A CROSS-LAYER INTRUSION DETECTION SYSTEM FOR RPL-BASED INTERNET OF THINGS

RPL TABANLI NESNELERİN İNTERNETİ İÇİN KATMANLAR ARASI SALDIRI TESPİT SİSTEMİ

ERDEM CANBALABAN

ASSOC. PROF. SEVİL ŞEN
Supervisor

Submitted to
Graduate School of Science and Engineering of Hacettepe University
as a Partial Fulfillment to the Requirements
for the Award of the Degree of Master of Science
in Computer Engineering

2020

ÖZET

RPL TABANLI NESNELERİN İNTERNETİ İÇİN KATMANLAR ARASI SALDIRI TESPİT SİSTEMİ

Erdem CANBALABAN

Yüksek Lisans, Bilgisayar Mühendisliği
Danışman: Doç. Dr. Sevil ŞEN
Haziran 2020, 82 sayfa

“Nesnelerin İnterneti” (IoT), hem birbirine hem de İnternet’e bağlı kısıtlanmış cihazlardan oluşan heterojen bir ağdır. IoT’nin önemi son yıllarda önemli ölçüde arttığı için, bu alanda, özellikle bu tür karmaşık sistemlere uygun yeni mekanizmalar ve protokoller geliştirilmesi konusunda birçok araştırma yapılmaktadır. “Düşük Güç ve Kayıplı Ağlar için IPv6 Yönlendirme Protokolü” (RPL), IoT için kabul edilen yönlendirme protokollerinden biridir. Birden çok noktaya yönlendirme sağlar ve temel olarak “çok noktadan tek noktaya” (MP2P) iletişimin yanı sıra “noktadan noktaya” (P2P) ve “tek noktadan çok noktaya” (P2MP) iletişimi de destekler. Ancak, RPL, IoT için önerilen birçok protokolde olduğu gibi, ağ güvenliği ön planda tutularak tasarlanmamıştır. Bu nedenle literatürde RPL’nin güvenliğini sağlamak için bazı çözümler geliştirilmiştir. Ağdaki hareketleri izleyen ve izinsiz girişleri tespit eden “Saldırı Tespit Sistemleri” (IDS’ler), bu tür güvenlik sistemlerinin gerekli bir parçası haline gelmiştir, çünkü önleme mekanizmaları tek başına yeterli değildir. Bu tez çalışmasında, bu tür düşük güç kullanan kayıplı ağlar için yeni bir IDS önerilmiştir.

IoT, akıllı evler, akıllı araçlar ve tıbbi bakım gibi çeşitli uygulama alanlarında kullanılır ve her bir uygulama farklı güvenlik gereksinimlerine ihtiyaç duyar. IoT'deki cihazlar kayıplı iletişim ağları ile birbirine bağlanır ve sınırlı kaynaklarla bağlantılı olduğundan saldırılara açıktırlar. Ayrıca, bu tür cihazların iletişimi RPL tarafından sağlandığından, iç saldırganların hedefi haline gelebilirler. Bu tez çalışmasında, RPL'ye yönelik versiyon numarası saldırısı, en kötü ebeveyn saldırısı ve merhaba sel saldırısı gibi RPL'e özgü saldırılar detaylıca analiz edilmiştir. Ayrıca bu tür saldırıları tespit etmek için yapay sinir ağı tabanlı bir saldırı tespit sistemi önerilmiştir. Bu aşamada yönlendirme ve bağlantı katmanından alınan özniteliklerin etkileri araştırılmış ve hem ikili sınıflandırma hem de çoklu sınıflandırma uygulanmıştır. Önerilen sistem daha sonra farklı saldırı yüzdeleri (%2, %6, %10 ve %20) kullanılarak da değerlendirilmiştir. Çalışmanın sonuçları, önerilen sistemin ikili sınıflandırma için %96,88 tespit oranı, %0,13 yanlış pozitif oranı ve çoklu sınıflandırma için %97,52 doğruluk oranı ile saldırıları etkili bir şekilde tespit edebildiğini göstermiştir. Daha spesifik olarak, tespit oranının versiyon numarası saldırısı için % 93,2, en kötü ebeveyn saldırısı için % 98,17 ve merhaba sel saldırısı için %99,96 olduğu gösterilmiştir. Ayrıca, yapay sinir ağı eğitiminde bağlantı katmanıyla ilgili özniteliklerin kullanılmasıyla, yanlış pozitif oranın %0.61'den % 0.13'e düştüğü gösterilmiştir. Bağlantı katmanınından alınan özniteliklerin olumlu etkisi özellikle versiyon numarası saldırısının tespitinde gözlenmiştir. Bildiğimiz kadarıyla, bu çalışma literatürdeki ilk katmanlar arası saldırı tespit sistemini sunmaktadır.

Anahtar Kelimeler: Nesnelerin İnterneti, Saldırı Tespiti, Yönlendirme Saldırıları, RPL, Yapay Sinir Ağları

ABSTRACT

A CROSS-LAYER INTRUSION DETECTION SYSTEM FOR RPL-BASED INTERNET OF THINGS

Erdem CANBALABAN

Master of Science, Computer Engineering Department
Supervisor: Assoc. Prof. Sevil ŞEN
June 2020, 82 pages

The "Internet of Things" (IoT) is a heterogeneous network of constrained devices connected both to each other and to the Internet. Since the significance of the IoT has risen remarkably in recent years, a considerable amount of research has been conducted in this area, and especially on, new mechanisms and protocols suited to such complex systems. "Routing Procotol for Low-Power and Lossy Networks" (RPL) is one of the accepted routing protocols for the IoT. It provides for multi-hop routing and is mainly proposed for "multipoint-to-point" (MP2P) communication as well as supporting "point-to-point" (P2P) and "point-to-multipoint" (P2MP) communication. However RPL, as with many protocols proposed for the IoT, was not purposefully designed with security in mind;hence, certain solutions for securing RPL have been developed in the literature. Intrusion detection systems known as IDSs, which monitor activities in systems and detect intrusions, have become an inevitable part of such security systems, since prevention mechanisms alone are never enough. In this thesis study, a new IDS is proposed for these types of low-power and lossy networks.

The IoT exists in a variety of fields such as smart homes, medical care and smart vehicles and with each having different security requirements. Devices in the IoT are interconnected to each other over lossy communication links, but with limited resources, they are also susceptible to attacks. Moreover, since the communication of such devices is provided by RPL, the protocol could also be targeted by internal attackers. In this thesis study, specific attacks against RPL, namely “version number attack”, “worst parent attack” and “hello flood attack” were deeply analyzed. Then, an IDS based on neural networks was proposed in order to detect such attacks. Here, the effects of features taken both from the routing layer and the link layer, were explored, and both binary classification and multi-class classification applied. The proposed system was then evaluated on simulated networks using different percentages of attackers (2%, 6%, 10%, 20%). The study’s results showed that the proposed system was able to detect the attacks effectively with a 96.88% detection rate, a 0.13% false positive rate for binary classification, and a 97.52% rate of accuracy for multi-class detection. More specifically, the detection rate was shown to be 93.2% for “version number attack”, 98.17% for “worst parent attack” and 99.96% for “hello flood attack”. Also, with the usage of features related to the link layer in training, the false positive rate was shown to decrease from 0.61% to 0.13%. The positive effect of the link layer’s features was especially noted in the detection of “version number attacks”. To the best of the author’s knowledge, this study presents the first cross-layer IDS in the literature.

Keywords: Internet of Things, Intrusion Detection, Routing Attacks, RPL, Neural Networks

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my supervisor, Assoc. Prof. Sevil Şen, for her guidance, advice and criticism throughout the research. Her wisdom and support motivated me and gave me a strength. Without her patience and understanding, completing this thesis would have been almost impossible.

I would like to thank Wireless Networks and Intelligent Secure Systems Lab (WISE), especially Selim Yılmaz and Emre Aydoğan, for their support and excellent advice that facilitates progress throughout the thesis.

I am very grateful to my circle of friends who have been with me throughout my graduate life and created such a cheerful environment: Aysel Akgemci, Sevil Çalışkan, Negin Razizadeh, Ece Büber, Gonca Gencer, Baturay Özel, Berkay Baykara, Barış Özkuşlar, Furkan Doğramacı, Emek Barış Küçüktabak, Emre Yiğit Alparslan, Nami Çağan Yüksel, Hakan Mert and Cenk Tüysüz.

I also thank Aselsan Inc. for supporting my graduate study, and my coworkers for encouraging and motivating me throughout the preparation of this thesis.

I would like to share my deepest gratitude to Selin Şahin for her endless support, understanding and encouragement throughout my study. I am very lucky to have you, thank you!

Last but not least, I would like to thank my parents, Seval and Metin Canbalaban, and my big brother Ersin Canbalaban, for their endless love, trust and support in every step of my life. My success wouldn't have been possible, without their encouragement and love.

CONTENTS

TABLES

FIGURES

SYMBOLS AND ABBREVIATIONS

Abbreviations

6BR	**6**LOWPAN **B**order **R**outer
6LoWPAN	IPv**6** over **Lo**w-Power **W**ireless **P**ersonal **A**rea **N**etwork
ADAM	**ADA**ptive **M**oment Estimation Algorithm
CSMA	**C**arrier **S**ense **M**ultiple **A**ccess
CoAP	**Co**nstrained **A**pplication **P**rotocol
DAG	**D**irected **A**cyclic **G**raph
DAO	**D**estination **A**dvertisement **O**bject
DIO	**D**ODAG **I**nformation **O**bject
DIS	**D**ODAG **I**nformation **S**olicitation
DODAG	**D**estination **O**riented **D**irected **A**cyclic **G**raph
DoS	**D**enial-**o**f-**S**ervice
ETX	**E**xpected **T**ransmission **C**ount
GUI	**G**raphical **U**ser **I**nterface
HFA	**H**ello **F**lood **A**ttack
IDS	**I**ntrusion **D**etection **S**ystem
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
IETF	**I**nternet **E**ngineering **T**ask **F**orce
IoT	**I**nternet **o**f **T**hings
KDE	**K**ernel **D**ensity **E**stimation
LLN	**L**ow-Power **L**ossy **N**etworks
MAC	**M**edia **A**ccess **C**ontrol
MANET	**M**obile **A**d Hoc **NET**work
MLP	**M**ultil**a**yer **P**erceptron
MP2P	**M**ulti**P**oint to **P**oint
MRHOF	**M**inimum **R**ank with **H**ysteresis **O**bjective **F**unction
OF0	**O**bjective **F**unction Zero

OSI	**O**pen **S**ystem **I**nterconnection
P2MP	**P**oint to **M**ulti**P**oint
P2P	**P**oint to **P**oint
PDR	**P**acket **D**elivery **R**atio
ReLU	**Re**ctified **L**inear **U**nit
RFC	**R**equest **F**or **C**omments
RPL	**R**outing **P**rotocol for **L**ow-Power and Lossy Networks
SVM	**S**upport **V**ector **M**achine
TPR	**T**rue **P**ositive **R**ate
UDP	**U**ser **D**atagram **P**rotocol
VNA	**V**ersion **N**umber **A**ttack
WPA	**W**orst **P**arent **A**ttack

1. INTRODUCTION

With the development of technology, the usage of the Internet and smart devices has increased with popularity on a seemingly daily basis. Advances in smart sensors, embedded devices, and wireless communication technologies have led to the emergence of a new concept known as the "Internet of Things" (IoT). With the introduction of the IoT, smart devices which have low-levels of power and processing resources have become able to monitor, sense, and collect information from the environment without need for human interaction. This technology has been expanding exponentially, and now pervades many different areas such as the smart grid, medical care, smart home appliances, smart vehicles, as well as automation in factories and in the area of logistics [1, 2]. According to research conducted by Statista [3], the number of devices exist in the IoT environment will exceed 50 billion by 2023 and 75 billion by 2025, as illustrated in Figure 1.1. This state of affairs and predictions for the near future's adds significant importance to the role of research based on the IoT.

The rapid growth of the IoT, however, has also raised certain concerns about security. Many IoT applications collect large amounts of data from various devices, which presents a challenge for the protection of information. Moreover, most devices which connect to the Internet for specific purposes have resource constraints related to power (i.e., they are battery powered), connectivity, and also of their physical size. IoT networks consist of a variety of devices such as smart phones, wireless sensors, and wearable devices, and each with differences in their computing and communication capabilities. This also presents a challenge in terms of the development of complex security solutions. However, existing security solutions are not suited to such heterogeneous and complex networks and, therefore, new solutions need to be developed, or existing ones adapted in order to service this new environment.

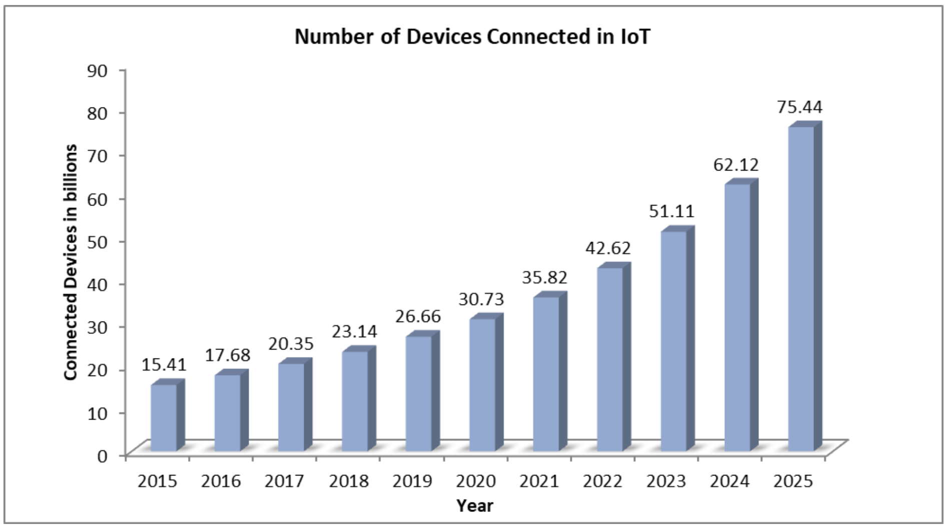

Figure 1.1. Number of devices connected in IoT

The traditional protocols for communication are also not suited to devices with limited resources in the IoT. In order to resolve this problem, new types of protocols, which are both less complex and consume less power, have been introduced for the IoT in the literature. The Internet Protocol, known as IPv6 is not suited to IoT networks. Therefore, 6LoWPAN was developed as an adaptation layer between the link layer and network layer for the IoT, and is a form of wireless network that sends packets and uses IPv6 over low power wireless personal area networks. 6LoWPAN, therefore, provides a communication environment among devices in the IoT and also the opportunity to access the Internet using border router. With the introduction of 6LoWPAN, devices which have constrained resources related to memory, battery power and computational power can be connected to the Internet by way of this simplified IPv6 structure. The protocol allows devices to perform their operations even if the scale of the network is considered to be large. The RPL was proposed by the IETF in 2012 [4]. The RPL was designed for the provision of efficient routing paths especially for resource-constrained devices which use a lower bandwidth, and have lower computational power and limited energy. The RPL protocol has the ability to adapt to changing network conditions, and can thereby provide alternative routes should an existing path become no longer available. However, even as one of the accepted protocols in the literature, it was not fully designed with security in mind. Therefore, considering that devices in the network have limited resources and the communication links are lossy, the RPL routing protocol may become under serious attack, and which can further affect the entire network. However, existing "Intrusion Detection Systems" (IDSs) proposed for wired and wireless networks may not be suitable for the IoT network, due to the complexity of IDSs as well as certain constraints of the devices themselves. It is therefore, important to understand the RPL routing protocol and to analyze the potential security attacks it may face, and also their downstream effects on the network. A need therefore exists to develop suitable security solutions for RPL-based IoT networks in order to prevent such attacks, wherever possible, and to detect and respond accordingly when these preventions fail.

This thesis study aimed to analyze specific attacks against RPL and to propose a solution in order to detect them. In developing a suitable IDS for RPL, it is important to investigate the behavior of networks under different types of attack. To obtain a detailed information of specific attacks against RPL, this study simulated attacks on networks with varying traffic patterns, with "version number attack", "worst parent attack", and "hello flood attack". Then, based on these analyzed attacks, an IDS based on neural networks was developed using both binary classification and multiclass classification. The proposed method was then evaluated on simulated networks with varying traffic patterns and attack types/attackers.

There are several studies that have analyzed attacks against RPL [5–8]. ; however, the majority of these have been conducted with just a limited number of devices [5, 8] . Realistically, at least 25 or 30 devices are needed in order to see the multihop characteristics of RPL[9]. Moreover, some of these studies have analyzed only a few attacks [5–8] , or more general types of attack which could be applied to any type of network. There has been no detailed and comprehensive study that covers attacks specifically against the RPL protocol. Therefore, in the current study, three attack types specific to RPL, namely "version number attack", "worst parent attack", and "hello flood attack" were analyzed based on realistic scenarios. According to the attack analysis, both the version number attack and hello flood attack degraded the network performance. In the version number attack, even if one attacker was used in the topology, both PDR and E2E delay were dramatically affected. In the hello flood attack, the network still continued to communicate with 60% "Packet Delivery Ratio" (PDR) for a 2% attacker density. However, in the cases based on 6%, 10%, and 20% attacker density, the network became nonfunctional. According to the analytical results, the "worst parent attack" was the least harmful in RPL, with RPL still operating for a 2% or 6% attacker density event, and even where there was no attack. However, for 10% and 20% attacker density, both PDR and E2E delay were negatively affected.

Intrusion detection systems have become inevitable parts of the modern security system. In the literature, there are security solutions that have been developed for RPL-based IoT networks, particularly intrusion detection systems, and which forms the main focus of this thesis study. However most of these are not suited to resource-constrained networks[10]. Moreover, an IDS may need to process a large amount of data which could be the result of an attack such as a DoS attack or due to the network condition itself. From this perspective, a neural-network-based model could be a good candidate for dealing with these larger amounts of data in order to find patterns as a means to distinguishing malicious network traffic from benign traffic. Therefore, the current study investigated the use of neural networks in order to detect attacks against RPL. There are few studies in the literature based on intrusion detection in RPL-based IoT networks that have utilized deep learning techniques [11, 12] . However, such studies have primarily focused on detecting a particular type of attack. In the current study, both binary classification and multiclass classification were employed. If the type of a detected attack is known, specific response mechanisms can be triggered in order to limit the damage inflicted. Furthermore, this helps to decrease the manual analysis time required following malicious behavior detection. Moreover, an extended feature set was employed in this study, with features extracted from both the link layer and the network layer. Effects of features collected from the link layer were explored first in this study, then a cross-layer IDS was proposed and evaluated. According to the results, link-layer features showed the most significant improvements on the detection of the version number attack type, and features collected from the link layer were also shown to help decrease the false positive rate.

This thesis is organized as six chapters. Whilst Chapter One briefly introduces the thesis, Chapter Two presents background information about the IoT and its fundamentals, RPL, routing attacks taxonomy for RPL, and Intrusion Detection Systems for IoT. Chapter Three then discusses other studies in the literature on attack analysis against RPL and on the detection and mitigation of routing attacks. Chapter Four explains the routing attacks analyzed by this study in detail, and also introduces the proposed method for attack detection, whilst Chapter Five details the simulation environment and the created dataset, followed by the experimental results that include routing attack analysis and the performance evaluation of the proposed IDS. Chapter Six concludes the study by highlighting the important results and outlining the study's contribution to the literature, and the author's future research plans.

2. BACKGROUND

This chapter provides background information related to the protocols used in the IoT such as 6LoWPAN and RPL. Existing routing attacks against RPL topology and IDSs for detecting such attacks are then introduced, with IDSs then explored under different classifications.

2.1. Internet of Things

IoT is a network type that consists of objects or devices connected to the Internet or to each other. These objects or devices communicate and share information with each other, gather data from the environment, and are connected to each other by way of an IP protocol. IoT devices can be a simple light bulb, a thermostat, or more complex devices such as a smartphone or personal computer. In the IoT network, devices usually have limited resources with low energy and low-rate wireless communications. The IETF designed a new communication and security protocol that matched the constraints of the IoT network and supports certain future uses. New standardized solutions work together with existing Internet standards, guaranteeing that IoT devices can communicate with other Internet entities without encountering problems. The protocol stack created by the IETF is presented in Table 2.1.

Table 2.1. Protocol stack in IoT

OSI Layer	**IoT protocol**	**Contiki Implementation**
Application	CoAP	CoAP
Transport	UDP	μIP
Network	RPL	ContikiRPL
Adaptation	6LoWPAN	SICSLoWPAN
Data Link	802.15.4 MAC	ContikiMAC
Physical	802.15.4 PHY	Contiki 802.15.4

For the "physical" (PHY) layer and "media access control" (MAC) layer, low-rate communication is standardized and supported by IEEE 802.15.4 [13]. The protocol sets the rules for communications at the bottom of the protocol stack and provides the basis of IoT communication protocols for higher layers. As an adaptation layer, 6LoWPAN uses IPv6 over a low-power WPAN, with its structure and methods gathered mainly under three standards, namely RFC (acronym for “Request For Comments”) 4919 [14], RFC 4944 [15] and RFC 6282 [16] as defined by the IETF. These standards define in detail the packet architecture, packet fragmentation and reassembly, and other functionalities of the protocol. The routing mechanism for IoT environments is defined by the RPL [17], and can support various link layers used in limited-resource devices. The protocol can create optimized network routes and efficiently adapt the topology to the network. The “Constrained Application Protocol” (CoAP) was designed as an Internet application protocol for constrained devices, with the details of the protocol defined in RFC 7252 [18]. CoAP was designed not only for use between devices on low-power and lossy networks, but also for use between constrained devices and general nodes of the Internet.

