Monitoring and Security for the RPL-based Internet of Things

Anthéa Mayzaud

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Monitoring and Security for the RPL-based Internet of Things

Supervision et Sécurité pour l'Internet des Objets utilisant le protocole de routage RPL

THÈSE

présentée et soutenue publiquement le 21/10/2016

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Anthéa Mayzaud

Composition du jury

Président :	Olivier Perrin	Professeur à l'Université de Lorraine
Rapporteurs :	Maryline Laurent	Professeur à Télécom SudParis
	Michele Nogueira	Professeur associé à l'Université fédérale de Paraná, Brésil
Examinateurs :	Vincent Nicomette	Professeur à INSA Toulouse
	Jürgen Schönwälder	Professeur à l'Université Jacobs de Brême, Allemagne
Encadrants :	Isabelle Chrisment	Professeur à TELECOM Nancy, Université de Lorraine
	Rémi Badonnel	Maître de conférences à TELECOM Nancy, Université de Lorraine

Inria Nancy Grand-Est

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

À ma famille,
À mes amis.

Remerciements

Cette thèse n'aurait pas pu voir le jour sans le concours et le soutien de certaines personnes que je tiens à remercier ici.

Tout d'abord je souhaite remercier Maryline Laurent et Michele Nogueira pour le temps qu'elles ont consacré à la relecture de cette thèse ainsi que tous les membres du jury pour l'intérêt qu'ils portent à mes travaux.

Je tiens ensuite à remercier Isabelle Chrisment ma directrice de thèse qui m'a offert cette grande opportunité de réaliser ma thèse au sein de l'équipe Madynes ainsi que Rémi Badonnel, mon co-encadrant de thèse. Je les remercie pour leurs conseils, leur patience et leur bienveillance durant ces trois années et demi.

I would also like to thank the Flamingo European network of excellence and all its members. This European project allowed me to share and exchange in a very productive way with its members. In particular, I want especially thank Jürgen Schönwälder, Professor at Jacobs University Bremen and its former student Anuj Sehgal who welcomed me in Bremen and taught me a lot.

Plusieurs personnes m'ont également apporté leur expertise tout au long de cette thèse et je tiens à les remercier: Bernardetta Addis, César Bernardini, Alexandre Boeglin, Thibault Cholez, François Despuaux, Gaëtan Hurel, Meihui Gao, Abdelkader Lahmadi, Emmanuel Nataf, Kévin Rousselle, Evangelia Tsiontsiou et tous les autres membres de l'équipe qui m'ont conseillé. Je tiens aussi à remercier d'autres personnes qui m'ont aidé techniquement durant cette thèse: Thibaut Delarozière, Pierre Kimmel et Sébastien Parisot.

Je remercie tous mes collègues qui m'ont permis de travailler dans une bonne ambiance durant ces trois années et notamment: Elian Aubry, Eric Finickel pour ceux que je n'ai pas déjà cité ainsi que tous les autres membres de l'équipe. Je remercie tout particulièrement Thibault Cholez qui a été celui qui m'a donné envie de suivre cette voie lors de sa soutenance de thèse il y a quelques années et à laquelle j'ai pu assister alors stagiaire dans l'équipe.

Finalement je remercie ma famille et mes amis qui m'ont soutenu, encouragé et supporté durant ces années de thèse, je ne fais pas de liste mais ils sont tous dans mon cœur et mes pensées.

Contents

1 Introduction **1**

1.1 Context 1

1.1.1 The Internet of Things 2

1.1.2 Low Power Lossy Networks and Routing Protocols 3

1.2 Problem Statement 5

1.2.1 Security Issues 5

1.2.2 Addressed Challenges 6

1.3 Overview of Contributions 6

2 Routing and Monitoring in RPL-based Internet of Things **9**

2.1 Introduction 9

2.2 The RPL Protocol 10

2.2.1 RPL Control Messages 11

2.2.2 DODAG Building and Maintenance 12

2.2.3 Loops, Inconsistencies and Repairs 12

2.2.4 Protocol Security 13

2.3 Monitoring RPL-based Internet of Things 14

2.3.1 Active Monitoring 14

2.3.2 Passive Monitoring 19

2.3.3 Comparison and Limits 21

2.4 Conclusions 23

3 Taxonomy of Attacks in RPL Networks **25**

3.1 Introduction 25

3.2 Attacks against Resources 26

3.2.1 Direct Attacks 26

3.2.2 Indirect Attacks 27

3.2.3 Analysis . . . 30
3.3 Attacks on Topology . . . 31
3.3.1 Sub-optimization Attacks . . . 32
3.3.2 Isolation Attacks . . . 34
3.3.3 Analysis . . . 35
3.4 Attacks on Traffic . . . 37
3.4.1 Eavesdropping Attacks . . . 37
3.4.2 Misappropriation Attacks . . . 38
3.4.3 Analysis . . . 39
3.5 Conclusions . . . 40

4 Impact Assessment of RPL Attacks 43
4.1 Introduction . . . 43
4.2 The DAG Inconsistency Attack . . . 44
4.2.1 Attack Description . . . 44
4.2.2 Simulation Setup . . . 46
4.2.3 Impact Quantification . . . 46
4.3 The Version Number Attack . . . 48
4.3.1 Attack Description . . . 48
4.3.2 Simulation Setup . . . 49
4.3.3 Impact Quantification . . . 50
4.4 Conclusions . . . 55

5 Local Strategy for Addressing DAG Inconsistency Attack 57
5.1 Introduction . . . 57
5.2 DAG Inconsistency Attack Mitigation . . . 58
5.2.1 Default Mitigation . . . 58
5.2.2 Adaptive Mitigation . . . 59
5.2.3 Dynamic Mitigation . . . 60
5.3 Mitigation Evaluation . . . 62
5.3.1 Simulation Setup . . . 63
5.3.2 Mitigation Performance . . . 64
5.3.3 Configuration Parameters Impact . . . 69
5.3.4 Resource Consumption . . . 73
5.4 Conclusions . . . 76

6 Security-oriented Distributed Monitoring Architecture 79
6.1 Introduction . . . 79
6.2 Proposed Architecture . . . 81
6.2.1 Overview and Components . . . 81
6.2.2 RPL-based Mechanisms . . . 82
6.3 Monitoring Node Placement Formalization . . . 85
6.4 Detection Modules . . . 86
6.4.1 DAG Inconsistency Attack . . . 87
6.4.2 Version Number Attack . . . 89
6.5 Conclusions . . . 93

7 Architecture Evaluation 95
7.1 Introduction . . . 95
7.2 Overhearing Evaluation . . . 96
7.2.1 Simulation Setup . . . 96
7.2.2 Performance Analysis . . . 97
7.2.3 Cost Analysis . . . 99
7.3 Detection Modules Evaluation . . . 100
7.3.1 DAG Inconsistency Attack . . . 101
7.3.2 Version Number Attack . . . 104
7.4 Scalability Evaluation . . . 109
7.5 Conclusions . . . 113

8 General Conclusions 115
8.1 Achievements . . . 115
8.2 Perspectives . . . 117

Publications 121

List of Figures 123

List of Tables 127

Résumé de la thèse en français 129
1 Introduction . . . 129
2 Protocole de routage RPL . . . 130

3 Taxonomie des attaques contre le protocole RPL 132
4 Analyse d'attaques visant le protocole RPL 132
4.1 Attaque d'incohérence DAG 132
4.2 Attaque sur le numéro de version 133
5 Détection locale d'attaques d'incohérence DAG 134
6 Architecture de supervision distribuée pour la sécurité 135
7 Évaluation de l'architecture 137
8 Conclusions 138

Bibliography **141**

Glossary **149**

Chapter 1

Introduction

This thesis on monitoring and security for the RPL-based Internet of Things was carried out as part of Flamingo[1], a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Program. The Flamingo project focuses on network service and management for the Future Internet. This European project gave us the opportunity to collaborate with the Jacobs University Bremen, Germany, especially with Anuj Sehgal and Jürgen Schönwälder, Professor in the CNDS research group.

1.1 Context

The Internet of Things (IoT) is a paradigm that is increasingly growing in the context of pervasive networks and services. It consists in the extension of the Internet to objects from the physical world, which are interacting with each other in order to reach common goals in many application domains (see Figure 1.1). The high interest for this paradigm has resulted in the large-scale deployment of Low power and Lossy Networks (LLN), such as wireless sensor networks and home automation systems. These networks suffer from scarce resources and unreliable links. As an effort to standardize protocols used in the IoT, a dedicated stack has been designed having in mind all these constraints. In particular for the routing layer, the IETF RoLL[2] working group has proposed a new protocol called RPL (Routing Protocol for Low power and Lossy Networks) based on IPv6 and specifically designed for Internet of Things networks [80].

We propose in this thesis to address security monitoring issues regarding the RPL routing protocol in the context of IoT networks. Indeed, the multiple constraints faced by these networks make them particularly vulnerable to security threats. Addressing security in such environments is a real challenge considering all their limitations. Therefore, the proposed solutions have to be as lightweight as possible in order to achieve the best trade-off between security and its induced cost for the network.

[1] http://www.fp7-flamingo.eu/

[2] Routing over Low power and Lossy networks

This introduction gives the context and the research questions of this thesis. First, we present the Internet of Things paradigm and the induced Low power and Lossy networks with their routing requirements. We then detail the challenges of this new paradigm and more specifically the issues addressed in this thesis. We finally provide an overview of the contributions.

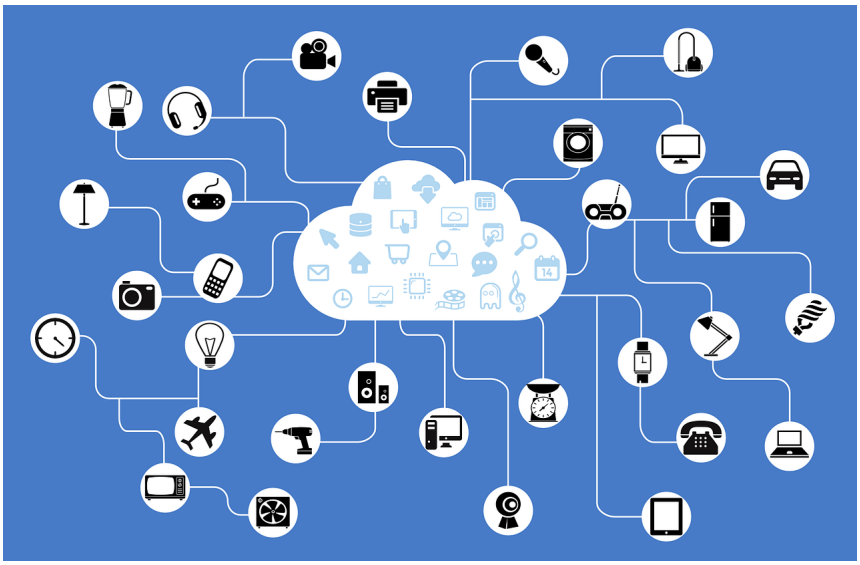


Figure 1.1: Extension of the Internet to everyday objects[3].

1.1.1 The Internet of Things

The Global Standards Initiative on Internet of Things (IoT-GSI) has defined the IoT as a global infrastructure for the information society which enables advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies [3]. This technology has been considered by the US National Intelligence Council (NIC) as one of the "*Disruptive Civil Technologies*" that will significantly impact the United States [58]. The NIC envisions that by 2025 Internet nodes will reside in everyday things such as food packages, furniture, paper documents. Furthermore, the Gartner group forecasts that 6.4 billions of objects will be deployed this year and 20.8 billions will be used by 2020[4]. The IoT enables huge number of new applications in various domains such as detailed in [7]:

- Smart environments,
- Transportation and logistics,
- Healthcare.

[3]Sources: pixabay.com, Creative Commons 0
[4]http://www.gartner.com/newsroom/id/3165317

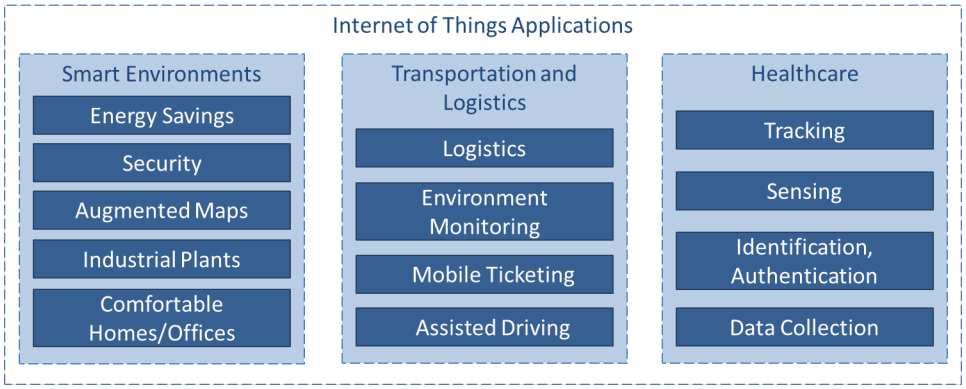


Figure 1.2: Example of IoT applications (based on [7]).

Figure 1.2 presents some applications for these domains. For example, in smart environments, IoT technologies permit to save energy by automatically turning off lights and heating when the building is empty, or provide security by monitoring areas at night. In logistics, the IoT enables real-time information processing technology which supports monitoring of products, raw materials and so on. The IoT also opens new possibilities in the healthcare domain by allowing assisted living for elderly or ill persons for instance. The deployed objects from the Internet of Things form specific networks called Low power and Lossy Networks (LLNs) which present particular characteristics.

1.1.2 Low Power Lossy Networks and Routing Protocols

LLN networks have strong constraints in terms of resources (energy, memory, processing) and their communication links are by nature characterized by a high loss rate and a low throughput. Even though several classes of devices can be employed in these networks, as described in Table 1.1, the available computing resources are quite minimal, when compared to standard computing devices used in most applications today. This means that protocols for the Internet of Things must operate within the resource constraints implied by these devices. Moreover, the traffic patterns are not simply defined according to a point-to-point schema. In many cases, the devices may also communicate according to point-to-multipoint and multipoint-to-point schemas.

Table 1.1: Classes of constrained devices used in the Internet of Things (IoT) [11].

Device Classes	RAM	ROM
C0	$<$ 10 KiB	$<$ 100 KiB
C1	$\sim$ 10 KiB	$\sim$ 100 KiB
C2	$\sim$ 50 KiB	$\sim$ 250 KiB

Several proprietary initiatives such as ZigBee[5] or Z-Wave[6] have been proposed to deal with LLN constraints. However, standards have to be designed in order to ensure interoperability among the variety of devices and objects. Regarding the routing layer, the IETF RoLL working group has therefore performed a survey of existing routing protocols specified in RFCs or mature drafts in order to answer whether any IETF standardized protocol can meet LLN requirements [47]. They have considered the following routing protocols: OSPF [57], IS-IS [6], RIP [50], OLSR [33], TBRPF [74], AODV [22], DSR [51], DYMO [15], OLSRv2 [20] and Triggered RIP [56]. The comparison has been performed using five criteria:

- **Routing state** indicates whether routing state scales reasonably within the memory resources of low-power nodes,
- **Loss response** indicates how the considered routing protocol deals with link failures and recompute paths,
- **Control cost** indicates if the considered routing protocol minimize power consumption regarding required control traffic.
- **Link cost** refers to the ability for a protocol to incorporate link properties into routing metrics,
- **Node cost** refers to the ability for a protocol to incorporate router properties into routing metrics and use node attributes for constraint-based routing.

Table 1.2: Protocol comparison results [47].

Protocol	Routing state	Loss response	Control cost	Link cost	Node cost
OSPF/IS-IS	fail	fail	fail	pass	fail
OLSRv2	fail	?	?	pass	pass
TBRPF	fail	pass	fail	pass	?
RIP	pass	fail	pass	?	fail
AODV	pass	fail	pass	fail	fail
DYMO	pass	?	pass	?	?
DSR	fail	pass	pass	fail	fail

Table 1.2 gathers obtained results regarding aforementioned criteria. For each criteria, pass indicates that a given protocol has satisfactory performance according to the considered criterion; fail meaning the opposite. The value ? indicates lacks in the protocol so that authors could not conclude if the test has succeeded regarding the criterion. We can observe that no existing routing protocols meet requirements of LLN networks as defined by the RoLL working group. As a result, RoLL has proposed a new proactive routing protocol called RPL standardized by RFC 6550 [80].

[5] http://www.zigbee.org/
[6] http://www.z-wave.com/modules/ZwaveStart/

Another effort regarding routing in LLN networks has also been proposed: the Lightweight On-demand Ad hoc Distance-vector - Next Generation routing protocol (LOADng) [49]. This protocol is a simplified version of the AODV protocol which has been extended to be used in Mobile Ad hoc NETworks (MANETs). Unlike the RPL protocol, LOADng is a reactive protocol which means that routes are built on demand by nodes which want to reach another node in the network. As such these solutions inherit advantages and disadvantages of these categories of routing protocols and their use must be chosen according to the considered scenarios. For instance, authors of [77] have compared the performance of the RPL and LOADng protocols in home automation scenarios. Their conclusion is that for applications in which the response time is important, RPL has performed better than LOADng even if some implementation improvements could be done regarding this protocol. Also in [29], authors have evaluated the LOADng protocol for bidirectional data flow in AMI (Advanced Measurement Infrastructure) mesh networks. Their study has showed that LOADng had better results than AODV protocol in all considered cases, however, RPL has outperformed them. Nonetheless, authors have pointed out that the RPL protocol has a more complex processing compared to their LOADng implementation. We also want to highlight the fact that the LOADng protocol is still currently an IETF draft whereas the RPL protocol is an IETF standard.

1.2 Problem Statement

The manifest importance of the Internet of Things opens new challenges for both industry and academia. One of the main challenges relates to security. We first present in this section major questions related to security in the Internet of Things. We then describe more specifically challenges addressed in this thesis.

1.2.1 Security Issues

The extension of the Internet to daily objects introduces a new major attack vector for IT technologies as confirmed by the NIC which stated that to the extent that everyday objects become information-security risks, the IoT could distribute those risks far more widely than the Internet has to date. As an example of this statement, in December 2013, a researcher at Proofpoint company has discovered a botnet not only composed of computers but also of household appliances, smart TVs and even a refrigerator[7]. These types of botnets are now referred as thingbots and show the lack of security which IoT networks suffer. The Gartner group also forecasts that worldwild IoT security spend will reach 348$ million in 2016 and 547$ million in 2018[8]. They foresee that more than 25% of identified attacks in enterprises will involve IoT infrastructures by 2020.

Several reasons can explain why the IoT is extremely vulnerable to attacks. First, limited resources of typical IoT devices prevent them from implementing traditional

[7] http://investors.proofpoint.com/releasedetail.cfm?releaseid=819799
[8] http://www.gartner.com/newsroom/id/3291817

security mechanisms usually deployed in Internet infrastructures. Second, deployments of things in many applications make these devices unguarded therefore allowing physical attacks. Third, as communications in the IoT are mostly wireless, they are exposed to eavesdropping attacks. Finally, the probably most important reason is about human interactions with these things. Indeed end users usually have little incentive to make IoT devices secure by changing device passwords for example, and most of vendors do not consider security as a key feature for their products. Some efforts have been made in order to design security solutions as presented in [7]. However, they all rely on cryptographic methods which take away resources and drastically affects the performance of constrained devices [72] likely to be used in IoT and WSN applications.

1.2.2 Addressed Challenges

The objective of this thesis is to design, implement and evaluate new strategies able to address security monitoring in the RPL-based Internet of Things. The RPL protocol is the IETF routing protocol standard for IoT networks. The proposed solutions should detect behaviors of malicious nodes in order to limit their effects.

The first challenge of this thesis is therefore to assess security in RPL networks by identifying and characterizing threats targeting the RPL protocol. We also classify them using several factors in order to stress threats to be addressed in preference and quantify their impact.

The second challenge consists in designing monitoring security solutions for previously identified attacks which minimize resource consumption of nodes deployed in the RPL network. These solutions should exploit the mechanisms and typical IoT deployments such as the RPL protocol features or heterogeneity of IoT networks in order to preserve node energy. We carefully evaluate their efficiency and their cost regarding the considered security attacks.

1.3 Overview of Contributions

We tackle in this thesis security monitoring for the RPL-based Internet of Things as depicted by Figure 1.3. As a first step we assess security threats targeting the RPL protocol. We then propose security solutions. These two main aspects are organized into specific chapters and are preceded by a state of the art on routing and monitoring in the RPL-based Internet of Things. General conclusions end this manuscript and point out future research perspectives.

State of the Art

Chapter 2 details the RPL protocol functioning from topology building and maintenance to inner repair mechanisms. We also present the main security concerns to which this protocol is exposed to. We provide a state of the art of existing approaches for monitoring and security in RPL-based IoT networks. They are classified

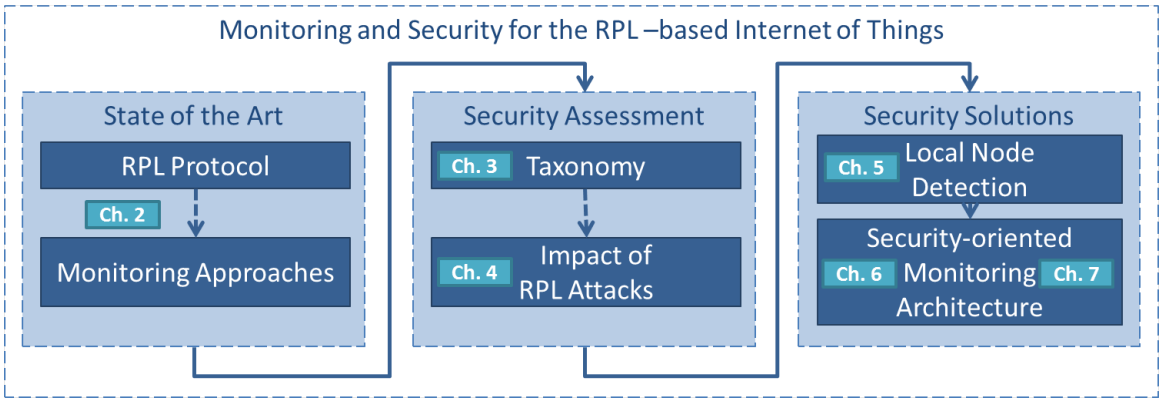


Figure 1.3: Road map of the contributions.

according to several criteria (active/passive, centralized/decentralized) which allows us to discuss their benefits and drawbacks in such environments.

Security Assessment

Chapter 3 introduces our taxonomy of the attacks against the RPL protocol considering three main categories including attacks targeting network resources, attacks modifying the topology and attacks related to the traffic. We describe these attacks, analyze and compare their properties, and discuss existing counter-measures [52].

Chapter 4 details and quantifies the consequences of two attacks exploiting RPL mechanisms: the DAG inconsistency attack and the version number attack [53, 71]. These attacks are chosen because they target node resources accordingly to the established taxonomy. The obtained results show the importance of addressing them because they can significantly shorten the network lifetime.

Security Solutions

Chapter 5 proposes different local mitigation approaches for DAG inconsistency attacks. A first solution relying on a fixed threshold has been introduced by the RoLL working group [32]. We show the limits of this solution and introduce new approaches called adaptive threshold and dynamic threshold [71, 54]. Performance of these mitigation solutions is evaluated through experiments. We also quantify their costs and benefits. Besides, we explain why such node-level approach cannot be used to detect or mitigate the version number attack because of its properties.

Chapter 6 presents our security-oriented distributed monitoring architecture which detects complex attacks such as the version number attack and complement our node-level approach. This architecture is composed of regular nodes and monitoring nodes and exploits RPL mechanisms to organize them. We detail our detection strategy which relies on dedicated modules deployed on monitoring nodes in order to detect both DAG inconsistency and version number attacks.

Chapter 7 evaluates this architecture and the detection strategy [55]. First, we quantify the performance of the overhearing mode of monitoring nodes in our simulation environment. We then study the detection results for our strategy. The monitoring nodes placement is also discussed through formalization of optimization problems.

Chapter 2

Routing and Monitoring in RPL-based Internet of Things

Contents

2.1 Introduction 9
2.2 The RPL Protocol 10
2.2.1 RPL Control Messages 11
2.2.2 DODAG Building and Maintenance 12
2.2.3 Loops, Inconsistencies and Repairs 12
2.2.4 Protocol Security 13
2.3 Monitoring RPL-based Internet of Things 14
2.3.1 Active Monitoring 14
2.3.2 Passive Monitoring 19
2.3.3 Comparison and Limits 21
2.4 Conclusions 23

2.1 Introduction

The Internet of Things specifics require new methods to perform routing and monitoring. In particular, the Routing Protocol for Low-power Lossy Networks (RPL) was designed by the IETF RoLL[9] working group, with capabilities of resource constrained nodes in mind [80]. This protocol is expected to form the basis of many Internet of Things (IoT) applications. It is integrated in a full standard protocol stack designed for the IoT as illustrated by Figure 2.1. The IEEE 802.15.4 protocol is used for both physical and link layers of wireless personal area networks (WPAN) and is appropriate for low power consumptions, short communication ranges and low flow rates. The 6LowPAN protocol also defines encapsulation and header compression mechanisms allowing IPv6 packets to be sent or received through the IEEE

[9] Routing over LLNs

802.15.4 communication protocol; it is thus used as an adaptation layer. The RPL routing protocol is capable to operate over the two previously mentioned protocols.

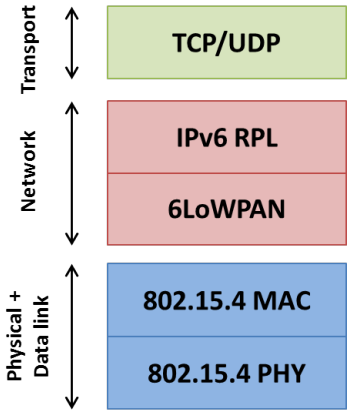


Figure 2.1: Overview of the IoT protocol stack.

Internet of Things infrastructures require lightweight methods and techniques to observe network devices and make sure services are properly running. Such monitoring information provides an important data source to identify failures, optimize the network functioning, and detect potential security attacks. In the meantime, monitoring such a type of networks should consume a minimal amount of resources to preserve node operation capacity.

Section 2.2 describes the routing protocol for LLNs (RPL), its functioning and its security mechanisms. In Section 2.3, we present and compare existing approaches dedicated to monitoring and security for RPL-based IoT.

2.2 The RPL Protocol

The RPL protocol is a distance-vector routing protocol based on IPv6. RPL devices are interconnected according to a specific topology which combines mesh and tree topologies called Destination Oriented Directed Acyclic Graphs (DODAG). A DODAG graph is built from a root node which is the data sink of the graph. A network can operate one or more RPL instances which consist of multiple DODAG graphs as showed in Figure 2.2. Each RPL instance is associated to an objective function which is responsible for calculating the best path depending on a set of metrics and/or constraints. For instance, this function can minimize energy consumption or simply compute the shortest path. A RPL node can join several instances at the same time, but it can only join one DODAG graph per instance such as nodes 13 and 17 in Figure 2.2. These multiple instances enable the RPL protocol to perform different optimizations, such as quality-of-service ones. The RPL packets can be forwarded according to three traffic patterns as showed in the third DODAG of Figure 2.2: (i) multipoint-to-point traffic (MP2P) from leaves to the root via upward routes; (ii) point-to-multipoint traffic (P2MP) from the root to leaves us-

ing downward routes; and (iii) point-to-point traffic (P2P) illustrated by red doted arrows using both upward and downward routes.

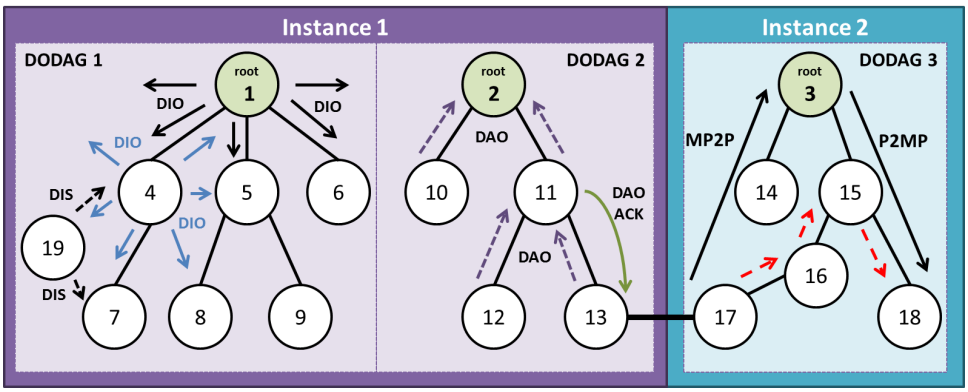


Figure 2.2: Example of a RPL network composed of two instances and three DODAGs.

We detail below the functioning and the main features of the RPL protocol. Section 2.2.1 presents the different control messages introduced by the RPL protocol. Section 2.2.2 explains how these messages are used to build and maintain the DODAG structures. We then describe loops and inconsistencies issues that may occur in RPL networks along with the RPL repair mechanisms designed to counter them in Section 2.2.3. Finally, Section 2.2.4 introduces security concerns regarding the RPL protocol.

