of multiagent q-learning with ε-greedy exploration. In *Proc. of Int'l Conf. on Machine Learning (ICML)*, pages 369–376, 2009.

[85] M. Saoudi, A. Bounceur, R. Euler, and T. Kechadi. Data mining techniques applied to wireless sensor networks for early forest fire detection. In *Proc. of Int'l Conf. on Cloud Computing and Internet of Things (CCIOT)*, pages 71:1–71:7, 2016.

[86] N. Schweitzer, A. Stulman, A. Shabtai, and R. D. Margalit. Contradiction based gray-hole attack minimization for ad-hoc networks. *IEEE Trans. on Mobile Computing (TMC)*, 16(8):2174–2183, 2017.

[87] J. Sen, M. G. Chandra, S. Harihara, H. Reddy, and P. Balamuralidhar. A mechanism for detection of gray hole attack in mobile ad hoc networks. In *Proc. of Int'l Conf. on Information and Communications Security (ICICS)*, pages 1–5, 2007.

[88] S. Shahabi, M. Ghazvini, and M. Bakhtiarian. A modified algorithm to improve security and performance of aodv protocol against black hole attack. *Wireless Networks*, 22(5):1505–1511, 2016.

[89] E. A. Shams and A. Rizaner. A novel support vector machine based intrusion detection system for mobile ad hoc networks. *Wireless Networks*, pages 1821–1829, 2017.

[90] S. K. Shandilya and S. Sahu. A trust based security scheme for rreq flooding attack in manet. *Int'l Journal of Computer Applications (IJCA)*, 5(12):4–8, 2010.

[91] H. Shen and Z. Li. A hierarchical account-aided reputation management system for manets. *IEEE Trans. on Networking (TON)*, 23(1):70–84, 2015.

[92] S. Shevade, S. Keerthi, C. Bhattacharyya, and K. Murthy. Improvements to the SMO algorithm for SVM regression. *IEEE Trans. on Neural Networks*, 11(5):1188–1193, 2002.

[93] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal (IoT–J)*, 3(5):637–646, 2016.

[94] Y. Shi, L. Xie, Y. T. Hou, and H. D. Sherali. On renewable sensor networks with wireless energy transfer. In *Proc. of Int'l Conf. on Computer Communications (INFOCOM)*, pages 1350–1358, 2011.

[95] T. Shu and M. Krunz. Privacy-preserving and truthful detection of packet dropping attacks in wireless ad hoc networks. *IEEE Trans. on Mobile Computing (TMC)*, 14(4):813–828, 2015.

[96] F. Simjee and P. H. Chou. Everlast: long-life, supercapacitor-operated wireless sensor node. In *Proc. of the Int'l symposium on Low power electronics and design (ISLPED)*, pages 197–202. ACM, 2006.

[97] P. K. Singh, R. R. Gupta, S. K. Nandi, and S. Nandi. Machine learning based approach to detect wormhole attack in vanets. In *Proc. of Workshops of the Int'l Conf. on Advanced Information Networking and Applications (AINA)*, pages 651–661, 2019.

[98] D. Subhadrabandhu, S. Sarkar, and F. Anjum. A statistical framework for intrusion detection in ad hoc networks. In *Proc. of Int'l Conf. on Computer Communications (INFOCOM)*, 2006.

[99] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Trans. on medical imaging (T-MI)*, 35(5):1299–1312, 2016.

[100] Y. K. Tan and S. K. Panda. Energy harvesting from hybrid indoor ambient light and thermal energy sources for enhanced performance of wireless sensor nodes. *IEEE Trans. on Industrial Electronics (TIE)*, 58(9):4424–4435, 2010.

[101] D. Tse and P. Viswanath. *Fundamentals of wireless communication*. Cambridge university press, 2005.

[102] F.-H. Tseng, L.-D. Chou, and H.-C. Chao. A survey of black hole attacks in wireless mobile ad hoc networks. *Human-centric Computing and Information Sciences (HCIS)*, 1(1):4, 2011.

[103] L. Van Der Maaten. Learning a parametric embedding by preserving local structure. In *Proc. of Int'l Conf. on. Artificial Intelligence and Statistics (AiStats)*, pages 384–391, 2009.

[104] M. Veerayya, V. Sharma, and A. Karandikar. Sq-aodv: A novel energy-aware stability-based routing protocol for enhanced qos in wireless ad-hoc networks. In *Proc. of Military Communications Conf. (MILCOM)*, pages 1–7, 2008.

[105] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proc. of Int'l Conf. on Machine Learning (ICML)*, pages 3540–3549, 2017.

[106] M. A. M. Vieira, C. N. Coelho, D. Da Silva, and J. M. da Mata. Survey on wireless sensor network devices. In *Proc. of IEEE Conf. on Emerging Technologies and Factory Automation. Proceedings (EFTA)*, volume 1, pages 537–544, 2003.

[107] K. Vishnu and A. J. Paul. Detection and removal of cooperative black/gray hole attack in mobile ad hoc networks. *Int'l Journal of Computer Applications (IJCA)*, 1(22):38–42, 2010.

[108] C. Wang, J. Li, Y. Yang, and F. Ye. A hybrid framework combining solar energy harvesting and wireless charging for wireless sensor networks. In *Proc. of Int'l Conf. on Computer Communications (INFOCOM)*, pages 1–9, 2016.

[109] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2004.

[110] B. Wu, J. Chen, J. Wu, and M. Cardei. A survey of attacks and countermeasures in mobile ad hoc networks. In *Wireless Network Security*, pages 103–135. Springer, 2007.

[111] L. Xiao, Y. Li, C. Dai, H. Dai, and H. V. Poor. Reinforcement learning-based noma power allocation in the presence of smart jamming. *IEEE Trans. on Vehicular Technology (TVT)*, 67(4):3377–3389, 2018.

[112] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *Proc. of Int'l Conf. on Machine Learning (ICML)*, pages 478–487, 2016.

[113] K. Xing and X. Cheng. From time domain to space domain: Detecting replica attacks in mobile ad hoc networks. In *Proc. of Int'l Conf. on Computer Communications (INFOCOM)*, pages 1–9, 2010.

[114] J. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proc. of Int'l Conf. on Artificial Intelligence (IJCAI)*, 2015.

[115] R. Zhang and C. K. Ho. Mimo broadcasting for simultaneous wireless information and power transfer. *IEEE Trans. on Wireless Communications (TWC)*, 12(5):1989–2001, 2013.

[116] H. Zhu, Z. Zhang, J. Du, S. Luo, and Y. Xin. Detection of selective forwarding attacks based on adaptive learning automata and communication quality in wireless sensor networks. *Int'l Journal of Distributed Sensor Networks (IJDSN)*, 14(11):1–15, 2018.