As previously mentioned, IoT networks contain numerous devices that vary based on their resources and on the capability of the node itself. However, with these limitations, securing the network presents a serious challenge. The devices in the IoT are vulnerable to both external and internal attackers, originating either from the Internet or the network itself. For the traditional network, various security solutions exist such as intrusion detection systems, trust mechanisms, mitigation techniques, as well as additional precautions over existing protocols. However, most of these methods are not considered to be suitable for the lossy and resource-constrained environment of the IoT. Since the primary aim of the current study is the proposal of a novel intrusion detection mechanism explicitly for the IoT network, the study’s focus is directed towards existing intrusion detection systems. Existing IDSs are not usually considered suitable, both due to their complexity and for being considered heavyweight. In order to propose a lightweight and effective security solution for the IoT, it is necessary to gather sufficient information related to the protocols used. Since the focus of this study is on the routing topology, both the 6LoWPAN and RPL protocols are investigated in detail, as shown in the following subsections.

2.1.1. 6LoWPAN

“6LoWPAN” is a low-cost, low-power communication network which connects wireless devices to each other using a compressed form of IPv6. The protocol defines the IPv6 header compression and specifies the routing details of the packet to the border router of the network using IEEE 802.15.4 protocol at the link layer and PHY layer.

6LoWPAN networks use multihop routing to communicate, with each node forwarding another node's packet to the "6LoWPAN border router" (6BR). These networks are then connected to the Internet through the 6BR, which handles the compression/decompression headers and fragmentation/assembly of the IPv6 messages while connecting to the Internet. A sample network structure is illustrated in Figure 2.1. One of the constraints of these devices in the network is their energy requirements. In order to keep power consumption to a minimum, the duty cycle system is employed during communication over the network. Thereby, for most of the time the device turned off its power. In order to receive and transmit messages, it turned on its power for short periods of time. Through this approach, devices with low-level energy storage can work for much longer durations.

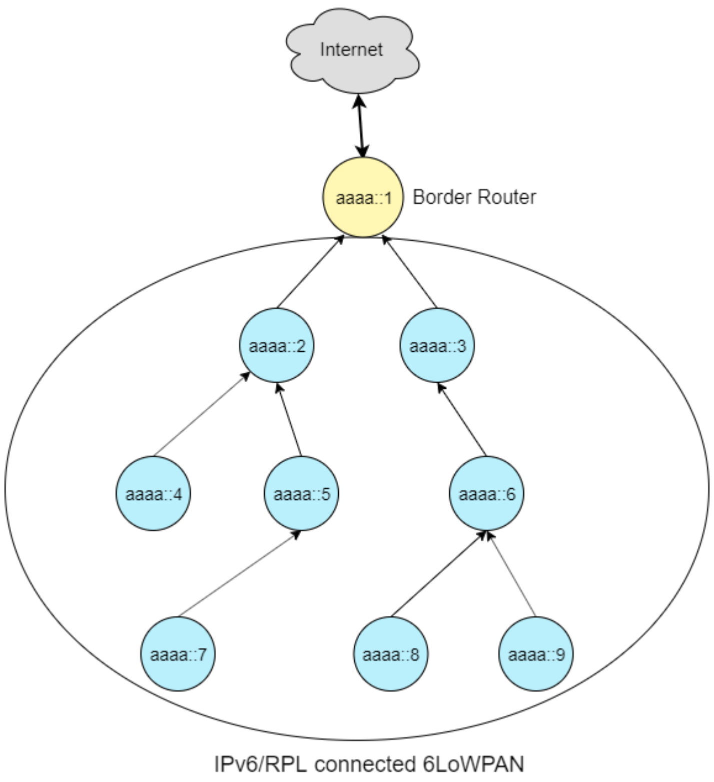

Figure 2.1. An IoT network with IPv6/RPL connected 6LoWPAN

Since 6LoWPAN is derived from the IPv6 Internet Protocol, it also has the same inherent weaknesses as the standard Internet protocol. The capability and complexity of IoT devices are also limited compared to the standard devices using the IPv6 protocol, which generates additional concerns in terms of securing the IoT network.

2.1.2. RPL

RPL is a standard routing protocol which can be used on top of link-layer mechanisms such as IEEE 802.15.4 PHY and MAC layers in IoT. The definition of the protocol and its standards are explained in RFC 6550. RPL topology supports different traffic types such as MP2P, P2MP and P2P. Especially, sensor networks where each node sends measurements to a central node periodically, can use the RPL to create routing paths in the concept of IoT. RPL was specifically designed for low-power and lossy networks which have limited resources such as bandwidth, computational ability, as well as energy. The protocol has an adaptive characteristic to varying conditions and can offer alternative paths when the existing path is no longer available.

RPL connects nodes to each other and to the 6BR border router by creating a DODAG. RPL protocol defines three different node types. The first is a "low power and lossy border router" (LBR), which is the root of DODAG which is a collection point for the multipoint-to-point network traffic, and can create a "Directed Acyclic Graph" (DAG). It also provides a connection between the Internet and the remaining nodes. The second is a "router", which is a device that generates data traffic and forwards packets in the network. Whilst unable to create a DAG, the router can join an existing DAG using topology messages. The third is a "host," which only generates data traffic, and can therefore be labeled as an end-device. Each node in a DODAG has a node ID which is the IPv6 address of the node, a list of neighbors, a parent node, and a rank value which indicates the relative distance of node itself to the 6BR border router. The rank of a node strictly decreases towards the DODAG root whereas it increases from the root towards the leaf nodes. The nodes on the top of DODAG have smaller ranks, whereas the bottom nodes have larger rank values. The Root Node has the smallest rank within the network.

In RPL there are two different modes in the route discovery phase. The first is the "non-storing mode," with each packet containing the complete path that the packet will follow through the network. The forwarding nodes do not keep the routing information, therefore, 6BR is only node who keeps that information. The second is the "storing mode," with each node maintaining a list of routing information for nodes in its subgraph. When a node gets a message, the node sends it to the next hop if the destination node is in the list. However, if the node is not in the list, the packet is forwarded to the parents. The rank is used in the topology in order to prevent construction of routing loops, and thereby allows nodes to identify the parent and child nodes in the neighborhood. Also, each node stores a list of neighbors and can use such lists in order to find a new parent should the connection be lost with the currently selected parent. The route is calculated according to the parameters such as energy resources, throughput, latency etc. To create a DODAG, each node decides its parent according to the better quality of path towards the root. The node selects the parent based on the route offers collected from its neighbors and at the end, each node has its best route to the root.

RPL uses three different control messages, namely DAO, DIS, and DIO in order to maintain the topology. Each node announces the route to its destination to the root by sending DAO control packets. The DAO message is propagated in an upward direction within the DODAG topology, via the parent of each node and border router, to become aware of the path to each node, with the help of the DAO message. DIS helps new nodes to ask for topology information prior to joining the network. Each new node sends a DIS message to its neighborhood. Once the neighbors reply with a DIO message, the node selects the best neighbor as its parent and joins the network. DIO helps to set and update the topology. A DIO message is sent by each node in order to inform other nodes about its routing condition, such as version number, rank, and Objective Function (OF). Rank relates to the quality of the path to the root. Each node in the network calculates its rank by using Objective Function. Objective Function defines how a node uses different metrics in order to calculate the rank value and how to select the best route in a DODAG. In order to calculate rank, the Objective Function can specify different parameters by considering routing metrics such as delay, link quality and connectivity. It is important to realize that rank of a node does not necessarily relate to the physical distance of the node to the root. For example, if load balancing is the most important parameter, even if the potential parent node has the closest distance to the root, the node can select another node as its parent if it has lower load potential at that time. When the parent or the rank of the node changes, it has to send the updated information in the next DIO message. In order to prevent a loop in the network, RPL organizes the network such that parents are always lower ranked than their children. The operation of a node can be seen as illustrated in Figure 2.2.

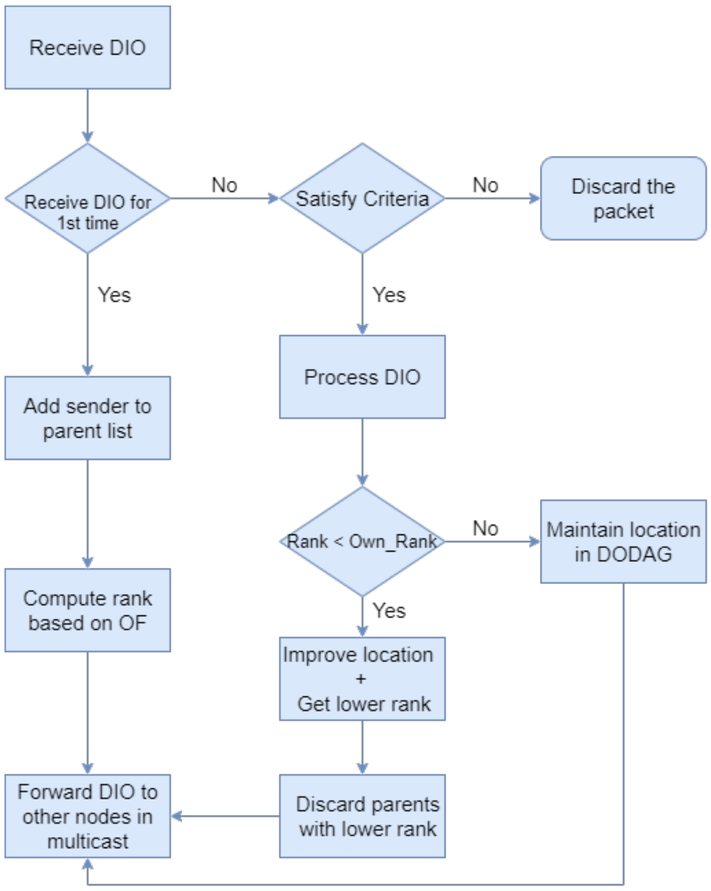
Receive DIO
Receive DIO for 1st time
No
Satisfy Criteria
No
Discard the packet
Yes
Yes
Add sender to parent list
Process DIO
Rank < Own_Rank
No
Maintain location in DODAG
Compute rank based on OF
Yes
Improve location + Get lower rank
Forward DIO to other nodes in multicast
Discard parents with lower rank

Figure 2.2. Operation of a node in DODAG

RPL protocol suggests two repair mechanisms: local and global. If there is a failure related to a node, link, or routing topology, a local repair mechanism is started in order to select a new parent for the affected nodes. For a node which loses connection with its parent node, it already has a list of its neighbors, as previously mentioned, so the node checks the list for an alternative parent. If there is no alternative parent node, it can use a neighboring node with the same rank value in order to forward packets. If the local repair mechanism is unable to provide stability to the network, a global repair mechanism can be used for this purpose. A global repair mechanism is controlled by the Root Node with the help of the DODAG version number. Once the Root Node decides to initiate a global repair, it simply increases the DODAG version number which is located in the DIO and then broadcasts the message. It is transferred through the network and received by the remaining nodes. When a DIO message with an incremented version number is received by a node, the node starts the parent selection algorithm and refreshes the parent node.The triggering condition of the global repair mechanism is left to the implementation according to the RFC document [17]. Therefore, a user should define a global repair mechanism condition and implement it within the RPL topology. For example, the DODAG root can perform a global repair mechanism if the number of local repair requests exceeds a specified threshold value.

In order to use resources effectively, the RPL employs the trickle algorithm [19] to adjust DIO message frequency. In order to decrease the number of topology messages, each node has a trickle time and DIO counter parameters. Trickle time parameter relates to the time interval that the node waits before sending the next DIO message. If the parameters which cause a topology change in the network are not modified in the incoming DIO message, then the DIO counter will be increased accordingly and the trickle timer duration of the idle state increases. When DIO message has a change which causes a trickle time to reset, the node will reset the DIO counter and minimize its trigger time. With the help of this mechanism, the number of DIO messages decreased and the network resources are used more efficient by decreasing the overhead in the network.The trickle timer is reset when there is an inconsistency found in the network. The detection of routing loops, joining of a node to the network, and changing rank values of nodes due to mobility are considered as inconsistencies whereby the trickle timer would be reset.

The RPL protocol is exposed to a large variety of attacks. Since "Low-Power Lossy Networks" (LLNs) have resource-constrained nodes, as well as limited physical security with unreliable links, they are considered vulnerable and difficult to protect against attack. The RPL protocol defines several security mechanisms, with integrated local and global repair mechanisms, loop avoidance and loop detection techniques. However, the protocol still has inherent weaknesses, especially in terms of insider attackers, and therefore requires the provision of additional security mechanisms.

2.2. Routing Attacks against RPL

The RPL is vulnerable to different kinds of routing attack, since it does not take security into consideration. Moreover, IoT devices often have limited resources and unreliable links, and IoT networks consist of numerous different devices such as smartphones, wireless sensors, and wearable devices with varying resources. Therefore, a security mechanism is required that can be applicable for each type and variation of device. In other words, any security solution proposed for such a system should be lightweight in terms of its computational and communication requirements, due to the known resource constraints of IoT devices and wireless links.

Even though the RPL has certain mechanisms such as local and global repair mechanisms, minimum rank increases limitations, as well as loop avoidance and loop detection techniques in order prevent malicious attempts and to recover the network, attackers can still circumvent these mechanisms. There are different kinds of attacks in RPL. While some are specific to this protocol, others are of a common type such as “spoofing attacks” and “denial of service attacks” which are known to exploit the shared vulnerabilities of both wired and wireless networks. Attacks made against RPLs are classified according to the target resource of the attack, and fall into one of three different categories, namely “resources,” “topology,” and “traffic” [20] as illustrated in Figure 2.3.

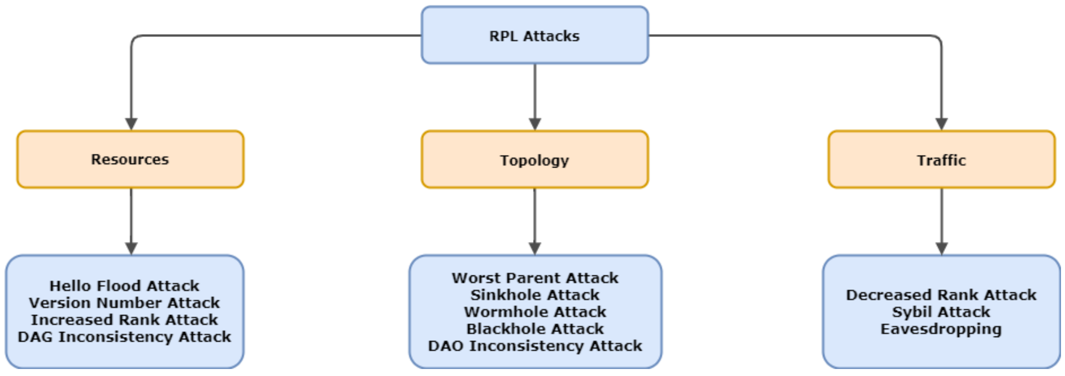

Figure 2.3. Classification of RPL attacks

Resource attacks mainly aim to consume resources of legitimate nodes, and/or the network, and result in poor network performance. These attacks can significantly accelerate the depletion of a node’s battery power, utilize the memory of nodes, and cause a lag to the remaining necessary operations. Such attacks can place the network in an unstable position and thereby shorten the lifespan of the network. The “hello flood attack” is one such attack in this category, whereby the attacker sends frequent DIS messages to the neighborhood to force them to reply with a DIO message. As a result, it increases the routing overhead by adding numerous routing control messages into the network, and by doing so exhausts the resources of nodes receiving periodic DIS messages from an attacker node. In the “version number attack,” the attacker changes the version number of the topology illegitimately by increasing the corresponding field of the DIO messages prior to sending it to other nodes. This action requires the rebuilding of the whole DODAG, and thereby results in a high number of routing control messages in the network, which may in turn cause congestion in the network. Another attack type which targets network resources is the "increased rank attack," in which the attacking node increases its own rank value and sends this malicious rank value in a DIO message as if it has a higher rank. In doing so, the attacker aims to create loops in the network, forcing other nodes to start local repair mechanisms in order to remove these loops. It is considered a resource attack since it overconsumes node energy and results in congesting the RPL network. The "DAG inconsistency attack" is detectable when there is an inconsistency between the direction flag and the rank of the Sender Node. Normally, if the direction of the received packet is downward, it should come from a node which has a lower rank value, and vice versa. When a node detects this inconsistency for the first time, it sets a flag for inconsistency in the packet. However, should this situation occur for a second time whilst forwarding a packet, the packet is dropped and the trickle timer is reset. If a malicious node modifies the flags deliberately while forwarding a packet, other nodes will subsequently detect that as an inconsistency and thereby drop the packet. However, since the nodes also reset the trickle timer, the frequency of the routing control messages increases and thereby causes additional network overhead.

RPL attacks can also aim to disrupt the topology of the network. The first attack in this category is the "worst parent attack," which is a specific type of rank attack. An attacker chooses a neighbor node which has the worst rank value as its parent by using the Objective Function. This selection affects the nodes as they select a path towards the Sink Node that now includes the malicious node. In a "sinkhole attack," an attacker attracts traffic by advertising itself as if it has a better link quality than it does in reality, or a shorter distance to a Sink Node. In doing so, it may be able to falsely place itself in the path. After that, messages could be dropped instead of being forwarded, or falsely modified, resulting in a disruption to the network communication. The "Wormhole attack" is another type of topology attack, whereby two or more malicious nodes create a tunnel between each them and packets are sent through this tunnel. Even though these malicious nodes could be physically distanced some way from each other, they can pretend to be very closely situated using the tunnel, which is usually achieved using a wired communication. Therefore, other nodes send their messages to the root via these malicious nodes, and the attack can thereby create non-optimized routes within the network. In a "blackhole attack," the attacker drops packets that pass over it, instead of forwarding them to the Root Node. Therefore, if an attacker is positioned in a strategic location, such as being a neighbor of the Sink Node, it could effectively isolate most of the legitimate nodes from the network.

The last category covers attacks targeted at disrupting network communication. The main aim in this category is to attract network traffic to a specific node. The first of these is the “decreased rank attack,” in which a malicious node claims a lower rank value than its real value in the DODAG structure in order to attract more traffic to itself. Through this behavior, many benign nodes may connect to the Sink Node using the attacker. Then, the malicious node, which takes control of the network traffic, could then start to drop or modify packets. In a “Sybil attack,” a malicious node creates fake identities to participate in network operations as a legitimate node. For instance, an attacker can create a Sybil identity in order to behave/appear like a Root Node by using its address and thereby manage the network traffic. The last of these attack types is the “eavesdropping attack,” in which an attacker node can passively listen to the network in order to illegitimately obtain information about the network such as its topology, frequency of data packets sent etc. This attack presents a significant risk when data of a confidential nature is transmitted, and has not been subjected to some form of encryption.

2.3. Intrusion Detection Systems (IDSs)

IDS is a security mechanism used to detect attacks against a network by analyzing the activity based on different methods. When an IDS detects malicious activity, it warns the authorities about the attack. Intrusion detection systems can be classified under three main areas according to its “detection resource,” “detection technique,” and “detection architecture”; as illustrated in Figure 2.4.

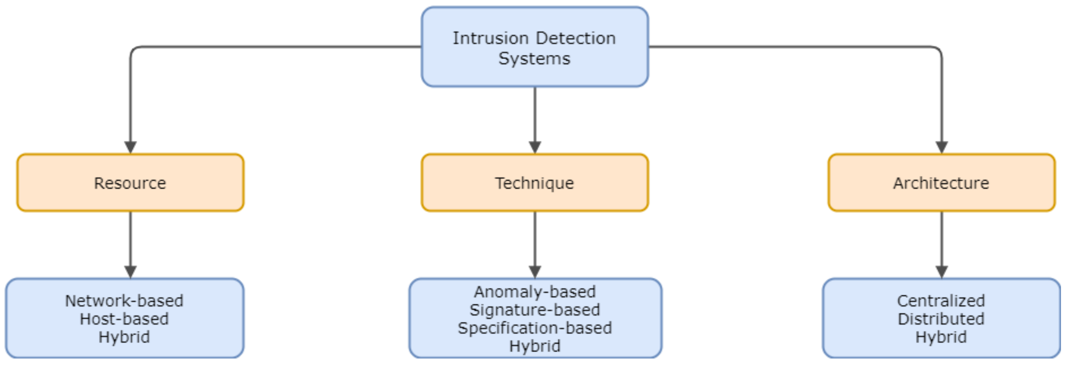

Figure 2.4. Classification of intrusion detection systems

2.3.1. IDS Classification with Detection Technique

Detection techniques can be classified according to three classes, namely "signature-based," "anomaly-based," and "specification-based." The anomaly-based and signature-based techniques are the most popular according to the literature. "Anomaly-based systems" detect malicious activities in the network by first determining the ordinary behavior of the network, and then comparing the current situation with this baseline. Where the current situation deviates from the baseline more than a predefined threshold, the system raises an alarm. Anomaly-based systems can detect all types of attack and also adapt to new types of environment, and are therefore able to detect new attacks. However, these systems can also produce high false positive rates. In certain conditions, even where there is no attack in the network, the system may raise a false alarm due to small deviations from the baseline. "Signature-based" systems create a set of signs which refer to a network's behavior according to certain types of known attack. The system uses a database of these signatures and compares the current behavior of the network with these signatures. If there is a correlation between the behavior of the network and these signatures, the system rises an alarm. Signature-based IDSs can produce high "true positive rates" (TPR) for known attack types. However, if the network is faced with a new type of attack, the signature-based IDS will not be able to successfully detect the attack since it has no predefined signatures for such an attack. Therefore, these systems can create a large number of false negatives. "Specification-based" systems observe the legitimate behaviors of the existing protocol (i.e., the RPL protocol in the current study), and uses the protocol specifications in order to detect the attacker activity. If a node does not follow the protocol's specifications, the system raises an alarm for the node. This approach is deemed effective in detecting topology attacks since it can detect malicious activities directly when a node breaks predefined protocol rules. In these systems, the development of these specifications and the coverage of the predefined rules are highly critical to the system's effectiveness and performance as an IDS, and can also be time-consuming for the user to define/update these rules. "Hybrid" techniques combine more than one type of detection system, by considering the advantages and disadvantages of each and then aiming to maximize the combined benefit of each of these systems. However, hybrid systems are usually considered to be higher in cost in terms of their computational, storage, and energy requirements.