2.2.1 RPL Control Messages

The RPL protocol defines four new ICMPv6 control messages in order to share routing information and manage DODAGs: DIS, DIO, DAO and DAO-ACK messages. The DIS messages (*DODAG Information Solicitation*) are typically used for asking routing-related information from the neighbor nodes. These neighbors then reply through the sending of DIO messages (*DODAG Information Object*). These messages contain the information required by RPL nodes to discover a RPL instance, get their configuration parameters, select a DODAG parent set, and maintain the DODAG graph. In particular, a DIO message is composed of five main fields corresponding to the RPL instance ID, the DODAG ID, the version number, the rank value and the Mode of Operation (MOP) field. The RPL instance ID and DODAG ID fields are defined by the root node. They indicate respectively the identifier of the RPL instance and the identifier of the DODAG graph which is the IPv6 address of the root. The version number of a DODAG is incremented by the root node, each time the DODAG is updated in order to synchronize nodes and maintain communications. The rank value of a node corresponds to its position in the graph with respect to the root. It must always be greater than its parents' rank in order to guarantee the acyclic nature of the graph. This value is always increasing in the downward direction. The MOP field given by the root allows the maintenance of

downward routes and multicast communications. Nodes joining the DODAG have to match the MOP field and therefore its properties, in order to participate as a router. If not, the RPL nodes can only join the graph as a leaf node. DIO messages are also used by the root in order to build a new DODAG graph. In that case, the messages are broadcasted by the root to its neighbors. Finally, the DAO messages (*Destination Advertisement Object*) are used to build downward routes along the DODAG graph. They may be acknowledged with DAO-ACK messages (*Destination Advertisement Object Acknowledgement*). In that case, a DAO-ACK message is sent back to the sender in unicast by the receiver of a DAO message (DAO parent or DODAG root). All these different control messages play an important role in the building and maintenance of DODAGs.

2.2.2 DODAG Building and Maintenance

The DODAG graph is built in a step by step manner. The root initially broadcasts a DIO message as depicted in Figure 2.2. Upon receiving a DIO message, a node adds the sender of the message to its parents list and determines its own rank value by taking into account the objective function referred in the DIO message. It then forwards updated DIO messages to its neighbors. Based on its parent list, the node selects a preferred parent which becomes the default gateway to be used when data has to be sent towards the DODAG root. At the end of this process, all the nodes participating in the DODAG graph have an upward default route to the DODAG root. This route is composed of all the preferred parents. The DIO messages are periodically sent according to a timer set with the trickle algorithm [46] which optimizes the transmission frequency of control messages depending on the network state. It consists in increasing the frequency of messages when an inconsistency is detected. This allows faster recovery. On the other hand, the frequency of messages may be reduced when the network shows stability. A new node may join an existing network by broadcasting a DIS message in order to solicit DIO messages from its neighbors.

The downward routes are then built using the DAO messages. Depending on the mode of operation specified by the root in the DIO messages, routing tables can be maintained by router nodes. In the storing mode, the child unicasts a DAO message to the selected parent which records it. The parent aggregates the routes received from other DAO messages and sends the information to its parent recursively through a DAO message. In the non-storing mode, DAO messages are unicasted to the DODAG root. Intermediate nodes do not store routing information but simply insert their own address to the message in order to complete the reverse path. The DAO messages can be acknowledged with DAO-ACK messages.

2.2.3 Loops, Inconsistencies and Repairs

The RPL protocol integrates mechanisms to avoid loops, detect inconsistencies and repair DODAGs. Count-to-infinity phenomena occur when a parent increases its rank value and selects its child as a new parent and the child does the same because

it cannot re-attach to another node and so on. Then, the rank value of both parent and child does not stop to increase. To prevent this, the RPL protocol limits the maximum rank value allowed within the graph. DODAG loops appear when a node does not respect the rank property which means that the DODAG is no longer acyclic. To avoid this, a leaving node must poison its sub-DODAG by advertising an infinite rank. The leaving node has also the possibility to use a detaching mechanism, which consists in forming an intermediary floating DODAG and rejoining the main DODAG later.

The RPL protocol can also detect inconsistencies using the datapath validation mechanism [80]. Routing information is included in data packets within a RPL Option carried in the IPv6 Hop-by-Hop header using several flags. The Down 'O' flag indicates the expected direction up or down of a packet. If a router sets this flag, the packet should be forwarded to a child node using downward routes, otherwise it should be sent to a parent with a lower rank towards the DODAG root. The Rank-Error 'R' flag indicates that a rank error is detected. It occurs when a mismatch is observed between the rank values and the direction of a packet indicated by the Down flag. These two flags are used to detect and repair a so called DAG inconsistency i.e. a routing loop in the network. Finally, the Forwarding-Error 'F' flag indicates the inability of a node to forward the packet towards the destination in case of downward packets [80]; this means that routing tables are not up-to-date and this flag is used to clean faulty entries of routing tables.

When inconsistencies are detected, the RPL nodes should trigger repair mechanisms. These mechanisms contribute also to the topology maintenance when node and link failures happen. The local repair mechanism consists in finding an alternative path to route the packets when the preferred parent is not available. A node chooses another parent in its parent list. It is also possible to route packets via a sibling node e.g. node with the same rank. This alternative path may not be the most optimized one. According to [37], this local repair mechanism is effective and enables the network to converge again within a reasonable time. When the local repair mechanisms fail due to multiple inconsistencies, the DODAG root can initiate a global repair by incrementing the version number of the DODAG graph. The RPL network is then completely rebuilt.

2.2.4 Protocol Security

The RPL protocol defines several mechanisms that contribute to its security. As previously mentioned, it integrates local and global repair mechanisms as well as loop avoidance and detection techniques. It also defines two security modes that can be used to ensure integrity and confidentiality of messages. The pre-installed mode consists in having nodes with pre-installed keys in order to send secure messages. The authenticated mode goes a step forward and considers that nodes with pre-installed keys can only join a DODAG graph as leaf nodes. They must obtain a key from an authenticated authority to join the graph as a router. However, important features like key-management are left out by the current standard [70]. Furthermore, cryptographic algorithms are known to occupy the most memory and take many CPU

cycles, thereby greatly affecting the performance of constrained devices likely to be used in IoT and WSN applications. Current RPL implementations, as such, do not enable secure operation modes [36]. Typical deployments of such networks base their security on link layer, transport/application layer [61, 9] or using end-to-end encryption [64, 65]. However, an attacker may bypass security at the link layer by either exploiting a vulnerability or gaining access to a shared key. The attacker can also be a misconfigured or faulty node whose behaviour can disturb network functioning.

The RPL protocol is exposed to a large variety of security attacks as showed by the taxonomy described in Chapter 3. The characteristics of LLN networks such as resource constraints, lack of infrastructure, limited physical security, dynamic topology and unreliable links make them particularly vulnerable and difficult to protect against attacks [76]. These ones can be specific to the RPL protocol, but can also be applied to wireless sensor networks or even to wired networks [10]. RPL-based networks thus require performance and security monitoring solutions that are lightweight and efficient.

2.3 Monitoring RPL-based Internet of Things

Many monitoring systems have been proposed for the traditional Internet. However these solutions necessitate to be adapted, or new approaches have to be designed in order to cope with the requirements of IoT networks. Since the IoT paradigm is quite recent, few approaches are specifically dedicated to these networks and in particular regarding the RPL protocol. In this study most of the presented monitoring solutions are inherited from the wireless sensor networks (WSNs) and from mobile ad hoc networks (MANETs). Some solutions also include frameworks that have been specifically designed for security.

The presented monitoring solutions are classified as presented in Figure 2.3. We distinguish two main categories: active monitoring architectures presented in Section 2.3.1 and passive monitoring architectures described in Section 2.3.2.

2.3.1 Active Monitoring

In what follows, target nodes and networks refer to nodes (respectively networks) to be monitored. We consider as active monitoring a solution that requires target nodes to perform monitoring tasks, for instance send or forward specific traffic messages over the network, collect or store monitoring information or decision-making process. We have divided active monitoring architectures into three categories: centralized, decentralized and hybrid approaches.

2.3.1.1 Centralized Approaches

Centralized approaches consist in solutions with a central manager. Monitoring agents are deployed on each node and have to collect, store information about the device and send collected data over the network to a global manager. This manager

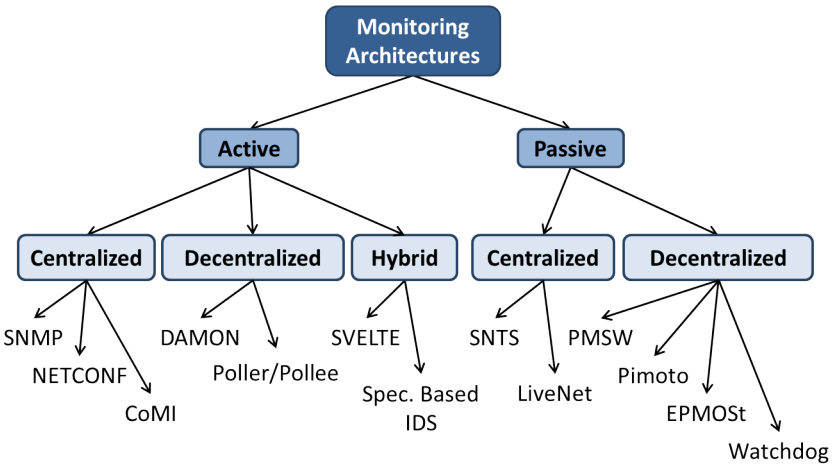


Figure 2.3: Classification of monitoring architectures.

is responsible for data aggregation and decision making about collected information. In IoT networks, it can be deployed on the sink or remotely to a server to which all messages are transmitted by the sink which is interconnected to the Internet.

We first consider in this category traditional management protocols, such as SNMP [14] and NETCONF [30] with their adaptation to resource constrained environments. SNMP permits to monitor, control, and also configure network devices. Each managed device implements an agent responsible for collecting and transmitting data about the device organized in a specific standardized database. NETCONF is used to install, delete and change configuration on network devices and needs persistent connections to operate. An analysis of the SNMP and NETCONF protocols shows their limits in the context of the Internet of Things [72]. The NETCONF protocol is quite resource heavy due to its reliance on XML. SNMP performs relatively well, as long as authentication and encryption are not utilized since these tasks occupy most of the device resources [38]. The integration of SNMP agents with their management information base (MIB) on resource constrained devices may take away valuable resources. This is especially true on C0 and C1 devices (see Table 1.1 from Chapter 1), where the amount of RAM available to nodes is quite restricted. It is important to note that these devices are likely to be the majority of deployed IoT devices [11].

Using the CoAP protocol [73] to perform network management and monitoring tasks can offer resource reduction since the protocol would be used by the application layer in most cases. As such, there are a few efforts under way to design CoAP based management and monitoring solutions. The ongoing CoMI (CoRE Management Interface) initiative in the IETF is aiming to make SMIv2 function over CoAP [24]. It utilizes also MIBs and does not rely on connection-oriented communications. Packets are encoded using the CBOR[10] format [12] which is similar to JSON [13] but optimized for constrained devices.

[10]Concise Binary Object Representation

We can notice that theses solutions focus on configuration management and do not track network events nor detect anomalies. They can however be exploited to perform security since the state of each node is recorded and certain types of malicious activities can be inferred from collected information. In these approaches, constrained nodes have to maintain internal information and send data. Also, in large networks, centralized manager may result in congestion of routes to the sink, and excessive load at the sink due to monitoring data.

2.3.1.2 Decentralized Approaches

Active decentralized approaches also typically rely on agents deployed on each node to collect and send monitoring data. However, other monitoring tasks (storing, aggregation, ...) are performed by distributed nodes in the network and are not dedicated to a central manager. Such approaches allow reducing target node load compared to active centralized architectures.

Authors of [62] propose a distributed monitoring solution called DAMON[11] for mobile ad-hoc networks (AODV routing protocol), composed of monitoring agents and data repositories storing monitored information as illustrated by Figure 2.4. The sinks, collecting monitoring data, vary over time based on their resources and locations, in order to maximize the network lifetime. In this approach, each agent is hosted by a target network node which sends information to the distributed monitoring sinks, thereby increasing the resource consumption of all nodes. DAMON supports sink auto-discovery using beacon messages and the resiliency of agents to sink failures.

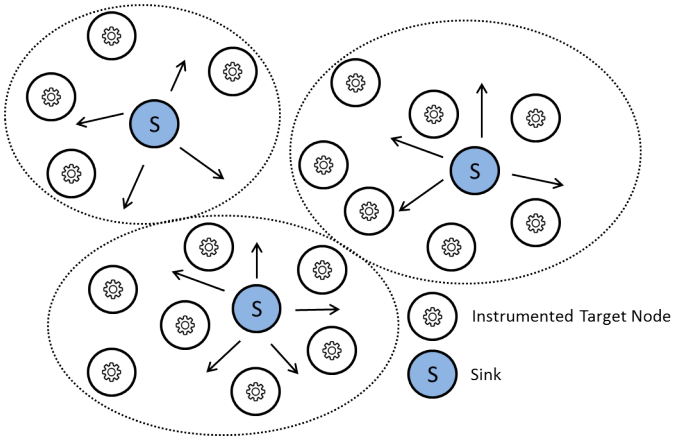


Figure 2.4: Architecture used in DAMON.

A poller/pollee strategy is introduced in [48, 39] to collect and aggregate monitoring data from a sensor network in a lightweight manner. In particular, authors of [48] present a distributed algorithm to select pollers among the WSN nodes while both minimizing the number of required monitoring nodes and the false alarm rate. A

[11] Distributed Architecture for MONitoring mobile networks

false alarm occurs if a pollee does not fail to send its reports but the poller misses all of them within a defined time period. This can happen when pollees are too far away from the poller. Aggregation algorithm is also proposed to reduce the communication overhead induced by such a solution. Each poller aggregates collected data and makes local decisions. Using a similar architecture as described in Figure 2.5, Lahmadi et al. [39] minimize monitoring communication overhead by embedding these communications in data packets. This work is designed for RPL-based LLNs networks. Not only the piggybacking process is proposed to tackle communication overhead issue, but authors also present a method to select pollers within the graph. Their evaluation has also showed that the proposed approach is robust to topology changes.

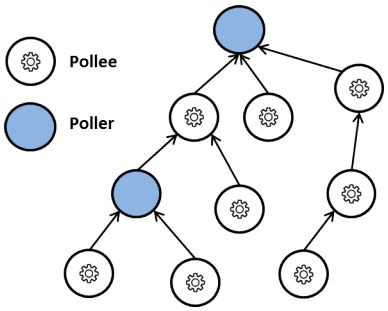


Figure 2.5: Poller-pollee architecture used in [39].

Since the monitoring data storing is performed by dedicated nodes, these decentralized solutions permit to reduce device resource usage on target nodes which might represent a better choice compared to active centralized architectures. However, agents still need to be deployed on target nodes to collect and send monitoring data thereby reducing their resources.

2.3.1.3 Hybrid Approaches

Hybrid approaches refer to architectures where monitoring data processing tasks are shared between a central entity and distributed nodes while target nodes are also instrumented to collect this data. The presented solutions below are Intrusion Detection Systems (IDS) and are therefore security oriented.

The SVELTE framework [63] is specifically designed for the RPL protocol. It is composed of three modules. One is responsible for rebuilding the topology at the sink nodes using requests, the second one carries out the intrusion detection process and the last one is a mini distributed firewall. The approach is hybrid since lightweight modules are deployed on each node of the network and modules responsible for heavy processing are run within the root as described by Figure 2.6. This IDS is designed to detect sinkholes and selective forwarding attacks and is robust to identity attacks.

A specification-based solution is described in [44, 41] to detect topological attacks in a RPL-based network. A model is generated remotely by learning the states, transitions and statistics based on analyzed traces. This one is then used to perform

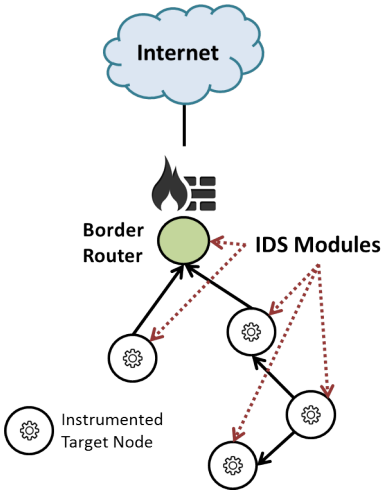


Figure 2.6: Architecture used in SVELTE.

the detection of abnormal situations. Distributed super nodes implement the finite state machine previously learned. They are then deployed to monitor target nodes through requests as illustrated by Figure 2.7. However, the super nodes do not participate in the target network.

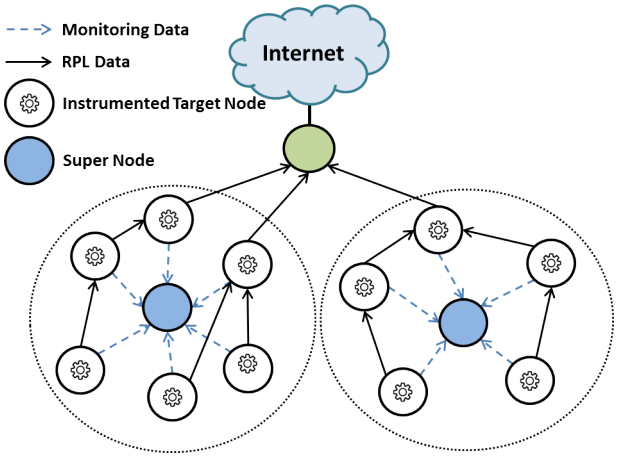


Figure 2.7: Architecture used in the specification-based IDS from [41].

In these hybrid approaches, even if data processing is performed by both central and distributed entities, target nodes are still instrumented to collect the required monitoring information which consumes their resources. As such, passive monitoring which relies on dedicated probes may offer a good compromise to perform network monitoring while preserving nodes energy.

2.3.2 Passive Monitoring

We define as passive monitoring, architectures where dedicated monitoring nodes called sniffers are deployed in the target network. They collect information about network events and the target nodes which are not instrumented. These solutions have been widely exploited in WSNs. As in the previous section, passive architectures are classified whether they are centralized or decentralized.

2.3.2.1 Centralized Approaches

Passive centralized monitoring architectures correspond to solutions where deployed monitoring nodes collect information which are transmitted to a central sink performing the analysis and decision process.

In particular, Khan et al. [35] introduce a troubleshooting suite called SNTS[12] to facilitate the identification of anomalies in sensor applications. The solution uses dedicated extra nodes which passively listen to communications. The gathered information is then sent in the back-end part of the architecture where data mining techniques are performed to automate analysis for troubleshooting.

In the same manner, LiveNet [16] proposes to reconstruct the complex behavior of a deployed sensor network using multiple passive packet sniffers collocated with the network as presented in Figure 2.8. Their work focuses on merging the monitoring traces obtained from the different sniffers, estimating the coverage of the monitoring nodes and deducing missing information.

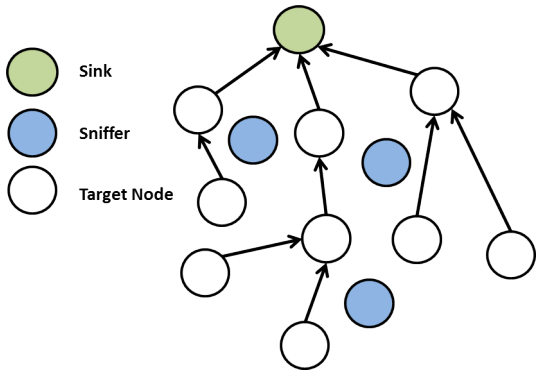


Figure 2.8: Architecture used in LiveNet.

In these examples, data analysis is performed offline remotely by a dedicated entity. This allows running complex algorithms, however it may introduce considerable delays in detecting failures or abnormal activities.

2.3.2.2 Decentralized Approaches

In passive monitoring, decentralized architectures refer to approaches where the monitoring tasks (data aggregation and analysis, decision making process) are not only

[12]Sensor Networks Troubleshooting Suite

performed by a central entity. These tasks can be achieved locally by the dedicated monitoring nodes or they can be shared in a two-level hierarchical system where the distributed monitoring nodes gather and aggregate data from the sniffers before forwarding them to a sink for further processing.

Authors of [82] design a passive monitoring system called PMSW[13] composed of four types of nodes: sensor nodes, sniffing nodes, monitoring nodes and a workstation. Sniffing nodes are deployed in the network to collect information regarding the sensors, and connect to a monitoring node. This one is responsible for aggregating collected data and sending them to the workstation. In the workstation, the traces are merged using clock-adjusting strategies, the missing traces are inferred based on a finite state machine. Authors also use descriptions of event-rules based on XML to perform fault diagnosis.

In Pimoto [8], sensor nodes are divided in so-called islands to which are assigned a monitoring node that passively listens to all communications in the area. These sniffers send their collected information over a hierarchical operating system to a server using a second radio channel operated on Bluetooth. The server processes the data afterwards.

Another passive solution called EPMOSt[14] [31] focuses on reducing energy consumption by passive monitoring in WSNs. The monitoring information is provided using an SNMP agent. Sniffers are deployed in the target network, send their collected information to a local monitoring node using a dedicated monitoring network as presented in Figure 2.9. The local monitoring nodes store this data in a distant server which performs the analysis. A sniffer election is run by sniffers and monitoring nodes, to choose which target nodes are monitored by a given sniffer. After having analyzed traces, the server generates reports which are stored in a MIB database.

While previous approaches rely on hierarchical architectures, the following solution use exclusively a local strategy to perform monitoring in WSNs. Authors of [25] propose a self-monitoring approach which relies on watchdog techniques: some nodes of the network are chosen to perform the monitoring tasks for the target nodes in communication range. Authors analyze the problem of self-monitoring in large-scale wireless sensor networks. They present two distributed algorithms to elect the monitoring nodes among the target nodes in an optimal topology reducing the induced overhead. They provide complexity analysis and cost evaluation of these algorithms. However, in this case, the monitoring is purely local because there is no monitoring data exchange between monitoring nodes. It is not possible in this solution to take into account the entire network, opening the possibility to miss major events. Also, energy issues are not considered in the selection process of monitoring nodes.

Decentralized passive architectures allow processing monitoring traffic locally. Most of these approaches are based on hierarchical systems where target nodes are monitored from local and global perspectives. In some cases, as in the watchdog approach, this processing is purely local which does not give a global view of the

[13] Passive Monitoring System for WSN
[14] Energy-efficient Passive MOnitoring System

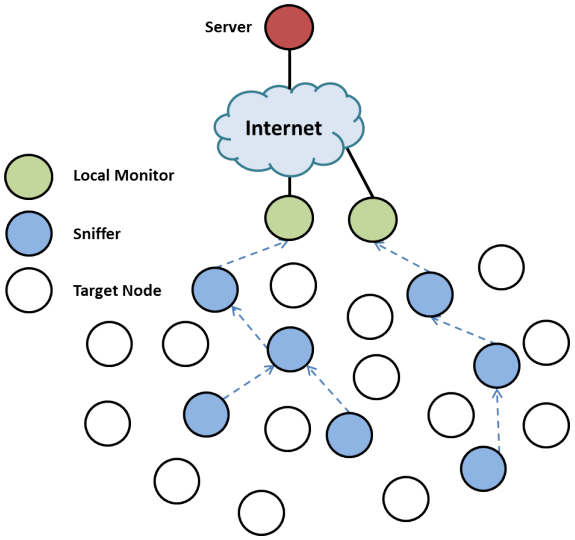


Figure 2.9: Architecture used in EPMOSt.

network. We can also observe that there is no direct interaction among the monitoring nodes which could perform a collaborative level of monitoring by crossing collected information.

2.3.3 Comparison and Limits

In this section, we compare the different identified solutions according to several relevant criteria for this thesis as presented in Table 2.1. Among those criteria, **storing** refers to the fact that target nodes have to store collected information. Gathered information can also be stored in dedicated nodes called data repositories according to the considered approach. The **pervasiveness** of the proposed solution indicates whether nodes performing monitoring activities are involved in the target network activities. The **security** property is also considered, it indicates whether the architecture has been designed for security purpose. It can be noted that each monitoring solution could be used to perform security but dedicated algorithms need to be deployed where data is processed. The next criterion is the **level of monitoring**. Two values are possible regarding the considered architecture: local which means that the monitoring process is performed locally by monitoring nodes based on local data; and global which indicates that the monitoring process is performed considering all monitoring data from the network. Finally, the **RPL** criterion indicates whether the considered architecture is designed for or utilize the RPL protocol (Yes/No) or whether it can be adapted to manage RPL protocol (Maybe). To provide security while being energy efficient for target nodes we need a framework that (i) does not require target nodes to be instrumented or to store their information; (ii) is pervasive; (iii) is designed for security; (iv) provides several level of monitoring; and (v) can be used in RPL environments.

In Table 2.1 we can observe that in active architectures, all the centralized solu-

Table 2.1: Comparison of monitoring approaches.

Solutions			Storing	Pervasiveness	Security	Level of monitoring	RPL
Active	Centralized	SNMP	Yes	Yes	No	Global	Maybe
		NetConf	Yes	Yes	No	Global	Maybe
		CoMI	Yes	Yes	No	Global	Yes
	Decentralized	DAMON	Data rep.	Yes	No	Local	No
		Poller/Pollee	Data rep.	Yes	No	Local + Global	Yes
	Hybrid	SVELTE	Yes	Yes	Yes	Local + Global	Yes
		Spec. based IDS	No	Yes	Yes	Local	Yes
Passive	Centralized	SNTS	No	No	No	Global	Maybe
		LiveNet	No	No	No	Global	Maybe
	Decentralized	PMSW	No	No	No	Local + Global	Maybe
		Pimoto	No	No	No	Local + Global	Maybe
		EPMost	No	No	No	Local + Global	Maybe
		Watchdog	No	Yes	No	Local	Maybe

tions and the SVELTE framework require nodes to store their own monitoring information while dedicated target nodes perform this task in the other decentralized monitoring approaches. On the contrary the specifically deployed monitoring nodes in passive monitoring are responsible for collecting monitoring information. We can also notice that all active monitoring architectures are pervasive, since they require target nodes to be instrumented. They are therefore involved in both monitoring and target network functioning activities. Except the watchdog strategy, monitoring nodes used in passive solutions do not contribute in the target network. The presented monitoring solutions are not designed for security purpose (except the two IDS), even if security related information could be inferred from the collected data. Centralized architectures (active and passive) only provide a global monitoring level which may lack of reactivity in case of attacks since all data are processed in a central entity remotely. Some architectures give only a local view of the network which implies the possibility to miss major events. Other decentralized approaches have the capability to perform local and global level of monitoring. Except the DAMON tool, which is based on the AODV protocol, other solutions have been designed for RPL networks or could be adapted to them. All passive solutions from the WSNs can use this protocol since their implementation does not rely on any particular routing protocol.

Based on these criteria, we can conclude that none of the described architecture meet our aforementioned requirements regarding a security-oriented monitoring framework. Indeed, active solutions have to be excluded since they require to instrument target nodes for collecting and storing (for most of them) monitoring information which can take away precious resources on constrained devices. We therefore argue in favor of implementing passive monitoring framework in RPL net-

works. However, in passive solutions the deployed monitoring nodes (or sniffers) do not contribute in the target network at all. Deploying an architecture only dedicated to monitoring might represent a considerable cost for the operator. Even if most of these architectures provide local and global level of monitoring, we also want to perform collaborative monitoring and allow monitoring nodes to interact with each other and get useful complementary information from neighboring monitoring nodes. While a large majority of mentioned solutions can be adapted to the RPL protocol, a solution able to exploit the RPL mechanisms will be more energy efficient.

2.4 Conclusions

In this chapter, we have presented the RPL protocol and detailed its main mechanisms. We have showed how DODAGs are built and maintained using the different control messages. First the upward routes are built to reach the sink using DIO messages. According to the advertised mode of operation, the downward routes are then formed through DAO messages in order to reach the different nodes participating in the graph. We have also described loops and inconsistencies that may occur in RPL networks and how the protocol standard manages them. In particular, a limited rank value is introduced to address count-to-infinity phenomena and detaching mechanisms such as poisoning are designed to prevent them. The data path validation is used to detect and fix possible loops in the network while local and global repair mechanisms prevent connectivity loss and ensure optimal topology. We have introduced security concerns regarding the RPL protocol. We have showed that the lack of nodes resources and no currently deployed secured transmissions leave the RPL protocol open to a large variety of attacks.

In such constrained networks, it is necessary to provide a monitoring solution with a minimal impact on the network resources. We have described several monitoring architectures mostly from ad-hoc networks and wireless sensor networks depending on their active/passive and centralized/decentralized natures. These approaches have been compared using several criteria. We have concluded that none of the presented solutions meets our requirements. In RPL-based IoT networks, in order to address security, we need a framework which does not require target nodes be instrumented in order to preserve their resources. To reduce the cost of deploying such an architecture, it should take benefit from the RPL protocol features and allows monitoring nodes to participate in the target network by relying on typical IoT deployments. This solution should also perform multi-level of monitoring to guarantee a satisfying reactivity and a fine granularity of data to address a large variety of threats.

These solutions are necessary to get performance indicators and to detect attacks to which the RPL protocol is exposed. In the next chapter, we propose to identify and classify these attacks.