2.3.2. IDS Classification with Detection Resource

IDSs are also classified according to the resource used for the detection algorithm and they are named as "network-based" and "host-based". "Network-based systems" are placed at a strategic point in order to monitor the network traffic. The system analyzes the passing traffic through the network and checks it based on the detection method. Network-based systems can use the obtained general view in order to boost its detection performance. However, these systems do not possess information related to individual resources' consumption or the logs of the nodes related to the environment, which can be a critical disadvantage whilst detecting specific types of attack. However, "host-based systems" can only observe the traffic received and transmitted by the host node. Since host-based systems do not possess general knowledge about the overall network, they may miss important information which can only be observed from a more global perspective. "Hybrid" systems attempt to combine the advantages of both network-based and host-based systems in order to make use of both network and node-based information.

2.3.3. IDS Classification with Detection Architecture

The architecture of IDS systems can be grouped as being either "centralized" or "distributed" systems. "Centralized systems" deploy a monitoring tool to a central location and then secure the network from this point. Especially in resource-constrained networks, not all nodes in the network may have adequate resources to operate an additional security system. Therefore, more powerful nodes such as the border router can run intrusion detection algorithms. There are two main options for centralized intrusion detection systems. First, a centralized device can monitor the network activity which passes through it. This approach is best suited to a wired network since the IDS can be placed at border routers from where all network traffic can be monitored. However, for some types of wireless network in which network traffic cannot be monitored at a central node, such as in “Mobile Ad Hoc Networks” (MANETs), the system may not have sufficient information to make an informed decision. In the second method, the central node collects information from other nodes, with some or all nodes sending periodic messages to the central node, and decisions are then made based on this collected information. In such a case, the communication overhead is the most significant disadvantage. Since, the nodes need to send monitoring information to the central node, the number of packets being sent through the network may increase significantly with this approach. If malicious nodes then block the monitoring data from reaching the centralized IDS, the performance of the IDS will likely drop significantly as a result. “Distributed systems” can utilize every node in the network, with each node working as an intrusion detection agent. Distributed IDSs are therefore carefully designed since not all nodes in the network may have adequate resources to run these types of IDS. If the nodes share information with each other, the system overhead increases, which can then cause communication problems in the network. This approach has no communication overhead if the system is not collaborative. In this case, each node makes its own assessment according to its own observations. However, the IDSs are required to use only localized information when analyzing the network situation, which can then increase the instances of false alarms being raised. The “Hybrid system” aims to utilize the advantages of both centralized and distributed systems. For example, the “cluster-based” approach can be classified as a type of hybrid architecture. In this approach, the network is divided into separate clusters in order to decrease the overhead. For each cluster, just one cluster-head is selected, with each member of the cluster gathering information from its neighbors and sending it to the cluster-head. Then, the cluster-head makes the final decision on the action to be taken according to the network situation. In certain cases, cluster-heads can also communicate with each other; however, that may also increase the overhead. There are some limitations in IoT networks which should be considered when designing an IDS. Since signature-based detection uses predefined conditions for different types of attack, the storage requirements for such an approach could be considered a problem in a network which has constrained nodes. Also, since signature-based systems use the signatures of already known attacks, they cannot by default detect unknown attacks. However, if an anomaly-based detection system is used, the computational costs will increase to a point where it can negatively affect the operation of the nodes, and deplete their limited energy resources. Moreover, it is difficult to define what constitutes the “normal behavior” of a network, especially if there are mobile nodes in a particular network. Therefore, false positive rates may also increase in anomaly-based systems. For specification-based systems, the protocol rules of the network are used. Development of these specifications and coverage of the created rules are highly critical and can be considered time-consuming for the user to define them. Also, in RPL topology, there are certain undefined events such as the global repair mechanism and default parameters which can be adjusted according to the implementation, and which thereby create a weakness in the specifications.

3. RELATED WORK

Traditional intrusion detection systems are not suited to IoT networks due to the connection of resource-constrained devices and lossy communication links. Therefore, suitable, lightweight intrusion detection algorithms should be developed, or existing systems adapted to this new environment. In the literature, some studies have proposed solutions for the detection of attacks against RPLs. However, traditional attack types such as DoS, Sybil, and other new attacks have emerged that target the RPL, especially its route discovery and route maintenance mechanisms such as the "rank attack" and "version number attack". Researchers have primarily focused on detecting these specific attacks in the RPL environment, whilst some studies have analyzed the effects of these attack types on the network. This thesis chapter reports on the various intrusion detection proposals for RPL to be found in the current literature.

3.1. Related Studies on Intrusion Detection for RPL-based IoT Networks

In order to secure RPL-based IoT networks, it is important to employ a type of security mechanism which protects the network from both internal and external attackers. As previously discussed, there are several techniques which have generally been used to secure these networks such as intrusion detection systems, mitigation methods, and trust-based evaluations; however, the current study primarily focuses on the analysis of intrusion detection systems for RPL-based IoT networks. Many IDSs have been proposed for both wired and wireless networks, but the main problem is that most of the traditional IDSs are not suited to low-power and lossy IoT networks. Therefore, the research community has had to refocus its attention to propose lightweight security solutions that specifically target the RPL-based IoT network.

SVELTE [21] was the first IDS proposed specifically for the IoT, with the authors having proposed a solution especially for the sinkhole attack and selective forwarding attack. Their proposed solution consisted of three phases: "6LoWPAN mapper," "intrusion detection," and a "distributed mini firewall." In the 6LoWPAN mapper phase, one node (i.e., border router) was selected as a mapper in order to visualize the constructed DODAG with the help of periodic messages gathered from each node. In this phase, Sink Node requests the following information from each node: node ID, rank, parent ID, and neighbor information of each node, and then uses this collected information for the purposes of intrusion detection. In the intrusion detection phase, inconsistencies in the network are discovered by checking violations of predefined intrusion rules. For example, since the 6LoWPAN mapper gathers rank information from the nodes, the rank of each node is checked by comparing the information from its neighbor. If any inconsistency is noticed, the node is subsequently blocked from the network operations. The minimum rank increase rule is also satisfied by the node while broadcasting its own rank. If the difference between the rank of the node and its parent is less than the predefined value , the node is marked as being malicious. Distributed mini firewall is another mechanism in the study and the malicious traffic coming from the external hosts is filtered by the system. In their study, the authors evaluated the model with a varying number of nodes, from eight to 32, as well as using lossless and lossy network environments in their test simulations. According to their published results, in the case of a lossless environment and eight nodes, the true positive rate was measured as being 100%. However, when the environment was reconfigured as lossy and the number of nodes was set to 32, the TPR decreased to 65-70%. Moreover, it was stated that false positives were found in their analysis, although the numeric results were not explicitly given in the published study.

In 2017, Medjek et al. [22] studied on a specification-based IDS equipped with both centralized and distributed architecture to counter a Sybil attack. In their proposed T-IDS (Trust-based IDS), the nodes in the network are also used to monitor the network and communicate with each other in order to detect abnormal activity, and then reports them to the border router. The authors examined the scenario of a Sybil attack against an RPL by defining an attacker model which was both mobile and steady. In the attack scenario, the attacker behaved as a honest node for a while, then chose a new location to move to and created a new identity once placed at the new location. Then, it broadcast DIS messages to its neighbors like a new node by advertising a different IP address. The attacker node repeated the same behavior periodically and for each attempt, it created a new identity with a different IP address and also a new location. In other words, the number of Sybil identities were equal to the number of moves of the attacker node. Simulations were conducted on a network using a total of 50 nodes, each with varying attacker numbers and Sybil identities by using the Cooja simulator. The authors analyzed the effects of the Sybil attack by using the following performance metrics: PDR, energy cost, and overhead. Then, they proposed a trust-based IDS for the Sybil attack within an RPL environment. In their proposed mechanism, an additional trusted entity was required to maintain the node authentication. Also, the nodes needed to send a message to the root when inconsistencies were detected, which brought about additional overhead. In their proposed IDS, each node in the network was equipped with a cryptographic coprocessor chip, which was used to build hardware support identification, store security parameters, and to handle the cryptography calculation. Therefore, the solution involved additional cost and power usage for the nodes. Simulation of the proposed IDS was, however, left as a piece of work planned for the future.

Napiah et al. [23] proposed a hybrid-based IDS which employed both anomaly-based and signature-based techniques, and utilized a 6LoWPAN compression header in order to detect hello flood attacks, sinkhole attacks, and wormhole attacks, and variations thereof. At the beginning, the packet compression header data was captured and analyzed in order to extract specific features to help distinguish malicious from benign activity. To select the most useful features, two different search techniques were employed: "best first search" and "greedy stepwise algorithm." Then, the "correlation-based features selection" (CFS) algorithm was employed in evaluating the features which shows the difference between normal and malicious behavior at mostFollowing the feature selection phase, the information extracted from the header was used to classify networks as "normal," "hello flood attack," "sinkhole attack," or "wormhole attack" by using six different machine learning algorithms ("logistic," "J48," "Multilayer Perceptron [MLP]," "naïve bayes," "Support Vector Machine [SVM]," and "random forest") using the WEKA tool. The proposed system then alerts the user when a malicious activity takes place. However, in the simulation, only eight nodes were used along with a single attacker, and it is therefore debatable whether the proposed system would be suited to a larger IoT network. Also, the number of samples used for training and testing consisted of unbalanced data using a small-sized dataset.

Yavuz et al. [12] studied a deep-learning-based IDS to address both the version number attack and hello flood attack against RPL-based IoT networks. The authors created a neural network model with five hidden layers. They used a feature set including reception/transmission rate, reception/transmission average time, received/transmitted packet counts, reception/transmission time, and DIO/DAO packet counts with a 1,000 millisecond window size. Simulations were run on networks varying from 10 to 1,000 nodes using the Cooja simulator. Their experimental results showed precision and recall of 94% for version number attack and 97% for hello flood attack. However, the false positive rates were not provided in the published study.

Muller et al. [24] proposed an anomaly-based intrusion detection system for version number attack and hello flood attack using the "Kernel Density Estimation" (KDE) algorithm.

The authors used a feature set which included a number of topology control messages (DIS-/DIO/DAO), different DODAG versions, and a "User Datagram Protocol" (UDP) forward ratio defined based on the number of transmitted and received packets. Simulations were conducted using a maximum of 12 nodes and a time varying from 800 up to 1,400 seconds. According to their experimental results, a 90% true positive rate was achieved for the hello flood attack and 96% for the version number attack, while a false positive rate of 0.5% was obtained on average.

In another study, Aydogan et al. [25] created an intrusion detection mechanism using a genetic programming technique. Their study reported the effectiveness of evolutionary-computation-based techniques by using a centralized IDS. The IDS showed high levels of accuracy with low false positive rates on the detection of hello flood attack and version number attack in RPL-based IoT networks. The authors also evaluated the same intrusion detection system in a distributed manner, reporting that for the hello flood attack especially, that the centralized IDS outperformed the distributed variant since it collected more information than the other nodes.

Kfoury et al. [26] suggested an IDS based on the "Self-Organizing Map" (SOM) neural network in order to cluster RPL routing attacks. The proposed method used self-organizing maps in order to cluster attacks and benign traffic. The authors used a number of topology control messages (DIS/DIO/DAO), the ratio of version number and rank changes, and the average power consumption of the nodes as features. However, details were not provided of the simulation parameters or the detection rate they found, only the visualization of the clusters is given in the reported results. Moreover, the mobility of the nodes was not considered as in other studies to be found in the literature.

In 2016, Le et al. [27] studied on a specification-based IDS for detection of worst parent attack and hello flood attack. They used simulation trace files generated by the Cooja simulator in order to generate the finite state machine for the RPL. A set of rules was created in order to check the network activity by using data gathered from the nodes; then, according to these rules, the system detected whether or not there was an attack. In order to decrease resource usage, the network was divided into clusters, with each cluster member reporting information about itself and its neighborhood to the cluster head. In turn, each cluster head then ran an IDS agent to analyze the information gathered from the cluster members; and where a node visited a state more than a predefined threshold of times during a certain interval, a threat alarm was generated by the IDS agent. In testing, the model achieved 100% for worst parent attack in terms of TPR; however, the FPR was reported to be around 5.5%. For the hello flood attack, the TPR decreased to around 94% and the FPR was 3% some 8 minutes after the attack started. After a period of 12 minutes on the network, TPR was reported as being 100%, but the FPR had increased up to 6%. Also, the power consumption in the network increased by 6.3% compared to the normal behavior of the network.

ELNIDS [28] was proposed in 2019, using an ensemble-based machine learning model to create an IDS against RPL-based attacks. The proposed model proved capable of detecting sink-hole attack, blackhole attack, Sybil attack, clone ID attack, selective forwarding attack, hello flood attack, and local repair attack. In their study, four different classifiers, namely "Boosted Trees," "Bagged Trees," "Subspace Discriminant," and "RUSBoosted Trees" were used and then evaluated using a dataset which consisted of all different types of attack; however, the detection accuracy at the attack type level is not specified in the study.

Very recently, the same authors, Verma and Ranga [29],conducted a survey which investigated existing security mechanisms for RPL-based IoT networks in the literature. Their survey reviewed more than 100 published studies, and reported that no study used a cross-layered security solution based on both link layer and network layer activities. Also, it was stated that no effective solution had been proposed for hello flood attack in the literature. According to their research, most of the published studies only used a few nodes whilst simulating their test IoT network, which may indicate a potential problem when the network size increases.

3.2. Related Studies on Attack Analysis for RPL-based IoT Networks

In order to propose effective security solutions for RPL-based IoT networks, it is critical to observe the effects of different RPL attacks on the network. Some attack types can be analyzed in other network types; however, some attacks such as version number attack, rank attack, and hello flood attack are specific to the RPL topology. Therefore, these attacks should be specifically and carefully analyzed in order not to overlook any behavioral actions noted during network monitoring.

In the literature, several studies have analyzed rank attacks; however, most of these have modified the rank value of the attacker nodes in order to simulate an attack. A malicious node could advertise a lower rank value than it actually has to attract traffic, and then use that traffic for different attack purposes such as dropping or modification. An attacker could also increase the rank value of itself and send this value with DIO messages. The aim here being to create loops within the network and to force the remaining nodes to start localized repairs in order to remove these loops. In the worst parent attack scenario, instead of modifying the rank value, the parent selection algorithm is modified in a way that an attacker selects the parent which has the worst rank value instead of the best rank value. As far as is known, the first implementation of the worst parent attack was performed by Le et al. [7] back in 2013, with simulations performed using 100 nodes on a grid topology and network performance analyzed by changing the location of the attacker. Their experimental results showed a security weakness found in the RPL so that the child nodes have to believe the routing information supplied by their parents via DIO packets and, the nodes have no other mechanism to verify the reliability of the parent nodes. Therefore, if the preferred parent is malicious, the performance of all nodes in the surrounding area of that node could be adversely affected, leading to increased overhead and number packet collisions, according to the study's results. In another study involving three of the same authors, Le et al. [30] showed the effects of different routing attacks on RPL-based networks. They performed four different type of attacks, namely worst parent attack, hello flood attack, local repair attack, and neighbor attack. Their experimental results showed hello flood attack to be the most influential in terms of degrading the IoT network's performance.

Mayzaud et al. [5] studied the effect of version number attack on the IoT network. Their simulations were conducted using 20 nodes and one attacker node located in a grid topology using the Cooja simulator. The nodes sent periodic data messages to the Sink Node every 20 seconds, with a simulation executed over a total of 50 minutes. The effects of the attack were investigated by changing the location of the attacker node to all possible locations in the network. According to the results, the effect of the attack increased when the malicious node was moved away from the Sink Node, since the attacker finds an opportunity to spread the damage.Moreover, it was observed that most of the time loops and rank inconsistencies due to the attack were located around the neighborhood of the malicious node, which can help in localizing the attacker. It was also shown that the unnecessary rebuilding of the DODAG as a result of the version number attack increased the number of topology control messages, which could in turn drain nodal resources. However, the study's simulations were only conducted on static networks with grid topology. The effects of the attacks were measured with the following performance metrics: overhead, PDR, and E2E delay. However, there was no analysis performed related to the power consumption of nodes.

Another recent study that investigated the effect of version number attack was proposed by Aris et al. [6], in which mobile and static nodes were placed on different network topologies using a probabilistic attacker model, with simulations conducted using the Cooja simulator. Both grid and random network topologies with 44 nodes were created, with simulations ran for 50 minutes, and where all nodes in the network sent periodic messages to the root every 60 seconds. However, only one attacker was used in the study's experimental testing. According to the analysis results, the effects of the attack increased when the malicious node was further away from the Root Node. Moreover, it was shown that mobile nodes harmed the network with the same impact of nodes located far away from the root. The authors also performed power consumption analysis for the network, and found that for the highest possibility of attacker, the average power consumption increased by 265%.

3.3. Other Studies

The literature also contains proposed security mechanisms other than IDSs in order to prevent and decrease the effects of RPL attacks. Trust mechanisms, mitigation techniques, and new rules in order to limit extraordinary changes in the topology are among the exemplar mechanisms to be found in the literature. These proposals are briefly introduced in this section.

Aris et al. [31] studied on a mitigation technique to prevent version number attacks. The mitigation architecture consisted of two different parts. First, nodes simply ignored the version number updates coming from the leaf nodes. Since legitimate version number updates should be performed using the Root Node via DIO messages, version number updates coming from leaf nodes can therefore simply be marked as malicious activity. However, this technique only mitigates attackers from limited locations. Second, the authors proposed a mitigation method that did not consider the position of the attacker. In order to accept a version number as being valid, most of the neighboring nodes with better ranks should validate the update of the version number. The proposed method was evaluated using the Cooja simulator on a network with 36 static nodes and one attacker. According to the results, it was possible to protect the network against version number attacks by employing two mitigation techniques, whilst performing legitimate version number updates. However, if the number of attackers increased, the malicious nodes were found to manipulate the voting scheme as they wanted. They also showed a trade-off between the resource constraints and mitigation performance, and in order to obtain improved mitigation performance, increased usage of resources was required.

In 2019, Verma and Ranga [8] proposed a mitigation method for DIS flooding attacks. The main idea was to set thresholds in order to restrict trickle timer resets and decrease the number of control message transmissions caused by hello flood attack. Simulations were performed in the Cooja network simulator with the number of nodes set to eight and 16, with a focus on overhead and power consumption. The simulation results showed a dramatic decrease in overhead when the proposed mechanisms were applied. For the nodal power consumption, there was a slight decrease noted when the model was employed; however, power consumption was still found to be much higher than for an attack-free RPL-based network. Furthermore, the number of nodes in the tested simulations was reported to be very limited.

There have also been a number of cryptographic mechanisms proposed for RPL-based networks. In 2011, Dvir et al. [32] studied on the VeRA (Version Number and Rank Authentication) application, which aimed to secure rank and version number with hash function, message authentication codes, and digital signatures. By using these mechanisms, the network was protected from illegitimate rank and version number updates. However, the system created traffic overhead and also increased computational power requirements which degraded the network performance. In 2013, Perrey et al. [33] studied on a mechanism called TRAIL, which was a generic scheme for topology authentication in RPL and detected and prevented inconsistencies in the network. This was achieved by using the nodes in order to validate their routes to the sink node, and also to detect rank spoofing, thereby detecting rank attack and version number attacks, as well as minimizing the overhead and nodal resource consumption.

In 2016, Glissa et al. [34] proposed a modified protocol called as SRPL, which placed a threshold on the rank parameter with the combination of hash chain authentication. The mechanism aimed to eliminate internal attackers that degrade the network's performance with illegal modification the rank parameter. In order to mitigate rank attacks, The protocol checks the number of times that a node increases or decreases its rank, and then limits them if necessary by setting thresholds. The evaluation was tested using the performance metrics of power consumption, network overhead, and packet reception rate. Simulations were conducted using 22 nodes in the Cooja simulator, with each simulation run for a period of 60 minutes. The authors then compared the number of topology messages used by SRPL with RPL, and it was noted that especially in the initial stage, proposed mechanism created an extra overhead. According to the experimental test results, the SRPL approach was able to detect an attack when a sharp change was seen in the rank value; however, may not be noticed when the rate of change was small.

Mayzaud et al. [35] studied on a distributed model to monitor the network and detect version number attacks and to identify those nodes involved in such malicious activities. New nodes were added to the RPL topology for the purpose of monitoring other nodes. Besides monitoring, these nodes also took part also in routing. These monitoring nodes then constructed a separate network and used that network to periodically forward information collected about the version number of incoming DIO messages to the root. Monitoring nodes were placed such that each node was directly monitored by at least one monitoring node. However, it was found that new nodes or mobile nodes might not be monitored if they are not in the transmission range of the monitoring nodes, and even where every node was monitored by one other node, FPR was reported to be 20%. As the coverage of the monitoring nodes increased, FPR was found to drop to 1%, as shown in a test where 66% of the network's nodes were monitored by two monitoring nodes. Furthermore, the proposed model was evaluated on a network with only one attacker.

There have also been studies which have provided trust-based mechanisms in order to provide network security. Airehrour et al. [36] proposed SecTrust-RPL, which was a detection and isolation mechanism developed to counter rank attack and Sybil attack. In their framework, nodes computed the trustworthiness of its neighbors based on two different metrics, namely direct and recommended trust. Each node chose parents with higher trust values for routing, whereas nodes with lower trust values were marked as malicious. In a rank attack scenario, while the packet loss rate of the nodes reportedly changed between 60% and 100% in the network, the SecTrust-RPL approach limited the packet loss to around 22%, and from 60-100% down to 15-25% for the Sybil attack.

One of the first RPL performance analyses was performed by Tripathi et al. [37] in 2010. Their study investigated the RPL performance of a network under both P2P and MP2P traffic, with a focus on the behavior of the trickle timer mechanism and its effects on network performance. Moreover, the effect of the usage of a periodic global repair mechanism with varying numbers of intervals against the loss of connectivity of the nodes in the network was studied. OmNET++ was used for the simulations, with the number of nodes set to 86.

No local repair mechanism was implemented during the simulations, therefore the nodes reportedly lost their connections for long periods in some cases.

3.4. Discussion

The related studies found in the literature on the security of RPL are summarized in Table 3.1.. There were a number of proposals found that aimed to detect malicious attempts in RPL-based networks, with these intrusion detection systems generally applying anomaly-based, specification-based, or hybrid techniques.

As can be seen from Table 3.1., intrusion detection proposals in the literature generally performed evaluations in a limited way, with most studies using a single malicious node in their simulations. Therefore, the effects of multiple attackers were generally omitted when evaluating the proposed solutions. A recent review study[29] also reported that most research in the literature used a low number of nodes in their experimental simulations. However, as pointed out in[9], at least 25 or 30 devices are actually needed in order to appraise the multihop characteristics of the RPL network. In these studies, simulations were usually only run for periods of up to 30 minutes at most. Considering the time required for a network to stabilize, this testing period is perceived to be very limited in terms of evaluating the real term effects of such attacks.

Most studies explored the effect following worst parent attack, version number attack, and hello flood attack RPL attacks; but again, the studies only employed a single malicious node in their simulations. The experiments have also always been conducted using a grid topology in order to see the clear effects of attackers from different locations. However, a scenario should more realistically include a distribution of nodes and attackers, and the partitioning of networks were also notably excluded in these studies. It was also noted that all except one of the published studies[37] used the Cooja network simulator in order to perform attack analysis on RPL.

Mobility was another important issue noted to affect the stability of the network. However, only one study [6] in the literature has found to have used a mobile attacker in their network simulation to show the effect of a mobile attacker on the performance of an IoT network. However, no studies were found that proposed a security mechanism that targeted a network with mobile nodes/attackers. Some of the studies proposed a security mechanism for RPL-based networks or analyzed routing attacks, however some were not implemented within a simulation environment, or at least detailed information related to the simulations were not provided [12, 26, 32], and some that claimed to be suited to RPL-based networks included no specific experimental results. Such studies should be recreated to include simulations using proper tools and known/validated metrics.

Although there are different types of solution proposed for securing RPL-based IoT networks, there have been no studies published in the literature that used a cross-layered security solution based on both link layer and network layer activities. This issue was also underlined in a recent review [29], which also stated that no effective solution has been put forward for the hello flood attack, which is one of the contributions of the approach proposed in the current thesis.

The reviewed studies show that more suitable solutions are needed in order to detect specific attacks such as version number attack, worst parent attack, and hello flood attack against RPL networks, and that offer more appropriate responses against such malicious attempts. These types of attack can cause significant harm to IoT networks if they are not detected and prevented. However, most of the solutions put forward in the literature have elected to focus only on detecting one routing attack at a time. However, in reality, whilst protecting the network from one type of attack, some other successful type of attack can place the network in an unstable position. Moreover, while the number of devices and attackers increases, the network can exhibit different responses to these varied attacks. However, most of the studies have opted to use a limited number of nodes and generally only one attacker in their simulated experiments. In the current study, the aim is to propose an IDS which can detect more than one attack type at the same time, and with different numbers of attackers within large and complex networks. Moreover, in another differentiation from previous approaches, the current study explores the application of a cross-layered intrusion detection system.

Table 3.1. Outline of the related studies

Proposed Solution	**Attack Type**	**Security Mechanism**	**Number of Nodes(Attacker/Total)**	**Simulation Time(min)**	**Traffic Pattern Type**
Raza et al. [21]	Sinkhole, Selective Forwarding	IDS(Hybrid)	1/8, 2/16, 4/32	10,20,30	Random
Le et al. [27]	Worst Parent, Hello Flood	IDS (Specification-based)	1/100	30	Random
Medjek et al. [22]	Sybil	IDS (Specification-based)	2/50, 4/50, 6/50, 8/50, 10/50	5.5	Random
Yavuz et al. [12]	Version Number, Hello Flood	IDS (Anomaly-based)	2/10, 4/20, 10/100, 100/1000	No info	No info
Muller et al. [24]	Version Number, Hello Flood	IDS (Anomaly-based)	1/12	13.3, 16.6, 20, 23.3	Grid/Random
Kfoury et al. [26]	Version Number, Hello Flood	IDS (Anomaly-based)	No info	No info	No info
Aris et al. [31]	Version Mumber	Mitigation	1/36	50	Grid/Random
Verma and Ranga [8]	Hello Flood	Mitigation	1/8, 1/16	5, 10, 15	Grid
Dvir et al. [32]	Version Number, Rank	RPL Improvement	No info	No info	No info
Glissa et al. [34]	Increased Rank, Decreased Rank	RPL Improvement	2/22	60	No info
Mayzaud et al. [35]	Version Number	Distributed Monitoring	1/20	10	Grid
Le et al. [7]	Worst Parent	Attack Analysis	1/100	No info	Grid
Le et al. [30]	Worst Parent, Hello Flood	Attack Analysis	1/50	5.5	Grid
Mayzaud et al. [5]	Version Number	Attack Analysis	1/20	No info	Grid
Aris et al. [6]	Version Number	Attack Analysis	1/44	50	Grid
Airehrour et al. [36]	Rank, Sybil	Trust-based	3/30	60	Random

4. PROPOSED INTRUSION DETECTION SYSTEM

This chapter describes the method used for analyzing and detecting attacks against RPL. First, the different types of attack addressed in this thesis are presented. Then, the attack dataset constructed for the purposes of training and testing the proposed cross-layer intrusion detection system is introduced. Finally, the proposed neural-network-based IDS is presented. In addition, the chapter details each of the features used for training the proposed detection system.

4.1. Target Attacks

The primary focus of the study is attacks that are specific to RPL-based IoTs. In the simulations, three attacks were used based on their potential effects on RPL: version number attack, worst parent attack, and hello flood attack, which are each detailed in the subsequent subsections.

4.1.1. Version Number Attack

In RPL, the version number is specified in DIO, and the version number change is triggered only by the Root Node where it is deemed necessary to apply a DODAG global repair. When the Sink (Root) Node changes the version number, this information is carried with DIO, and new DODAG is consequently reconstructed. In this attack scenario, a malicious node illegally changes the version number field before forwarding the received DIO message on to its neighbors. Unfortunately, RPL does not have a security mechanism which protects the protocol from such abnormal behavior. The “version number attack” results in unnecessary reconstruction of the DODAG graph, and thereby introduces unnecessary overhead to the network due to excessive routing control messages used in reconstructing the DODAG. Hence, rebuilding of the graph repeatedly results in the unnecessary usage of nodal resources and the network’s performance is decreased as a result. Moreover, it could create routing loops in the network that might result in packet loss.

4.1.2. Worst Parent Attack

“Rank attack” is one of the most dangerous attacks that aims to change the topology of a DODAG. A rank value is calculated by each node, and indicates the quality of a path between the node itself and the root of the DODAG. The rank value has some important roles in RPL such as creating an optimal topology, prevention of loop formation, and managing the overhead of routing control messages. Since RPL assumes that IoT network consists of reliable nodes and follow the specifications of itself, there is currently no mechanism defined to check for nodal misconduct related to the determination and assignment of the rank value. In a rank attack scenario, the attacker falsifies its rank information and sends a DIO message to its neighbors using a rank that differs from its genuine rank. As a result, a rank attack affects the parental node selection mechanism in the network and degrades network performance due to the selection of non-optimal parents on routing paths. Worst Parent attack is a type of rank attack in which a node selects the worst parent according to the Objective Function. As a result, a child node could find itself in a non-optimal routing path having been deceived into seleccting the attacker as its parent. Therefore, the child node will experience a degraded network performance compared to being on the optimum path. However, this attack cannot be detected easily, since child nodes depend on their parents to send data packets to the Root Node. Furthermore, this attack cannot be monitored or detected by the attacker’s neighboring nodes since they can only see the rank value calculated by the malicious node itself and cannot decide whether or not the rank value is genuine.

4.1.3. Hello Flood Attack

In RPL, a node which wants to join the network broadcasts DIS messages to the neighborhood in order to inform them. Since there is no specific time interval defined in RFC 6550 [17] for the transmission of DIS messages, it may therefore differ according to the RPL implementation. A new node transmits DIS messages using fixed time interval and awaits a reply from other nodes within its transmission range. When one of the neighboring nodes answers the DIS message with a DIO message, the new node joins the network by responding with a DAO message to nodes who send a DIO message. The main aim of the hello flood attack is to generate large amounts of network traffic, and thereby keeping nodes and routes to the Root Node excessively busy, with the attacker’s aim to consume the network’s resources. The malicious node behaves like a new node who wants to join the network and broadcasts DIS messages periodically to the neighborhood. Hence, neighboring nodes are forced to reset their trickle timers or to to respond with a DIO. This situation can overload RPL nodes by increasing the overhead of the control message, and hence result in network congestion.

4.2. Proposed Neural-Network Based Intrusion Detection System

This section details the proposed neural-network-based IDS for RPL-based IoT networks. Before introducing the IDS model, the feature set used in the training of the neural network is detailed, along with other related information.

4.2.1. Feature Selection

In order to obtain an effective security solution against routing attacks in the IoT, it is important to determine the features that are suitable for training a machine learning system. The selected features should contain sufficient information to distinguish malicious from benign activities. Furthermore, they are preferred to have non-redundant information, as the presence of too many features could adversely affect the training process. In order to select suitable features for training the neural network, first the behavior of a network not under attack is analyzed in terms of different performance metrics. Then, each attack is implemented using different parameters and their effects on the network are analyzed. In a recent study [25], a feature set which covers most of the features related to RPL control messages and data packets in the network was employed for the purposes of intrusion detection. In the current study, in addition to this feature set, features related to the link layer were employed in the training process. A list of the features used as the input to the neural network is presented in Table 4.1., after which follows an explanation of each feature.

Table 4.1. Features

COUNT_DATA	Number of data packets
COUNT_DIO	Number of DIO messages
COUNT_DIS	Number of DIS Messages
COUNT_DAO	Number of DAO Messages
COUNT_MAC_DROP	Number of dropped packets due to link-layer collisions
COUNT_NEIGH_DROP	Number of dropped packets due to link-layer neighbor allocation
COUNT_QUEUEBUF_DROP	Number of dropped packets due to link-layer queueing
COUNT_PACKET_DROP	Number of dropped packets due to link-layer packeting
RADIO_TRANSMIT_ENERGY	Energy consumed during transmission in the link layer
RADIO_LISTEN_ENERGY	Energy consumed during listening
MAX_VERSION	Maximum version number
MIN_VERSION	Minimum version number
DIF_VERSION	Difference between minimum and maximum version numbers
MAX_RANK	Maximum rank value
MIN_RANK	Minimum rank value
AVG_RANK	Average rank value
MAX_DATA_LEN	Maximum length of data message
MIN_DATA_LEN	Minimum length of data message
AVG_DATA_LEN	Average length of data message
MAX_TIME_BTW_DATA	Maximum time difference between data messages
MIN_TIME_BTW_DATA	Minimum time difference between data messages
AVG_TIME_BTW_DATA	Average time difference between data messages
MAX_TIME_BTW_DIO	Maximum time difference between DIO messages
MIN_TIME_BTW_DIO	Minimum time difference between DIO messages
AVG_TIME_BTW_DIO	Average time difference between DIO messages
MAX_TIME_BTW_DIS	Maximum time difference between DIS messages
MIN_TIME_BTW_DIS	Minimum time difference between DIS messages
AVG_TIME_BTW_DIS	Average time difference between DIS messages
MAX_TIME_BTW_DAO	Maximum time difference between DAO messages
MIN_TIME_BTW_DAO	Minimum time difference between DAO messages
AVG_TIME_BTW_DAO	Average time difference between DAO messages

COUNT_DATA is the total number of data messages destined to the Root Node. Every node which joins the network sends a periodic data message to the Root Node each minute. As a result of different situations such as the failure of a link connected to a parent node, local or global repairs in the network, some nodes cannot join the network or send a message to the Sink Node periodically for a specific time interval. Therefore, this situation provides information related to the network's stability.

COUNT_DIO is the total number of DIO messages destined to the root in the network. As previously mentioned, DIOs are sent periodically to the neighborhood to provide information about the node itself. The frequency of these messages can change according to the stability of the network, which could signal anomalous behaviors in the network.

COUNT_DIS is the total number of the DIS messages sent by nodes in the Root Node's transmission range in the network. DIS messages are sent by nodes who want to join the network or somehow to renew its parent node, and are seeking a path to the root. Therefore, the number of DIS messages is an important feature which shows the state of the network. Since the hello flood attack uses DIS packets in order to choke the network, this feature can be effective in the identification of a hello flood attack, especially if the monitored node is within transmission range of the attacker.

COUNT_DAO is the total number of DAO messages sent to the Sink Node in the network. Since destination information is propagated in the upward direction with the help of DAO messages, it can also provide clues about the state of the network.

COUNT_MAC_DROP is the number of dropped packets due to collisions in the link layer, which uses CSMA/CA as a multiple access protocol. After a specific number of retransmissions, the packet which has still to be transmitted is subsequently dropped. This feature and the consecutive three features are important to distinguishing the reason for packets being dropped in the network, especially for those dropped in different layers.

COUNT_NEIGH_DROP is the number of dropped packets due to neighbor allocation in the link layer. In the link layer, when there is a drop related to not being able to allocate buffer space in the full neighbor queue, this counter is incremented.

COUNT_QUEUEBUF_DROP is the number of dropped packets in the link layer due to queuing. Therefore, when the number of packets in the network increases, packets are dropped due to non-allocation of space in the transmitter buffers.

COUNT_PACKET_DROP is the number of dropped packets due to problems in the buffer management module that manages incoming and outgoing packets in the CSMA/CA protocol. Once the packet is ready, 6LoWPAN passes it to the link layer for transmission. Likewise, when packets are received, the link layer passes packets to 6LoWPAN via this module.

MAX_VERSION, MIN_VERSION and **DIF_VERSION** are features related to the repair mechanisms in the RPL, hence they concern version number updating. As previously mentioned, the version number is only incremented by the Sink Node in order to create a new version of the DODAG. However, in several types of RPL attack, the version number can be incremented by an attacker directly or indirectly. Therefore, the maximum and minimum version number encountered during the feature extraction time interval is also selected as a feature. The difference between the maximum and minimum version numbers is also selected as a feature in order to take into account the changing rate of the version number. In "version number attacks," since the version number changes more frequently than is considered normal network behavior, it can be a strong indicator for the existence of this attack type.

MAX_RANK, MIN_RANK and **AVG_RANK** are features related to the rank value calculated by each node. The rank value of a Sender Node is obtained from DIO messages broadcast to the Root Node. As previously described, the rank of a node represents the quality of the route between the node itself and the root. The rank value could change according to different routing metrics given in the Objective Function. A malicious node could change the rank of a node directly or indirectly. Moreover, it could falsely advertise its own rank value. Therefore, it is important to inspect the Sender Node's maximum, minimum, and average rank values in order to observe the effects of RPL attacks. It could be especially effective in detecting "worst parent attacks," as the attacker node chooses the worst neighbor (with the worst rank value) as its parent in this attack scenario, which subsequently increases the rank value of the attacker node. Since other nodes use the rank of their parent in calculating their own rank value, the rank of nodes in the downward direction is increased cumulatively, and thereby affects the MAX RANK and AVG RANK feature values.

MAX_DATA_LEN, MIN_DATA_LEN and **AVG_DATA_LEN** report the length of the data frame for all of the packets received by the Root Node. Since data messages could have different data lengths, these three features provide information about the density of messages transmitted within a specific time period.

The other features are time-related, providing information about the time intervals between data and routing control protocol packets in the network. For each packet type, the minimum and maximum time between two packets and also the average time difference is extracted and recorded. Each of these features were extracted from ".pcap" files, which are the output files of the Wireshark [38] packet analyzer tool. Other features could also be extracted from the ".pcap" files such as source address, destination address, and timestamp, however, these features are considered network-topology-specific, and are therefore not used for training purposes.

Two more features were selected, but have yet to be implemented, with their evaluation to be the subject of study in the future. First, **RADIO_TRANSMIT_ENERGY** represents the energy consumed by nodes during packet transmission, and is a feature extracted from the simulation log files via the "CollectView" tool in the Cooja simulator[39]. Due to the destructive effects of RPL attacks, nodes may have reason to attempt retransmission of packets due to collisions. As a result of these retransmissions, the energy consumed by each node could increase, which in turn could be taken as a sign of the network being attacked. Second,**RADIO_LISTEN_ENERGY** is a feature for measuring power consumption as a node listens to the network in order to receive packets destined for that node. When the network becomes unstable, the trickle timer is reset more frequently than usual and the energy spent by each node for listening increases by differing ratios. Therefore, this is an important feature that can be used to identify attacks within RPL-based network.

4.2.2. Proposed IDS Model Based on Neural Networks

Today, numerous devices require connection to the Internet. However, this has increased the level of traffic and consequently the risk of data leakage where local devices are connected to the wider outside network. This can pose a major problem in terms of information sharing, with data potentially easily accessible to third parties if the necessary precautions are not employed. In wireless networks, the situation can become more acute, since such networks use an open medium. Whilst communication is conducted using physical cables in wired networks, in wireless networks the communication signals are openly broadcast, leaving a potential open door to potential attackers. Another issue in IoT networks is the use of lossy and weak communication links. Especially in RPL-based IoT networks, differentiating anomalous from normal network behavior is much more demanding than for other network types, since packet loss could result in false positives and false negatives. Lastly, the nodes in IoT networks could be affected by resource constraints, meaning that complex security solutions might not be readily adaptable to such networks. For all these reasons, new intrusion detection systems should be proposed that are specifically designed for RPL-based IoT networks. Most RPL-based IoT networks are generally used for multipoint-to-point (MP2P) communication, therefore data (such as sensor data collected from the environment) flows are from leaf nodes to the Root Node. Root Node is usually responsible either for forwarding collected data to other applications, or for analyzing the data locally. Since it is a single point of failure, many IoT applications use a backup node. For instance, one of the Sink Node's neighbors may take the place of the Sink Node and it results in a failure of the design. Moreover, this node is generally a more powerful device than other nodes, and in the network layer, the Root Node has a better view of the network than any other individual node in the network due to the use of RPL as a routing protocol. Based on all of these assumptions, the current thesis study proposes a centralized intrusion detection system located in the Root Node for applications based on MP2P communication. Moreover, taking into consideration the resource-constrained nature , a centralized intrusion detection architecture is expected to be more efficient for computation and communication costs than a distributed system.

As the IoT network is getting larger, the RPL protocol generates a large number of routing control messages collected in the Root Node. In order to process such a large volume of data, this thesis proposes a neural-network-based IDS. With increasing popularity, deep learning techniques are used in a number of wireless network domains in areas such as network security, mobility analysis, network control, signal processing, and mobile data analysis. Recent survey studies [40, 41] have also detailed the domains in which these techniques can be applied, as well as specifying their usage. The current thesis study investigates its usage in developing an intrusion detection system for RPL-based IoT. The problem at hand is a classification task in machine learning, with the aim being to differentiate malicious attempts from normal network behaviors, utilizing data collected in the Root Node. The aim is not only to predict whether or not the network is being attacked, but to predict the type of RPL attack with a high degree of accuracy. Therefore, the problem is explored within this thesis study using both binary and multiclass classification.

First, the training and testing datasets were constructed. The steps taken for constructing these datasets are detailed in the following chapter. Then, as a preprocessing step, data normalization was applied to the datasets. It was considered important to perform feature scaling prior to the training in order to achieve the quickest and most appropriate results. Therefore, the Standard Scaler function of the Scikit library [42] was used to normalize the features so that each feature has a mean value of zero ("0") and standard deviation of unit variance ("1"). The neural-network-based learning algorithm was implemented using the following libraries: Scikit [42], Pandas [43], Numpy [44] and Keras [45]. Keras is a neural network API which can be used within Tensorflow [46]. It can also be used as a deep learning library. Moreover, Keras allows users to model and test a neural network quickly, since it supports algorithms run on both CPU and GPU. The current study therefore selected Keras for the proposed neural network implementation.