Chapter 3

Taxonomy of Attacks in RPL Networks

Contents

3.1 Introduction 25
3.2 Attacks against Resources 26
3.2.1 Direct Attacks 26
3.2.2 Indirect Attacks 27
3.2.3 Analysis 30
3.3 Attacks on Topology 31
3.3.1 Sub-optimization Attacks 32
3.3.2 Isolation Attacks 34
3.3.3 Analysis 35
3.4 Attacks on Traffic 37
3.4.1 Eavesdropping Attacks 37
3.4.2 Misappropriation Attacks 38
3.4.3 Analysis 39
3.5 Conclusions 40

3.1 Introduction

Attacks targeting RPL networks require to be identified and prioritized according to their consequences in order to design efficient security solutions. In this chapter, we propose to establish a taxonomy of routing attacks against the RPL protocol. We take into account the goals of the attack and which element of the RPL network is impacted. We also describe in this classification existing security solutions we have found in the literature. The taxonomy is depicted in Figure 3.1 and considers three categories of security attacks. The first category covers attacks targeting the exhaustion of network resources (energy, memory and power). These attacks are

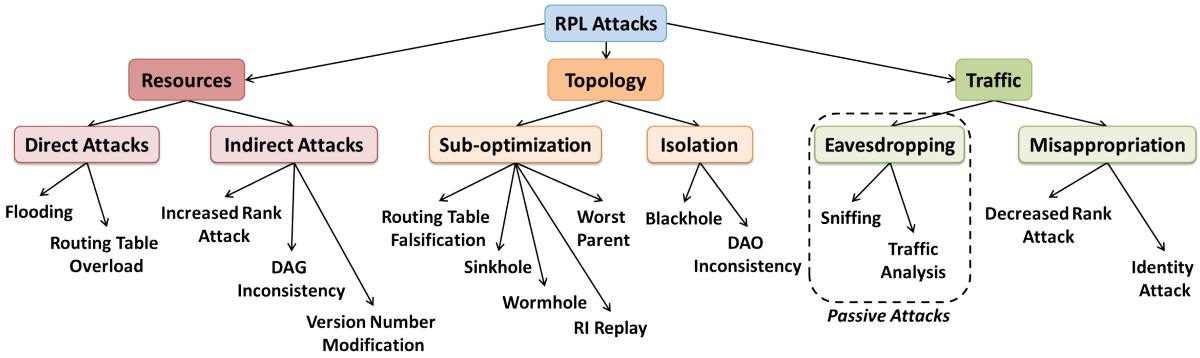


Figure 3.1: Taxonomy of attacks against RPL networks.

particularly damaging for such constrained networks because they greatly shorten the lifetime of the devices and thus the lifetime of the RPL network. The second category includes attacks targeting the RPL network topology. They disturb the normal operation of the network: the topology may be sub-optimized in comparison with a normal convergence of the network or a set of RPL nodes may be isolated from the network. The third category corresponds to attacks against the network traffic, such as eavesdropping attacks or misappropriation attacks. Each section of this chapter focuses on a specific category of RPL security attacks.

3.2 Attacks against Resources

Attacks against resources typically consist in making legitimate nodes perform unnecessary processing in order to exhaust their resources. This category of attacks aims at consuming node energy, memory or processing. This may impact on the availability of the network by congesting available links and therefore on the lifetime of the network which can be significantly shortened. We distinguish two subcategories of attacks against resources. The first one gathers direct attacks where a malicious node will directly generate the overload in order to degrade the network. The second one contains indirect attacks where the attacker will make other nodes generate a large amount of traffic. For instance, such an attack can be performed by building loops in the RPL network so that other nodes produce traffic overhead.

3.2.1 Direct Attacks

In case of direct attacks, the attacker is directly responsible for resource exhaustion. This can typically be done by performing flooding attacks or by executing overloading attacks with respect to routing tables, when the storing mode is active.

3.2.1.1 Flooding Attacks

Flooding attacks consist in generating a large amount of traffic in a network and make nodes and links unavailable. These attacks can be performed by an external or

internal attacker. They exhaust the resources of all the network nodes in the worst case. More specifically, using solicitation messages to perform the flooding is called an HELLO flood attack. In RPL networks, an attacker can either broadcast DIS messages to its neighboring nodes which have to reset their trickle timer, or, unicast DIS messages to a node which has to reply with DIO messages. In both cases, this attack leads to network congestion and also to the saturation of the RPL nodes. The consequences of such attacks have been studied in [45], the authors show that the control message overhead significantly increases but the delivery ratio is not affected. However, no solution especially designed for RPL has been proposed.

3.2.1.2 Routing Table Overload Attacks in Storing Mode

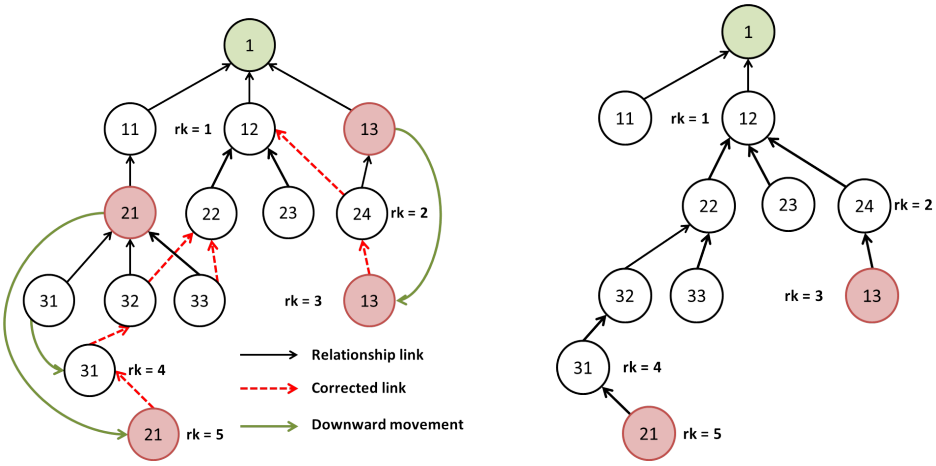
It is also possible to perform direct attacks against resources by overloading the RPL routing tables. The RPL protocol is a proactive protocol. This means that the RPL router nodes build and maintain routing tables when the storing mode is enabled for those nodes. The principle of routing table overload is to announce fake routes using the DAO messages which saturate the routing table of the targeted node. This saturation prevents the build of new legitimate routes and impacts network functioning. It may also result in a memory overflow. Let us consider the example of the DODAG 2 described in Figure 2.2 in Section 2.2 and assume that node 12 plays the role of the attacker. Nodes 12 and 13 send a DAO message in order to add the corresponding entries in the routing table of node 11. The attacker, node 12, sends multiple forged DAO messages to node 11 with false destinations. As a consequence, node 11 builds all the corresponding entries in its routing table. Afterwards, when the other nodes including node 13 are sending legitimate DAO messages with respect to new routes, the node 11 is no longer able to record them because its routing table is overloaded. This attack is not specifically mentioned in the literature but it is part of overload attacks more generally [66].

3.2.2 Indirect Attacks

Indirect attacks correspond to attacks where the malicious node makes other nodes generate an overload for the network. It includes: increased rank attacks, DAG inconsistency attacks and version number attacks.

3.2.2.1 Increased Rank Attacks

The increased rank attack consists in voluntarily increasing the rank value of a RPL node in order to generate loops in the network. This attack has been studied in [81] through ns-2 simulations. The authors showed that their loop avoidance mechanisms costed more than the attack itself. Concretely, in a RPL network, a rank value is associated to each node and corresponds to its position in the graph structure according to the root node. As previously mentioned, the node rank is always increasing in the downward direction in order to preserve the acyclic structure of the DODAG. When a node determines its rank value, this one must be greater than



(a) Initial State (b) Final State

Figure 3.2: Rank increased attack in a RPL network.

the rank values of its parents. If a node wants to change its rank value, it has first to update its parents list by removing the nodes having a higher rank than its new rank value. Once a node has established the set of parents in a DODAG, it selects its preferred parent from this list in order to optimize the routing cost when transmitting a packet to the root node. A malicious node advertises a higher rank value than the one it is supposed to have. Loops are formed when its new preferred parent was in its prior sub-DODAG and only if the attacker does not use loop avoidance mechanisms. In that case, two attack scenarios are possible as illustrated in Figure 3.2. In the first scenario, the attacker is node 13 and the new preferred parent (node 24) has already a substitute parent (node 12) to re-attach to. The node 13 increases its rank value to 3 and chooses node 24 as the new preferred parent. This operation generates a routing loop in the DODAG graph, because the node 24 was in the prior sub-DODAG of node 13. The formed loop is composed of nodes 13 and 24 and is easily repaired because the node 24 can re-attach to node 12 after sending few control messages. However, this attack becomes more problematic when the node does not have a substitute parent such as node 31 in the second scenario. As depicted in Figure 3.2, the attacker, node 21, increases its rank value which requires node 31 to also increase its own in order to find a new parent. Meanwhile nodes 32 and 33 have to connect to a substitute parent (node 22) so node 31 selects node 32 as new preferred parent. At the end, node 21 increases its rank value to 5 in order to add node 31 as its preferred parent. The count-to-infinity problem is avoided because of the limitation of the maximum rank value advertised for a DODAG, as seen in Section 2.2.3. The increased rank attack is more damaging in this second scenario, because more routing loops are built at the neighborhood. In that case, the loop repair mechanism requires to send many DIO messages (resets of the trickle timer) and requires a longer convergence time. The more the number of affected nodes increases, the longer the convergence time is. We consider this attack as part of the

resource consumption attacks because the churn is exhausting node batteries and is congesting the RPL network.

To mitigate this attack, the number of times a RPL node is increasing its rank value in the DODAG graph should be monitored to determine if a node can be considered as malicious or misconfigured. It is important to notice that a node can legitimately increase its rank value if it no longer matches the objective function and/or cannot manage the amount of received traffic. However, it must use the loop prevention techniques or it can wait for a new version of the DODAG graph. Also, thanks to the data path validation mechanism, the RPL protocol is able to deal with these loops even if resources are consumed to repair them [80].

3.2.2.2 DAG Inconsistency Attacks

A RPL node detects a DAG inconsistency when it receives a packet with a Down 'O' bit set from a node with a higher rank and vice-versa [80] e.g. when the direction of the packet does not match the rank relationship. This can be the result of a loop in the graph. The Rank-Error 'R' bit flag is used to control this problem. When an inconsistency is detected by a node, two scenarios are possible: (i) if the Rank-Error flag is not set, the node sets it and the packet is forwarded. Only one inconsistency along the path is not considered as a critical situation for the RPL network, (ii) if the 'R' bit is already set, the node discards the packet and the trickle timer is reset [46]. As a consequence, control messages are sent more frequently. A malicious node has just to modify the flags or add new flags to the header. The immediate outcome of this attack is to force the reset of the DIO trickle timer of the targeted node. In that case, this node starts to transmit DIO messages more frequently producing local instability in the RPL network. This also consumes the battery of the nodes and impacts the availability of links. All the neighborhood of the attacker is concerned by the attack, since it has to process unnecessary traffic. Moreover, by modifying legitimate traffic, all the packets are discarded by the targeted node. This causes a blackhole and isolates segments of the network. To mitigate the flooding induced by this attack, [32] proposes to limit the rate of trickle timer resets due to an RPL Option to no greater than 20 resets per hour, however no reasoning is provided regarding this value. We will present in Chapter 5 two solutions that takes into account network characteristics to detect and mitigate such attacks.

3.2.2.3 Version Number Attacks

The version number is an important field of each DIO message. It is propagated unchanged down the DODAG graph and is incremented by the root only, each time a rebuild of the DODAG is necessary which is also called global repair. An older value indicates that the node has not migrated to the new DODAG graph and cannot be used as a parent node. An attacker can change the version number by illegitimately increasing this field in DIO messages when it sends them to its neighbors. Such an attack causes an unnecessary rebuilding of the whole DODAG graph thereby exhausting node resources. Dvir et al. [28] proposed a security mechanism called VeRa

(standing for Version Number and Rank Authentication) that prevents compromised nodes from impersonating the root and from sending an illegitimate increased version number. The solution uses authentication mechanisms based on hash operations. In that case, a node can easily check if the version number has been modified by the root node or by another malicious node, which can no longer usurp the identity of the DODAG root. Also, authors of [40] proposed an improvement of the previous solution solving some issues they discovered in VeRA. In this thesis we will also propose a detection strategy for these attacks in Chapter 6.

3.2.3 Analysis

We discuss in this section the properties of the identified attacks as well as methods and techniques to address them. Table 3.1 summarizes attacks against resources. A first property to be analyzed is the internal (I) or external (E) nature of the attacks. Internal attacks are initiated by a malicious or compromised node of the RPL network. External attacks are performed by nodes that do not belong to the RPL network or are not allowed to access it. We can observe that only the flooding can be performed externally because the attacker does not need to join the graph to perform the DIS flooding since DIS message are used to discover the DODAG. For the rest of the attacks, the malicious node needs to be part of the DODAG to have enough knowledge in order to launch its attacks.

Table 3.1: Summary of attacks on resources.

Attacks	I/E	A/P	Prerequisites	Impact	CIA	Mitigation/ Protection	Overhead
Flooding	I/E	A	-	Link/Battery	A	None	None
Routing Table Overload	I	A	Storing Mode	Memory/Battery	A/I	None	None
Increased Rank Attack	I	A	-	Battery/Link	A	RPL loop detection and avoidance mechanisms [80]	None (by default in RPL)
DAG Inconsistency Attack	I	A	Option Header	Battery/Link	A/I	Limitation of timer resets [32]	Low
Version Number Attack	I	A	-	Battery/Link	A/I	VersionNumber and Rank Authentication [28], TRAIL [40]	Low (for both solutions)

A second property is to determine if the attack is passive (P) or active (A). Passive attacks do not modify the behavior of the network. On the contrary, active attacks require the malicious node to perform operations that are observable by other nodes in the network. They are usually more critical than passive attacks which mainly target data confidentiality or topology information. Attacks targeting

the resources are all active since the attacker has to send packets. A third property is the prerequisites property. The prerequisites are the required conditions to initiate the attack besides the internal/external nature of the attack, such as particular configuration of the network. The storing mode which means maintaining routing tables has to be enabled to launch routing table overload and the RPL option header has to be implemented to run DAG inconsistency attacks.

The next property corresponds to the impact of the attacks. The objective is to quantify the consequences of a successful attack on the network. The impact in this category is evaluated as the type of over-consumed resources (e.g. memory, battery, link availability). We observe that all the attacks consume node battery as they imply additional processing for the nodes. Most of the time, the link availability is also impacted since the attacks require sending a large number of control messages. The memory is also over-consumed in case of routing table overload attacks.

The fifth property corresponds to the CIA acronym standing for confidentiality, integrity and availability, and refers to a security reference model. In the context of the RPL protocol, confidentiality means the protection of routing information and exchanges. Integrity involves the protection of routing information from unauthorized modification, and availability requires that forwarding services and routing information exchanges are accessible for the nodes. Regarding the identified attacks targeting resources, they systematically impact network availability. Indeed, these attacks involve that the attacker jeopardizes resources of the network (battery, memory, processing, link availability). The integrity is also impacted when the result of the attack supposes that a legitimate resource or legitimate traffic is corrupted e.g. routing table of legitimate nodes is altered during routing table overload attacks. Version number modifications and DAG inconsistency attacks induce that the integrity of packets is jeopardized.

The two last columns of tables indicate respectively the possible security mechanisms to address the attacks, and their overhead (according to their authors). We saw that RPL provides internal mechanisms which contribute to counter attacks. For instance, the loop avoidance mechanisms prevent increased rank attacks. The protocol also proposes an optional mitigation mechanism that limits inconsistency attacks impact [32]. Specific approaches have been designed for the RPL protocol. The VeRa [28] and the TRAIL [40] approaches address version number modifications. In many cases, it is difficult to evaluate the overhead induced by the security mechanisms because they are still at a conceptual level. Moreover, we cannot really consider that the mechanisms which are inherent to the RPL protocol operation introduce an overhead.

3.3 Attacks on Topology

Attacks against the RPL protocol can also target network topology. We distinguish two main categories among these attacks: sub-optimization and isolation.

3.3.1 Sub-optimization Attacks

In case of sub-optimization attacks, the network will not converge to the optimal form (i.e optimal paths) inducing poor performance.

3.3.1.1 Routing Table Falsification Attacks in Storing Mode

In a routing protocol, it is possible to forge or modify routing information to advertise falsified routes to other nodes. This attack can be performed in the RPL network by modifying or forging DAO control messages in order to build fake downward routes. This can only be done when the storing mode is enabled. For instance, a malicious node advertises routes towards nodes that are not in its sub-DODAG. Targeted nodes have then wrong routes in their routing table causing network sub-optimization. As a result, the path can be longer inducing delay, packet drops or network congestion. This has not been studied yet in the context of the RPL protocol.

3.3.1.2 Sinkhole Attacks

An alternative attack consists in building a sinkhole. Such an attack takes place in two steps. First, the malicious node manages to attract a lot of traffic by advertising falsified information data (for instance, up and downward links of superior quality). Then, after having received the traffic in an illegitimate manner, it modifies or drops it. In RPL networks, the attack can be easily performed through the manipulation of the rank value as showed in Section 3.4.2.1. Because of this falsified advertisement, the malicious node is more frequently chosen as preferred parent by the other nodes, while it does not provide better performance. Thus, the routes are not optimized for the network. The attack modifies the topology and degrades network performance. Moreover, if the attacker decides to drop all the traffic, it also performs a blackhole attack.

This attack was studied in [78] and [63], the authors proposed an IDS called SVELTE to counter it. A functionality of this IDS is to build a global view of the network and as a consequence the possibility to detect inconsistencies in the network such as sinkholes as presented in Chapter 2. In [79], the authors investigated defense techniques against sinkholes. The first technique is called Rank verification and restricts the possibility for the attacker to decrease its rank value. It allows legitimate nodes to check if another node along the path has a fake rank. The second technique is called parent fail-over and operates as an end-to-end acknowledgement. When a root node does not receive enough traffic from a node (according to a certain threshold value), it adds this node's address in a DIO message field. When the node receives the DIO message with its own identity, it blacklists its parent and selects another one. The authors show that a combination of these two techniques provides efficient results in a RPL network.

3.3.1.3 Wormhole Attacks

Wormhole attacks are defined as the use of a pair of RPL attacker nodes, nodes A and B, linked via a private network connection. An example is depicted in Figure 3.3. In this scenario, every packet received by node 13 is forwarded through the wormhole to node 21 in order to be replayed later. Since the roles are interchangeable, node 21 may perform the same operations than node 13. In the case of wireless networks, it is easier to perform this attack because the attacker can send through the wormhole the traffic addressed to himself as well as all the traffic intercepted in the wireless transmissions. The wormhole attack distorts the routing path and is particularly problematic for RPL networks. If an attacker tunnels routing information to another part of the network, nodes which are actually distant, see each other as if they are in the same neighborhood. As a result, they can create non-optimized routes according to the objective function.

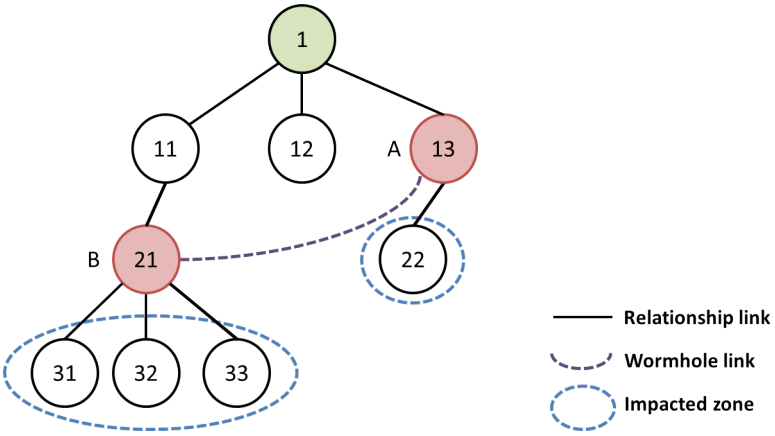


Figure 3.3: A wormhole attack in a RPL network.

This attack was studied in [78] which showed that the RPL protocol cannot solve this attack by itself. The authors explained that countering this type of attack is a research challenge if one node of the wormhole is in the Internet. If both attackers are in the RPL networks, the authors suggested to use geographical data and different cryptographic keys at the MAC layer for different segments to solve this threat issue. Also the authors of [34] proposed to prevent this attack by using Merkel trees to authenticate nodes and paths.

3.3.1.4 Routing Information Replay Attacks

A RPL node can also perform routing information replay attacks. It records valid control messages from other nodes and forwards them later in the network. In case of dynamic networks, this attack is quite damaging because the topology and the routing paths are often changed. Replay attacks cause nodes to update their routing tables with outdated data resulting in a false topology. The RPL protocol uses

some sequence counters to ensure the freshness of the routing information such as the version number for DIO messages or the path sequence number present in the transit information option of DAO messages [80]. This attack is mentioned in [66] however the authors neither study the consequences of such an attack nor explained how it can takes place in RPL networks.

3.3.1.5 Worst Parent Attacks

This attack described in [43] and termed as "Rank attack" consists in choosing systematically the worst preferred parent according to the objective function. The outcome is that the resulting path is not optimized inducing poor performance. This attack cannot be easily tackled because children node rely on their parent to route packets and this attack cannot be monitored by neighbors. However, using a security solution which rebuilds a global view of the graph based on nodes information should detect this attack such as the proposed solution in [63].

3.3.2 Isolation Attacks

The attacks against the topology can also isolate a node or a subset of nodes in the RPL network which means that these nodes are no longer able to communicate with their parents or with the root.

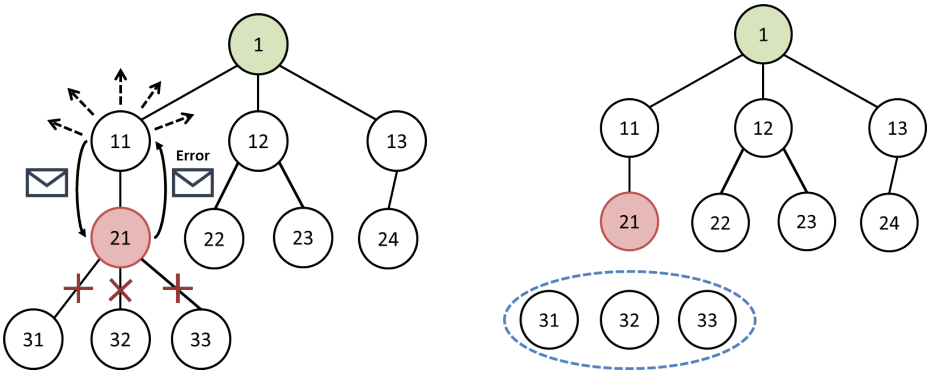
3.3.2.1 Blackhole Attacks

In a blackhole attack, a malicious intruder drops all the packets that it is supposed to forward. This attack can be very damaging when combined with a sinkhole attack causing the loss of a large part of the traffic. It can be seen as a type of denial-of-service attack. If the attacker is located at a strategic position in the graph it can isolate several nodes from the network. There is also a variant of this attack called gray hole (or also selective forwarding attack) where the attacker only discards a specific sub-part of the network traffic. Chugh et al.[18] investigated the consequences of blackhole attacks in RPL networks through a set of Cooja simulations. They highlighted different indicators to detect these attacks such as rate and frequency of DIO messages, packet delivery ratio, loss percentage and delay. The SVELTE IDS proposed in [63] was designed to detect selective forwarding attacks in such networks.

3.3.2.2 DAO Inconsistency Attacks in Storing Mode

DAO inconsistencies occur when a node has a downward route that was previously learned from a DAO message, but this route is no longer valid in the routing table of the child node [80]. RPL provides a mechanism to repair this inconsistency, called DAO inconsistency loop recovery in the data path validation. This optional mechanism allows the RPL router nodes to remove the outdated downward routes using the Forwarding-Error 'F' flag in data packets which indicates that a packet can not be delivered by a child node. The packet with the 'F' flag is sent back to the parent in order to use another neighbor node, as depicted in Figure 3.4. Once

a packet is transmitted downward, it should normally never go up again. When it happens the router sends the packet to the parent that passed it with the Forwarding-Error 'F' bit set and the Down 'O' bit left. When the parent receives the packet with 'F' set it removes the corresponding routing state, clear the 'F' bit, and try to send the packet to another neighbor. If the alternate neighbor still has an inconsistent state the process reiterates.



(a) Initial State (b) Final State

Figure 3.4: Illustration of a DAO inconsistency attack.

In this scenario, the malicious node is represented by node 21. It uses the 'F' flag to make RPL routers remove legitimate downward routes and thus isolate nodes from the DODAG graph. Each time node 21 receives a packet from node 11, it only changes the RPL 'F' flag and sends it back to node 11. As a consequence, the other nodes of the network (nodes 31 to 33) are isolated from the graph. The objective of this attack is to make router nodes discard available downward routes. This makes the topology of the DODAG graph sub-optimal. One possible consequence of this attack is to isolate the sub-DODAG bound to the attacker which can no longer receive packets, as in our example. This also leads to additional congestion (if the packets are forwarded through sub-optimal paths), partitions and instabilities in the network. The consequences for the children nodes include starvation and delay [10]. To reduce the effects of this attack on the network, RFC 6553 proposes to limit the rate of the downward routing entries discarded due to an 'F' flag to 20 per hour [32]. Once this value is reached 'F' flag packets are no longer taken into account.

3.3.3 Analysis

Table 3.2 synthesizes attacks targeting the topology. We notice that the attacker has to be both internal and active for these attacks. Indeed, the malicious node has to join the graph to manipulate the topology. The attacks related to routing tables such as routing table falsifications and DAO inconsistency attacks need the storing mode to be enabled. Also the RPL option header has to be implemented for the second attack since the malicious node misuses the data path validation which

relies on this header. At least two malicious intruders are required to perform the wormhole attack.

Table 3.2: Summary of attacks on topology.

Attacks	I/E	A/P	Prerequisites	Impact	CIA	Mitigation/ Protection	Overhead
Routing Table Fals.	I	A	Storing Mode	Target's Subnet, D	A/I	None	None
Sinkhole	I	A	-	Attacker's Subnet and Neighborhood, D/U	A/I	SVELTE [63], Rank verification [28] and Parent fail-over [79]	Low, No evaluation
Wormhole	I	A	2 intruders min.	Attackers Subnet, D/U	A/I	Geographical data [78], Merkel trees [34]	No evaluation (for both)
Routing Information Replay	I	A	-	Attacker's Neighborhood, D/U	A/I	Sequence Number [80]	None (by default in RPL)
Worst Parent	I	A	-	Attacker's Subnet, D/U	A/I	None	None
Blackhole	I	A	-	Attacker's Subnet, D/U	A/I	Monitoring of counters [18], Parent fail-over [79], SVELTE[63]	Depends on the solution, No evaluation
DAO Inconsistency	I	A	Storing Mode, Option Header	Target's Subnet, D	A/I	Limitation of discarding routing state [32]	Low

In this table, the impact characterizes how the topology of the network is affected (modified or isolated) and what type of traffic is concerned (e.g. downward (D) or upward traffic (U)). We consider two main areas that may be impacted: (1) the neighborhood of a RPL node corresponding to nodes in the direct vicinity such as parents, children, and siblings nodes, and (2) the subnet of a node. We can observe in that table that routing table falsification and DAO inconsistency attacks are characterized by a similar impact. Indeed, these two attacks corrupt the routing tables of the target. Only downward traffic is concerned because routing tables are only used for downward routing. Therefore, the subnet of the target is modified but the upward traffic is not disturbed. All the other attacks can have consequences on both upward and downward traffic since they concern all types of packets. In that case, both the subnet and/or the neighborhood can be damaged. These attacks do not target a specific node but try to impact on the overall network traffic in general, even if some filtering can also be performed.

Regarding the next property, the availability is impacted in all attacks because

the malicious node modifies the topology and then isolates nodes or degrades network performance through sub-optimization. The integrity is also impacted by attacks targeting topology. For instance, routing table falsification attacks and DAO inconsistency attacks alter routing tables. Decreased rank attacks induce that the integrity of packets is jeopardized. Moreover, the routing information held by legitimate nodes such as parent identity, freshness or routing path are corrupted during routing information replay, sinkhole, wormhole, blackhole and worst parent attacks.

Replay attacks can be countered by sequence numbers implemented by default in the RPL protocol; also the optional mechanism proposed in RFC 6553 mitigates the effects of DAO inconsistency. The cost of this mitigation is low because it consists in implementing a fixed threshold. Different authors proposed several counter-measures to topology attacks such as the Rank verification [28], the Parent fail-over [79] or Merkel trees [34], however the costs of these solutions have not been evaluated yet. Chugh et al. [18] have defined methods for efficiently detecting blackholes in these networks. The SVELTE IDS [63] is also designed to detect the sinkhole and blackhole attacks. We notice that there is no solution for routing table falsification since this attack has not been studied in the context of the RPL protocol. The worst parent attack also does not have any counter-measures although this threat has been studied [43].

3.4 Attacks on Traffic

This third category concerns the attacks targeting the RPL network traffic. It mainly includes eavesdropping attacks on the one hand, and misappropriation attacks on the other hand.

3.4.1 Eavesdropping Attacks

The pervasive nature of RPL networks may facilitate the deployment of malicious nodes performing eavesdropping activities such as sniffing and analyzing the traffic of the network.

3.4.1.1 Sniffing Attacks

A sniffing attack consists in listening to the packets transmitted over the network. This attack is very common in wired and wireless networks and compromises the confidentiality of communications. An attacker can perform this attack using a compromised device or directly capture the packets from the shared medium in case of wireless networks. The information obtained from the sniffed packets may include partial topology, routing information and data content. In RPL networks, if an attacker sniffs control messages, it can access information regarding the DODAG configuration such as DODAG ID, version number, ranks of the nodes located in the neighborhood. By sniffing data packets, the attacks can not only discover packet content but also have a local view of the topology in the eavesdropped area by looking at source/destination addresses. This attack is difficult to be detected due

to its passive nature. The only way to prevent sniffing is encryption of messages when the attacker is external. Even if RFC 6550 mentions encryption of control messages as an option, the technical details are left out from the specification making implementation difficult.

3.4.1.2 Traffic Analysis Attacks

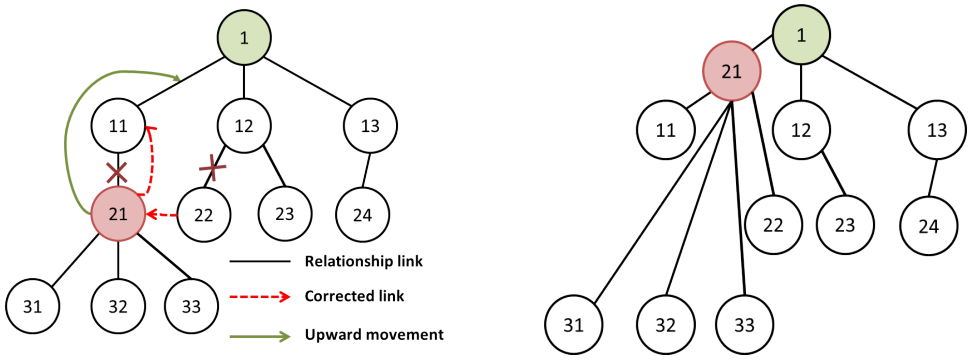
Traffic analysis aims at getting routing information by using the characteristics and patterns of the traffic on a link. This attack can be performed even if the packets are encrypted. The objective is, like sniffing attacks, to gather information about the RPL network such as a partial view of the topology by identifying parent/children relationships. Thanks to this attack, a malicious node can possibly perform other attacks with the gathered information. The consequences depend on the rank of the attacker. If this one is close to the root node, it can process a large amount of traffic and therefore can get more information than when the node is located on the edge of a sub-DODAG.

3.4.2 Misappropriation Attacks

In misappropriation attacks, the identity of a legitimate node is usurped or performance are overclaimed. These attacks are not so damaging for the RPL network per se. However, they are often used as a first step for other attacks such as those seen in the previous two main categories. They allow the attacker to gain a better understanding of the network and its topology, to get better access or to intercept a large part of the traffic.

3.4.2.1 Decreased Rank Attacks

In a DODAG graph, the lower the rank is, the closer the node is to the root and the more traffic this node has to manage. When a malicious node illegitimately advertises a lower rank value, it overclaims its performance. As a result, many legitimate nodes connect to the DODAG graph via the attacker. This results in the attraction of a large part of the traffic, as showed in Figure 3.5. Thanks to this operation, the malicious node is capable of performing other attacks such as sinkhole and eavesdropping attacks. In the RPL protocol, an attacker can change its rank value through the falsification of DIO messages. The VeRa [28] solution as well as the Rank verification method [79] described in Sections 3.3.1.2 and 3.2.2.3 are able to address this issue. However, authors in [40] have showed that VeRa is not secure regarding rank authentication and they proposed improvements to address this issue called TRAIL. They also showed another way to perform this attack by replaying the rank of the attacker's parent which allows it to decrease its rank by one. Since SVELTE [63] can detect sinkhole attacks it can also detect the decreased rank attack.



(a) Initial State (b) Final State

Figure 3.5: Illustration of a decreased rank attack.

3.4.2.2 Identity Attacks

Identity attacks gather both spoofing and sybil attacks. A spoofing attack also called Clone ID attack occurs when a malicious node pretends to be a legitimate existing node. In RPL networks, the root node plays a key role in a DODAG graph. It builds and maintains the topology by sending routing information. An attacker may sniff the network traffic to identify the root node. Once this identification is performed, it can spoof the address of the DODAG root and take the control over the network. During sybil attacks [26], one malicious node uses several logical entities on the same physical node. Identity attacks are used as a premise to perform other operations. In [78], the authors have showed that the RPL protocol cannot solve this issue by itself and proposed to consider geographical data to detect such attacks.

3.4.3 Analysis

As we can observe in Table 3.3, eavesdropping attacks can be performed externally. They are usually exploited to gain access to the internal network. As for the other categories, the attacker has to be an insider to perform misappropriation attacks. Only the eavesdropping attacks have been classified as passive attacks. All the other identified attacks induce that the attacker generates or modifies packets. Passive attacks are quite difficult to detect, in particular in RPL networks which are often supported by wireless links. There is no particular prerequisite for attacks on traffic.

Regarding attacks on traffic, Table 3.3 describes the cases where the consequences can be considered as critical for the RPL network. The effect of eavesdropping attacks depends on the nature of the listened data. For instance, data content may be of high importance for patients in the area of healthcare sensor networks, while the criticality is lower when the objective of the RPL network is simply to collect weather temperatures. In case of misappropriation attacks, the consequences are determined by the location of the malicious or spoofed node. Indeed, when the malicious node has a lower rank, it is closer to the root. It is therefore capable of intercepting a

Table 3.3: Summary of attacks on traffic.

Attacks	I/E	A/P	Prerequisites	Impact	CIA	Mitigation/ Protection	Overhead
Sniffing	I/E	P	-	Critical data	C	Encryption [80]	Depends on the algorithms
Traffic Analysis	I/E	P	-	Critical data	C	None	None
Decreased Rank	I	A	-	Node's rank	I	VeRA [28], TRAIL [40], Rank verification [79], SVELTE [63]	Low, Low, Not evaluated, Low
Identity attack	I	A	-	Node's rank	I	None	None

larger amount of data and the opportunities to attack the RPL network are bigger.

The next property to be discussed is the classification according to the CIA model. The confidentiality aspect concerns eavesdropping attacks, where the goal of the attacker is to obtain information about the network configuration. Due to the nature of misappropriation attacks, the integrity property is affected in these cases.

The only way to prevent sniffing is to use encryption. However, in our security model we assumed that the attacker is able to break the cryptography due to the physical constraints of RPL networks. As mentioned previously, even if cryptographic mechanisms are suggested in the standard, it is difficult to implement them because important feature like key-management are left out by the RFC. Moreover, cryptographic algorithms are known to be resource consuming in terms of memory and computation. Current RPL implementations, as such, do not enable secure operation modes. To our knowledge, there is no current existing solution to prevent traffic analysis and identity attacks in RPL networks. However, the decreased rank attack has been widely studied because it is also used in sinkhole attacks and several counter-measures has been proposed.

3.5 Conclusions

Considering the nature of RPL networks it is mandatory to identify and analyze the security attacks to which this protocol is exposed. We have therefore proposed, in this chapter, a taxonomy classifying the attacks against the RPL protocol in three main categories. The attacks against resources reduce network lifetime through the generation of fake control messages or the building of loops. The attacks against the topology make the network converge to a sub-optimal configuration or they isolate nodes. Finally, attacks against network traffic let a malicious node capture and analyze large part of the traffic. Based on this taxonomy, we have compared the properties of these attacks and discussed methods and techniques to avoid or prevent

them.

We have showed how the RPL protocol is exposed to a large variety of attacks. In order to complement this analysis, we need to quantify the consequences of these attacks. We therefore want to study further attacks in order to evaluate their behavior and their impact on a RPL network according to relevant metrics.

Chapter 4

Impact Assessment of RPL Attacks

Contents

4.1 Introduction 43
4.2 The DAG Inconsistency Attack 44
4.2.1 Attack Description 44
4.2.2 Simulation Setup 46
4.2.3 Impact Quantification 46
4.3 The Version Number Attack 48
4.3.1 Attack Description 48
4.3.2 Simulation Setup 49
4.3.3 Impact Quantification 50
4.4 Conclusions 55

4.1 Introduction

Quantifying the impact of attacks is important to design efficient and accurate security strategies. We consider in this chapter two attacks specific to RPL: the DAG inconsistency attack and the version number attack because they target the node resources, as presented in our taxonomy, which implies a shortened lifetime of the network.

We have showed in Chapter 3 that the DAG inconsistency attack exploits the RPL data path validation feature which is used to avoid and detect possible loops within the network. Packet information is transported in an IPv6 option header. Three flags are defined: the down 'O' flag indicating the expected direction of a packet, the rank-error 'R' flag signaling if a mismatch occurred between the down flag and the actual direction of the packet and the forwarding-error 'F' flag used to indicate whether a node cannot reach a destination. The attack consists in manipulating the 'O' and 'R' flags in the IPv6 option header of regular data packets to introduce

fake loops in the network. Meanwhile the version number attack exploits the global repair mechanism provided by the RPL protocol to ensure an optimized topology. A malicious node may modify the version number associated to a topology, thereby forcing a rebuild of the entire routing tree. Since the version number is included in control messages by parents, there is no mechanism provided by the standardized protocol to guarantee the integrity of the advertised version number.

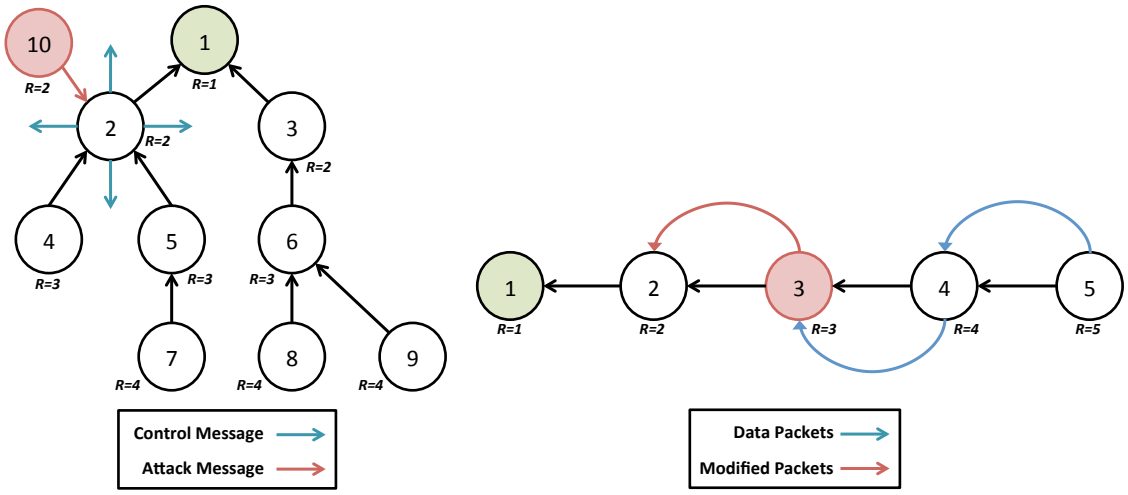
We focus on the DAG inconsistency attack in Section 4.2 and present two possible ways for a malicious node to perform it. The consequences of these attack scenarios are evaluated through experiments. We then analyze in Section 4.3 the version number attack and also assess its impact on a RPL network.

4.2 The DAG Inconsistency Attack

In this section, we first present the different scenarios for the DAG inconsistency attack. After describing our simulation setup, we detail the evaluation results.

4.2.1 Attack Description

The RPL data path validation mechanism was designed to improve reliability of the protocol. However, a malicious node can misuse it in order to attack the network; this is called a DAG inconsistency attack. This attack can be used to directly harm a targeted node, or to manipulate packet headers and force the next-hop node to drop the modified packets.



(a) Direct attack scenario.

(b) Indirect attack scenario.

Figure 4.1: DAG inconsistency attack scenarios.

4.2.1.1 Direct Attack Scenario

A malicious intruder can directly attack its neighborhood by sending packets that have the 'R' flag and the wrong direction set. For instance, if a parent is targeted, the attacker can send packets with the 'O' and 'R' flags set, since packets with 'O' flag are intended for descendant nodes. The parent will detect an inconsistency and thus, drop the packet and restart its trickle timer.

Resetting the trickle timer causes control messages to be sent more frequently which leads to local instability in the network. This increased control message overhead reduces channel availability and increases energy consumption which can lead to a shortened network lifetime in case nodes are battery operated. Since nodes in RPL networks are likely to be resource constrained, they are unlikely to support multi-tasking or large packet buffers. As such, time spent on processing malicious packets could lead to loss of genuine ones.

Figure 4.1(a) depicts a scenario where such an attack takes place. In this case, a stable network topology of ten nodes is formed using RPL. Node 10 assumes the role of an attacker by sending messages, with the 'O' and 'R' flags set, to node 2, its parent. Node 2 resets its trickle timer, thereby flooding its neighborhood with control messages and affecting nodes 4 and 5 as well.

4.2.1.2 Packet Manipulation Scenario

In this scenario, the malicious intruder modifies the IPv6 header of packets it forwards such that the 'R' flag and the 'O' flag representing the wrong direction are set. The receiving node assumes that a DAG inconsistency has taken place and discards the packet. As a result, the malicious node succeeds in forming a black-hole at the next-hop node. This attack could either be carried out on all packets forwarded by the malicious node, or selectively based on source, destination, or even type of message.

The nodes originating the message cannot easily detect this forced black-hole because the packet is not dropped by their next-hop, but by a node that is at least two hops away. If the malicious node itself were to drop the packets, its children could detect this by enabling the promiscuous mode. But the promiscuous mode is not an option in such a scenario since in most RPL networks, a node that is two hops away is usually out of radio range as well [18]. This means, that only the attacker is within the radio range of both the sender and the node that drops the packet. The modification of header could be detected by nodes with the promiscuous mode enabled, however, it is known to consume node energy, which is not suitable for C0 devices.

In general this approach is an advantageous strategy for the attacker to force another node to drop the packets. Furthermore, if the control packets originating from the malicious node are normal, then the malicious activity is completely hidden. In this scenario, not only the delivery ratio decreases, but the control overhead of RPL nodes also increases along with deteriorating channel availability and increasing energy consumption.

For example, in the DODAG depicted by Figure 4.1(b) node 3 is the attacker.

Before forwarding data packets from its descendants, nodes 4 and 5, it modifies them such that the 'O' and 'R' flags are set. As a consequence, node 2 drops them, thereby becoming akin to a black-hole. This causes the delivery ratio for nodes 4 and 5, descendants of node 3 to be severely harmed. Node 2 also resets its trickle timer causing an increase in overhead as well.

4.2.2 Simulation Setup

The Contiki 2.6 operating system has been used for evaluating the DAG inconsistency attack since it provides an RPL implementation that works on multiple platforms. However, the RPL implementation provided by Contiki did not support correct handling of data path validation mechanism. While packets with the 'R' flag are dropped, the trickle timers are not reset. As such, the implementation was modified to correct this functionality.

The TelosB, also known as the TMote Sky, has been used as the development platform since its computational resources allow it to function as an RPL router node with the Contiki RPL implementation. To permit evaluation under multiple scenarios, instead of building a topology of actual nodes, the compiled binary for a TelosB was used in the Cooja [59] simulator provided by Contiki with Unit Disk Graph radio attenuation and scattering model (UDGM). This approach provides a method to quantify the DAG inconsistency attack impact under conditions where the lossy IEEE 802.15.4 channel does not cause packet loss. This allows evaluation under ideal conditions, with no external characteristics causing bias in the results. The Cooja simulator is quite close to real hardware since it uses the MSPSim software to emulate the MSP430 architecture and the performance of a MSP430F1611 microcontroller, which is utilized by the TelosB. A validation of the simulation tool will be provided in Chapter 5.

4.2.3 Impact Quantification

The consequences of the direct attack scenario are first evaluated using the control message overhead metric. We then study the impact of the packet manipulation scenario based on control message overhead and overall delivery ratio metrics.

4.2.3.1 Direct Attack Scenario Evaluation

To evaluate the control message overhead caused by the DAG inconsistency attack, the topology showed in Figure 4.1(a) has been setup in Cooja, with node 1, the DODAG root, acting as the sink. Other nodes have been configured to send messages to the sink to generate a background traffic (every six seconds). To avoid packet collisions and add a degree of irregularity to the transmission scenario, an additional back-off period of up to six seconds has been added. The RPL implementation has been setup to always reset trickle timers as indicated by RFC 6550 [80]. The attacker, i.e., node 10 in Figure 4.1(a), has been setup to periodically send packets, between 15 to 90 msgs/hr, with the 'O' and 'R' flags towards the sink. Each simulation has been

executed for one hour and the attacks have begun after 2 minutes so as to allow the network to stabilize. A simulation with no attacks has also been performed to obtain a baseline measurement for comparison. More complex topologies and scenarios have not been studied because the effects of such attacks are limited to the target and its immediate neighborhood. As such, our scenarios are carefully designed to study the possible effects of such attacks.

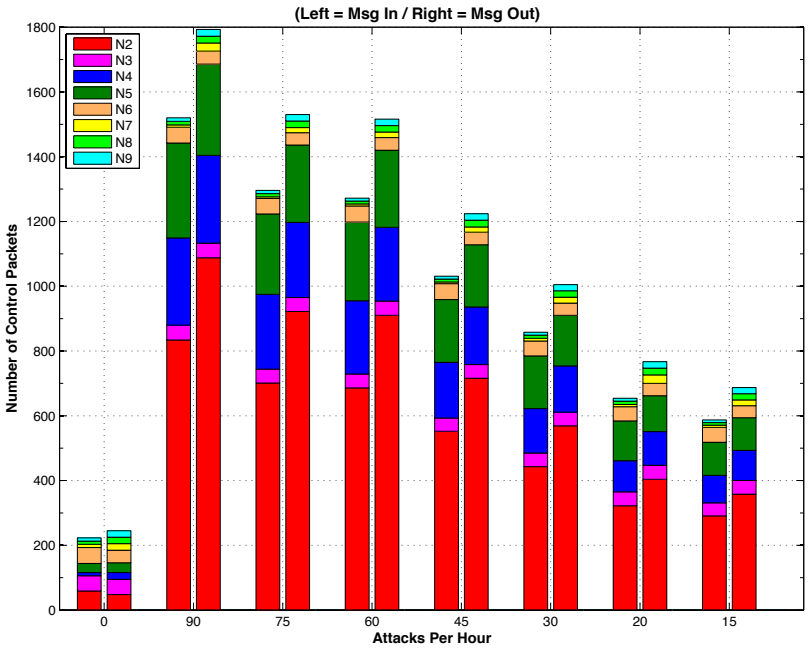


Figure 4.2: Total control message overhead experienced by network presented in Figure 4.1(a) under DAG inconsistency attacks.

Results of this experiment can be seen in Figure 4.2. As expected, the more aggressive the attacker, the higher the overall message overhead in the network. Node 2 experiences the largest increase in control messages since it is directly targeted. Nodes 4 and 5 also experience an increase due to being direct descendants of node 2. The most aggressive attacker tested can increase overhead at the targeted node by over 2000%. We can therefore conclude on the importance of mitigating DAG inconsistency attacks.

4.2.3.2 Packet Manipulation Scenario Evaluation

To evaluate the effect of the packet manipulation scenario, the topology showed in Figure 4.1(b) was setup in Cooja, with node 1, the DODAG root, acting as the sink. All other nodes, except the attacker, were configured to send messages to the sink at rates varying from 5 to 20 packets per minute. The packet sending rate is varied, because the attacker, i.e., node 3, silently modifies the option headers of the packets it forwards, rather than originating a direct attack.

The primary effect of packet manipulation attacks is not to increase overhead, but to cause the next-hop node to drop all packets of the attacker's descendants as explained earlier. The emergence of this black-hole can severely impact the overall delivery ratio of packets, since none of the packets from the attacker's descendants will ever reach the sink. Without a black-hole mitigation approach the overall delivery ratio is only about 33% in our results. This is because only packets from node 2 reach the sink, while the attacker forces node 2 to drop all packets sent by nodes 4 and 5.

4.3 The Version Number Attack

In this section we first describe the version number attack. We then detail our simulation setup. Finally, we study the consequences of this attack in terms of control message overhead, delivery ratio, end-to-end delay, number of loops and inconsistencies.

4.3.1 Attack Description

The version number is used by the root to control the global repair process of RPL and to ensure that all nodes in the DODAG are up-to-date with the routing state. Every DIO message carries the version number so that nodes which are part of an outdated DODAG version, can join the new DODAG by recalculating their rank and then updating their stored version number.

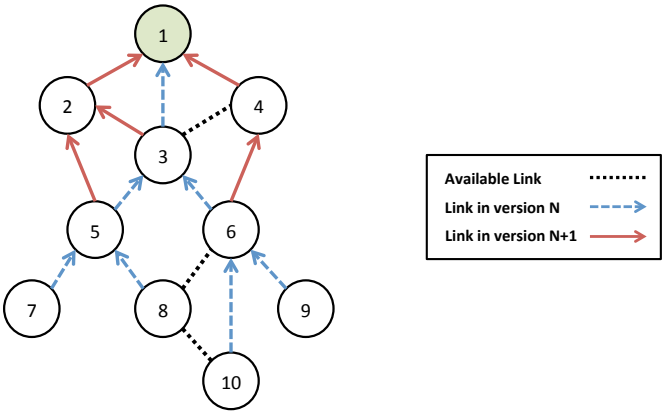


Figure 4.3: Example of new DODAG iteration.

Figure 4.3 illustrates a rebuild of a DODAG due to a global repair process. In red solid arrows, the new DODAG is being built, in blue dashed arrows the old topology is showed. An older value of the version advertised in DIO messages indicates that the node has not migrated to the new version of the DODAG. Such a node should not be considered as a preferred parent by other nodes. While the global rebuild process is ongoing, it is possible for two versions of a DODAG to temporarily coexist. To avoid loops, data packets are permitted to transit from the old version to the new

one (from blue to red in Figure 4.3) but not the other way, as showed in Figure 4.3. This is because the old version in blue is no longer a DAG and loop free topologies cannot be guaranteed in this situation.

To avoid possible inconsistencies in the network, the version number should be propagated unchanged through the DODAG. However, there is no mechanism in RPL to check if the integrity of the version number is maintained in received DIO messages. A malicious node may change this field in its own DIO messages to harm the network. Nodes receiving a malicious DIO, with a new version number, will reset their own trickle timer, update the version in their own records and advertise this new version through DIO messages to their neighborhood as well. This can cause the illegitimate version number to propagate through the network.

Such a manipulation of the version number in the DIO packets does not only cause an unnecessary rebuild of the whole DODAG but it also generates loops in the topology. This can negatively impact energy reserves of the nodes, routing of data packets and channel availability.

4.3.2 Simulation Setup

We have used the same operating system (Contiki 2.6) and the same target platform (TelosB) to perform an evaluation of the version number attack as previoulsy described in Section 4.2.2. A grid topology of 20 nodes using the UDGM radio model [59], showed in Figure 4.4, was setup for all experiments in Cooja. Across all experiments, node 1 is the DODAG root and also the sink to which all other nodes send messages every twenty seconds to generate background traffic. The attacker is designed to constantly send incorrect version numbers, which are greater than the root's. This scenario is adopted because it allows relocation of the attacker to multiple positions easily, making it possible to study the consequences of the attack from different locations and neighborhood scenarios within a network. A random back-off of up to six seconds is also added to this periodic transmission time on all nodes so that packet collisions are avoided when possible. The nodes are placed at a regular distance of 30 meters from their vertical and horizontal neighbors. The transmission strength is set such that packets are received successfully by nodes within a 30 meters radius and the signal causes interference with other nodes for a radius of 60 meters. This ensures that every node only has vertical and horizontal neighbors reachable during the simulation, thereby adding predictability and ease of analysis to the results.

Each simulation lasts for a lifetime of fifty minutes. One simulation was executed without any attacker in the network to obtain a baseline for comparisons. Further nineteen simulations are also run, with the location of the attacker being fixed to one of nodes 2 to 20, such that at least one simulation with the attacker located at every node between 2 and 20 is executed. Moving the position of the attacker in the network allows us to study the impact that the position of an attacker and the size of the neighborhood have upon the behavior of the RPL network. This entire set of twenty simulations is repeated five times to obtain some statistical significance in order to ensure dependability. Attacks start after five minutes of simulation time, so

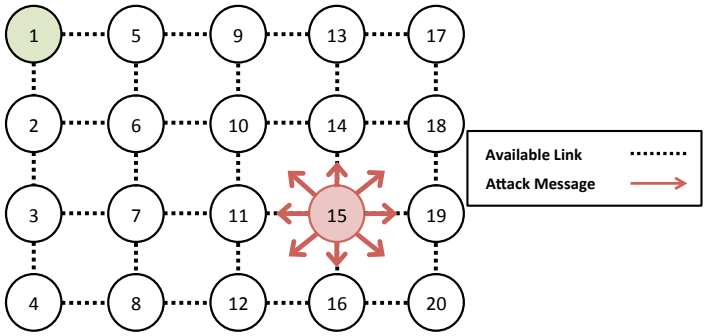


Figure 4.4: Grid topology used for performing experimental evaluation of the version number attack.

that the network has enough time to settle and a stable RPL topology emerges.

4.3.3 Impact Quantification

The following metrics are used to perform this study: (1) *Packet overhead*, which is the total number of RPL control packets, i.e. DIS, DIO and DAO message, transmitted (outgoing overhead) and received (incoming overhead) in the network. As such, in the no attacker scenario these are the messages necessary to form and maintain an RPL DODAG. (2) *Delivery ratio*, which is the number of data packets successfully delivered to the sink (node 1) compared to the total number of data packets generated by all nodes in the network. (3) *Average end-to-end delay*, which is the average amount of time it takes for all packets, from every node in the network, to be successfully delivered to the sink. Lost and dropped packets are not considered in this calculation. (4) *Inconsistencies*, which are the number of packets detected by a node that are destined for a descendant but also arrive from a child or vice versa. (5) *Loops*, which are the number of packets detected by a node that not only indicate an inconsistency but also have the 'R' flag enabled, i.e. a possible loop was previously detected on this path.

Packet overhead. The average incoming and outgoing packet overhead experienced by the entire network, for each location of the attacker, is depicted in Figure 4.5. The error bars show the standard deviation between the five simulation runs. The incoming and outgoing overhead when there is no attacker (attacker ID 0 in the figure), both are about 1250 packets, which can be considered reasonable for a network of 20 nodes that functions for 50 minutes. However, as soon as an attacker is introduced, the overhead can increase by up to 18 times in the network. At first glance it appears that the overhead increases as the attacker moves into regions where it has more neighbors. Closer inspection of Figure 4.5 reveals that mostly attacker positions in the bottom row of the topology (4, 8, 12, 16 and 20) produce localized maximums in their column of the topology. For example considering attacker position at node 8 and the other nodes of its column (5,6,7), we can

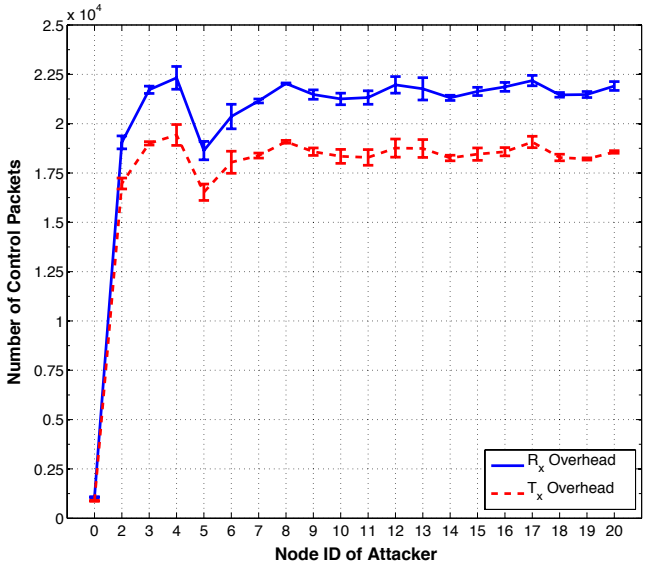


Figure 4.5: Incoming and outgoing control message overhead for every location of the attacker.

observe that the packet overhead is the highest for node 8 compared to nodes 5, 6 and 7. Since each of these nodes ends up towards the tail-end of their section of the DODAG, it also implies that the further away an attacker is from the root, the more damage it can spread. When the attacker is located at nodes 2 and 5, the produced topologies are analogous to each other, thereby leading to results that are similar in both cases. As such, not only the number of neighbors, but also the distance from the root impacts the level of overhead increase.

Since the position of the attacker can impact the overhead, it is interesting to also investigate which nodes are particularly affected. Only the per node outgoing packet overhead is plotted in Figure 4.6, because the incoming and outgoing packet overhead is closely related. While it is intuitive to assume that the largest increase in overhead would be contributed by nodes neighboring the attacker, because these are most likely to form loops, the results from Figure 4.6 indicate otherwise. The version number attack, by design, is propagated across all neighbors, even if they are not relatives of the current attacker. This causes a significant increase in control packets to cascade all across the network, leading to the observed results. As such, a version number attack is worse than many others because it does not only impact the attacker's neighborhood but also the entire network.