An illustration of the architecture is shown in Figure 4.1. In order to calculate the weights of the input set, four fully-connected neural network layers, each with different output sizes, was used. As an activation function, the “Rectified Linear Unit” (ReLU) function was employed. The output size of the first layer started with the highest number of neurons, and decreased by a factor of two in each consecutive layer. In other words, the number of neurons for the four hidden layers were 128, 64, 32, and 16, respectively. Dropout layers were situated between each fully-connected layer, with a 0.5 drop rate applied in order to prevent overfitting. Then, a fully-connected layer with the Softmax function was included. The output size of this layer depends on the problem type, namely binary and multiclass classification. Two neurons were used in binary classifications, representing normal and abnormal network behaviors. Four neurons were used in multiclass classifications, with one neuron corresponding to normal network behavior and the others corresponding to each of the various attack types addressed in this thesis.

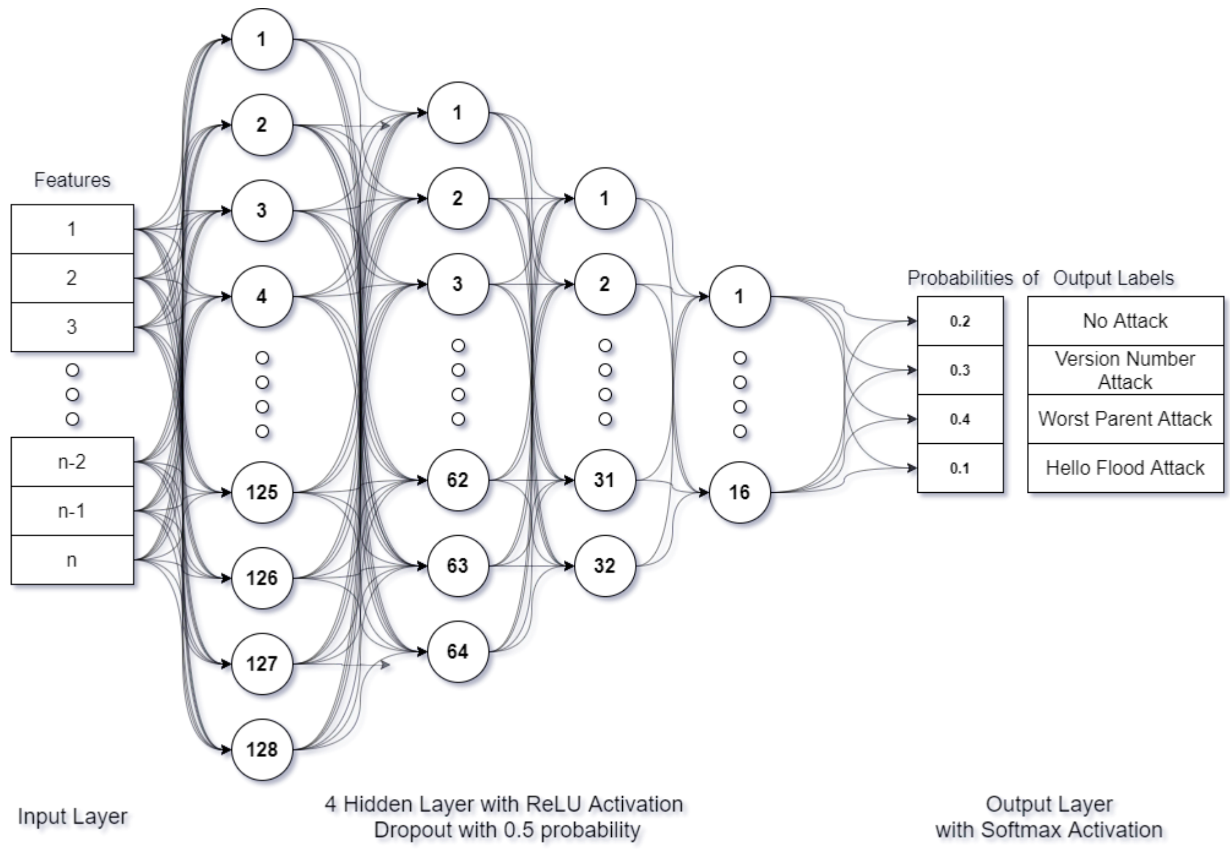

Figure 4.1. Proposed neural network architecture

The number of hidden layers and neurons used in each of the layers were defined by trial and error method. Although different approaches for selecting the number of hidden layers have been used in the literature [47, 48], none were proven to be an absolute solution. In [49], it was stated that more than two hidden layers could be used in order to learn complex representations (sort of automatic feature engineering). In considering this approach, different numbers of hidden layers with various numbers of neurons were used in training the model. In the preliminary experiments, networks from one single hidden layer up to five hidden layers were evaluated separately for their accuracy. Also the number of neurons and their orders were changed in order to observe the corresponding effect. Ultimately, the model with four hidden layers and a decreasing number of neurons, which gave the best result, was selected for the remainder of the experiments.

One of the important features of the proposed neural network architecture was the use of the ReLU activation function for determining the weights of neurons. A number of different types of activation function have been used in the literature such as ReLU, linear, sigmoid, and hyperbolic tangent, but linear (ReLU) activation is the simplest form of activation function used in machine learning. It is also very easy to train with this function, although it is known that it cannot learn complex structures. Sigmoid and hyperbolic tangent functions are nonlinear activation functions which are used in different kinds of application. However, the problem related to these functions is that sigmoidal function is likely to be saturated in certain situations which creates a challenge when adapting weights during training [50]. Moreover, the calculation of the error function in back-propagation becomes more challenging as the derivative of these functions becomes more complex. ReLU is an activation function that appears to be a simple linear function; however, due to its unique structure, it also allows complex relationships to be taken into account. A demonstration of the ReLU function is presented in Equation 1, and the activation functions, along with their derivatives, are shown in The activation functions and their derivatives are demonstrated in Figure 4.2. The derivative of the ReLU is simple, which enables the neural network to calculate the weights easier than for other activation functions. Therefore, due to of all these reasons, the ReLU function was selected as the activation function for the calculation of the weights in the current study.

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{1}$$

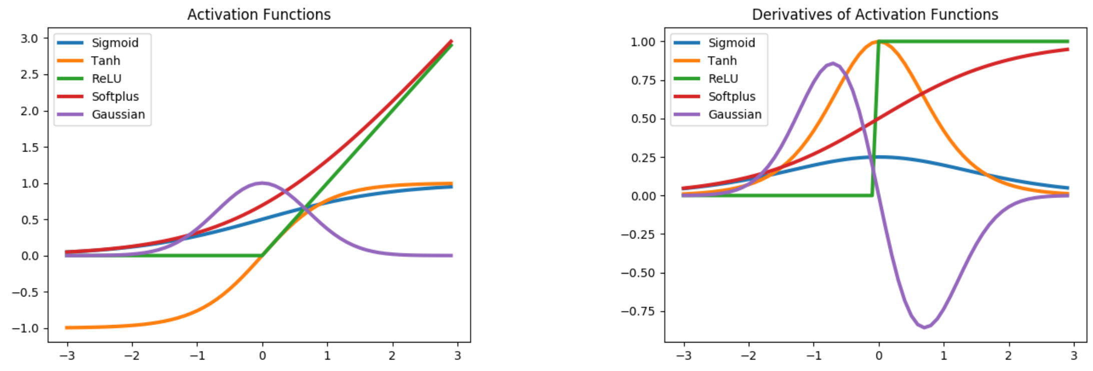

Figure 4.2. Activation functions and their derivatives

One of the biggest problems researchers face whilst training a neural network is overfitting. If the neural network produces a high level of accuracy during training and a low level of accuracy during testing, this is a strong indication of overfitting. This means that the neural network has not learned the patterns of the data but memorized them, and when a new dataset is employed for the evaluation of the trained model, it subsequently fails to meet the expected output. In order to avoid this problem, there is a method called "dropout," which randomly drops the output of certain neurons according to a predefined dropping rate. In each epoch, each dropout layer will drop different neurons' outputs, which allows the training process to continue with a different configuration, and results in a network which has better generalization for different datasets. In the proposed method, a 0.5 drop rate was used, since some studies [51, 52] in the literature have shown that a drop rate of approximately 0.5 provides the best results for regularization. As the dropout rate increases, the learning process slows down and the training time needed for the convergence of the model subsequently increases. Conversely, if the training time is limited or the training does not continue until the point of convergence, a high dropout rate may be accountable for inferior results. The use of dropout usually degrades the accuracy of the neural network at the start of training; however, by the end, the usage of dropout provides superior results in converged neural networks [51, 52].

Another important component of the network is the Softmax activation function, which was utilized in the last layer. The Softmax activation function is considered the best choice for both binary and multiclass classification problems because it takes the output of the previous layer and converts it into a probabilistic interpretation by normalizing them according to the number of classes at the output. It provides an array of probability scores for label prediction where the sum of these probability scores are equal to 1. The accuracy of the neural network is also calculated by taking into account the highest probability between predicted labels.

In the literature, there are many optimizers used in order to adjust weights and learning rates. In the current study, a popular optimizer known as "Adaptive Moment Estimation Algorithm" (ADAM) is an optimizer used for updating weights in training. The ADAM optimizer calculates learning rates according to the average of both the first and second moments of the gradients, which refer to the mean and uncentered variance [53]. In order to compare the ADAM optimizer with existing options, a neural network model with two fully-connected hidden layers of 1,000 hidden units, and that uses the ReLU activation function, was employed in order to compare the ADAM optimizer with existing optimizers such as AdaGrad, RMSProp, Stochastic gradient descent (SGD) with Nesterov momentum, and AdaDelta. In the comparative results, the ADAM optimizer showed superior convergence over the other optimizers [53].

As a loss function, "Sparse Categorical Crossentropy" was used in the design, since it can be used in both binary and multiclass classification. Categorical Crossentropy requires that the label for the output should be one-hot encoded. However, with the help of the Sparse Categorical Crossentropy loss function, the dataset output labels are left as they are and there is no need to modify the labels.

The model was trained for 100 epochs, which was also selected experimentally. As the dropout rate between the hidden layers increases, the convergence time needed for training also increases. Therefore, the proposed neural network model, using the architecture as shown in Figure4.1., was trained and tested using a varying number of epochs with the 10-fold cross-validation technique. According to the results, the best accuracy on the test dataset was obtained with 100 epochs. Therefore, 100 epochs was used for training the neural network for the remainder of the experimentation.

For training and testing the model, two different schemes were employed: "10-fold cross-validation" and "60% percentage split." In the percentage split scheme, the same dataset was divided into two subsets, as training and testing sets, according to the given percentage rates. Because of the sample variability between test and training data, while the training dataset should provide a low error rate and with a high rate of accuracy, the test dataset can give a higher error rate but with a lower accuracy level. Also, the model could overestimate the training data since it uses a specific portion of the data, which results in a failed evaluation due to overfitting. In order to avoid these kinds of problems, besides defining specific percentages to be applied for training and testing, the k-fold cross-validation technique [54] was employed. This technique randomly divides the dataset into *k* different groups of equal size and iterates the training and testing for *k* times. In each iteration, it keeps a different fold for testing, and performs the training using the remaining dataset. With the help of this technique, the aim is to cover all of the dataset for both training and testing purposes and to predict more accurate and non-variable results.

5. EXPERIMENTS AND RESULTS

In this chapter, the simulation environment with its parameters and the performance metrics applied in the analysis of the experimental routing attacks are detailed. Also, the performance of the proposed IDS solution is analyzed and discussed in this chapter.

5.1. Experimental Settings

5.1.1. Simulation Environment

To simulate IoT networks, different simulators have been used by researchers in the literature such as Cooja [39], Omnet++ [55], NetSim [56], and NS-3 [57]. After some research into these simulation tools, the “Cooja Contiki Simulator 2.7” was decided to be used, since it has RPL implementation. Moreover, the Cooja Contiki simulator has a user-friendly Graphical User Interface (GUI) and has also been used in numerous other research studies. Cooja can be used both with hardware on a real-time basis, and also with programmable nodes within a simulated environment. For RPL attack analysis, a simulator interface is used to implement different routing attacks. Cooja allows the researcher to simulate the RPL operation in detail, and also to implement routing attacks with a high degree of flexibility. The RPL protocol is implemented in the Contiki OS using the name “ContikiRPL.” By modifying this protocol, it is then possible to simulate attacks and, by using built-in measurement tools in the simulator, the effects of each attack on the network can be analyzed.

In the simulation, “Tmote Sky” [58] nodes were employed to simulate IoT devices. Tmote Sky is a low-power wireless module used in monitoring applications. The Cooja simulator creates a node type for each sky node and allows the user to simulate the same node virtually. “Sink Mote” is a border router in the network which connects other nodes to the Internet by collecting data from the network and helping the nodes to create the DODAG structure. “Sender Mote” represents IoT devices in the network by sending periodic data messages to the Sink Node by using its preferred parent constructed in the DODAG. When the preferred node has some data packets to forward, it sends it to its own parent and through this approach, the packet is forwarded until the Root Node received it. The “Malicious Mote” is another Sender Mote, but its mission is to manipulate the network and decrease the performance of the network. In order to perform a realistic simulation in Cooja, the simulation parameters given in Table 5.1. were used in the experiments.

Table 5.1. Cooja simulation parameters

Simulation Parameters	
Simulation tool	Contiki 2.7 Cooja simulator
Mote type	Sky Mote
Simulation run time	60 mins
Total number of Sender Nodes	50
Sink Node	1
Radio medium	Unit Disc Graph Medium: Distance Loss
Transmission range	50m
Interference range	100m
Mote start delay	60 secs
Seed type	Random Seed
Positioning	Random Positioning
Simulation area	125x125m

For the performance evaluation, additional data-gathering methods are used during the simulation. There is a tool in Cooja called "CollectView" which helps users to extract detailed information related to the network. While sending a periodic data packet to the Sink Node, a Sender Node includes additional information related to its condition such as consumed energy and ETX metric. The CollectView interface allows the user to both access and use these data. To obtain information regarding all the transmitted packets in the network, "Headless Radio Logger" is used, which creates a ".pcap" file in the Cooja directory and records every sent packet into the file. The "Simulation Script Editor" is used to start/stop simulations automatically, and is also used to log all of the "print" commands. These "print" commands can also be used in order to gather information related to network behavior.

5.1.2. Performance Metrics

The effects of attacks on the network were analyzed using four metrics:

- Packet delivery Ratio
- End-to-end delay
- Overhead

- Power Consumption

All four of these parameters are the average of the corresponding parameters collected from each node in order to observe the general behavior of an IoT network.

Packet delivery ratio (PDR): This provides an indication of packet loss due to interference, bandwidth, congestion, or some other types of problem in either a wired or wireless environment. It is represented as a ratio of the number of packets having reached a destination node over the number of packets destined to this node, and can be used to measure the overall health of the network. In order to provide continuous data flow from IoT devices in the network, the packet delivery ratio should be as high as possible. The formula for PDR is given in Equation 2:

$$PDR = \frac{Total\ number\ of\ packets\ received\ by\ the\ Root\ Node}{Total\ number\ of\ packets\ transmitted\ by\ other\ nodes} * 100 \tag{2}$$

End-to-end delay (E2E Delay): This represents the time passed for a packet to successfully reach the destination node from the source node. Since, the communication pattern of the proposed network is multipoint-to-point (MP2P) and the nodes use other nodes to relay their packets to the Sink Node, the end-to-end parameter consists of transmission delay, processing delay, and routing delay for each hop. Therefore, in each hop from source node to the destination, these processes append additional time to the E2E delay. Lost and dropped messages are not taken into account in calculating the E2E delay. The formula used to calculate the E2E delay is given in Equation 3:

$$End\text{-}to\text{-}end\ delay = \frac{\sum_{n=1}^{n} received\ time - sent\ time}{n} \tag{3}$$

where n is the total number of received packets by the Root Node.

Overhead: In order to manage the networking operations, RPL protocol uses different types of message packets such as DIO, DAO, and DIS. In a stable network, these packets are used with decreasing frequency. When the stability of a network is disturbed, the number of these packets increase, and this increase can then start to detrimentally affect the network. The total number of these topology control messages are noted as “Overhead” in the IoT. The formula for calculating overhead is given in Equation 4:

$$Overhead = total\, number\, of\, DIO + total\, number\, of\, DAO + total\, number\, of\, DIS \quad (4)$$

Power Consumption: Due to the limited resources of IoT devices, power consumption becomes critical for the nodes. In RPL topology, the nodes receive and transmit for a certain amount of time with smaller duty cycles. If the stability of the network is disrupted, the nodes need to increase the duty cycle of being active in order to perform certain necessary operations, and this results in increased power consumption. The power consumption is the sum of the listen and transmit power in the network.

The proposed IDS was evaluated with the following four metrics:

- Precision
- Recall
- F1-score
- False Positive Rate (FPR)

Precision: This is a ratio of the number of correct positive predictions divided by the total number of positive predictions. The formula for precision is given in Equation 5:

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

where TP is True Positives and FP is False Positives.

Recall: This is a ratio of the number of correct positive predictions to the total number of positives. It is also known as the detection rate. The formula for recall is given in Equation 6:

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

where TP is True Positives and FN is False Negatives.

F1-score; This is a measure of a test's accuracy. It uses both precision and recall in order to compute the F1-score, which is the harmonic mean of the precision and recall. The F1-score can reach the value of 1 with perfect precision and recall and 0 as the worst condition. The formula for F1-score is given in Equation 7:

$$F1\text{-}score = 2.\,\frac{Precision\ .\ Recall}{Precision + Recall} \tag{7}$$

False Positive Rate (FPR):: This is a ratio of the number of incorrect positive predictions divided by the total number of negatives. The formula for FPR is given in Equation 8:

$$FalsePositiveRate = \frac{FP}{FP + TN} \tag{8}$$

where FP is False Positives and TN is True Negatives.

5.2. Analysis of Routing Attacks

In this section, the effects of routing attacks against RPL, namely "version number attack," "worst parent attack," and "hello flood attacks" are investigated. Three different node types are used for each attack simulation. The first is the "Sink Node," which is responsible for connecting data from the remaining nodes. The second is the "Benign Node," which is a reliable and trustworthy node operating in the network. The third is the "Malicious Node," which behaves maliciously and aims to harm the network.

In the literature, there are different simulations conducted with varying numbers of nodes. Most of them use fewer number of nodes while simulating the network [5, 8] . However, in order to observe the effects of these attacks properly, higher number of nodes should be employed for the simulation. Therefore, in the current study, one Sink Node and 50 Sender Nodes were elected to be used. In order to address different types of distribution, the simulations were run several times with "Random Seed," which creates nodes using random coordinates according to predefined intervals. Simulations with different network topologies were repeated five times for each attack type. For the network under no attack, 20 different network topologies were created. Using this approach, the simulations can address different types of network topology. While simulating a network under attack, the attacks were started at the 10th minute of the simulation in order to leave enough time for the network to regain stability by using RPL.

5.2.1. Analysis of Network under No Attack

To compare the effects of each attack, it is also necessary to simulate the same network set-up without an attack occurring to observe the “normal” network behavior as a baseline. For this purpose, a simulated IoT network with one Sink Node and 50 Benign Nodes was executed 20 times, and the log files of each network were recorded. In the topology design, except for the Root Node, the nodes send periodic data messages to the root at the rate of one packet per minute. This simulation emulated a typical sensor network application.

An illustrated sample network topology can be seen in Figure 5.1. The RPL protocol starts to create routing paths according to the Objective Function, and each node in the network chooses its parent according to this function. Several Objective Functions are presented for parent selection in RPL. The most popular are MRHOF [4] and OF0 [59]. According to research conducted in 2016 [60], MRHOF has a better PDR and lower E2E delay results for 50 nodes and 50 packets/min network density. Since MRHOF is therefore a better fit for the current study’s set-up, it was selected for further simulations. ETX is the “expected transmission count” which measures the quality of the path between two nodes, and the MRHOF Objective Function decides the parent of each node by taking into account the ETX value [61] and Energy parameters.