Delivery ratio. This increased overhead decreases channel availability, thereby impacting delivery ratio. The delivery ratio, averaged over five runs, for the entire network, with respect to the location of the attacker, is showed in Figure 4.7. It is immediately apparent that the version number attack can have a significant impact

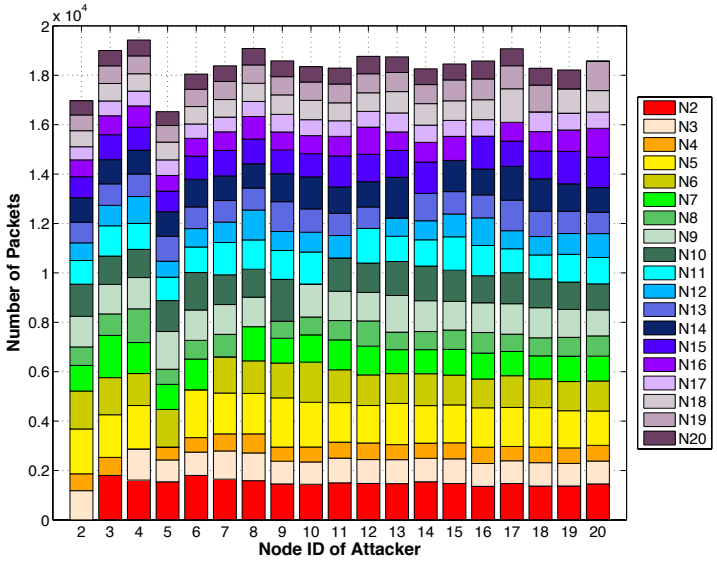


Figure 4.6: Per node outgoing packet overhead for every location of the attacker.

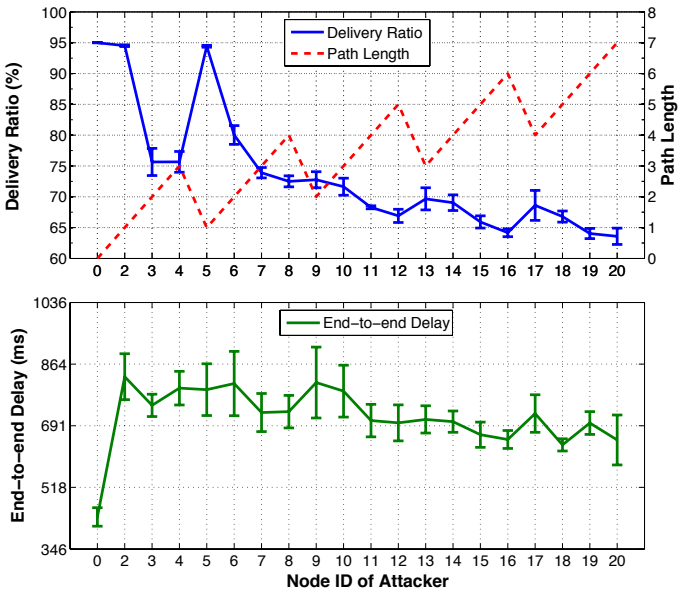


Figure 4.7: Total delivery ratio and end-to-end delay for every location of the attacker.

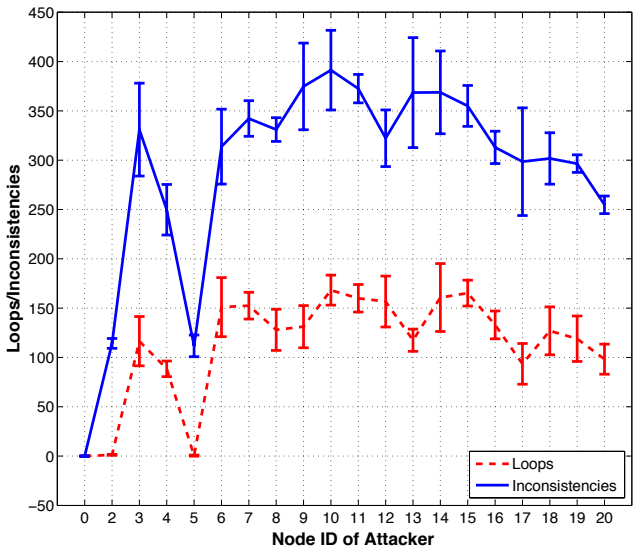


Figure 4.8: Total number of loops and inconsistencies in the network for every location of the attacker.

on delivery ratio, with it being reduced by up to about 30%. More interestingly, we can see that the path length of the location of the attacker has an effect on the delivery ratio as we have seen on the overhead. However, instead of local maximums for their column, we observe local minimums. There is a strong correlation between the distance of the attacker from the root and the impact of the version number attack on delivery ratio. So, when the attacker is located on nodes 2 and 5, the delivery ratio is exactly the same, and they both also have a path length of 1 to the root. In fact, the correlation between the delivery ratio and path length can be seen across all positions of the attacker, with the attacker located at the bottom of the topology in Figure 4.4, i.e. farthest from the root, leading to worst impact on delivery ratio.

End-to-end delay. The end-to-end delay is also a good measure of a network's performance. The average end-to-end delay for different attacker locations can also be seen in Figure 4.7. Any lost packet did not contribute towards calculation of the end-to-end delay. As with other metrics, it is obvious that an attack significantly impacts end-to-end delay, by almost doubling it as against no attack within the network. Unlike overhead and delivery ratio, there is no strong correlation between location of the attacker and the delay. This is because the delay is affected by a number of elements, such as the channel availability, number of loops, possible alternate routes, neighborhood density, etc. which also cause the high variations observed in Figure 4.7.

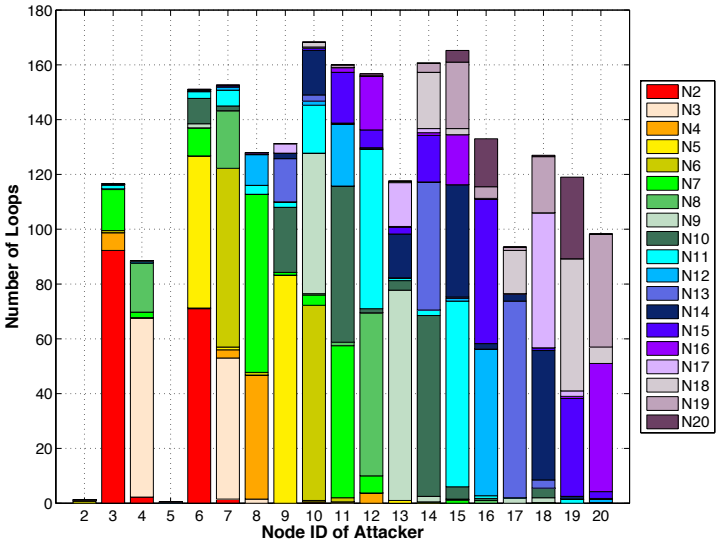


Figure 4.9: Total number of loops detected per node for every location of the attacker.

Loops and inconsistencies. Since the version number attack creates loops (packets encountered with the 'R' flag) and rank inconsistencies (packets that mismatch actual direction and have 'O' flag set) in the network, it is important to also understand their impacts. The number of such inconsistencies can be seen in Figure 4.8. The pattern of rank inconsistencies and loops are closely related because loops are included in inconsistencies. Unlike previously, attacker locations farthest from the root generally lead to the least number of inconsistencies and loops in the network. On the other hand, attacker locations closest to the root, but with most amount of neighbors lead to the highest number of inconsistencies and loops. For example, nodes 3 and 6 create more inconsistencies than nodes 2 and 5, which are closer to the root node. This is because both nodes 2 and 5 have fewer neighbors than nodes 3 and 6. As such, the number of loops are closely related to the number of neighbors an attacker has and increases with proximity to the root. Closer proximity to the root likely has this behavior because it forces a rebuild from the root faster as well, causing this to cascade into the rest of the DODAG before a new attack cycle can begin.

This interesting relationship of inconsistencies and loops with the location of the attacker means that it would be useful to understand where in the network most of this effect is centered. As such, the number of loops and inconsistencies per node are plotted in Figures 4.9 and 4.10 respectively. It is immediately apparent from these plots that as the location of the attacker changes, the loops and inconsistencies location changes accordingly. Closer analysis reveals that while there might be some loops created in nodes that are farther away from the attacker, the majority of them are located within the direct neighborhood of the attacker. In fact, the bulk of these

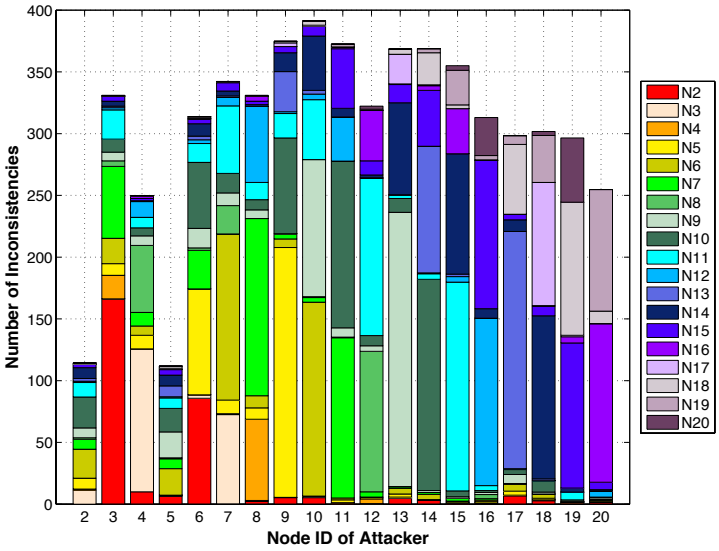


Figure 4.10: Total number of inconsistencies detected per node for every location of the attacker.

loops and inconsistencies are detected by the parent, and alternate parents, of the attackers. This is because most of the packets will be routed towards the preferred parent, or the alternate parent in case of the preferred parent being unavailable. The next highest quantity of loops and inconsistencies is detected at the children of the attackers. For example, when the attacker is located at node 11, the highest number of loops and inconsistencies are detected by parent nodes 7 and 10. The children, nodes 12 and 15, account for the majority of the rest of these anomalous situations.

4.4 Conclusions

In this chapter, we have presented two attacks targeting the node resources: the DAG inconsistency and the version number attacks. We have also analyzed their impact.

The DAG inconsistency attack exploits the data path validation to forge fake loops in the network forcing nodes to reset their trickle timer. We have showed two ways of performing this attack: a direct attack scenario where a malicious node directly sends attack messages to a targeted node and a packet manipulation scenario where the malicious node illegitimately modifies data packets from other nodes creating a black-hole in the network. Through experiments, we have observed that this attack can increase significantly the control message overhead of the targeted node and its descendants, reducing their lifetime. In the packet manipulation scenario we have also seen that this attack may have a significant impact on the delivery ratio.

The version number attack misuses the global repair mechanism provided by the

repair protocol to propagate within the network malicious version number forcing a global rebuild of the DODAG to occur. We have quantified its effects in RPL networks. Through simulations we have discovered that this type of attacks that control message overhead can increase by up to 18 times, thereby impacting energy consumption and channel availability. This in turn can reduce the delivery ratio of packets by up to 30% and nearly double the end-to-end delay in a network. A strong correlation between the position of the attacker and the effect on the network has been also observed. An attacker located as far away from the root as possible causes the highest increase in overhead, and similarly a higher path length between the attacker to the root also causes the higher packet loss. It has been also discovered that loops and rank inconsistencies created by the attack are generally located around the neighborhood of the attacker, with parents or alternate parents experiencing the maximum loops, followed by the descendants.

Through these analyses, we can conclude on the importance of addressing these attacks whose impact may considerably shorten network lifetime. The designed security strategy should be as lightweight as possible to preserve the scarce resources of nodes while being efficient to tackle these threats.

Chapter 5

Local Strategy for Addressing DAG Inconsistency Attack

Contents

5.1 Introduction 57
5.2 DAG Inconsistency Attack Mitigation 58
5.2.1 Default Mitigation 58
5.2.2 Adaptive Mitigation 59
5.2.3 Dynamic Mitigation 60
5.3 Mitigation Evaluation 62
5.3.1 Simulation Setup 63
5.3.2 Mitigation Performance 64
5.3.3 Configuration Parameters Impact 69
5.3.4 Resource Consumption 73
5.4 Conclusions 76

5.1 Introduction

We have previously demonstrated the importance of addressing DAG inconsistency and version number attacks because they damage the network by increasing the overhead and reducing the delivery ratio. By analyzing properties of these attacks we have observed that the first one targets a particular node and impacts mostly the target's descendants while the second one is propagated through the entire DODAG graph. In that context, we can assume that it is possible to locally deploy defense mechanisms to counteract the DAG inconsistency attack, but it is not feasible for the version number attack due to its propagation properties. In this chapter, we propose to investigate local detection and mitigation approaches to limit DAG inconsistency attack impact.

A sided RFC of RPL [32] proposes to counteract this attack by using a fixed threshold, beyond which all subsequent packets with erroneous header options are ignored. However, the fixed threshold is arbitrarily set and does not resolve the black-hole issue either. In this chapter, we present a new solution called adaptive threshold (AT) to mitigate effects of such an attack. This initial AT approach is further improved to dynamically take into account network characteristics while deriving an appropriate threshold for counteracting the DAG inconsistency attack. Our experimental results show that both our proposed approaches can lead to improvements over the fixed threshold. Furthermore, our approaches are able to counteract the black-hole scenario while still outperforming the default RPL mitigation strategy.

This chapter is organized as follows. Section 5.2 presents the different mitigation approaches for the DAG inconsistency attack which are the fixed threshold, the adaptive threshold and finally the dynamic threshold. Section 5.3 details our experimental evaluation by comparing the different mitigation strategies, discussing the values of the parameters and analyzing the energy consumption of the proposed approaches.

5.2 DAG Inconsistency Attack Mitigation

We introduce three detection and mitigation approaches to counter the DAG inconsistency attack. The first one is proposed by a sided RFC [32] of the RPL protocol. It is based on a fixed threshold. Since no justification is provided on the threshold value in the RFC, we have proposed an adaptive threshold relying on set parameters which mimics the fixed approach under normal conditions. We then improve this solution to be fully dynamic with auto-configured parameters.

5.2.1 Default Mitigation

The default DAG inconsistency attack mitigation strategy of RPL [32] can be seen in Algorithm 1, where i is a node within the graph with a rank of r_i. M represents a packet received by node i from a neighbor j with rank r_j. O and R indicate the 'O' and 'R' flags present in M. The variable $count_R$ is the number of received data packets with the 'R' flag set and is initialized to 0. λ is a constant set to 20.

Algorithm 1 Default DAG inconsistency mitigation strategy of a node.

```
if (O = 1 and r_i < r_j) or (O = 0 and r_i > r_j) then
    if R = 1 then
        count_R++
        drop(M)
        if count_R < λ then
            reset(trickle_timer)
        end if
    end if
end if
```

Upon receiving a packet with an inconsistency, the node drops it and resets its own trickle timer. To limit the effects of an attack, the number of trickle timer resets is limited to the recommended constant $\lambda = 20$ [32]. Upon reaching this threshold, malformed packets are dropped but the trickle timer is not reset. The variable $count_R$ is reset every hour, allowing attackers to once again have a higher impact.

This approach limits the impact of a DAG inconsistency attacks, but the value of the threshold $\lambda = 20$ is arbitrarily set. No reasoning is provided to justify this choice or how performance could be improved in case of varying attack scenarios.

5.2.2 Adaptive Mitigation

In order to take into account the current network state and react to varying attack patterns we have developed an adaptive threshold (AT), which determines when to stop resetting the trickle timer. Instead of a constant λ, a function $\lambda(r)$ is used, which takes the following form:

$$\lambda(r) = \lfloor \alpha + \beta \cdot e^{-\gamma \cdot r} \rfloor \tag{5.1}$$

where,

$$r = \frac{count_R}{D_{pkt}}, \quad \alpha = 5, \quad \beta = 15$$

The variable $count_R$ stands for the number of received data packets with the 'R' flag set and D_{pkt} represents the number of normal forwarded data packets. To allow comparison with the default strategy, the value of β was chosen such that the default $\lambda(r) = 20$. The value of α is an asymptote to ensure that threshold never reaches 0. This guarantees that data packet validation is not disabled upon encountering the first packet with an 'R' flag, but only when the situation is deemed an attack. Since γ impacts the threshold rate of change, a value is not chosen here. In general, a larger value for γ leads to a smaller threshold being reached quicker. The effect of choosing different values for γ is discussed in Section 5.3.3.1.

The adaptive threshold causes $\lambda(r)$ to change based on network conditions. If an attacker is aggressive, the threshold drops quickly and increases slowly once the attacks stop. Unlike with the fixed threshold, $count_R$ is not reset every hour, but rather allowed to increase in the absence of attacks. As such, not only is this approach likely to be better than a fixed threshold within the first hour of an attack, but it should perform significantly better against long running attacks. This also ensures that greater trust is placed in networks where problems have not been encountered for a long time. Of course, a natural limit upon the value of the counter is the bit-length of the variable imposed by the platform. In this case, the counter will reset when the value overflows. If any of the counters overflows, we recommend resetting all counters ($count_R$ and D_{pkt}) so that the algorithm functions as though it was started in a new network.

To counter the packet manipulation DAG inconsistency attack, an extension was made to the adaptive threshold. Nodes behave normally until the number of messages indicating an inconsistency becomes greater than the threshold obtained from Equation 5.1 and this threshold reaches the α value. This situation indicates either an attack against the node, or a malfunction of the node forwarding such packets. To rectify the situation, the node clears the 'O' and 'R' flags before forwarding the packets normally. The complete packet manipulation mitigation strategy, combined with the adaptive threshold mitigation approach, can be seen in Algorithm 2. The complexity of this algorithm depends on the complexity of the exponential function used to compute the threshold. Section 5.3.4.1 will discuss the cost of the proposed algorithm. Since no additional resources are used by this approach, the cost of protecting the network against black-hole scenarios is quite low.

Algorithm 2 Adaptive DAG inconsistency mitigation of a node.

```
if (O = 1 and r_i < r_j) or (O = 0 and r_i > r_j) then
    if R = 1 then
        if count_R < λ(r) then
            count_R++
            drop(M)
            reset_trickle_timer()
        else if λ(r) = α then
            O ← 0
            R ← 0
            forward(M)
        end if
    end if
end if
```

The adaptive threshold approach relies on set parameters, which need to be chosen in the implementation. This can lead to sub-optimal optimizations and so we have improved our mitigation approach via the design of a fully dynamic threshold, which is based on network characteristics.

5.2.3 Dynamic Mitigation

We have designed the dynamic threshold (DT) in order to avoid using pre-configured parameters while benefiting from advantage of the adaptive threshold. The new dynamic threshold $\lambda(r)$ used to determine whether the trickle timer should be reset is:

$$\lambda(r) = \lfloor \delta \cdot e^{-\epsilon \cdot r} \rfloor \tag{5.2}$$

where,

$$r = \frac{count_R}{D_{pkt}}, \quad \delta = 2 \cdot \epsilon, \quad \epsilon = Card(neighbors)$$

As before, $count_R$ is the number of received data packets with the 'R' flag set. D_{pkt} represents normal data packets forwarded by the node.

Normally, packets with the 'R' flag set do not arrive at any nodes, because the network is stable and functions as intended. It has been observed, via experiments carried out during this study, that packets with the 'R' flag set arrive only when an attack is performed on the network, or loops form due to serious malfunction of nodes, which is unlikely, unless a software bug exists. Even when the root node initiates a rebuild of the entire network, i.e., a global repair, a maximum of one or two packets containing 'R' flags are received from each child. Any given local neighborhood in an RPL network always returns to stability within two packets containing an 'R' flag, if the problem is a genuine topological inconsistency.

As such, setting δ to twice the number of neighbors (parents and children represented by $Card(neighbors)$ or ϵ in Equation 5.2) permits for each link to send up to two packets with an 'R' flag set in case of legitimate loops. $\lambda(r)$ corresponds to the value of δ in a steady state, i.e., when no packets with 'R' flags are received.

Even though not observed during our experiments, it is possible for multiple packets with an 'R' flag to arrive as a result of the same inconsistency. This can be especially true in case a node malfunctions, leading to a loop being formed. Resetting the trickle timer each time a malfunctioning node sends packets with 'R' flags leads to unnecessary overhead, especially since a single trickle timer causes aggressive transmissions of DIO messages in any case. To avoid this situation, a convergence timer is introduced in this algorithm. This timer is used to ensure that no further trickle timer resets take place within the amount of time it takes for an RPL neighborhood to typically converge. Previous experiments have showed that time for convergence of a DODAG neighborhood increases by about 2 seconds for every additional 10 neighbors [21]. The $convergence\_timer$ is, as such, set to 2 seconds by default but grows based on neighborhood size of a node.

Since the purpose of introducing a $convergence\_timer$ is to block trickle timer resets caused by 'R' flag packets arriving within the time it takes for the neighborhood to converge, it no longer makes sense to compare $count_R$ with $\lambda(r)$ to determine whether a trickle reset must occur. Rather, a new counter that keeps track of the number of trickle timer resets, $count_T$, is introduced. The value of $count_T$ is reset one hour after the first 'R' flag packet is encountered to allow the repair of genuine loops and because $\lambda(r)$ does not depend on it. Instead of λ representing the number of 'R' flag packets allowed before a trickle timer reset occurs, as with the default mitigation approach, it is now the number of trickle timer resets allowed to be caused by 'R' flag packets that arrive while the neighborhood is already considered to be converged. The overall dynamic threshold approach can be seen in Algorithm 3.

The dynamic threshold allows $\lambda(r)$ to change based on network conditions. Like the adaptive threshold approach, this mitigation strategy should perform better against long running attacks. This dynamic threshold approach not only does away with arbitrary constant thresholds, as in the case of the default strategy, but by being based purely upon network characteristics it does away with the need for constant parameters to be chosen before deployment [71] and thereby is more useful in case

Algorithm 3 Dynamic DAG inconsistency mitigation strategy of a node.

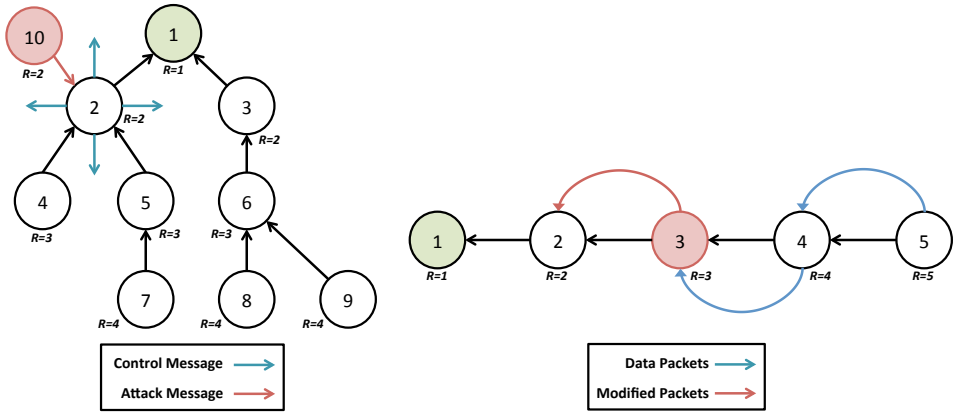
```
if (O = 1 and r_i < r_j) or (O = 0 and r_i > r_j) then
    if R = 1 then
        count_R++
        if count_T < λ(r) then
            if timer_expired(convergence_timer) then
                drop(M)
                start(convergence_timer)
                reset(trickle_timer)
                count_T++
            end if
        else if r >= 1/ε then
            O ← 0
            R ← 0
            forward(M)
        else
            drop(M)
        end if
    end if
end if
```

of unforeseen network conditions as well.

In order to counter the packet manipulation scenario using the dynamic threshold approach, Algorithm 3 allows a node to forward packets with the inappropriate flags in some situations. First, as long as $count_T$ is lesser than $\lambda(r)$, i.e. as long as a direct DAG inconsistency attack or genuine topological error is being corrected, all packets containing the incorrect flags are dropped. However, once this mitigation is over, it is deemed that the network should have returned to normal and any further inconsistency could be a packet manipulation DAG inconsistency attack. As such, if more than $1/\epsilon$ traffic received by the node contains the 'R' flag, then this is considered a packet manipulation attack. It means that one neighbor only sends messages containing 'R' flags. In this case, having given enough chances for the network to fix itself, the node clears the flags and forwards the message normally. The complexity of this algorithm also relies on exponential function such as for the adaptive threshold. The cost of the dynamic threshold computation will be discussed in Section 5.3.4.1.

5.3 Mitigation Evaluation

In this section, we first present our simulation setup along with a validation of our simulation tool. We then detail performance results of the different mitigation approaches. After this, we discuss the effect of the introduced parameters before analyzing resource consumption of our local strategy.



(a) Direct attack scenario. (b) Indirect attack scenario.

Figure 5.1: Topologies for mitigation approaches evaluation (same as Figure 4.1).

5.3.1 Simulation Setup

Such as in the previous chapter, the Contiki 2.6 [27] operating system and the TelosB platform have been chosen. To allow evaluation under multiple scenarios, instead of building a topology of actual nodes, the compiled binary for a TelosB was used in the Cooja [59] simulator provided by Contiki with Unit Disk Graph radio attenuation and scattering model (UDGM). This approach provides a method of testing the different thresholds under conditions where the lossy IEEE 802.15.4 channel does not cause packet loss. This allows evaluation of our approach under ideal conditions, with no external characteristics causing bias in the results.

Simulation Validation. Even though the Cooja approach is expected to be close to real performance, we need to confirm this assumption and validate our simulation approach. As such, the topology showed in Figure 5.1(a), which is the same used for attack analysis in Chapter 4, was setup using real TelosB motes, with node 1, the DODAG root, acting as the sink. All other nodes were configured to send messages to the sink every six seconds. An additional per transmission random back-off period of up to six seconds was utilized to avoid packet collisions and add a degree of irregularity to the transmission scenario. The dynamic threshold mitigation mechanism was deployed to all nodes.

The attacker node, i.e., node 10 in Figure 5.1(a), was setup to periodically send packets with the 'O' and 'R' flags towards the sink. This period was varied from 20 to 90 messages sent per hour. The experiment was repeated five times for each attack frequency and lasted for a duration of one hour each time. The amount of outgoing packet overhead at the attacked node, which is the number of DIS, DIO and DAO messages, for varying number of attacks per hour can be seen in Figure 5.2. The same experiment was carried out in Cooja as well. Error-bars represent standard deviation for average from 5 runs.

It is clear from the plot that the results provided by Cooja are within the deviation

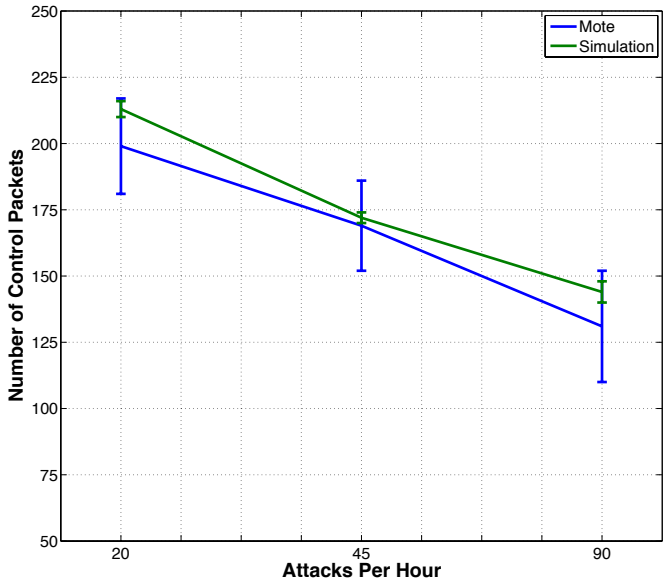


Figure 5.2: Comparison of per node outgoing packet overhead for node 2 in the topology from Fig. 5.1(a).

range of the overhead seen in a network of real motes. This indicates that the Cooja simulations provide results which closely mimics reality. Furthermore, the overhead reported by Cooja is on average higher than in reality because the IEEE 802.15.4 channel causes packets to be lost in a deployment of real motes, whereas this does not occur in Cooja.

A larger topology was not used since the effect of a DAG inconsistency attack is limited mostly to the targeted node. Its children and further descendants are affected only to a small degree. A larger topology would only make the overhead greater, but not change the patterns observed with this topology.

5.3.2 Mitigation Performance

We first evaluate our mitigation algorithms in the direct attack scenario with control message overhead metric. We then analyze the packet manipulation mitigation with the control message overhead and the delivery ratio metrics.

5.3.2.1 Direct Attack Mitigation

Using the same experimental setup as in Section 5.3.1 the performance of the fixed, adaptive and dynamic threshold mitigation approaches have been evaluated using simulations. The attack frequency was varied from 15 to 3600 attacks per hour.

Packet Overhead. When using the fixed threshold to mitigate the DAG inconsistency attack, we can see from Figure 5.3 that worst case overhead for fixed threshold (30 attacks/hr) is reduced by nearly 15%. Aggressive attacks cause the threshold to

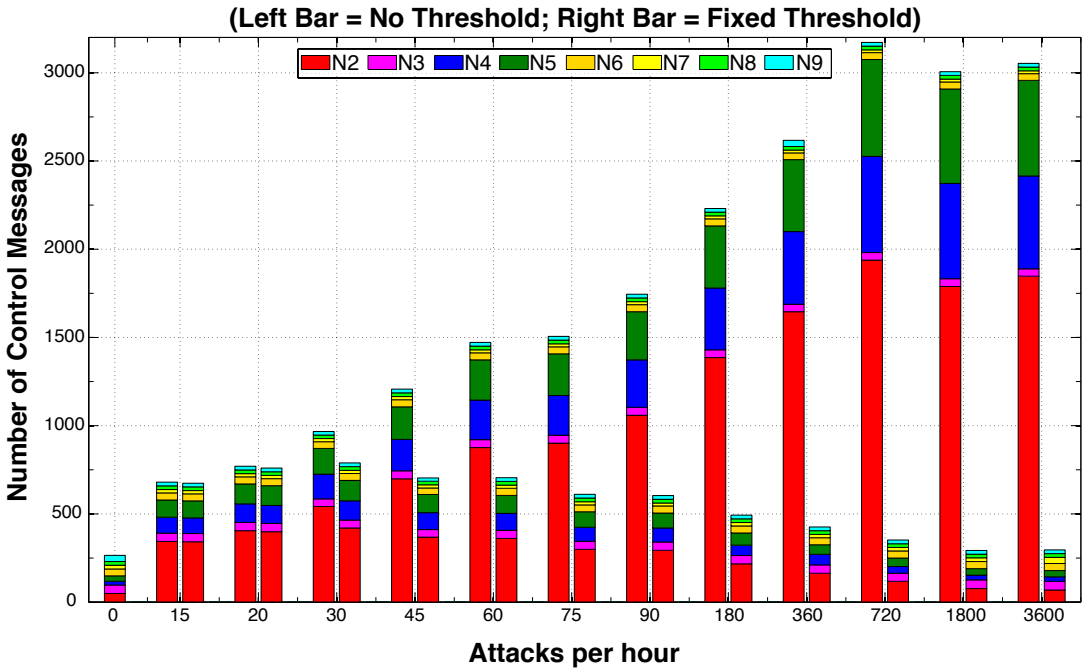


Figure 5.3: Total control message overhead per node when no mitigation strategy and default mitigation strategy are used.

be reached faster, causing lower overhead in these scenarios. As such, the best strategy for an attacker is to remain as close to the threshold as possible, as is evident from the 20 attacks/hr scenario. Since the counter for DODAG inconsistencies is reset every hour, by remaining close to the fixed threshold the attacker can do maximum damage and the nodes have no recourse. While a threshold is undoubtedly useful in mitigating such attacks, adapting it to current network conditions would not allow an attacker to keep just below a well-known value and neither would counter resets give the attacker another window of opportunity. The adaptive threshold approach provides such a solution.

From Figure 5.4, we can observe that the adaptive threshold is more successful in reducing control message overhead than a fixed threshold. An aggressive attack causes the adaptive threshold to reduce rapidly, thereby limiting the impact of the attack. This results in slower attacks being the best strategy. We can also see that 20 attacks/hr is the best strategy for an attacker because the values of α and β were chosen to model the default value of 20 in a steady state. However, if the values of these coefficients are changed, so will the periodicity of the optimal attack pattern. For the most aggressive attacks the differences are not so significant since the fixed threshold is quickly reached. The adaptive threshold is between 8% ($\gamma = 20$) to 13% ($\gamma = 25$) better, even in the worst case scenarios.

Figure 5.5 shows that the dynamic threshold is able to reduce overhead by 20% for aggressive attacks and 50% for slow attacks, when compared to the default fixed threshold approach. Comparing Figures 5.4 and 5.5, we can see that the advantage

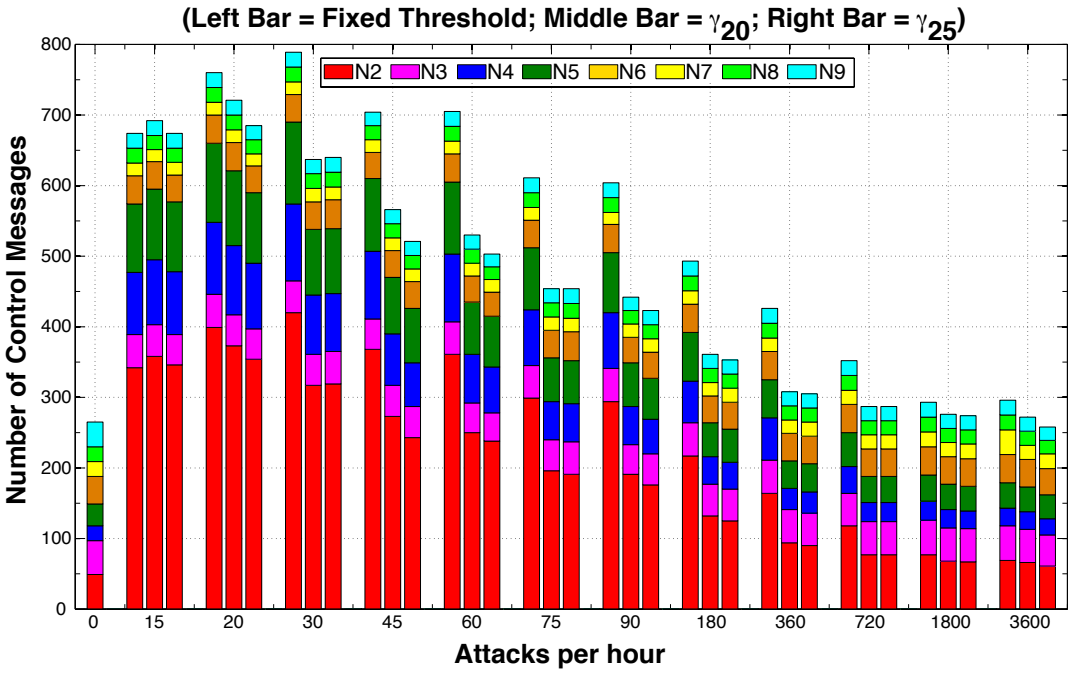


Figure 5.4: Total control message overhead per node when default mitigation strategy and adaptive threshold $\gamma = 20$ and $\gamma = 25$ are used.

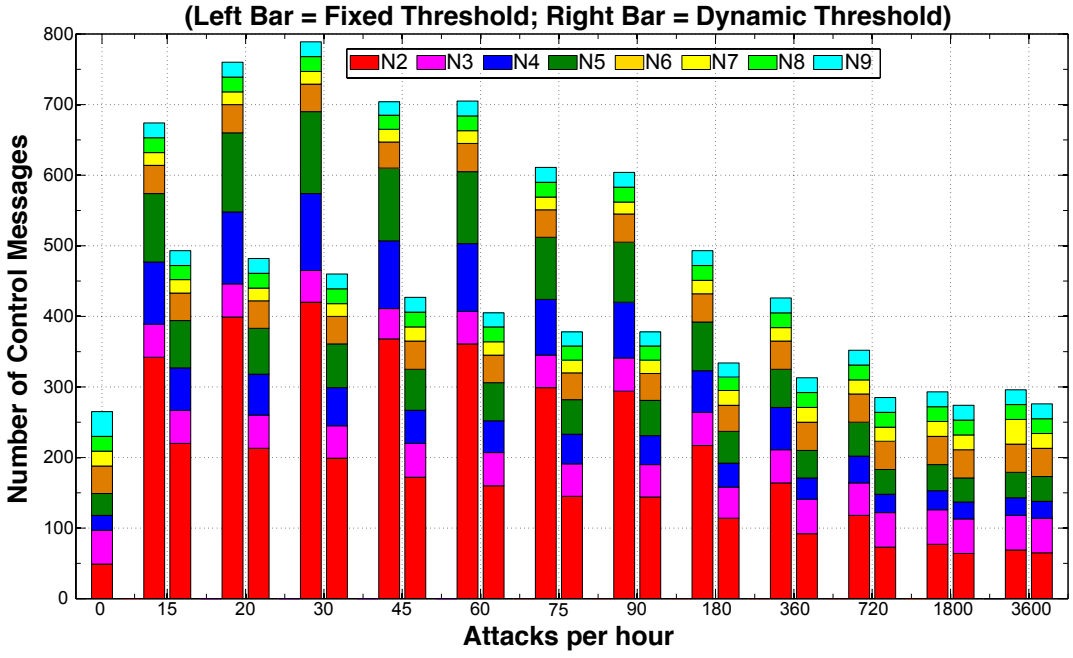
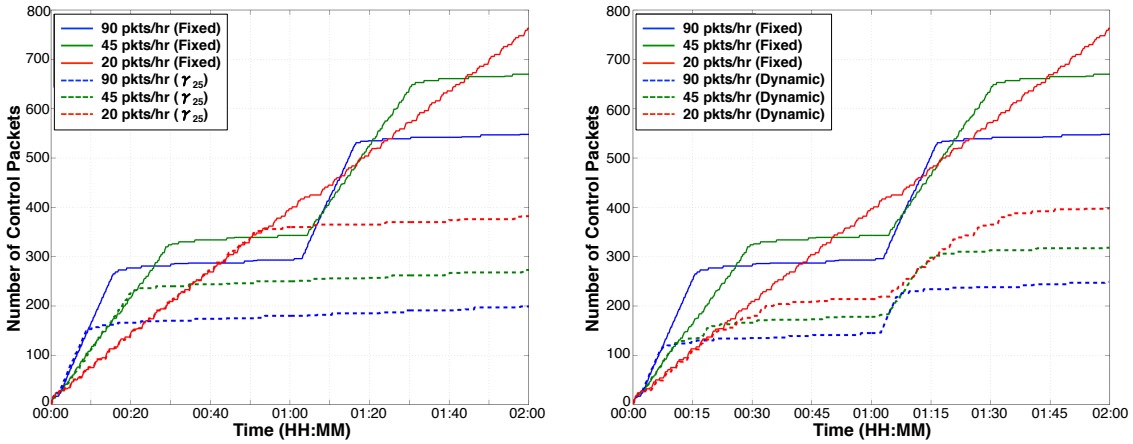


Figure 5.5: Total control message overhead per node when default mitigation strategy and dynamic threshold are used.



(a) Fixed threshold and adaptive threshold time-lines (b) Fixed threshold and dynamic threshold time-lines. ($\gamma = 25$).

Figure 5.6: Time-lines of outgoing packet overhead of node 2 in topology of Fig. 5.1(a).

of both approaches is almost the same for aggressive attacks (above 90 attacks per hour). However, the dynamic threshold has better results for slower attacks making a strategy to overcome the mitigation mode difficult.

Time-lines of packet overhead. Since the value of $count_R$ is not reset every hour for the adaptive and dynamic thresholds, the attacker does not have a future window of opportunity for causing increased damage. Both these thresholds increase in the absence of an attack, and as such the adaptive and dynamic approaches mitigate long running attacks even better. Results from a two hour long experiment can be seen in Figure 5.6. Only results from the directly attacked node 2 are depicted.

In Figure 5.6, we compare the fixed threshold to adaptive and dynamic thresholds. When using a fixed threshold the control messages increase quickly till the threshold is encountered. They then grow at a slow rate, following the trickle timer pattern until the 1 hour mark, when the counter is reset. Once again, the control messages increase quickly until the threshold is encountered. This behavior causes a high control message overhead. The only exception is the period of 20 attacks per hour, because at this rate the threshold is never encountered, thereby causing the largest overhead growth.

On the other hand in Figure 5.6(a), the limit is reached much faster with the adaptive threshold, due to the exponential growth of the function. Coupled with a non-reseting counter, this leads to between 45%-55% savings in the control message overhead. Those results depend on the value chosen for γ (discussed in Section 5.3.3.1). We notice a similar phenomenon in Figure 5.6(b) with the dynamic threshold approach. Instead of rising quickly in the second hour, as happens in case of the fixed threshold, overhead increases slowly with the dynamic threshold since the r variable

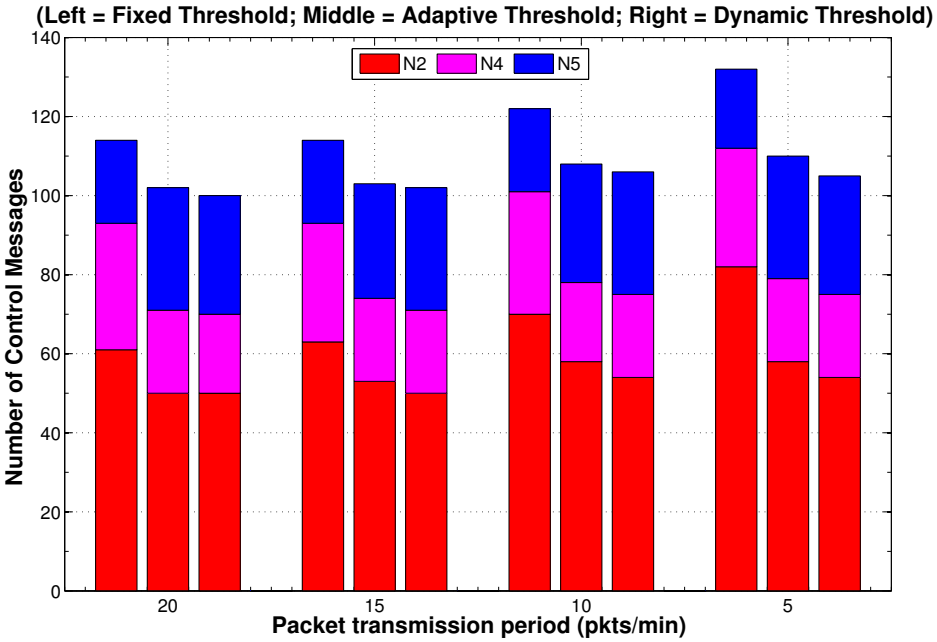


Figure 5.7: Total control message overhead per node with fixed threshold, adaptive threshold ($\gamma = 25$) and dynamic threshold (Fig.5.1(b)) .

from Equation 5.2 increases slowly. The saving for the different attack patterns is around 45%. In comparison with Figure 5.6(a), we can see that the adaptive threshold has slightly better results. This is due to the γ chosen here and also because $count_T$ in Algorithm 3 is reset every hour to allow legitimate 'R' flag packets from neighbors to be correctly handled. While using the dynamic threshold, the increase in overhead will continue after the second hour. This should argue in favor of using the adaptive threshold. However, the adaptive threshold requires setting the γ value, which needs to be learned empirically for every node in the network if optimal performance is desired. The dynamic threshold does not require any such empirically learned values to be configured. As such, because we gain more flexibility, the dynamic threshold algorithm is recommended over the adaptive threshold.

5.3.2.2 Packet Manipulation Mitigation

To evaluate the effect of our mitigation approaches on packet manipulation attacks (Algorithms 2 and 3), the topology showed in Figure 5.1(b) was setup in Cooja, with node 1, the DODAG root, acting as the sink. All other nodes, except the attacker have been configured to send messages to the sink at rates varying from 5 to 20 packets per minute. The packet sending rate is varied, because the attacker, i.e., node 3, silently modifies the option headers of the packets it forwards, rather than originating a direct attack.

Packet overhead. The nodes have been configured to use the adaptive threshold, then the dynamic threshold for mitigating packet manipulation. Results in Figure 5.7 show that the adaptive and dynamic thresholds reduce overhead in the network as demonstrated in Section 5.3.2.1. We can also note that the dynamic threshold performs slightly better than the adaptive threshold because, in this configuration, the dynamic threshold is reached faster than the adaptive one due to the number of neighbors of node 2. Compared to the default fixed threshold approach a reduction up to 30% can be achieved.

Delivery ratio. The black-hole created at the next-hop node in this scenario can severely impact the overall delivery ratio of packets, since none of the packets from the attacker's descendants will reach the sink. Without a black-hole mitigation approach such as the fixed threshold which does not counter this scenario, the overall delivery ratio is only about 33% as showed in Section 4.2.3. This is because only packets from node 2 reach the sink, while the attacker forces node 2 to drop all packets sent by nodes 4 and 5.

On the other hand with the adaptive threshold strategy the overall delivery ratio increases to just above 99%, because node 2 no longer drops packets from nodes 4 and 5 once the threshold is reached. The dynamic threshold approach also has a similar performance, with the delivery ratio being above 99%. These results speak strongly in favor of mitigating packet manipulation based DAG inconsistency attacks via an adaptive or dynamic threshold approach.

However, since the adaptive and dynamic thresholds depend upon different parameters, it is also important to check the effect they can have upon the performance of these approaches.

5.3.3 Configuration Parameters Impact

In the adaptive and dynamic threshold, the computation of the threshold $\lambda(r)$ are based on several parameters. In this section we discuss the effects of γ for the adaptive threshold and ϵ and δ for the dynamic threshold on the mitigation efficiency.

5.3.3.1 Adaptive Threshold

Given the same attack periodicity, the value of γ in Equation 5.1 determines the rate at which the threshold changes. Experiments were run with $20 \leq \gamma \leq 35$ to gain insights into its impact. Values larger than 35 have not been used because larger values of γ result in the threshold dropping too quickly. This leads to situations where even a single packet with the 'R' flag causes the trickle timer resets to stop. This means that genuine malfunctions will no longer be repaired either. In our tests we observed that for values over 35, this situation was encountered frequently. Below 20, the threshold reduces too slowly, thereby making it too permissive and increasing the likelihood of a successful attack.

As can be seen in Figure 5.8, by increasing the value of γ, even in the case of the most efficient attacker, the overhead can be further reduced by around 10%.

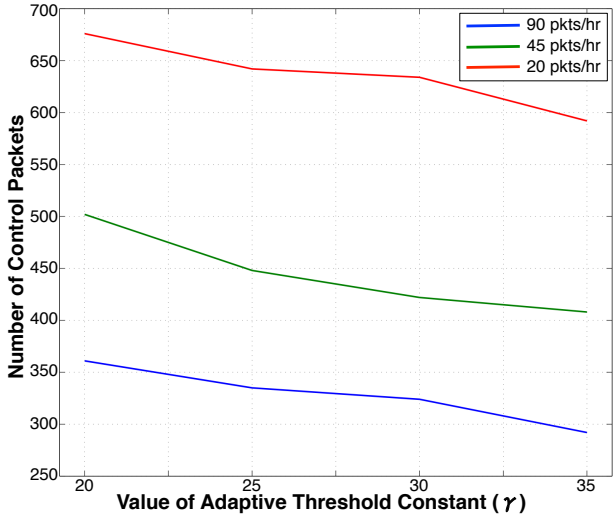


Figure 5.8: Effect of γ parameter on total control packet overhead experienced by node 2 in topology of Fig. 5.1(a).

This means that higher values of γ are able to offer more significant savings in the overhead. However, a rapidly reducing threshold might also impact the repair of genuine loop conditions. Our recommendation is to keep the value of γ between 20 and 35 so that the algorithm is neither too permissive nor too aggressive. The exact value has to be chosen according to the topology configuration by running tests.

5.3.3.2 Dynamic Threshold

The performance of the dynamic threshold approach is closely tied to the size of an attacked node's neighborhood since two parameters ϵ and δ depend on it. We have therefore to study the effect of varying neighborhood sizes on the dynamic threshold.

Packet Overhead. The same attack and data packet transmission scenarios from Section 5.3.1 have been used with the topology from Figure 5.9 to evaluate the impact of changing neighborhood sizes. The number of neighbors for node 2, targeted by attacker node 3, was set to 4, 8, 16 and 32 neighbors. A larger neighborhood size was not evaluated since Contiki can only track about 30 neighbors [23]; furthermore, due to the limited resources on the TelosB mote, maintaining a list of large number of neighbors can lead to a node being out of resources.

The overhead experienced by node 2 under different neighborhood sizes and attack patterns can be seen in Figure 5.10. The dynamic threshold outperforms the default fixed threshold approach, in all neighborhood sizes. In fact, the savings are between 20-50% and are mostly impacted by the variation of r because r depends on the number of genuine data packets (cf. Equation 5.2) which increases with the number of neighbors. The major advantage of the dynamic threshold is that after reaching a neighborhood size of at least 16 nodes, the control overhead does not

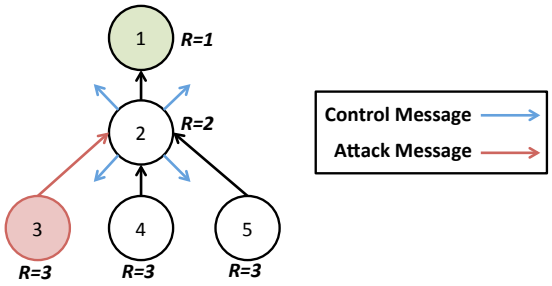


Figure 5.9: DAG inconsistency attack scenario used to study the effect of neighborhood size on the dynamic threshold.

increase more significantly in case of a larger neighborhood.

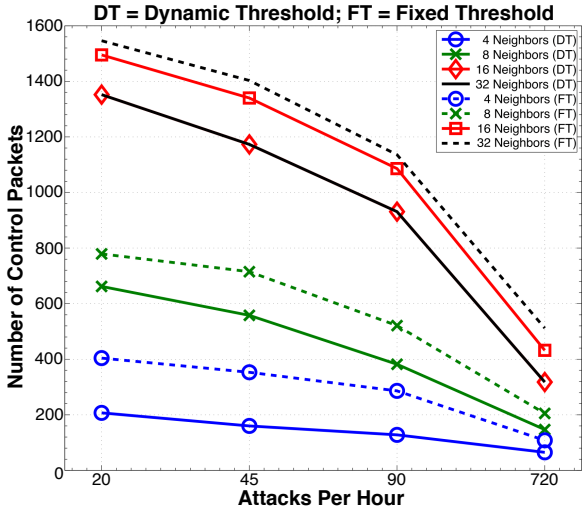


Figure 5.10: Outgoing packet overhead experienced by node 2 in the topology of Fig. 5.9 with varying neighborhood sizes.

In Figure 5.10, the curves for neighborhood sizes of 16 and 32 nodes, while using the dynamic threshold, are the same as their threshold values are very close. Since larger neighborhood sizes cause the threshold to reduce quickly, in case of 16 and 32 nodes, the threshold reaches its minimum value at the same time. As such, the dynamic threshold leads to lesser overhead in large neighborhood sizes.

The effect of varying number of neighbors has been also studied in the packet manipulation scenario. The same simulation scenario as in Section 5.3.2.2 has been used, the number of neighbors for node 2 has been set to 2, 4, 8 and 16 neighbors. Larger neighborhoods have not been studied since, as previously mentioned, their impact are not significant. Figure 5.11 shows the overhead experienced by the targeted node 2 for different packet transmission patterns. The overhead increases according to the number of neighbors since a larger neighborhood size allows more resets to occur as specified in the Equation 5.2.

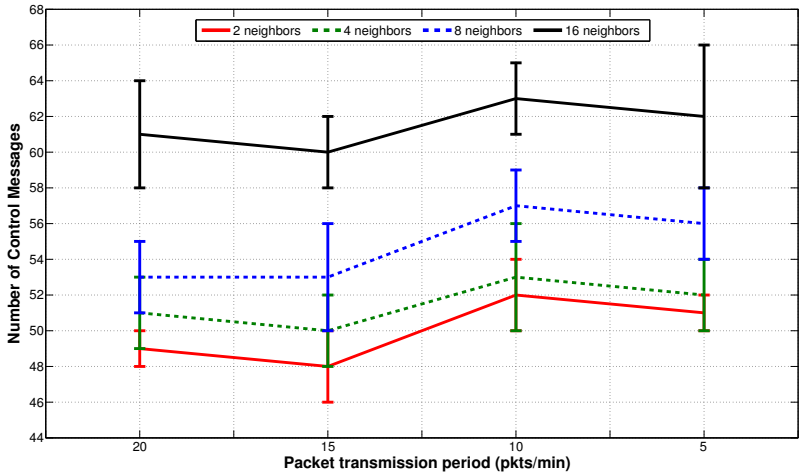


Figure 5.11: Outgoing packet overhead experienced by node 2 in the topology of Fig. 5.1(b) with varying sending frequencies and neighborhood sizes.

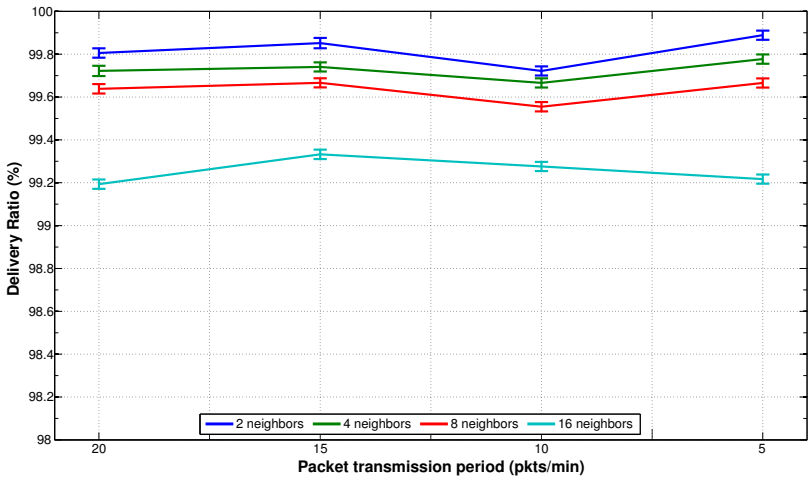


Figure 5.12: Global delivery ratio for different neighborhood sizes of node 2 (Fig.5.1(b)) with dynamic threshold.

Delivery Ratio. Figure 5.12 shows the delivery ratio for different neighborhood sizes of node 2 from the packet manipulation scenario described in Figure 5.3.2.2. The experiment has been repeated five times in order to obtain a standard deviation. In case of two neighbors, which corresponds to the simple scenario used in Section 5.3.2.2, we can see that the delivery ratio is above 99%. The delivery ratio decreases when the number of neighbors is increasing in accordance with the Algorithm 3. However even if the size of the neighborhood is 16 the delivery ratio stay above 99%.

5.3.4 Resource Consumption

To evaluate the efficiency of a countermeasure designed for constrained environments it is necessary to assess the cost of our solution.

5.3.4.1 Memory and Computational Costs

Since Equations 5.1 and 5.2 replace a constant threshold whose complexity is $\mathcal{O}(1)$, we have to also quantify the impact using an exponential function has upon the overall computation costs. The implemented function relies on an approximation of the actual exponential function using a simple loop making the complexity in $\mathcal{O}(n)$. Measuring this impact is even more important since these approaches are expected to be used on resource constrained devices with limited computing abilities. While

Table 5.1: Average computation time (ms) to calculate adaptive and dynamic thresholds for different attack patterns.

Type of threshold	***20 attacks/hr***	***45 attacks/hr***	***90 attacks/hr***
Adaptive threshold (γ=20)	28 *ms*	31 *ms*	31 *ms*
Adaptive threshold (γ=25)	28 *ms*	30 *ms*	31 *ms*
Dynamic threshold	26 *ms*	25 *ms*	24 *ms*

running the aforementioned experiments, the time taken to calculate the threshold has been also obtained. Table 5.1 shows the average computation time required to obtain the thresholds for multiple attack patterns (20, 45 and 90 attacks/hour) while using a MSP430F1611 microcontroller operating at 1 MHz on the TelosB platform.

Calculation of the dynamic threshold appears to add about 25 ms of computational overhead, and 30 ms for the adaptive threshold. This is because the value of the exponential part of the equation in the dynamic approach is lower than in the adaptive approach.

Using the `msp430-size` tool, we have determined the memory occupancy of nodes implementing the different thresholds. Table 5.2 gathers the obtained results. A node using the fixed threshold occupies 41.96 kB (87.4%) of flash memory and 8.63 kB (86.3%) of statically allocated RAM. The adaptive threshold approach requires 45.61 kB (95%) of flash memory and 8.62 kB (86.2%) of statically allocated RAM. The dynamic threshold approach requires 45.73 kB (95%) of flash memory

and 8.64 kB (86.4%) of statically allocated RAM. It is important to note that the base Contiki system is also already a part of this. The almost 8% increase in flash usage, for both approaches, can be reduced by optimization. On the other hand, there is almost no change in the amount of statically allocated RAM required.

The almost 4 kB increase in flash memory occupancy is due to the usage of a floating point library for calculation of the thresholds. This negative impact can be reduced greatly by using certain optimization, for example, a lookup table with linear interpolation will save not only flash space but also CPU execution time. Results using such optimizations have not been presented here so that the worst case performance of the algorithms can be quantified.

From the measured worst case values, we observe that the overall impact of both adaptive and dynamic threshold approaches on computational overhead is quite minimal, especially when taking the gains into consideration. Even though the dynamic threshold approach uses a little extra memory, the gains in having auto-configured parameters and providing good performance make it a good choice.