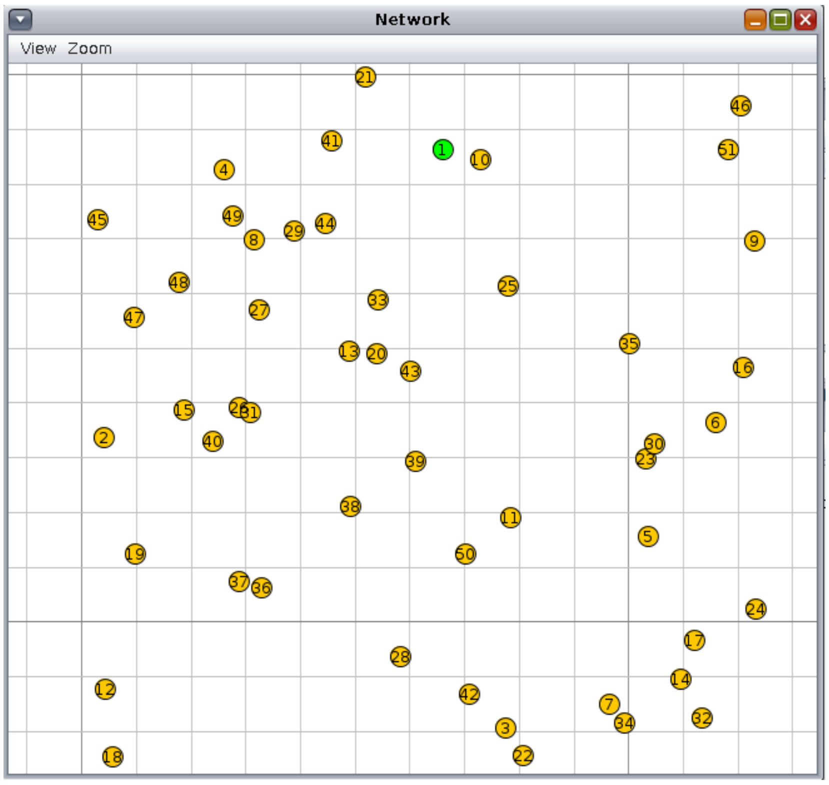

Figure 5.1. A sample simulated network

Simulation time is also an important parameter which has to be decided carefully. As can be seen from the literature, most studies are conducted over short time intervals (5-20 min) [7, 8, 35] ; however, due to the nature of the RPL topology, it also takes time to create a DODAG and therefore time for a network to restabilize following an attack having been resolved. The number of nodes also impacts on the time required to restabilize the network. Moreover, the attacker nodes could also slow down the construction of the DODAG. Therefore, in the current study, a 60 minute simulation period was chosen in order to better observe the changes affecting the network over time.

Each simulation was therefore conducted for a period 60 minutes, and were each recorded using ".pcap" and ".log" files. The result of the 20 simulations are given in Figure 5.2., where PDR started at nearly 96% during the first 10 minutes, converged to 99% after around 20 minutes and then remained at that level. The end-to-end delay was nearly constant (approximately 300 milliseconds) throughout the simulation.

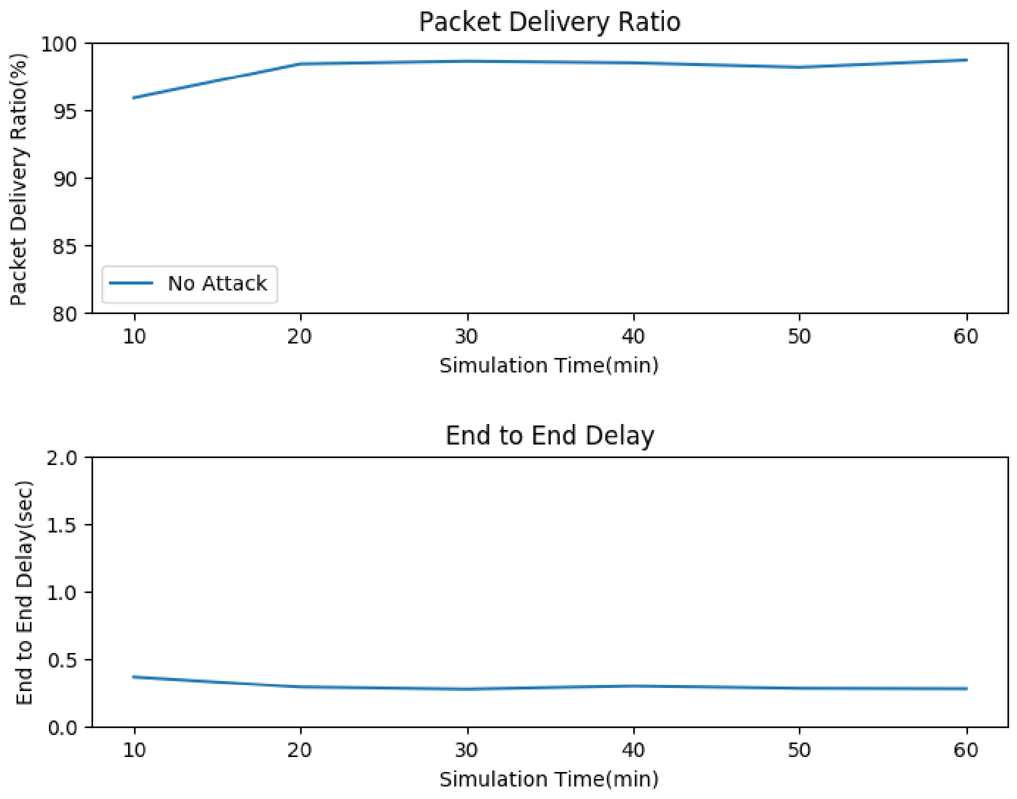

Figure 5.2. PDR and E2E delay metrics of network without attack

5.2.2. Analysis of Network under Version Number Attack

It is one of the most effective attack types against RPL-based IoT networks. The "Version Number" is one of the parameters in the DIO, and is used to trigger global repair mechanism. Normally, the version number can only be modified by the Sink Node; however, an attacker can manipulate this number whilst forwarding the incoming DIOs to other nodes. When DIO message with a different version number is received by a node, the repair mechanism is automatically started, resulting in the network moving to an unsteady state.

One of the most important parameters while simulating the version number attack is the frequency of version number increase triggered by the malicious node. According to the current study's experimental results, it was observed that increasing the version number in every transmitted DIO causes the network to become choked, and thereby impossible to observe the effect of the attack. Experiments were therefore conducted in order to select the most appropriate version number frequency of increase for the purposes of the experimental testing. According to the results, it was decided to use two different variations for the attack simulation. First, an attacker increases the version number by 1 each minute, and second, the malicious node increases the version number by 1 every five minutes. After the attack frequency was selected, it was implemented by modifying the "dio output()" function within the "rpl-icmp6.c" file with the help of a timer.

To observe the effects of the version number attack with different numbers of attackers, four different cases were created for the purposes of executing test simulations, as shown in Table 5.2. For each attacker type, five simulations were conducted.

Table 5.2. Number of attackers in Version Number Attack

Attack Case	Number of Attackers (%)
Case 1	1 (2%)
Case 2	3 (6%)
Case 3	5 (10%)
Case 4	10 (20%)

In the version number attack implementation, one Sink Node and 50 Sender Nodes with varying number of attackers were used. These nodes were placed randomly in each simulation in order to represent more network settlements. The effect of the version number attack on a minute-by-minute basis for four different attacker densities is illustrated in Figure 5.3. Even with just a 2% attacker ratio, the PDR can be seen to drop dramatically down to 10%. After the 40th minute of simulation, the PDR increased slightly up to 20%. For the simulations using a 6%, 10%, and 20% attacker ratio, the PDR was seen to remain constant at around 10% until the end of the simulation. Therefore, where the malicious node increased the version number by a factor of 1 every minute, even in the lightest attack case (2%), the network was then unable to rebuild the DODAG and reachieve a stable position. Because of the increase in the number of topology control messages the Sender Nodes mostly could not transmit their periodic messages to the Sink Node.

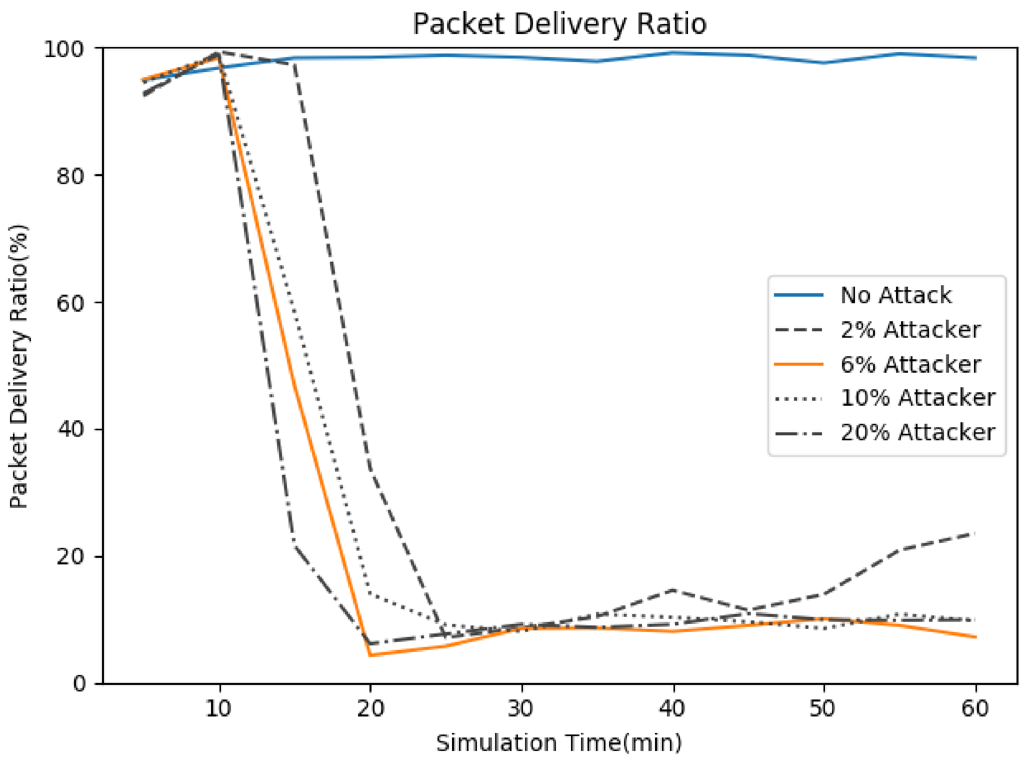

Figure 5.3. PDR for Version Number Attack - 1 min. variation

End-to-end delays for the four different attacker densities are illustrated in Figure 5.4. In all of the cases, the E2E delay increased dramatically from 300 ms up to a maximum of 4 seconds. On average, the number of attackers did not significantly impact the E2E delay value. Even for the lightest attack case (2%), the average E2E delay increased by 1,000% compared to the baseline.

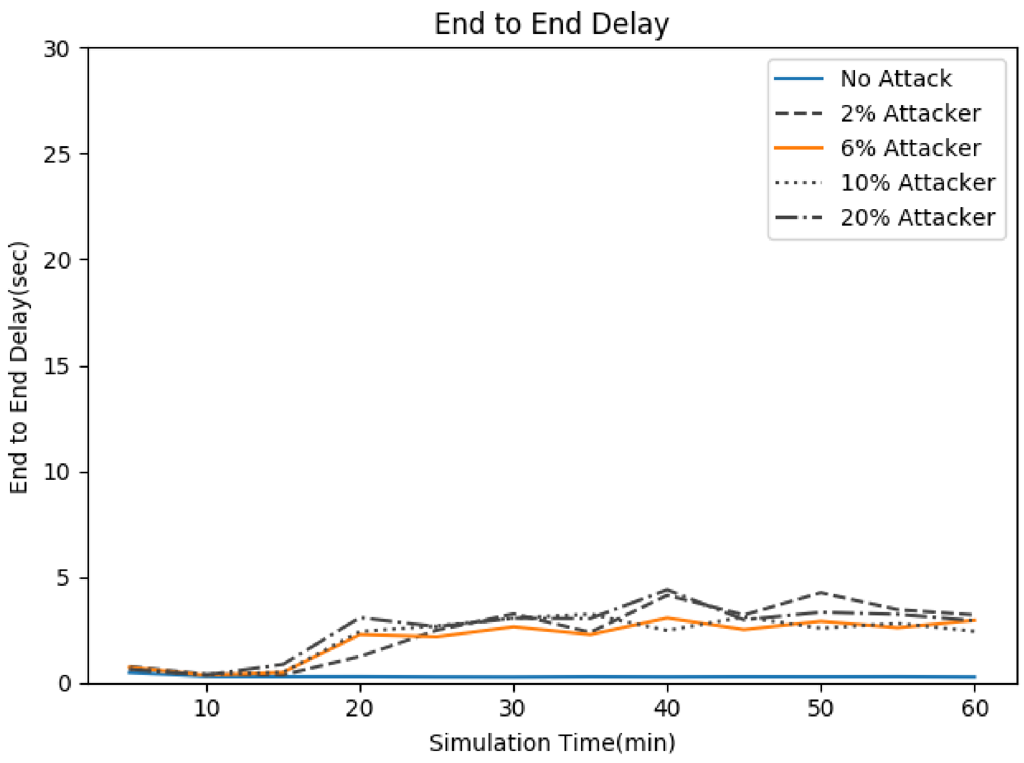

Figure 5.4. E2E delay for Version Number Attack - 1 min. variation

The second type of attacker increased the version number every 5 minutes. The PDR variance for this second type of attacker is illustrated in Figure 5.5. During the first 5 minutes of attack, the PDR dramatically dropped to around 40% for all attack cases. Following the 20th minute of simulation, the PDRs started to increase, recording different levels for each case. In the 2% attacker case, the PDR stabilized at around 70%. For the 6% and 10% attacker cases, the PDRs oscillated between 55% and 65%, whereas the PDR in the 20% attacker case oscillated between 50% and 60%.

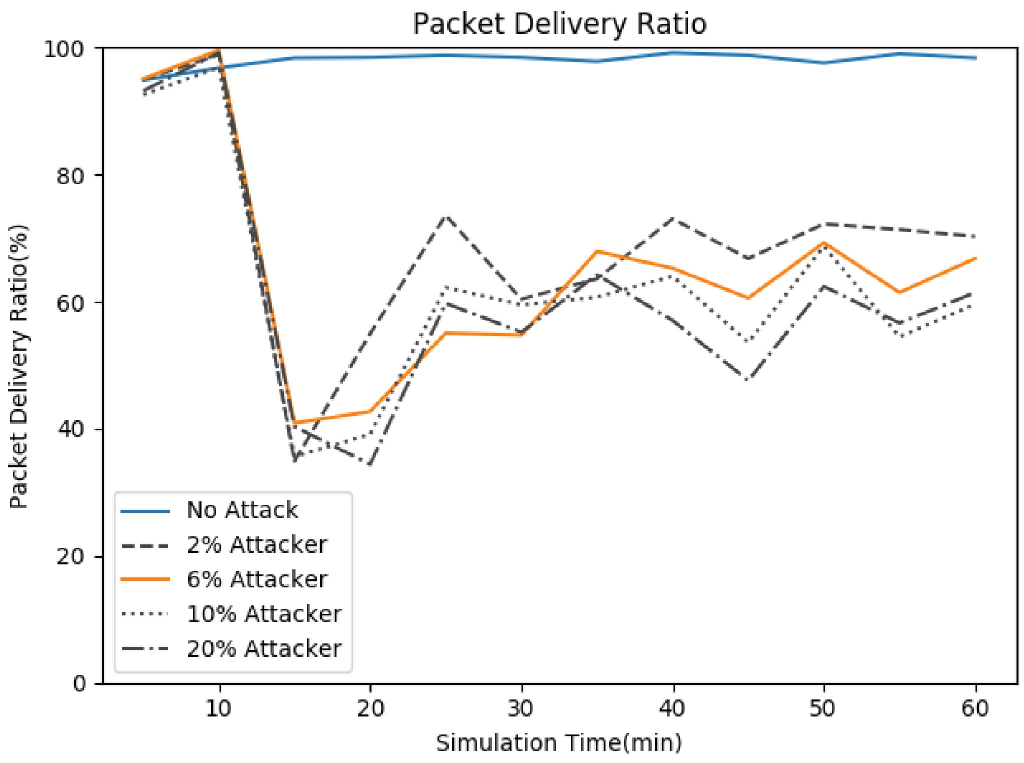

Figure 5.5. PDR for Version Number Attack - 5 min. variation

The E2E delay for the second type of attacker is illustrated as shown in Figure 5.6. Up until the 20th minute of simulation, the E2E delay increased constantly up to 2 seconds, which is nearly seven times the baseline value. After the 20th minute of simulation, the end-to-end delays started to decrease according to the attacker density. For the 2% and 6% attacker cases, the end-to-end delay stabilized at around 1.2 to 1.4 seconds interval. For the 10% and 20% attacker cases, the interval of stabilization for the E2E delay was around 1.5 to 1.8 seconds.

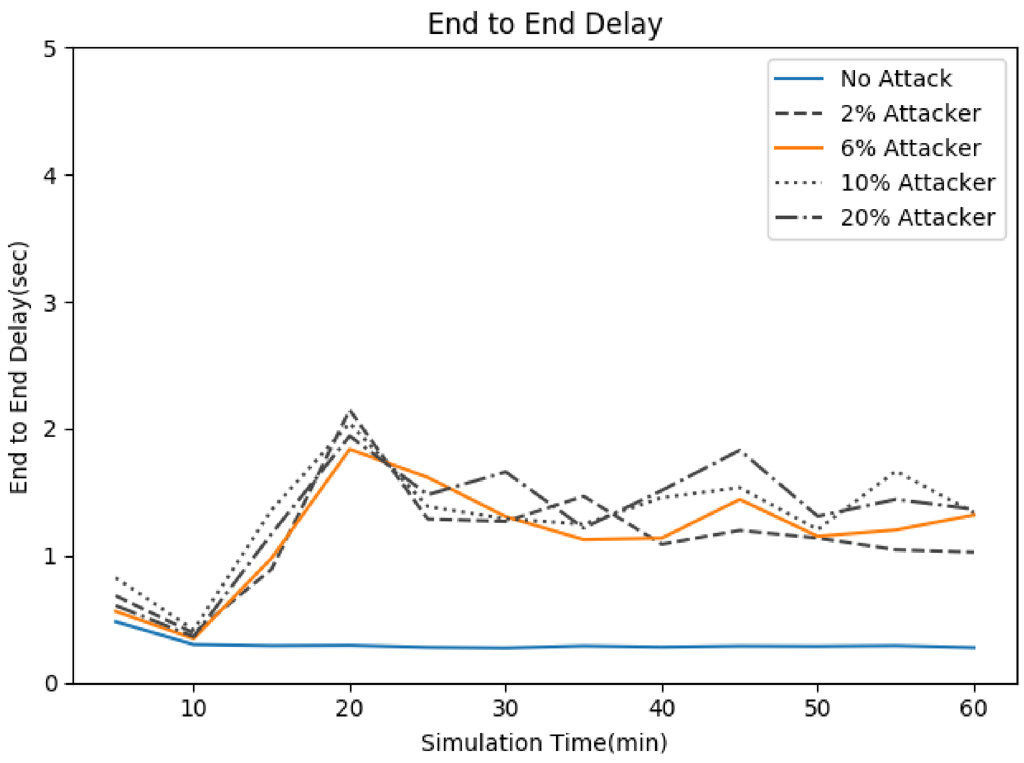

Figure 5.6. E2E delay for Version Number Attack - 5 min. variation

5.2.3. Analysis of Network under Worst Parent Attack

The worst parent attack is a different version of rank attack which aims to change the rank parameter by manipulating the calculation method and thereby causing harm to the network. As previously mentioned, the rank parameter is very important in helping child nodes to select the best parent for themselves and to create routes to the Sink Node from each node. Therefore, if a malicious node manages to successfully manipulate the rank parameter, the action can result in the network becoming unstable.

Changing rank values can be detected by the RPL protocol since RPL has certain control mechanisms in place to note illegal rank parameter changes. The “Loop Detection,” “Loop Avoidance,” and “Greedy DODAG Parent Selection” rules can limit the malicious node from changing its rank value to some other desired value. In this attack type, instead of changing the rank value, the parent selection algorithm is manipulated. There is a function called “best parent” in the “rpl-mrhof.c” library which is used to select the parent with the lowest rank.

Using this function, the attacker chooses the parent node with the highest rank instead of the lowest. Since rank parameter increases when the quality of link between a node and the Root Node decreases, selecting a parent with a higher instead of a lower rank can weaken the established link between the node and the root. Through the modified parent selection algorithm, child nodes which will start to use the malicious node as a parent node. If there is more than one attacker in the path between the node itself and the root, the effect of a worst parent attack is also increased. This attack aims to decrease the link quality between nodes, decrease PDRs and increase end-to-end delays, since the average number of hops can also increase.

In the simulated worst parent attack, one Sink Node and 50 Sender Nodes were used. To increase the variety of the attack and to observe the effect of different numbers of attackers in the network, four different attacker densities were used. For each attacker type, a total of five simulations were conducted with a varying number of attackers. The number of attackers and the corresponding case name are presented in Table 5.3. These nodes were placed randomly in each simulation.

Table 5.3. Number of attackers in Worst Parent Attack

Attack Case	Number of Attackers (%)
Case 1	1 (2%)
Case 2	3 (6%)
Case 3	5 (10%)
Case 4	10 (20%)

The PDRs for four different attacker densities can be observed in the illustration in Figure 5.7. The 2% attacker scenario PDR dropped slightly for the first 5 minutes; however, after that point, the network tolerated the effects of the attack, the PDR increased, and the network performed as if there was no attack taking place. In the 6% attacker case, the PDR was slightly below what was considered the normal behavior of the network, but did not drop below 95%. In the 10% attacker case, the attack caused the PDR to drop below 90% during the first 5 minutes of the attack, and continued to drop to 85% and then remained at that level throughout the simulation. In the 20% attacker case, the PDR immediately dropped to 70% in the first 5 minutes. At a point 20 minutes from the start of the attack, the PDR slightly increased to 75% and the remained at that level until the end of the simulation. The PDR decreased from 99% to nearly 70% in the worst case, which means in excess of one packet in every four was dropped and therefore unable to reach the Sink Node, which can cause serious problems where the data being sent from the node is considered to be of some importance.