5.3.4.2 Energy Costs

From the energy model showed in Table 5.3, we have determined that the amount of energy taken up by the adaptive threshold (γ=25) computation is approximately 22.68 μJ, which is the amount of energy required to keep the processor running for the computation time of 31.25 ms. On the other hand, computing the dynamic threshold uses about 18.14 μJ, since the time to compute the threshold is about 25 ms. This means that for attack frequencies of 20, 45 and 90 attacks per hour, the total energy spent over a period of one hour on computing the adaptive threshold is about 0.45 mJ, 1.02 mJ and 2.04 mJ respectively, for the adaptive threshold (γ=25). On the other hand, this is about 0.36 mJ, 0.81 mJ, 1.63 mJ respectively for the dynamic threshold. Figure 5.13 presents the energy consumed at the attacked node to calculate the adaptive and the dynamic thresholds. The consumed energy increases by a significant amount when the attacker becomes more aggressive. This is because aggressive attacks lead to more threshold calculations, as such, more energy is consumed.

However, looking only at the energy consumed in calculation of the overhead is not a good measure for energy consumption since such attacks also cause additional packet overhead, which leads to additional consumption by the radio. Since the radio tends to be the most energy hungry device on constrained nodes, we have to factor this into the energy consumption as well. The upper part of Figure 5.14 shows the energy consumption caused by the control message overhead and threshold computation for all the nodes in the network. The lower part of Figure 5.14 presents the change in energy consumption caused by the control message overhead and threshold computation for all the nodes in the network, while the adaptive (γ=25) and dynamic thresholds approaches are used in comparison to the fixed approach. We see that in case of our adaptive and dynamic thresholds the energy spent by the network to process the control message overhead and the computation of thresholds is less than the energy used for the fixed threshold strategy. However, when the attacker is the

Table 5.2: Memory occupancy of the different thresholds.

Type of threshold	***Flash memory (kB)***	***RAM (kB)***
Fixed threshold	41.96 (87.4%)	8.63 (86.3%)
Adaptive threshold	45.61 (95%)	8.64 (86.4%)
Dynamic threshold	45.73 (95%)	8.64 (86.4%)

Table 5.3: Energy model for the CC2420 radio and MSP430F1611 microcontroller operating at 1 MHz on the TelosB platform.

Operation	***Current***	***Voltage***	***Part***
Transmit (T_x)	18.8 mA	2.2 V	CC2420 [17]
Receive (R_x)	17.4 mA	2.2 V	CC2420 [17]
Processing	0.33 mA	2.2 V	MSP430F1611 [75]

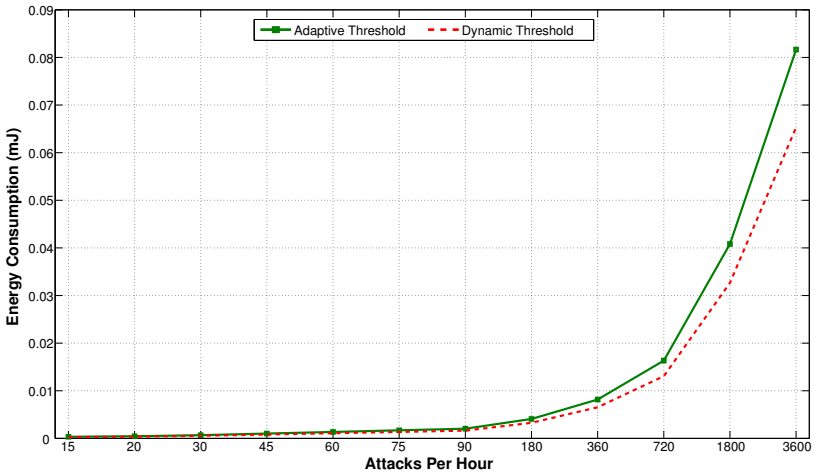


Figure 5.13: Energy required for adaptive (γ=25) and dynamic thresholds computation under different attack patterns.

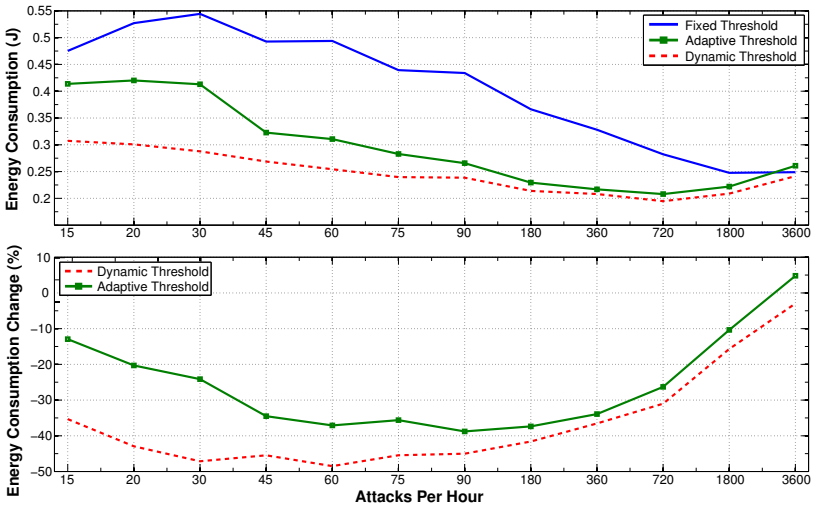


Figure 5.14: Energy consumption caused by control message overhead and thresholds computation under different attack patterns.

most aggressive (3200 attacks per hour) the curves become closer. This is because, in case of aggressive attacks the threshold is computed more often, leading to a higher energy cost. For all attack patterns, the dynamic algorithm has better results than the adaptive threshold as observed in the lower part of Figure 5.14. In fact, the dynamic threshold approach can provide nearly 50% energy savings in certain attack scenarios.

5.4 Conclusions

While the RPL protocol provides a fixed threshold based approach as an option to mitigate the DAG inconsistency attack, the value of the threshold is arbitrarily chosen and can be improved by taking into account network characteristics. Towards this goal, we have first designed an adaptive mechanism that mimics the fixed threshold when there is no attack. This approach has been improved by a fully dynamic solution. Both of these approaches have been evaluated in our analysis and they have both outperformed the fixed threshold. In particular, the overhead can be reduced between 20%-50% and in case of packet manipulation scenarios, which are not mitigated by the default RPL approach, our method can improve the delivery ratio to 99% as against 33% for the default RPL mitigation approach. We have discussed the configuration parameters impact. We have concluded that we should remain cautious for the γ parameter in the adaptive threshold in order to repair genuine loops and that the dynamic approach still outperforms the default mitigation even with large size of neighborhood. We have performed a cost analysis of our proposed strategy and we have showed that energy savings of up to 50% can be obtained. Due to the drawback of picking pre-deployment constants that need to be determined empirically for the adaptive approach, the dynamic approach is recommended since

it derives all parameters from the network neighborhood size. The performance of our two approaches is quite similar in case of aggressive attacks. However, in all other scenarios the dynamic threshold has outperformed the adaptive threshold. We therefore recommend the dynamic threshold for use.

We have proposed a node-level solution to counteract the DAG inconsistency attack. We can note that specific attack patterns such as the version number attack cannot be addressed by such a node-based approach because the attack is propagated through the entire network making it impossible to detect in a node-level perspective. In that context, we propose in the next chapter of this thesis to complement this approach through a distributed security-oriented monitoring architecture for RPL-based IoT environments able to address these cases.

Chapter 6

Security-oriented Distributed Monitoring Architecture

Contents

6.1 Introduction 79
6.2 Proposed Architecture 81
6.2.1 Overview and Components 81
6.2.2 RPL-based Mechanisms 82
6.3 Monitoring Node Placement Formalization 85
6.4 Detection Modules 86
6.4.1 DAG Inconsistency Attack 87
6.4.2 Version Number Attack 89
6.5 Conclusions 93

6.1 Introduction

We have previously proposed a node-level mitigation approach to address the DAG inconsistency attack. While we have demonstrated that the cost of this solution was reasonable, in some cases, attacks, such as the version number attack, have specific characteristics such that similar local-node methods cannot be efficient nor feasible in that context. We therefore propose to extend our solution to these cases with a passive distributed monitoring architecture designed for security. The originality of our approach comes in particular from the fact that the architecture exploits the RPL protocol to efficiently organize monitoring nodes. Node-level security strategy and distributed strategy are complementary according to network characteristics. Indeed, on one hand, the distributed strategy allows providing a global view and raising alarms to a network administrator, while, on the other hand, the node-level strategy permits to limit the impact of ongoing attacks. Thanks to this architecture, we can also handle cases where specific code, such as the node-level mitigation, cannot be deployed on nodes (no physical access, proprietary motes, no more capacity, etc.).

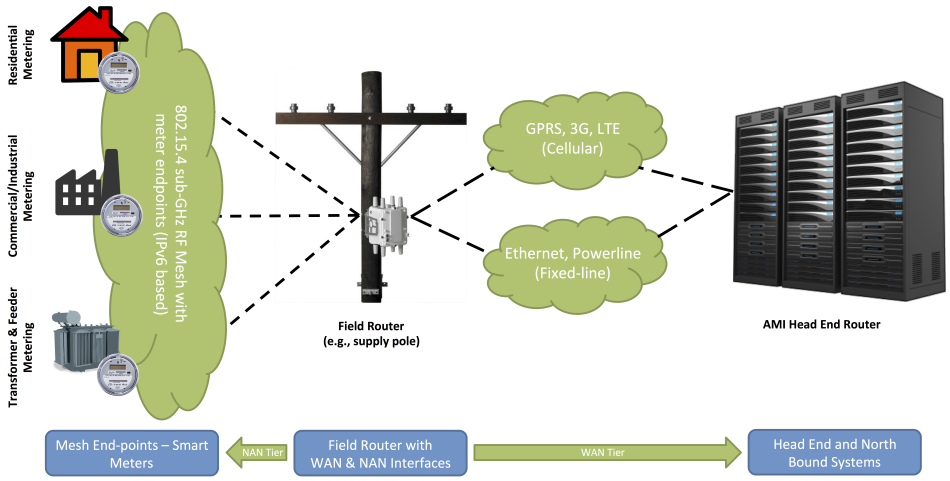


Figure 6.1: Typical AMI network [19].

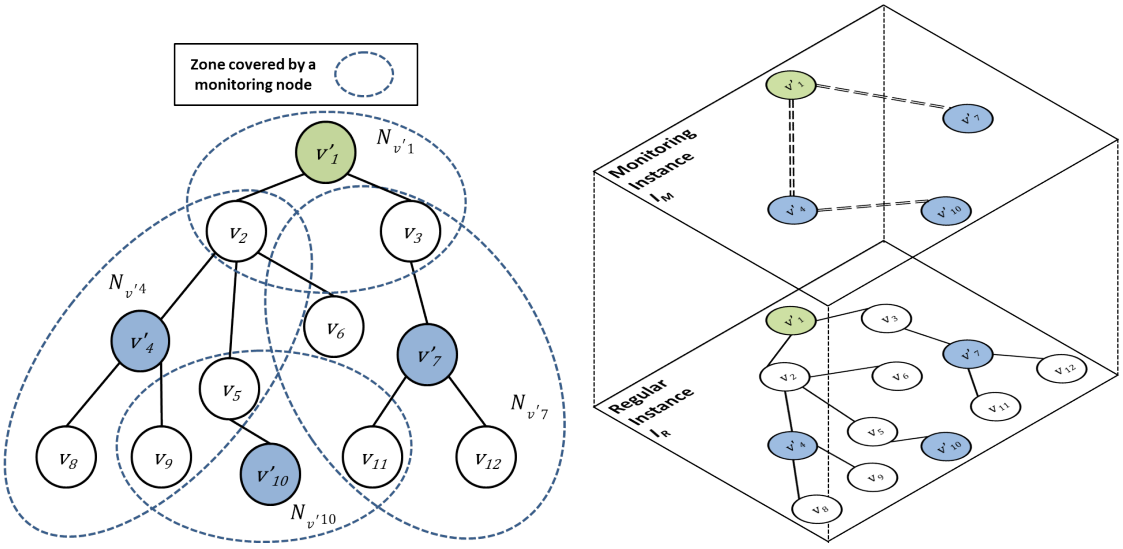
In order to preserve node resources, we exploit typical deployments of IoT infrastructures relying on higher-order devices. This is the case for advanced measurement infrastructures (AMI) which is expected to be organized as showed in Figure 6.1 [19]. This network can be divided into two tiers, i.e., the Neighborhood Area Network (NAN) and the Wide Area Network (WAN). The NAN consists of the smart meters that are deployed at (1) residential premises, (2) commercial and industrial buildings and (3) electricity transformer and feeder points in a specific neighborhood. These smart meters typically communicate by forming an IEEE 802.15.4 based mesh network that uses IPv6 for addressing individual devices. The RPL routing protocol is likely to be used to form the routing topology in the NAN tier. The WAN tier usually consists of the utility providers head end systems where metering data is typically collected. Unlike the NAN tier, systems in the WAN tier communicate using high-speed wireless or fixed-line access technologies. Field routers controlled by the utility providers, deployed on supply poles in a neighborhood, act as a bridge between the NAN and WAN tiers. These field routers have two interfaces, one that allows it to communicate with the low-power lossy network (typically IEEE 802.15.4) on the NAN side and another one that provides access to the high-speed wireless or fixed-line networks on the WAN side. It is also possible for these field routers to participate in a NAN-to-NAN mesh, such that the final interconnection of smart meters with head end systems occurs only via the low-power lossy communication channel. We therefore want to outsource monitoring and anomaly detection activities on these higher-order devices that are field routers. These ones can be interconnected to form an independent network from the LLN network in order to share their information which constitute our monitoring architecture.

This chapter presents our monitoring architecture concepts and details detection algorithms which can be integrated in our solution to address attacks. Section 6.2 introduces our solution, describes its main components and mechanisms based on the RPL protocol. We then propose in Section 6.3 to formalize the placement of

monitoring nodes through an optimization problem. Finally, Section 6.4 presents algorithms deployed on monitoring nodes in order to detect DAG inconsistency and version number attacks.

6.2 Proposed Architecture

We propose a security-oriented distributed monitoring architecture for the Internet of Things that passively observes the network. This one allows us to detect threats complementary to the local-node approach, for specific complex attacks which cannot be detected locally or even when dedicated code cannot be implemented on nodes. It is based on dedicated nodes and relies on the RPL protocol mechanisms to perform monitoring operations, so the target nodes do not have the charge of this activity. We describe both the main components of this architecture and the RPL-oriented features that are exploited to support it on an IoT network.



(a) Monitoring nodes snooping packets transmitted by nodes in radio range.

(b) Building of two RPL instances.

Figure 6.2: Example of our passive monitoring architecture exploiting the RPL multi-instance feature.

6.2.1 Overview and Components

Our monitoring architecture described in Figure 6.2 is composed of two types of nodes participating in the network, *regular* nodes also called target nodes which are monitored, plotted in white, and *monitoring* nodes plotted in blue. The sink plotted in green is also a monitoring node.

The regular nodes are typically lower order devices that fit into the C0 or C1 class of constrained devices. Their primary function is to carry out their assigned sensing

or actuation task. They form the so called regular network. They communicate with a sink/controller, where all collected sensing data is forwarded or from where actuation commands might be periodically received. This communication occurs over low-power lossy channels and a multi-hop mesh network might be formed in order to enable interconnection between all nodes.

The monitoring nodes are higher-order devices that should be at least C2 or better. As such, their monitoring activities might not have an effect upon their ability to serve their primary purpose of routing information in the regular network. These monitoring nodes are capable of passively listening to the regular nodes in their radio communication range, while also recording required information.

Since the higher-order devices, instrumented as monitoring nodes, are expected to be deployed in many IoT applications, those nodes participate in the regular network. As such, they are able to intercept and analyze packets sent by regular nodes. A monitoring node can only monitor its own low-power lossy network neighborhood as represented by circled areas in Figure 6.2(a). However, network-level monitoring information is useful to track the topology and inconsistencies in the network, e.g. topological, security, etc. As such, these monitoring nodes must periodically forward the collected monitoring data towards a sink. To avoid using the resource of constrained nodes, the monitoring nodes form a second routing topology as illustrated by the upper part of Figure 6.2(b). This second network, known as the *monitoring network*, has access strictly limited to monitoring nodes. Two possibilities can be considered to build the monitoring network depending on the use case. If the monitoring nodes have an high-speed high-bandwidth access network they can interlink with each other and with the sink, this can be the case in AMI (Advanced Measurement Infrastructure) deployments [19]. The second possibility is to share the same medium as the regular network, the interconnection is feasible using different radio ranges for monitoring nodes which is possible considering higher-order devices. The monitoring network will form an overlay network.

A monitoring node has two possibilities to collect monitoring data:

- the processing of traffic (data and control messages) legitimately sent to it by other nodes;
- the overhearing of packets exchanged between other nodes.

Indeed, complementary to the data collected by a monitoring node based on packets it has to process, i.e. data and control messages that are legitimately sent to it, it may enable the promiscuous mode in order to cover a larger amount of packets. The promiscuous mode allows a node to overhear packets, it is particularly useful for detecting anomalies and potential attacks by snooping data traffic not transmitted to it.

6.2.2 RPL-based Mechanisms

This passive monitoring solution is instantiated using the RPL protocol mechanisms. Figure 6.2(a) presents an example of a DODAG where node v_1' is the root. Multiple

instances of RPL, each being an execution of RPL with a specific objective function, can be run within a network. Each instance has its own DODAG graph [80] as illustrated in Figure 6.2(b) where the network is composed of two instances I_M (on the upper plane) and I_R (on the lower plane). While a node may be a member of multiple instances, it can only join a single DODAG in an instance such as nodes v'_4, v'_7 and v'_{10} in Figure 6.2(b) which are part of both instances.

Table 6.1: Summary of considered notations.

Notation	**Meaning**
I_R, I_M	regular instance, monitoring instance
V, v_i	set of all regular nodes, particular regular node i
V', v'_k	set of all monitoring nodes, monitoring node k
r_i	rank value of node i
N_{v_i}	neighborhood of node v_i

The following introduced notations are gathered in Table 6.1. Applying the RPL protocol to a network leads to the building of a DODAG in a instance I noted D^I. We note as $D^I(V, E)$ the DODAG graph composed of V nodes linked using E edges. Every node participating in the DODAG has an access to the root or sink S using the E edges which are chosen among all the links available to cope with the objective function. We note N_{v_i} the neighborhood of a node v_i which is the set of nodes $\{v_k\}$ in the communication range of v_i. The neighboring nodes of v_i can be parents whose rank is lesser than the rank value of v_i, children whose rank is greater or siblings with the same rank value. The rank value of a node v_i is noted r_i. We exploit the multi-instance feature of RPL to build two networks: a regular network and a monitoring network. An instance in RPL can be seen as a network optimized for specific metrics or constraints given by an objective function. The RPL multi-instance principle is an example of VRF (Virtual Routing and Forwarding): multiple instances of a routing table coexist on the same router at the same time. Those instances are completely independent which means if one network breaks down at some point because of regular node failure or attacks, the second network can operate normally. Therefore, two instances are running at the same time in our solution. One instance is used for the regular service noted I_R wherein the DODAG built is noted D^{I_R} composed of $V \cup V'$ nodes i.e. with both regular and monitoring nodes. The second instance called the monitoring instance, I_M where the DODAG is noted as D^{I_M} also is composed of V' nodes as showed in Figure 6.2(b). Using the RPL multi-instance feature presents two main advantages. First, it allows us to preserve regular nodes' resources because monitoring nodes forward their data on their monitoring instance/network. Second, if the regular network malfunctions, the monitoring data will still be forwarded thanks to the monitoring network, which is independent of the regular one.

The sink S is also a monitoring node. A monitoring node, v'_k is able to collect information regarding its neighborhood $N_{v'_k}$ as illustrated in Figure 6.2(a), the zone

covered by a monitoring node is its neighborhood. The collected information allows it to monitor the network and also detect possible anomalies by implementing locally detection algorithms. In Figure 6.2(a), monitoring node v'_{10} is able to monitor $N_{v'_{10}} = \{v_5, v_9, v_{11}\}$ using passive listening and overhearing. The monitoring node supports the detection of local anomalies based on dedicated detection modules. Collected information as well as detection results can then be aggregated and forwarded to its monitoring neighbor v'_4. Since neighborhoods of nodes v'_{10} and v'_4 overlap, node v'_4 checks if information gathered by node v'_{10} matches its own information in order to refine the detection in a collaborative manner. Node v'_4 performs the same process as its predecessor: collects information, runs detection algorithms, aggregates the different sources of data and reports it to the next monitoring node which is here the sink. Since the sink collects data from the different monitoring nodes, it may detect inconsistencies only observable at a global level.

A monitoring node is able to record the following RPL statistics from intercepted messages:

- **Information about the DODAG:**
  - **Instance ID** observed in messages originating from regular nodes.
  - **DODAG ID** observed in messages originating from regular nodes.
  - **DODAG Root**: destination address observed in all data packets from regular nodes.
  - **DODAG Version** observed from RPL control messages originating at each regular node.
  - **Node Objective Function** observed from RPL control messages originating at each regular node.
- **Information specific to a node:**
  - **Node DODAG Rank** observed from RPL control messages originating at each regular node.
  - **Minimum Rank Increase**: the option observed in control messages advertised by a non-root node.
  - **Maximum Rank Increase** the option observed in control messages advertised by a non-root node.
- **Information about the repair mechanisms:**
  - **Local Repairs Triggered** the number of local repairs triggered by a node.
  - **Global Repairs Triggered** the number of global repairs triggered by a node, i.e. higher DODAG version advertised by a non-root node.
- **Information about Control Messages:**

  - **DIO Message Count** the number of RPL DIO control messages observed from a node.
  - **DIS Message Count** the number of RPL DAO control messages observed from a node.
  - **DAO Message Count** the number of RPL DAO control messages observed from a node.
  - **Delay between DAO messages** observed by timing the frequency of DAO message reception from each regular node.

- **Information related to the Data Path Validation:**
  - **O-Bit Set** the number of packets observed from a regular node with the O-Bit set.
  - **F-Bit Set** the number of packets observed from a regular node with the F-Bit set.
  - **R-Bit Set** the number of packets observed from a regular node with the F-Bit set.

These statistics allow detecting potential misconfigurations as well as misbehaviors in the RPL functioning.

6.3 Monitoring Node Placement Formalization

We assume that all regular nodes are covered by at least one monitoring node because information about each node needs to be collected to monitor correctly the network. A configuration under this constraint can be calculated with the resolution of an optimization problem thanks to integer linear programming. The problem can be formulated as follows: for a given topology and a given connectivity matrix for all possible monitoring nodes placement in this topology, find a configuration of monitoring nodes placement that minimizes the number of monitoring nodes needed to cover all regular nodes.

Table 6.2: Required inputs for monitoring node placement.

Domain	Parameter	Description
$[\![1, N]\!]$	N	Number of nodes in the topology
$[\![1, N]\!] \times [\![1, N]\!]$	A	Connectivity matrix for monitoring nodes, $A_{i,j} = 1$ if node i covers node j

As input to solve this problem we need two parameters detailed in Table 6.2. The first parameter is the number of nodes (size of the topology) and the second one is the connectivity matrix detailing the links of possible monitoring nodes with other nodes, $A_{i,j} = 1$ if node v_i can listen to node v_j. We set the diagonal of this matrix to 0, i.e. $\forall i, A_{i,i} = 0$ which means that we consider that a possible monitoring node

Table 6.3: Considered variables for modeling.

Domain	Variable	Description
$[\![1, N]\!]$	Y	Binary variable indicating whether Y_i is a monitoring node (= 1) or not

does not cover itself. This facilitates the formalization of more complex problems by excluding monitoring nodes from specific constraints focusing on regular nodes only (as it can be seen in the next chapter). Only one variable is used here (cf. Table 6.3), Y, which represents whether node v_i is a monitoring node ($Y_i = 1$) or not ($Y_i = 0$).

The constraints are detailed in Equations 6.1 and 6.2:

$$Y_1 = 1 \tag{6.1}$$

$$\forall i \in [\![1, N]\!] : \sum_{j=1}^{N} (A_{i,j}.Y_j) + Y_i \geq 1 \tag{6.2}$$

The objective function f_{obj} is given by Equation 6.3:

$$f_{obj} = min \sum_{j=1}^{N} Y_j \tag{6.3}$$

Equation 6.1 indicates that node v_1' is a monitoring node, because in our case node v_1' is the sink and the sink is always a monitoring node in our architecture. It is possible, depending on the topology, to force particular nodes to be monitoring nodes for the configuration calculation. Equation 6.2 specifies that each regular node is covered by at least one monitoring node. Since the diagonal of the connectivity matrix is 0, we need to add $+Y_i$ in the equation so that the model is correct. The objective represented by Equation 6.3 is to minimize the number of monitoring nodes under these different constraints. Minimizing the number of monitoring nodes allows us to reduce the cost for their deployment. This solution permits to guarantee that each regular node is covered by at least one monitoring node, however it is also possible to extend the model with new constraints to meet other requirements.

Our architecture is able to monitor network traffic but it has been designed having in mind anomaly detection. We therefore propose detection modules deployed on monitoring nodes to identify threats targeting RPL networks.

6.4 Detection Modules

We have considered an IoT infrastructure where monitoring nodes can enable the promiscuous mode and implement detection modules to identify unusual behaviors and potential attacks. In this section, we present two algorithms: the first one allows monitoring nodes to detect the DAG inconsistency attack by overhearing data traffic, and the second one to detect and localize the launcher of a version number attack.

6.4.1 DAG Inconsistency Attack

The DAG inconsistency attack exploits the data path validation feature of RPL which is used to avoid and detect possible loops within the network. We have previously proposed a mitigation strategy deployed on each regular node in the network to counter such threats. When specific code cannot be deployed on regular nodes and/or when a global view is needed, we want to exploit our designed architecture to perform the security strategy using distributed detection algorithms. In order to detect such an attack, the promiscuous mode has to be enabled on monitoring nodes since it targets data traffic. In our architecture each monitoring node implements Algorithm 4 to identify this type of anomaly.

Algorithm 4 Detection algorithm implemented on monitoring nodes $\{v'_k\}$, $k \in \{1, 4, 7, 10\}$ to detect DAG inconsistency attacks

```
for each data packets received from N_{v'_k} do
    if R_flag is set then
        identify sender v_i  count_R_i ++
        if count_R_i == THRESHOLD then
            alone = 1
            for each v_j ≠ v_i in N_{v'_k} do
                if count_R_j > 0 then
                    alone = 0
                end if
            end for
            if alone == 1 then
                anomaly is detected
            end if
            end if
        end if
    end for
```

In Algorithm 4, R_flag represents the rank error flag in data packets. A monitoring node v'_k tracks for each neighbor $v_i \in N_{v'_k}$ the number of rank error flags $count\_R_i$ they have set. If this counter reaches the threshold value then the monitoring node has to check if v_i is the only node which has sent such packets. In this case, the 'R' flag packets are not originated from a legitimate loop and the anomaly is detected. The different $count\_R_i$ values are reset every hour in order to allow nodes to send legitimate 'R' flag packets.The threshold value has to be chosen carefully because it limits the number of 'R' flag packets a node can send, legitimately or not. We have adopted a fixed threshold in this approach such as the default mitigation presented in Chapter 5 for three reasons. First, it allows us to test easily the ability of our architecture and our algorithm to detect the DAG inconsistency attack. Second, the varying thresholds (adaptive and dynamic thresholds) as presented in Chapter 5 cannot be used in this context because they were designed for a local-node detection, as such, the chosen parameters depending on node-level information are

inappropriate in a distributed detection strategy. Finally, the idea of this algorithm is to detect the DAG inconsistency attack in a global view and not to mitigate it. The main point of the varying thresholds (AT and DT) is to limit the consequences of such an attack. The value of the threshold used in this algorithm is discussed in Section 7.3.1. In our architecture when an attack is identified, the malicious node is reported to an operator who can isolate it.

In order to determine whether the attack is a direct scenario or not, the monitoring node has just to compare the IP address of the direct sender of 'R' flag messages and the IP address of the source.

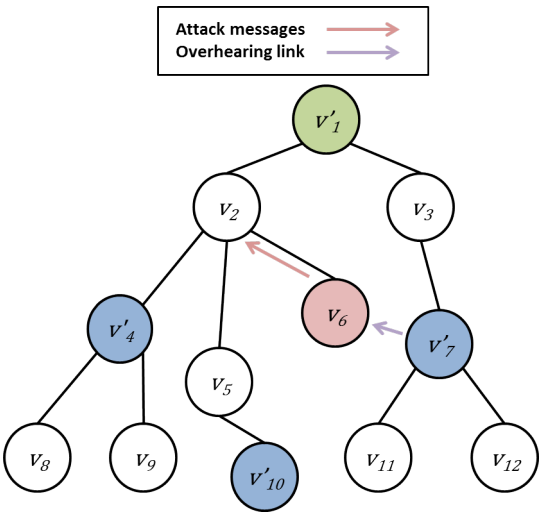


Figure 6.3: DAG inconsistency attack illustration where node 6 is the malicious node.

Figure 6.3 shows an example of a direct attack scenario. In this scenario, the node v_6 is the attacker and sends attack messages with the 'R' flag enabled to its preferred parent v_2. The monitoring node v'_7 is able to overhear these messages since the attacker is in its neighborhood. As soon as the number of 'R' flag messages reaches the threshold value and if the attacker is the only node to have sent such messages, then it is considered as anomalous.