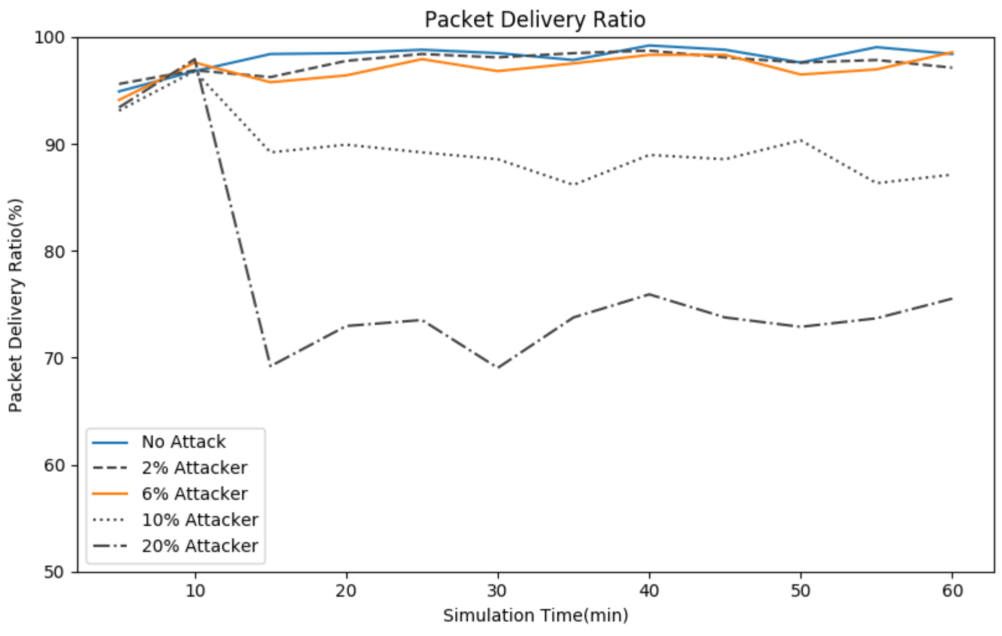

Figure 5.7. PDR for Worst Parent Attack

E2E delays for the four different attacker densities can be observed in Figure 5.8. The effect of the attack with a 2% attacker density in unrecognizable by examining the E2E delay. In the 6% attacker case, the E2E delay is slightly above that considered the normal behavior of the network, but there was still no recognizable difference from the baseline. However, in the 10% attacker case, the attack caused the E2E delay to increase by 300% in the first 5 minutes of the attack, and remained at that level for the remainder of the simulation. Similarly, in the 20% attacker case, the E2E delay peaked at 600% of the baseline level during the first 5 minutes of the attack, and then decreased slightly up until the end of the simulation.

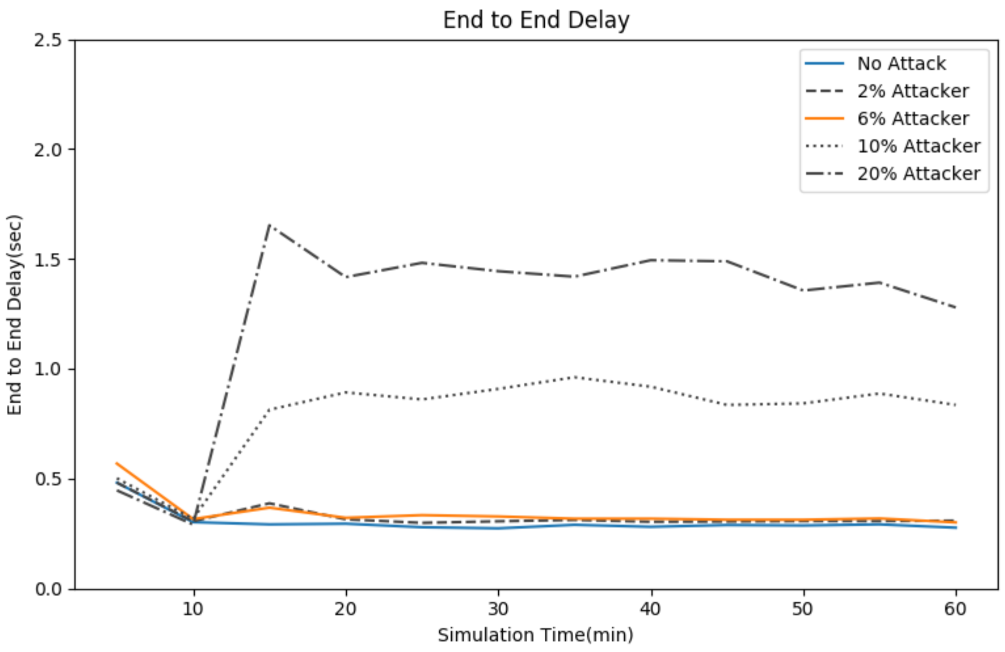

Figure 5.8. E2E delay for Worst Parent Attack

In both the 2% and 6% attacker cases, the PDR and E2E delay metrics were not found to deviate from the baseline. Instead of changing the rank value directly, the malicious node tried to indirectly impact the network parameters, making the attack undetectable and duplicitous. Since these two cases did not affect the performance metrics being monitored, both the 2% and 6% attacker cases for worst parent attack were not considered as an attack during training, validation, or testing of the developed intrusion detection system.

5.2.4. Analysis of Network under Hello Flood Attack

The hello flood attack is a generic attack type that can be applied to any type of wireless or wired network. In RPL-based networks, a flooding attack is carried out by using DIS messages. They are used by nodes who want to join the network, and nodes which receive a DIS message respond with a DIO message in order to help the new node to select its parent node and start communicating using the network. However, this feature can be manipulated by a malicious node by sending out DIS messages unnecessarily. In this attack type, the DIS message sending method is manipulated in order to create a flooding effect, with a malicious node repeatedly sending out DIS messages to force neighbors to reply with a DIO and thereby increase the overhead.

One of the most important parameters whilst simulating the hello flood attack is the selection of the frequency of broadcasting DIS messages to neighboring nodes. According to the experimental results, it was observed that sending DIS messages more frequently than 500 milliseconds caused the network to become choked, and it was then impossible to observe the effect of the attack through the network. According to the results, it was decided to use two different attacker densities for the attack simulation. In the first case, the attacker sends out a DIS message every 500 milliseconds, and this was changed to every second in the second case. Through this approach, the network's conditions according to different variations of attack can be observed. After selection of the DIS message broadcast frequency, the "handle periodic timer" function in the "rpl-timers.c" library was modified for the attack implementation.

In the hello flood attack implementation, one Sink Node and 50 Sender Nodes with different attacker numbers were used. These nodes were then placed randomly for each different simulation. The number of attackers for each case is presented in Table 5.4. along with its corresponding case name. Each case is simulated five times with random seed.

Table 5.4. Number of attackers in Hello Flood Attack

Attack Case	Number of Attackers (%)
Case 1	1 (2%)
Case 2	3 (6%)
Case 3	5 (10%)
Case 4	10 (20%)

The effect of the a hello flood attack, with 500 milliseconds frequency for four different attacker densities can be observed as shown in Figure 5.9. For the 2% attacker case, the PDR decreased from 95% down to 50%. After the 25th minute of simulation, the PDR increased slightly up to 60%. In the 6% attacker case, the PDR dropped dramatically to 10%. After the 30th minute, it had increased slightly at 20%, but then continued at this level for the remainder of the simulation. In both the 10% and 20% attacker cases, the PDR dropped to

10% and remained constant at this level until the end of the simulation. As can be seen, when the malicious node sent periodic DIS messages with 500 ms frequency, except for in the 2% attacker case, the remaining attacker cases were heavily affected and were unable to perform a stable network operation. As the number of attacker increases, the DIS messages reach most nodes in the network and they respond in turn with DIO messages. Because of the increased frequency of DIS messages sent out, the nodes can no longer allocate adequate resources for other types of network activity. For the 2% attacker case, the attack affected the network locally. However, the part of the network that did not receive the broadcast DIS messages continued to perform normal network operations.

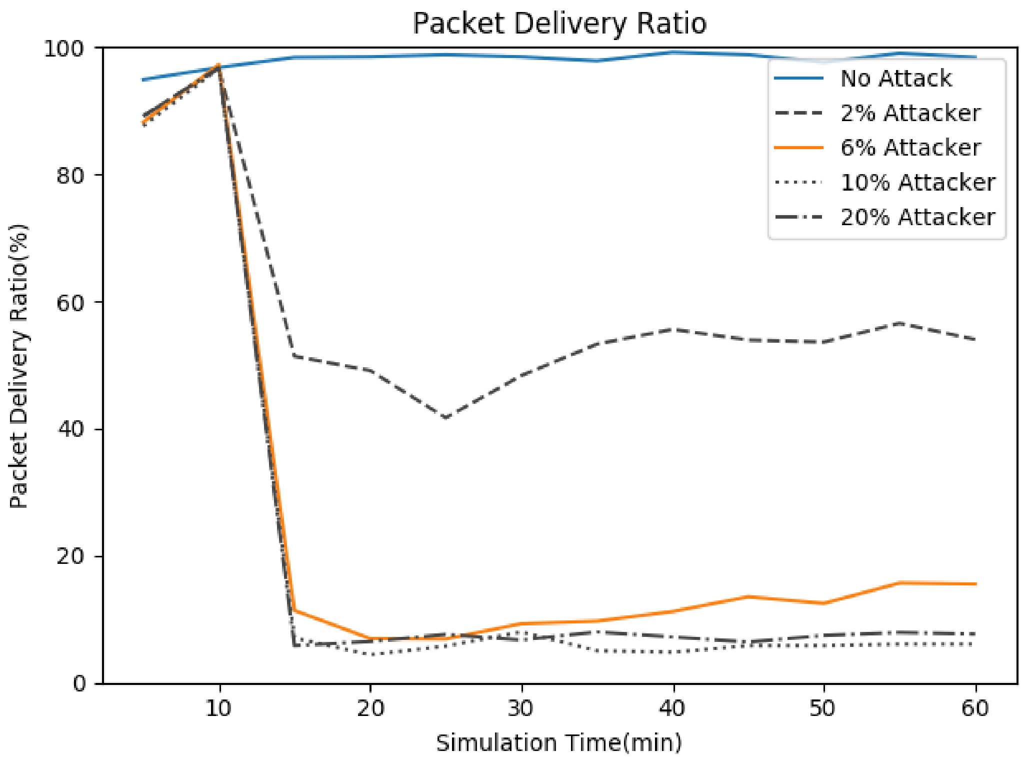

Figure 5.9. PDR for Hello Flood Attack

The end-to-end delays for the four different attacker densities are illustrated in Figure 5.10. In both the 2% and 6% attacker cases, the E2E delay increased dramatically from 300 ms to 5 seconds up until the 30th minute of the simulation. After this point, the E2E delay for the 2% attacker case remained constant at around 4 seconds, whereas the E2E delay for the 6% attacker case peaked at 10 seconds at the 35th minute, and then normalized to a level of 6 seconds up until the end of the simulation. Surprisingly, the 10% attacker case showed the lowest end-to-end delay average at around 4 seconds. The reason for this difference may be attributable to the attackers' location. If the attackers did not reach a specific region in the network, the average E2E delay remained at a low level when compared to the other attacker cases. However, in the 20% attacker case, the E2E delay increased aggressively right up until the end of the simulation, reaching the 25 second level, which is approaching 8,500% over baseline.

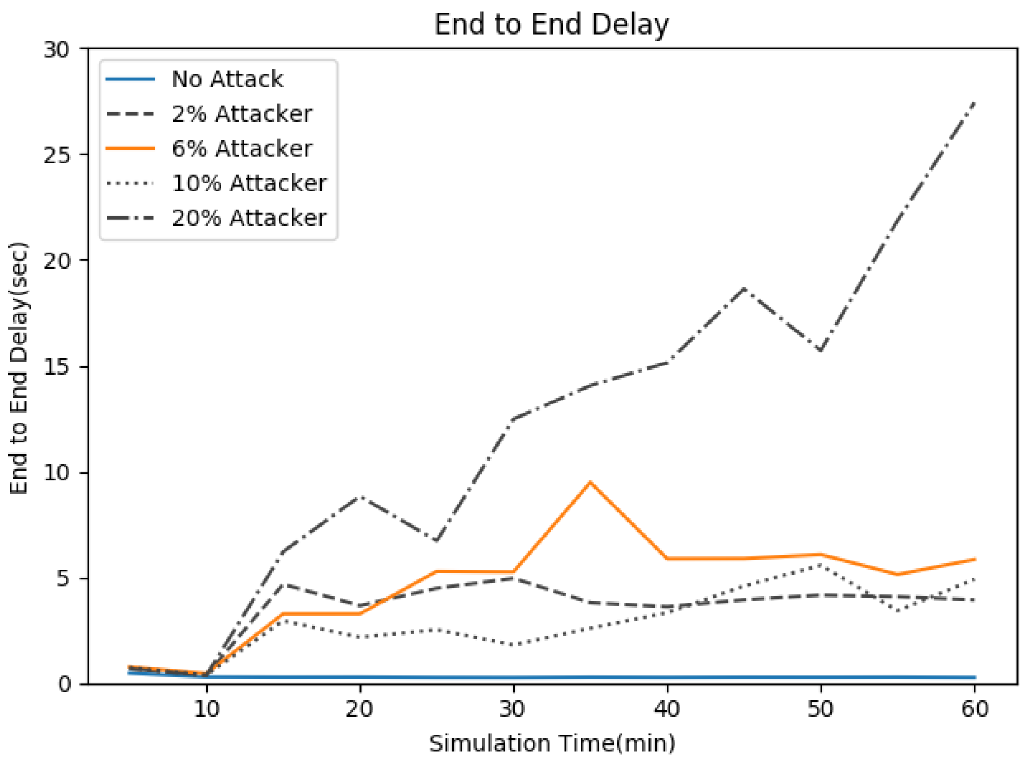

Figure 5.10. E2E delay for Hello Flood Attack

5.3. Attack Dataset

This section details the dataset which was used for the purposes of both training the neural network and for testing the proposed model. Even though a few attack datasets exist in the literature for the IoT [62, 63], none cover attacks against RPL. There was an RPL attack framework proposed in 2018 [64], which provided an interface in order to simulate built-in attack types without using the Cooja simulator interface. However, whilst it was possible to generate a database by using this framework, the provided attack types were not deemed sufficient for the current study as a multiple attacker option was not provided in the framework. Therefore, in the current study, a new dataset of routing attacks against RPL is introduced using the Cooja simulator.

The dataset was created from both ".pcap" files recorded from each simulation and ".log" files created by the Cooja simulator during the simulation process. Most of the features of the dataset such as rank and those related to version number or time-based features were collected from the ".pcap" files. However, the features based on the link layer which can help identify a benign network from an attacked network were captured from the ".log" files.

Different network analyzers have been utilized in order to investigate ".pcap" files to extract the required information. Tshark [65] is one of the network analyzers that has been used in the literature for similar purposes, and includes a variety of filters, options, and tools to extract the necessary information. Since the proposed IDS was planned to be centralized rather than distributed, it is therefore important to extract information from the packets which were destined to the Sink Node. Hence, the ".pcap" files were filtered according to the IP address of the Root Node and the "-Y" filter command of Tshark was used to filter according to the broadcast address. Also both the "-T" and "-E" commands were used in order to define options for the output file. Using this configuration, each ".pcap" file was processed and the necessary raw data extracted and saved as a ".csv" file.

In order to extract meaningful information from the log files, similar steps were applied. Whilst simulating the network, data gathered for feature extraction was printed to the log file using identifiers. These specific identifiers are then searched for in the log file and the pertinent information extracted and saved to a ".csv" file for each simulation log file.

Another important element of the raw data extraction is the selection of the data collection interval. In order to achieve meaningful data for training purposes, the data collection interval should be carefully defined. For this purpose, the time interval parameter was defined as a variable. The method for selecting the optimum time interval is provided in the following subsections.

5.3.1. Feature Extraction and Dataset Construction

After defining the feature set to be used for the purposes of training and testing the proposed IDS, each feature was calculated and recorded with a corresponding timestamp. These features were periodically collected. The time interval set for the periodic collection of features is important, because the data collected during the selected time interval should contain sufficient information in order to provide a reliable signal for detecting each type of RPL attack. For real-time attack detection, the time interval should not be specified as too long in order not to miss an attack's detection. The time interval's selection is performed according to the basic "trial and error" method, since no study exists for this purpose in the literature. For varying the time interval, from 500 milliseconds up to 10 seconds, the data from the network with normal behavior (i.e., "no attack"), together with the three different RPL attack simulations were extracted and recorded. Using the proposed neural-network-based IDS structure, each dataset was given to the network for the purposes of training. At the end of the training, the neural network was then evaluated for detection accuracy. The results of this evaluation are presented in Figure 5.11. Accordingly, the clear best option was a time interval of 5 seconds to achieve the highest detection rate for the proposed design.

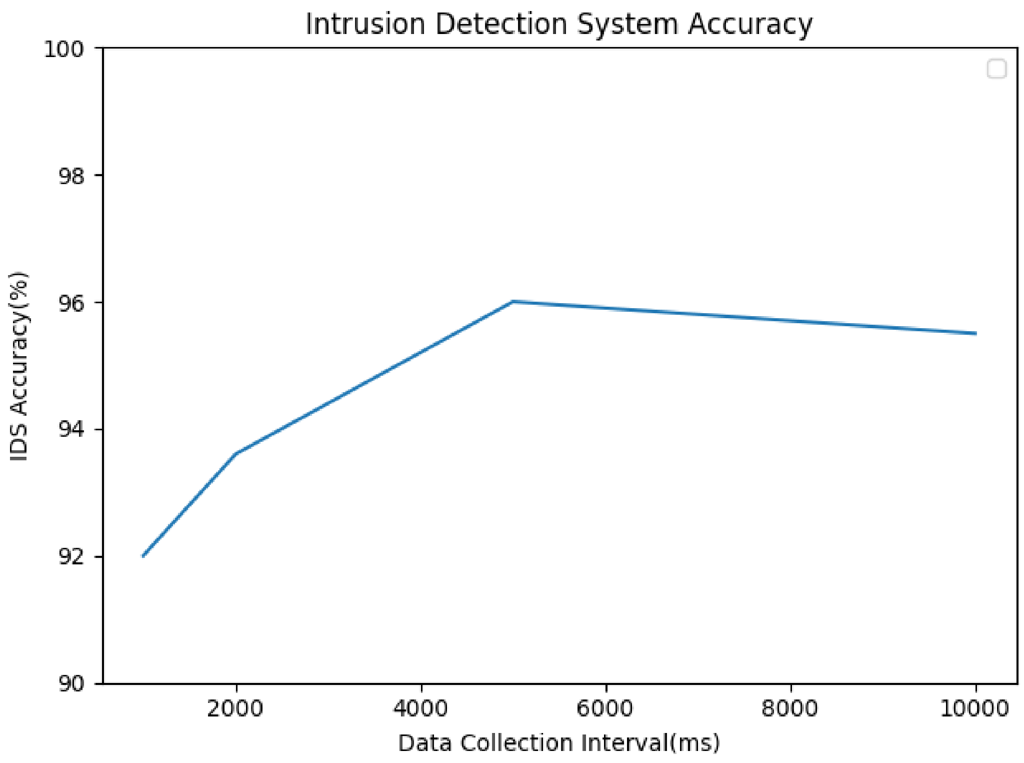

Figure 5.11. Accuracy of IDS with varying time interval duration

5.4. Performance Evaluation for Intrusion Detection System

In this section, the performance of the proposed neural-network-based model for intrusion detection is tested for both binary classification and multiclass classification. Slight changes are made to the design of the intrusion detection model according to the classification type. For binary classification, the output layer with Softmax activation was set to “2” (attack or benign). For the multiclass classification problem, the output size was set to “4” (benign network, version number attack, worst parent attack, and hello flood attack).

5.4.1. Performance of Binary Classification

In order to test the model with different routing attacks and attacker densities, the model was trained using two different network class types. The first was the baseline topology of a benign network operation, in which there was no malicious activity and no attacker node present. The second consisted of the dataset with three routing attacks, namely version number attack, worst parent attack, and hello flood attack. For evaluation of the proposed neural-network-based model, two different techniques were used, a 10-fold cross validation technique and train/test split method. For the second case, in order to separate the training and test data with an integer number of simulations, 60% of the dataset was set aside for training purposes, whereas 40% of the dataset was used for testing purposes. The evaluation of the trained network using the entire test data is given in Table 5.5.

Table 5.5. Performance of IDS with binary classification

Evaluation Technique	Precision	Recall	F1-score
Percentage Split (60%/40%)	0.999	0.969	0.984
10-fold cross validation	0.998	0.971	0.984

Detection and FPRs are also shown in Table 5.6.

Table 5.6. Detection rate and FPR for binary classification

Evaluation Technique	Detection Rate	False Positive Rate
Percentage Split (60%/40%)	96.88%	0.13%
10-fold cross validation	97.11%	0.34%

According to the results, the proposed model showed nearly the same results for both evaluation techniques. Although the detection rate was slightly higher in the 10-fold cross validation case, the false positive rate for the train/test split technique was lower than for the 10-fold cross validation case, as can be seen in Table 5.6.

In testing the model with smaller datasets of attack types based on varying densities, the following results were obtained for the train/test split technique based on a 60%/40% split in order to test the trained model using different sets of data. After obtaining the performance of the IDS on the overall test data, consisting of all four categories, the neural-network-based IDS was also tested for each category separately. The performance of the detection system for each attack type is presented in Table 5.7.

Table 5.7. Performance of IDS based on attacker class

Attack Class	Detection Rate
Version Number Attack	93.20%
Worst Parent Attack	98.17%
Hello Flood Attack	99.96%

As it can be seen from the Table 5.7., the proposed system achieved a detection rate of 93.20% for the version number attack, whereas it was 98.17% for the worst parent attack. In the hello flood attack, the neural-network-based IDS detected every second of the attack with a 99.96% detection rate.

For measuring the performance of the detection mechanism against varying attacker densities, the neural-network-based IDS was also tested for each attacker density variation for each of the three different attack cases. The performance of the developed IDS for the version number attack is presented in Table 5.8.

Table 5.8. Performance of IDS with Version Number Attack (VNA)

Class	Attack Frequency	Detection Rate
VNA 2% Attacker	1 minute	86.66%
VNA 6% Attacker	1 minute	92.99%
VNA 10% Attacker	1 minute	98.58%
VNA 20% Attacker	1 minute	94.83%

The results showed that even where there is only one attacker present in the network, the proposed IDS was able to detect a version number attack with 86.66% accuracy. For the 10% attacker case, the “version number attack” was detected with high detection rate of up to 98.58%. In general, when the number of attackers increases, their effects on the network becomes more observable.

Since the 2% and 6% cases in the “worst parent attack” did not affect the performance of the network and did not deviate from the baseline, these two cases were not taken into consideration as attack cases. The performance of the IDS for the 10% and 20% attacker cases for the “version number attack” are presented in Table 5.9.

Table 5.9. Performance of IDS with Worst Parent Attack

Class	Detection Rate
WPA 10% Attacker	96.91%
WPA 20% Attacker	99.42%

The performance of the IDS for the hello flood attack is given in Table 5.10.

Table 5.10. Performance of IDS with Hello Flood Attack

Class	Detection Rate
HFA 2% Attacker	99.83%
HFA 6% Attacker	100%
HFA 10% Attacker	100%
HFA 20% Attacker	100%

The experimental test results showed that the hello flood attack has characteristic features which made its detection easier than the two other attack types tested in the current study. Even in the 2% attacker case, which was the lightest for the hello flood attack, the proposed model was still able to detect the attack with almost 100% accuracy.

In order to see the effects of routing layer and link layer features, two models are trained with different groups of features and compared in Table 5.11. Detection and false positive rates are also shown in Table 5.12. Link layer features have caused a decrease in false positive rate since they help to discriminate normal cases from attack case in case of collisions in link layer.

Table 5.11. Performance of IDS with binary classification

Feature Set	Precision	Recall	F1-score
without link-layer features	0.997	0.971	0.983
with link-layer features	0.999	0.969	0.984

Table 5.12. Detection rate and FPR for binary classification varying link-layer features

Feature Set	Detection Rate	False Positive Rate
without link-layer features	97.06%	0.61%
with link-layer features	96.88%	0.13%

After evaluation of the proposed IDS using the entire test data consisting of all four different attack categories, each category was also tested separately. The performance of the detection system for each attack type is presented in Table 5.13.

Table 5.13. Performance of IDS with attacker classes varying link-layer features

Feature Set	Class	Detection Rate
without link-layer features	Version Number Attack	91.52%
with link-layer features	Version Number Attack	93.20%
without link-layer features	Worst Parent Attack	99.08%
with link-layer features	Worst Parent Attack	98.17%
without link-layer features	Hello Flood Attack	100%
with link-layer features	Hello Flood Attack	99.96%

When the model was trained with link-layer features, the detection rate for the version number attack also increased, from 91.52% to 93.20%. These features have slightly increased the detection rate of version number attacks, since this attack is the main cause of packet drops at the routing layer [31]. It is shown that while most of the packets are dropped at the routing layer as a result of version number attack [31], the packet drops in normal networks (under no attack) are mainly resulted from link layer issues. Therefore, it is believed that link layer features could help distinguishing normal cases from malicious activities. Hence they are employed for the first time in intrusion detection in RPL in this thesis. For the worst parent attack and hello flood attack, the detection rates remained almost the same. However, these attack types already have high rates of detection, therefore a clear effect of the link-layer features was not observed. However, there is a trade-off between false positive rate and the detection rate of an attack, and it is preferable to achieve a lower false positive rate (0.13% compared to 0.61%) instead of a slightly elevated detection rate (97.06% compared to 96.81%).

5.4.2. Performance of Multiclass Classification

For the multiclass classification, the size of the output layer in the model was adjusted to four, that being the number of classes in the dataset. The 10-fold cross validation method was then used for the evaluation. The evaluation technique repeated the train and test process 10 times, and for each operation, 90% of the dataset was used for training purposes, whereas the remaining 10% was used for testing. In each repetition, a different part of the dataset was used for the training. The results were obtained by averaging the values from the 10 different output sets. The performance of the model is given in Table 5.14.

Table 5.14. Performance of IDS with multiclass classification

Evaluation Type	Accuracy	FPR
10-fold cross validation	97.52%	0.26%

Confusion matrix for the multiclass classification is given in Table 5.15.

Table 5.15. Confusion matrix for multiclass classification

True Label\Predicted as	NA*	VNA**	WPA***	HFA****
No attack	99.7%	0%	0.3%	0%
Version Number Attack	6.94%	92.42%	0.48%	0.17%
Worst Parent Attack	1.42%	0.42%	97.71%	0.46%
Hello Flood Attack	0.04%	0.02%	0.04%	99.9%

* No Attack Implemented

** Version Number Attack

*** Worst Parent Attack

**** Hello Flood Attack

According to the results, the no-attack case was classified correctly with 99.7% accuracy, the version number attack was correctly classified with to an accuracy of 92.42%, whilst 6.94% of the version number attack sample was misclassified as being attack-free. When the version number attack started in the network, it needed time in order to affect the network, rather than creating an effect on the network immediately after initiation. Therefore, the model misclassified the version number attack as an attack-free case at the beginning of the initiation. This may be attributed as a reason for having the highest misclassification ratio in the multiclass classification model. According to the results, the hello flood attack was correctly classified to the highest accuracy with almost no false alarms raised.

6. CONCLUSION

As global usage of the IoT increases rapidly, the protection of IoT networks has become a significant challenge. Many IoT applications collect large amounts of data from various devices. In most cases, the data within the IoT can often include confidential device-related information. Moreover, most of the devices which need to connect to the Internet for specific purposes also have resource constraints related to battery, connectivity, and size. Today's IoT networks consist of many different devices such as smartphones, wireless sensors, and wearable devices. This also raises challenges in trying to apply traditional complex security mechanisms. Furthermore, the existing security solutions are not suited to such heterogeneous and complex networks. Therefore, IoT networks require new, simpler solutions that are based on lower complexity but higher effectiveness. In the current study, a novel cross-layer IDS based on neural networks was introduced. Both binary classification and multiclass classification were then applied, plus features from both the link layer and network layer employed within the design. To the best of the researcher's knowledge, the current study presents the first cross-layer IDS for RPL-based networks that explores the effect of features obtained from the link layer on intrusion detection.

For the evaluation of the proposed IDS, a dataset of routing attacks that consisted of "version number attacks," "worst parent attacks," and "hello flood attacks" was constructed by creating simulated networks with the Cooja 2.7 network simulator. The network analyses were then performed with one Root Node and 50 Sender Nodes for a duration of 60 minutes. Also, four different attacker densities were used for each attack class. The results provided an opportunity to observe the effect of different routing attacks on an RPL network with varying attacker density. Also, the dataset with three types of routing attack was created with information extracted from the network log files.

After designing the cross-layer neural-network-based IDS, the proposed design was trained and then tested by using the routing attack dataset. According to the results, the proposed IDS has a higher detection rate (96.88%) and a low FPR (0.13%) for binary classification. For measuring the accuracy of the IDS, it was also separately evaluated using each of the three attack types. The proposed method detected the "version number attack" with a detection rate of 93.20%. For the "worst parent attack," the detection rate achieved was 98.17% with different attacker densities. The "hello flood attack" was detected by the proposed IDS at a rate of 99.96%. This showed that “hello flood attacks” have distinctive characteristics, which makes their detection easier than the other two routing attacks types.

As previously mentioned, both link-layer-based and network-layer-based features were used in the design. In order to investigate whether or not the link-layer activity of the nodes provided clues about an attacker, the training and testing procedures were also applied without the link-layer features. Without the extracted link-layer features, the proposed detection system returned a false positive rate of 0.61%. However, by including the link-layer features, the false positive rate dropped down to 0.13%, representing a significant improvement introduced from the proposed intrusion detection model.

The proposed IDS was also evaluated for multiclass classification. According to the experimental results, the proposed cross-layer IDS also achieved a high level of accuracy (97.52%) for multiclass classification. The “no attack” and “hello flood attack” cases were classified with the highest ratios, whilst the “version number attack” detection produced the lowest ratio of detection (92.42%). At the start of the simulated test, when there was a “version number attack” in progress in the network, it was not recognized and “no attack” was noted by the proposed IDS (6.94%). Finally, the “worst parent attack” was also detected with a high degree of accuracy (97.71%).

For future work in this area, the researcher plans to develop a neural-network-based IDS by adding new routing attack sets to the dataset developed in the current study. The ultimate aim being to model an intrusion detection system which can predict the attacker type with a high degree of detection accuracy, whilst maintaining a FPR as low as possible. Also, to increase the detection ability of the neural network, new features will be attempted to be added to the dataset.

REFERENCES

[1] P.P. Ray. A survey on internet of things architectures. *EAI Endorsed Transactions on Internet of Things*, 2:151714, **2016**. doi:10.4108/eai.1-12-2016.151714.

[2] Carsten Maple. Security and privacy in the internet of things. *Journal of Cyber Policy*, 2:155–184, **2017**. doi:10.1080/23738871.2017.1366536.

[3] Iot:number of connected devices wworldwide 2012-2025—statistica. `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`. Accessed: 2020-01-08.

[4] Omprakash Gnawali and P Levis. The Minimum Rank with Hysteresis Objective Function. RFC 6719, **2012**. doi:10.17487/RFC6719.

[5] Anthéa Mayzaud, Anuj Sehgal, Remi Badonnel, Isabelle Chrisment, and Jürgen Schönwälder. A study of rpl dodag version attacks. In *AIMS*. **2014**.

[6] Ahmet Arış, Sema Oktug, and Berna Ors. Rpl version number attacks: In-depth study. pages 776–779. **2016**. doi:10.1109/NOMS.2016.7502897.

[7] Anhtuan Le, Jonathan Loo, A. Lasebae, Alexey Vinel, Yue Chen, and Michael Chai. The impact of rank attack on network topology of routing protocol for low-power and lossy networks. *Sensors Journal, IEEE*, 13:3685–3692, **2013**. doi:10.1109/JSEN.2013.2266399.

[8] Abhishek Verma and Virender Ranga. Mitigation of dis flooding attacks in rpl-based 6lowpan networks. *Transactions on Emerging Telecommunications Technologies*, pages 1–25, **2019**. doi:10.1002/ett.3802.

[9] Hyung-Sin Kim, Jeonggil Ko, David E Culler, and Jeongyeup Paek. Challenging the ipv6 routing protocol for low-power and lossy networks (rpl): A survey. *IEEE Communications Surveys & Tutorials*, 19(4):2502–2525, **2017**.

[10] Bruno Bogaz Zarpelão, Rodrigo Sanches Miani, Cláudio Toshio Kawakani, and Sean Carlisto de Alvarenga. A survey of intrusion detection in internet of things. *Journal of Network and Computer Applications*, 84:25 – 37, **2017**. ISSN 1084-8045. doi:https://doi.org/10.1016/j.jnca.2017.02.009.

[11] Hamid Bostani and Mansour Sheikhan. Hybrid of anomaly-based and specification-based ids for internet of things using unsupervised opf based on mapreduce approach. *Computer Communications*, 98:52 – 71, **2017**. ISSN 0140-3664. doi:https://doi.org/10.1016/j.comcom.2016.12.001.

[12] Furkan Yusuf YAVUZ, Devrim ÜNAL, and Ensar GÜL. Deep learning for detection of routing attacks in the internet of things. *International Journal of Computational Intelligence Systems*, 12:39–58, **2018**. ISSN 1875-6883. doi: https://doi.org/10.2991/ijcis.2018.25905181.

[13] Ieee standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (lr-wpans). *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, pages 1–314, **2011**. ISSN null. doi:10.1109/IEEESTD.2011.6012487.

[14] Gabriel Montenegro, Christian Schumacher, and Nandakishore Kushalnagar. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919, **2007**. doi: 10.17487/RFC4919.

[15] Gabriel Montenegro, Jonathan Hui, David Culler, and Nandakishore Kushalnagar. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, **2007**. doi:10.17487/RFC4944.

[16] Pascal Thubert and Jonathan Hui. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282, **2011**. doi:10.17487/RFC6282.

[17] Roger Alexander, Anders Brandt, JP Vasseur, Jonathan Hui, Kris Pister, Pascal Thubert, P Levis, Rene Struik, Richard Kelsey, and Tim Winter. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, **2012**. doi: 10.17487/RFC6550.

[18] Zach Shelby, Klaus Hartke, and Carsten Bormann. The Constrained Application Protocol (CoAP). RFC 7252, **2014**. doi:10.17487/RFC7252.

[19] P Levis, Thomas H. Clausen, Omprakash Gnawali, Jonathan Hui, and JeongGil Ko. The Trickle Algorithm. RFC 6206, **2011**. doi:10.17487/RFC6206.

[20] Anthéa Mayzaud, Rémi Badonnel, and Isabelle Chrisment. A Taxonomy of Attacks in RPL-based Internet of Things. *International Journal of Network Security*, 18(3):459 – 473,, **2016**.

[21] Shahid Raza, Linus Wallgren, and Thiemo Voigt. Svelte: Real-time intrusion detection in the internet of things. *Ad Hoc Networks*, 11(8):2661 – 2674, **2013**. ISSN 1570-8705. doi:https://doi.org/10.1016/j.adhoc.2013.04.014.

[22] Faiza Medjek, D. Tandjaoui, Imed Romdhani, and Djedjig Nabil. A trust-based intrusion detection system for mobile rpl based networks. **2017**. doi:10.1109/iThings-GreenCom-CPSCom-SmartData.2017.113.

[23] M. N. Napiah, M. Y. I. Bin Idris, R. Ramli, and I. Ahmedy. Compression header analyzer intrusion detection system (cha - ids) for 6lowpan communication protocol. *IEEE Access*, 6:16623–16638, **2018**.

[24] Nicolas M. Müller., Pascal Debus., Daniel Kowatsch., and Konstantin Böttinger. Distributed anomaly detection of single mote attacks in rpl networks. In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications - Volume 2: SECRYPT,*, pages 378–385. INSTICC, SciTePress, **2019**. ISBN 978-989-758-378-0. doi:10.5220/0007836003780385.

[25] Emre Aydoğan, Selim Yılmaz, Sevil Sen, Ismail Butun, Stefan Forsström, and Mikael Gidlund. A central intrusion detection system for rpl-based industrial internet of things. pages 1–5. **2019**. doi:10.1109/WFCS.2019.8758024.

[26] Elie Kfoury, Julien Saab, Paul Younes, and Roger Achkar. A self organizing map intrusion detection system for rpl protocol attacks. *International Journal of Interdisciplinary Telecommunications and Networking*, 11:30–43, **2019**. doi:10.4018/IJITN.2019010103.

[27] Anhtuan Le, Jonathan Loo, Michael Chai, and Mahdi Aiash. A specification-based ids for detecting attacks on rpl-based network topology. *Information*, 7, **2016**. doi:10.3390/info7020025.

[28] A. Verma and V. Ranga. Elnids: Ensemble learning based network intrusion detection system for rpl based internet of things. In *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pages 1–6. **2019**.

[29] A. Verma and V. Ranga. Security of rpl based 6lowpan networks in the internet of things: A review. *IEEE Sensors Journal*, pages 1–1, **2020**.

[30] Anhtuan Le, Jonathan Loo, Yuan Luo, and A. Lasebae. The impacts of internal threats towards routing protocol for low power and lossy network performance. pages 000789–000794. **2013**. ISBN 978-1-4799-3755-4. doi:10.1109/ISCC.2013.6755045.

[31] Ahmet Arış, Sıddıka Yalçın, and Sema Oktug. New lightweight mitigation techniques for rpl version number attacks. *Ad Hoc Networks*, 85, **2018**. doi:10.1016/j.adhoc.2018.10.022.

[32] Amit Dvir, Tamás Holczer, and Levente Buttyán. Vera - version number and rank authentication in rpl. pages 709–714. **2011**. doi:10.1109/MASS.2011.76.

[33] Heiner Perrey, Martin Landsmann, Osman Ugus, Thomas Schmidt, and Matthias Wählisch. Trail: Topology authentication in rpl. **2013**.

[34] Ghada Glissa, Abderrezak Rachedi, and Aref Meddeb. A secure routing protocol based on rpl for internet of things. **2016**. doi:10.1109/GLOCOM.2016.7841543.

[35] Anthea Mayzaud, Remi Badonnel, and I. Chrisment. A distributed monitoring strategy for detecting version number attacks in rpl-based networks (invited paper). *IEEE Transactions on Network and Service Management*, PP:1–1, **2017**. doi:10.1109/TNSM.2017.2705290.

[36] David Airehrour, Jairo A. Gutierrez, and Sayan Kumar Ray. Sectrust-rpl: A secure trust-aware rpl routing protocol for internet of things. *Future Generation Computer Systems*, 93:860 – 876, **2019**. ISSN 0167-739X. doi:https://doi.org/10.1016/j.future.2018.03.021.

[37] Joydeep Tripathi, Jaudelice De Oliveira, and Jp Vasseur. A performance evaluation study of rpl: Routing protocol for low power and lossy networks. pages 1–6. **2010**.

[38] The wireshark network protocol analyzer. `https://www.wireshark.org`. Accessed: 2020-05-09.

[39] Fredrik Osterlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with cooja. *Local Computer Networks, Annual IEEE Conference on*, 0:641–648, **2006**. doi:10.1109/LCN.2006.322172.

[40] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. Deep learning in mobile and wireless networking: A survey. *IEEE Communications Surveys Tutorials*, PP, **2018**. doi:10.1109/COMST.2019.2904897.

[41] Alessio Zappone, Marco Di Renzo, and Mérouane Debbah. Wireless networks design in the era of deep learning: Model-based, ai-based, or both? *IEEE Transactions on Communications*, 67:7331–7376, **2019**.

[42] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, **2011**.

[43] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, **2010**.

[44] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, **2006**.

[45] François Chollet et al. Keras. `https://keras.io`, **2015**.

[46] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283. **2016**.

[47] Howard B. Demuth, Mark H. Beale, Orlando De Jess, and Martin T. Hagan. *Neural Network Design*. Martin Hagan, Stillwater, OK, USA, 2nd edition, **2014**. ISBN 0971732116.

[48] Jeff Heaton. *Introduction to Neural Networks for Java, 2nd Edition*. Heaton Research, Inc., 2nd edition, **2008**. ISBN 1604390085.

[49] Jeffrey Heaton. The number of hidden layers. *Heaton Research Inc*, **2008**.

[50] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, **2016**. `http://www.deeplearningbook.org`.

[51] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, **2014**.

[52] Pierre Baldi and Peter J Sadowski. Understanding dropout. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2814–2822. Curran Associates, Inc., **2013**.

[53] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, **2014**.

[54] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, page 1137–1143. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, **1995**. ISBN 1558603638.

[55] András Varga. The omnet++ discrete event simulation system. *Proc. ESM'2001*, 9, **2001**.

[56] Netsim-network simulator emulator. `https://www.tetcos.com/index.html`. Accessed: 2020-01-10.

[57] ns-3, a discrete-event network simulator for internet systems. `https://www.nsnam.org`. Accessed: 2020-01-10.

[58] Tmote sky from moteiv. `https://insense.cs.st-andrews.ac.uk/files/2013/04/tmote-sky-datasheet.pdf`. Accessed: 2020-01-13.

[59] Pascal Thubert. Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL). RFC 6552, **2012**. doi:10.17487/RFC6552.

[60] Nurrahmat Pradeska, Widyawan Widyawan, Warsun Najib, and Sri Kusumawardani. Performance analysis of objective function mrhof and of0 in routing protocol rpl ipv6 over low power wireless personal area networks (6lowpan). pages 1–6. **2016**. doi:10.1109/ICITEED.2016.7863270.

[61] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, MobiCom '03, page 134–146. Association for Computing Machinery, New York, NY, USA, **2003**. ISBN 1581137532. doi:10.1145/938985.939000.

[62] Hyunjae Kang; Dong Hyun Ahn; Gyung Min Lee; Jeong Do Yoo; Kyung Ho Park; Huy Kang Kim. Iot network intrusion dataset, **2019**. doi:10.21227/q70p-q449.

[63] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). **2015**. doi:10.1109/MilCIS.2015.7348942.

[64] Alex, bahmadh, and MatM-G. dhondta/rpl-attacks: Multiple bugfixes doc refinement. **2018**. doi:10.5281/zenodo.1694632.

[65] tshark - the wireshark network analyzer 3.2.0. `https://www.wireshark.org/docs/man-pages/tshark.html`. Accessed: 2019-12-29.