This algorithm has been designed in case only one malicious node performs the attack. However, if multiple attackers are present at the same time, our algorithm is still able to detect them as long as they are in different monitoring node neighborhoods. If several attackers are in the same neighborhood of a monitoring node v'_k and launch their attacks simultaneously, node v'_k cannot determine there is an attack because there are several senders of 'R' packets, which is considered as legitimate loops. New conditions should be introduced to manage attackers coalition.

We can observe in this algorithm that monitoring nodes do not have to exchange data in order to identify the attack. However, when the attack is spread all across the network such as in the version number attack, monitoring nodes have to collaborate in order to detect the malicious node.

6.4.2 Version Number Attack

While the DAG inconsistency attack exploits information contained in data packets, the version number on the other hand misuses the version number included in DIO control messages. Due to the fact that an incremented version number is propagated within the entire graph, a monitoring node cannot decide by itself whether there is an attack or not. The monitoring nodes have therefore to share monitoring information to identify the malicious node. Our monitoring architecture is designed to allow monitoring nodes to collaborate together thanks to the monitoring instance. Also, the monitoring nodes can track information regarding their neighborhood, so the regular nodes do not have to carry out this task. To detect this attack and locate the malicious node we propose Algorithms 5, 6 and 7. The local assessment algorithm presented in Algorithm 5 is deployed on monitoring nodes except the root and allows monitoring nodes to report to the root the sender of an incremented version number in their neighborhood. Algorithms 6 and 7 are implemented on the root node. The first one detects the attack and gathers all monitoring node information into tables. The last algorithm performs the attacker identification by analyzing the collected information.

Algorithm 5 Local assessment algorithm.

```
potential_att = NULL;
for each DIO received by v'_k from v_i ∈ N_{v'_k} do
    if (VN_{v_i} > VN_{v'_k}) and (potential_att == NULL) then
        potential_att = v_i
        send_root(M_k = (VN_{v_i},v_i,N_{v'_k}))
    end if
end for
```

In Algorithm 5, a monitoring node v'_k, upon receiving a greater version number VN_{v_i} from v_i than its own version number $VN_{v'_k}$, sends to the root a message containing the address of the sender v_i and the list of its neighbors $N_{v'_k}$ obtained from the different received DIO messages. The monitoring node only sends a message the first time it receives an incremented version number. Indeed, since the attacker is in the direct neighborhood of at least one monitoring node there is no need in sending further messages because senders of other incremented version number messages are relays. We also consider the other neighbors of the monitoring node as safe. Complementary to the algorithms, the root could have the possibility to send a signal message indicating that the monitoring nodes reset the *potential_att* value, in order to restart the detection process, in case another attacker appears in the network.

Algorithm 6 is supported by the root and is used to detect the attack and gather all related monitoring data. Upon receiving either a monitoring message or an incremented version number, the root starts a detection timer to allow all monitoring nodes to send their messages. Two lists are managed by the root node: the *potential_att_list* list which is composed of all v_i nodes reported by the different monitoring nodes and the *neigh_list* list which is composed of each monitoring

node neighbors $N_{v'_k}$. Once the lists are completed, the root starts the localization procedure described in Algorithm 7.

Algorithm 6 Distributed detection algorithm.

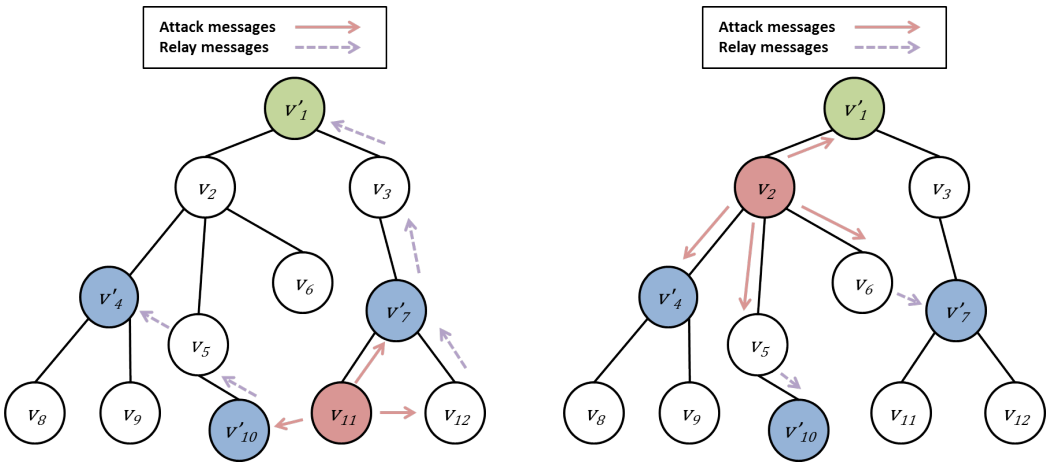
anomaly_detected = 0
if ($VN_{v_j} > VN_{v'_1}$ in DIO received from $v_j \in N_{v'_1}$) **then**
  anomaly_detected = 1
  add(potential_att_list, v_j)
  add(neigh_list, $\{N_{v'_1}\}$)
  start(detection_timer)
end if
if ($VN_{v_i} > VN_{v'_1}$ in M_k received) *and* (anomaly_detected == 0) **then**
  anomaly_detected = 1
  start(detection_timer)
end if
while (potential_att_list.nb != $Card(V')$) *or* (!timer_expired(detection_timer)) **do**
  for each message M_k received from $v'_k \in V'$ **do**
    add(potential_att_list, v_i)
    add(neigh_list, $\{N_{v'_k}\}$)
  end for
end while
Localization

This procedure exploits the two previous lists in order to produce two new lists: att_list list composed of nodes considered as malicious and the $safe\_list$ list containing all nodes classified as safe. The objective of this procedure is to compare neighborhoods of monitoring nodes in order to eliminate potential attackers. At initialization, the first element of the potential attacker list is added to the attacker list, and the other neighbors of the corresponding monitoring node are added to the safe list. While iterating, when the next potential attacker is already in the attacker list or in the safe list, it is ignored, and only the other neighbors are added to the safe list. This means that different monitoring nodes have detected the same node as a potential attacker, or that a monitoring node has detected a node as a potential attacker while being chosen as safe by another monitoring node. If the potential attacker is neither in *att_ list* nor in *safe_ list*, it is added to the attacker list. The final test consists in verifying if some elements of the neighbor list are in the attacker list. This can happen when monitoring messages are received in a disordered manner. In this case, these elements v_m have to be removed from the attacker list. We can notice that at the end of the algorithm it is possible to obtain several nodes considered as attackers, when senders of incremented version number are monitored by only one monitoring node.

Algorithm 7 Localization algorithm.

```
procedure LOCALIZATION
    att_list = NULL
    safe_list = NULL
    for (i=0, i<potential_att_list.nb, i++) do
        if (att_list == NULL) then
            add(att_list,potential_att_list[i])
            add(safe_list,{neigh_list[i] \ potential_att_list[i]}
        else
            if (potential_att_list[i] ∈ att_list) then
                add(safe_list,{neigh_list[i]\potential_att_list[i]}
            else if (potential_att_list[i] ∈ safe_list) then
                add(safe_list,{neigh_list[i]\potential_att_list[i]}
            else
                add(att_list,potential_att_list[i])
                add(safe_list,{neigh_list[i] \ potential_att_list[i]})
            end if
            if (neigh_list[i] ∩ att_list = v_m, v_m ≠ ∅) then
                remove(att_list,v_m)
            end if
        end if
    end for
end procedure
```

In order to illustrate these algorithms, we provide two examples to describe the different possibilities using the topology presented in Figure 6.2(a). The first scenario shows our detection strategy functioning under normal conditions. The second scenario is used to present a use case where the detection strategy produces false positive results (normal node considered as malicious). In Figure 6.4(a), the attacker is located at position 11, it sends DIO malicious messages to all its neighborhood (plain red arrows) which are relayed by other nodes (in purple dotted arrows). Monitoring nodes v'_7 and v'_{10} receive attack messages from attacker v_{11} and send to the sink a message containing the sender of the anomalous message and their neighbors as illustrated by Table 6.4(a). Nodes v'_1 and v'_4 do the same with relays v_3 and v_5. Once all data are gathered, the sink can start the localization procedure to establish the list of attackers and the list of safe nodes. At initialization, the first entry of potential attacker list, v_{11}, is added to the attacker list and the corresponding neighbors without the potential attacker $\{v_3, v_6, v_{12}\}$ are added to the safe list as described by Table 6.5(a). Then, since the second entry of Table 6.4(a), v_{11}, is already in the attacker list, only the safe list is updated with the neighbors of monitoring node v'_{10}, $\{v_5, v_9\}$. The third entry of Table 6.4(a) is v_3 which is already in the safe list, so only the safe list is updated with the corresponding neighbor $\{v_2\}$. The same process is repeated for the last entry v_5 which is also already in the safe list. At the end of the algorithm, the only element of the attacker list is v_{11} which is correct and all the



(a) Attacker located at position 11.

(b) Attacker located at position 2.

Figure 6.4: Version number attack illustrations.

Table 6.4: Potential attacker list and neighbors list obtained by the root after messages aggregation.

(a) Attacker located at position 11.

Monitoring node	Potential attacker	Neighbors list
v'_7	v_{11}	$\{v_3\ v_6\ v_{11}\ v_{12}\}$
v'_{10}	v_{11}	$\{v_5\ v_9\ v_{11}\}$
v'_1	v_3	$\{v_2\ v_3\}$
v'_4	v_5	$\{v_2\ v_5\ v_8\ v_9\}$

(b) Attacker located at position 2.

Monitoring node	Potential attacker	Neighbors list
v'_1	v_2	$\{v_2\ v_3\}$
v'_4	v_2	$\{v_2\ v_5\ v_8\ v_9\}$
v'_7	v_6	$\{v_3\ v_6\ v_{11}\ v_{12}\}$
v'_{10}	v_5	$\{v_5\ v_9\ v_{11}\}$

other regular nodes are considered as safe.

The second scenario, illustrated by Figure 6.4(b), where the attacker is located at position 2 shows the case where the localization procedure produces two attackers. Table 6.4(a) details how the monitoring data are aggregated by the root v'_1 and Table 6.5(b) shows the localization process. Until step 1 we can see that only v_2 is considered as the attacker, however in step 2 node v_6 is also added. The latter is not in the safe list meaning that no other monitoring node could exculpate it. As such, this detection algorithm may generate false positive results, a false positive corresponding to a normal node being detected as malicious by our strategy. The next chapter will show how to minimize the number of false positives.

Table 6.5: States of attacker list and safe list during the localization procedure.

(a) Attacker located at position 11.

Step	Attacker list	Safe list
Initialization	$\{v_{11}\}$	$\{v_3\ v_6\ v_{12}\}$
Step 1	$\{v_{11}\}$	$\{v_3\ v_5\ v_6\ v_9\ v_{12}\}$
Step 2	$\{v_{11}\}$	$\{v_2\ v_3\ v_5\ v_6\ v_9\ v_{12}\}$
Step 3	$\{v_{11}\}$	$\{v_2\ v_3\ v_5\ v_6\ v_8\ v_9\ v_{12}\}$

(b) Attacker located at position 2.

Step	Attacker list	Safe list
Initialization	$\{v_2\}$	$\{v_3\}$
Step 1	$\{v_2\}$	$\{v_3\ v_5\ v_8\ v_9\}$
Step 2	$\{v_2\ v_6\}$	$\{v_3\ v_5\ v_8\ v_9\ v_{11}\ v_{12}\}$
Step 3	$\{v_2\ v_6\}$	$\{v_3\ v_5\ v_8\ v_9\ v_{11}\ v_{12}\}$

6.5 Conclusions

We have proposed in this chapter a security-oriented monitoring architecture for RPL-based Internet of Things. This distributed passive architecture exploits the RPL protocol to support monitoring in a lightweight manner for the regular nodes. We have described its main components and how they interact based on the RPL protocol. It uses higher-order monitoring nodes typically deployed in many infrastructures such as AMI networks. These nodes are able to passively listen to the network while participating in its operation. The instantiation of our architecture takes benefit from the RPL protocol mechanisms such as the multi-instance feature in order to establish two separated routing topologies: a first instance corresponding to the regular network, and a second instance supporting the security monitoring activities. The regular nodes do not require to be instrumented so that monitoring tasks are operated by higher-order nodes.

We have also proposed dedicated modules to address DAG inconsistency and version number attacks in RPL networks. The first detection module which targets the DAG inconsistency attack relies on the promiscuous mode of monitoring nodes since the attack takes place in the option header of data packets. The module used to detect the version number attack exploits the ability of monitoring nodes to share their collected information since any local approach cannot be adopted. It permits to identify the malicious node, the attacker localization process being performed by the root after gathering detection information from all monitoring nodes. We have illustrated this strategy through examples showing the possibility for false positive results to occur. As our approach is passive and does not rely on regular nodes, it allows minimizing the impact on the Internet of Things infrastructure. The next chapter details the experimental evaluation of our architecture and its detection modules, in particular performance of the overhearing mode and the efficiency of our detection algorithms.

Chapter 7

Architecture Evaluation

Contents

7.1 Introduction 95
7.2 Overhearing Evaluation 96
7.2.1 Simulation Setup 96
7.2.2 Performance Analysis 97
7.2.3 Cost Analysis 99
7.3 Detection Modules Evaluation 100
7.3.1 DAG Inconsistency Attack 101
7.3.2 Version Number Attack 104
7.4 Scalability Evaluation 109
7.5 Conclusions 113

7.1 Introduction

We have described a whole strategy to detect attacks targeting RPL networks which relies on a distributed architecture and dedicated algorithms.

We propose here, to evaluate our architecture and its detection modules. Figure 7.1 presents the different part of this evaluation. We analyze performance and costs of the overhearing mode. The impact of three parameters in the Cooja environment is examined: distance, traffic load and neighborhood size. We then investigate the DAG inconsistency attack detection performance with the instantiation of a dedicated scenario in Cooja. We then evaluate the detection strategy for the version number attack. We analyze the number of false positive results according to monitoring node placement configuration in the topology. Since this detection strategy is distributed on the different monitoring nodes, we finally investigate the scalability of our approach with the formalization of an optimization problem for dealing with monitoring node placement.

The following of the chapter is organized as follows. Section 7.2 presents our experimental results regarding the overhearing mode. Section 7.3 details results of

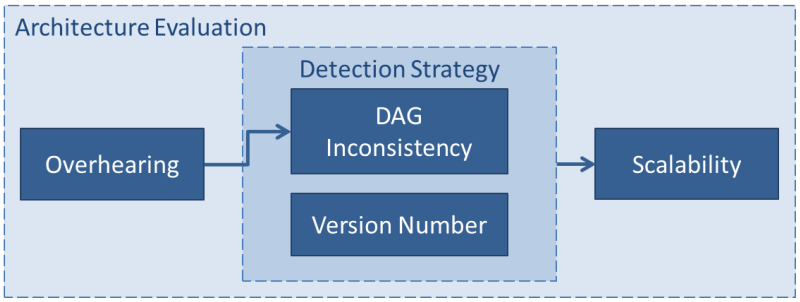


Figure 7.1: Monitoring architecture and detection strategy evaluation.

our detection modules. Finally, Section 7.4 describes our monitoring node placement strategy.

7.2 Overhearing Evaluation

Our monitoring architecture can exploit the ability of monitoring nodes to overhear their neighbors. A first study is dedicated to the evaluation of the performance and the cost of this overhearing activity.

7.2.1 Simulation Setup

We have set up a simple scenario composed of three nodes to quantify overhearing performance: a root node, a regular node configured to send data traffic periodically and directly to the sink, and a monitoring node with the promiscuous mode enabled which is equidistant from the previous nodes. This scenario allows us to study three parameters separately and measure their potential impact on the overhearing capacity of a monitoring node: (i) the distance from the monitoring node to the node it monitors, (ii) the sending frequency of a regular node and (iii) the number of neighbors of a monitoring node.

This scenario has been implemented in the Contiki 2.7 operating system [27]. The TelosB has been chosen as a target platform for regular nodes as in previous experiments. The Cooja simulator has been used during this analysis to execute the code written for the TelosB platform. Since Cooja does not have access to a device model for any higher-order devices, the TelosB platform has not only been used for regular nodes, but also for monitoring nodes. The monitoring node has been configured to enable the promiscuous mode. Each simulation has lasted for a lifetime of eight hours and has been repeated six times for accuracy reasons.

The metrics used to evaluate the different parameters are: (i) the *success ratio* which is the number of overheard data packets over the number of data packets sent by regular nodes in percentage and (ii) the *number of overheard packets*. Through the simulations it was experienced that the monitoring node can overhear two types of packets: data packets and point-to-point control messages (not destined to itself) which were ICMPv6 Neighbor Solicitation and Neighbor Advertisement messages.

7.2.2 Performance Analysis

In a first series of experiments, we have analyzed the impact of the distance. For this scenario, the position of the monitoring node has been varied from 15 to 50 meters from both the regular node and the sink. The regular node has been configured to send data every 20 s (180 msg/hr). Table 7.1 gathers results for different distances. Looking at the ratio which is around 36%, we can observe that a monitoring node cannot overhear all messages. This can be explained by the fact that the monitoring node has to process its legitimate traffic (control messages) in priority and when its queue is already full, the new arriving overheard packets are dropped. We could deduce that being able to overhear approximately 1/3 of data packets is low. However, the target platform used is a TelosB, a C1 class device, so it is expected that for a higher-order device the overhearing success ratio would be higher. Also, as explained in Section 6.2, overhearing is not the only source of monitoring data in our strategy, since a monitoring node can also gather data from packets that it legitimately forwards. In this scenario only monitoring data coming from the promiscuous mode is evaluated. As it will be presented in the next section, even if the success ratio seems low it is good enough to detect certain type of anomalies such Denial of Service (DoS) attacks. As we can observe the number of overheard packets do not change over distance such as the ratio, which implies that the distance does not affect the overhearing in the Cooja environment.

Table 7.1: Performance while varying the distance.

Distance (m)	15	20	30	40	50
Success ratio (%)	36	35.5	35.5	36.5	36
Number of listened data packets/hr	64	65	65	65	65

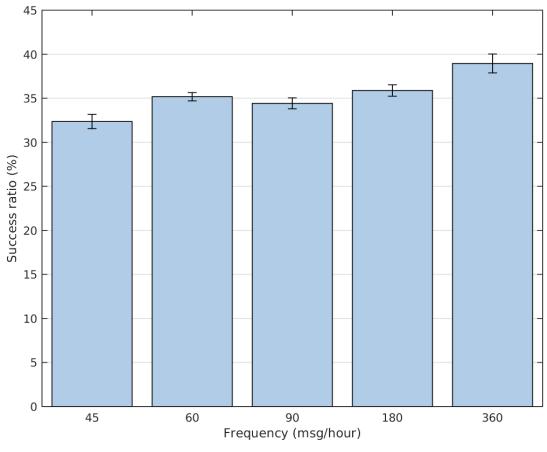


Figure 7.2: Average success ratio while varying sending frequency.

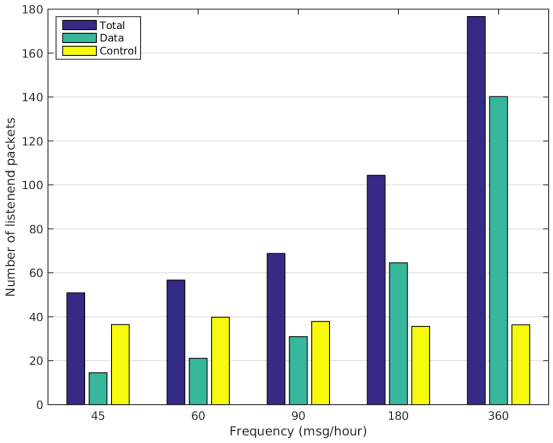


Figure 7.3: Average number of listened packets in one hour while varying sending frequency.

A second series of experiments have focused on the sending frequency. For this analysis the same topology as before has been setup, and the sending frequency of the regular node has been varied from 45 messages per hour to 360 messages per hour. Since the distance has no influence on the overhearing performance the monitoring node has been placed at 25 meters. Figure 7.2 presents the average success ratio of the monitoring node when the sending frequency varies from 45 msg/hr to 360 msg/hr. Figure 7.3 shows the number of listened packets using the promiscuous mode by the monitoring node. From Figure 7.2, we can see that the ratio lays between 32% and 39% and is slightly increasing with the frequency. We can therefore conclude that the success ratio is relatively stable (low variation). Since the ratio is stable and the number of sent data packets is increasing, it means that the number of overheard data packets also increases. This is confirmed by Figure 7.3 where the number of listened data packets increases with the frequency. We can also note that the number of listened control messages is stable (less than 40 messages), which makes sense because the number of nodes does not change over the simulations and the number of overall exchanged control messages is almost the same for each simulation. As a conclusion it can be said that for this environment the overhearing mode has slightly better results with a heavier load, although this improvement is limited (+7% when the traffic is multiplied by 8). As such, the overhearing success ratio can be considered as stable with the frequency.

Finally, the number of neighbors has been analyzed in a third series of experiments. For this scenario the topology has been modified so the number of regular nodes has been varied from 2 to 10 including the sink. The new neighbors have been directly connected to the root in the range of the monitoring node. The sending frequency has been set to 180 messages per hour. Table 7.2 gathers the different results regarding frequency and number of messages. We can see that the ratio stands between 36.5% and 38.5% which is quite stable. For these simulations we can ex-

plain the relative stability of the ratio by the fact that not only the number of data packets but also the number of control messages exchanged increases significantly with the number of neighbors. Indeed, if we consider the number of control messages exchanged between two nodes as stable over the simulations and if we multiply the number of nodes, the number of listened control messages is multiplied as well. The increase of listened data packets is also proportional to the number of neighbors: for two neighbors we have only one regular node which is sending data packets; if we multiply the number of listened data packets by the number of regular nodes we can see that we are close to the results given by the simulation. For instance, $66 \times 3 = 198$ which is close to 208, the number of listened data for 4 neighbors (3 regular nodes + 1 sink). Larger neighborhood sizes have not be simulated since the results can be extrapolated from previous observations.

Table 7.2: Performance while varying the neighborhood size.

Number of neighbors	2	4	6	8	10
Success ratio (%)	36.5	38.5	37.5	38.5	38
Number of listened data packets/hr	66	208	337	482	612

The different results obtained on the performance of the promiscuous mode for TelosB platform in a Cooja environment are useful information. Indeed, even if the monitoring node can overhear slightly more than a third of transmitted data packets, thanks to the different results, an estimation of the actual number of sent data packets can be achieved. Also these results can be helpful when developing detection algorithms. As we know, from the different studied scenarios, distance, sending frequency and neighborhood size do not affect much the success ratio.

7.2.3 Cost Analysis

Overhearing packets implies a cost for the monitoring node. From the energy model provided by Table 7.3, we calculated the cost for receiving monitored packets. Figure 7.4 shows the energy consumption of a monitoring node with two neighbors while the sending frequency of regular node is varied from 45 to 360 packets per hour.

Table 7.3: Energy model for the CC2420 radio and MSP430F1611 microcontroller operating at 1 MHz on TelosB.

Operation	Current	Voltage	Part
Receive (R_x)	17.4 mA	2.2 V	CC2420

Figure 7.5 presents the energy consumption while the neighborhood size is varied from 2 to 10 and the sending frequency on regular nodes is set to 180 msg/hr. Since the energy is proportional to the number of packets, the different results are similar to the ones presented in the previous section. We can see in Figure 7.4 that the

cost in total varies between 65 mJ for 45 msg/hr to 240 mJ for 360 msg/hr. Until 90 msg/hr the monitoring node spends more energy to overhear control messages than data packets. We can observe from Figure 7.5 that both overhearing data cost and overhearing control messages cost are increasing with the size of the neighborhood as explained earlier. The cost in total varies from 140 mJ (for two neighbors) to 1250 mJ (for ten neighbors) which is up 5 times more than costs presented in Figure 7.4. We see that the cost increases linearly with the size of the neighborhood. In more realistic conditions it is unlikely that regular nodes send so many data packets for their applications (360 msg/hr represents one message every 10 s) and have so many neighbors, we have included extreme cases in our analysis.

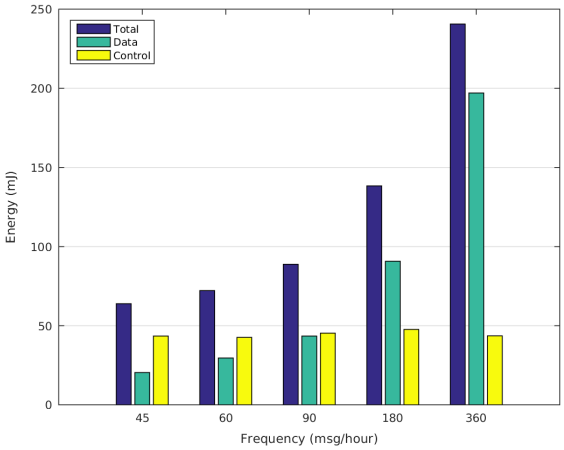


Figure 7.4: Energy consumed by the overhearing mode for a monitoring node while varying sending frequency (2 neighbors).

A trade-off has to be made between cost and efficiency. For instance we can optimize the number of monitoring nodes to be deployed but it means that they have to cover more nodes and consequently it costs more in terms of energy. This is also why we proposed in this architecture that the monitoring tasks are supported by higher-order devices so the energy is not as restrictive as it can be on usual C0/C1 devices.

7.3 Detection Modules Evaluation

This section aims at evaluating the performance of the detection modules used in our strategy which are deployed on each monitoring node. We first present results regarding module detecting the DAG inconsistency attack. We then focus on version number attack detection module.

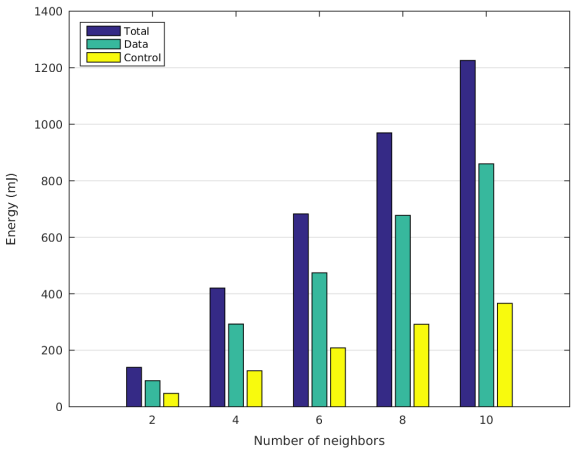


Figure 7.5: Energy consumed by the overhearing mode while varying neighborhood size for one hour.

7.3.1 DAG Inconsistency Attack

We present how the DAG inconsistency attack can also be detected using the monitoring architecture. After describing our simulation setup, we detail our detection results. In addition to detect such an attack for every node in the network, we also propose to mitigate it when monitoring nodes are targeted by implementing the dynamic threshold as described in Chapter 5 on them.

7.3.1.1 Simulation Setup

The topology showed in Figure 7.6 has been implemented in Cooja using the same configuration as Section 7.2. To emulate the behavior of monitoring nodes, Cooja has been setup to ensure that they all (i.e. nodes v'_1, v'_2, v'_8 and v'_{11}) were within radio range of each other. Across all experiments, node v'_1 is the DODAG root and also the sink, nodes v'_1, v'_2, v'_8 and v'_{11} are monitoring nodes. Regular nodes v_i have been configured to send data packets to the sink every 20 s. The attacker is designed to send attack messages (packets with down and rank error flags set) every 5 s in average to its preferred parent which means that the attacker is very aggressive. This corresponds to the extreme case showed in Chapter 5. This frequency has been chosen to study the ability of our architecture to deal with aggressive situation. The location of the attacker has been varied within the network replacing a regular node in order to analyze the detection performance of our monitoring architecture. Attacks start after two minutes of simulation time, so that the network has enough time to settle. Each simulation lasts for a lifetime of two hours and is repeated three times for accuracy reasons.

For this scenario, the threshold of Algorithm 4 in Section 6.4 has been set to twice the maximum number of neighbors of a monitoring node (cf. Table 7.4). This value gives the possibility for neighbors to send packets with an 'R' flag set in case of

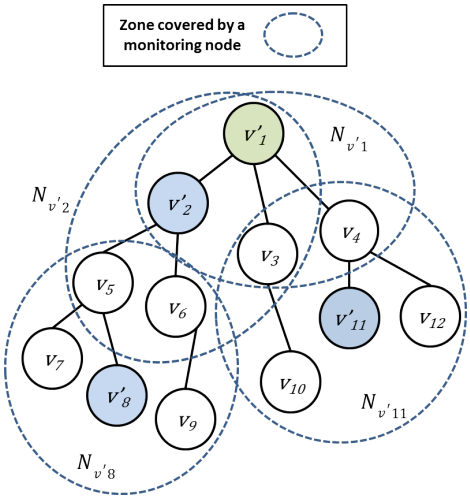


Figure 7.6: Scenario used to perform the DAG inconsistency attack.

legitimate loops without considering them as malicious. It can be envisioned that the threshold could be set dynamically according to each monitoring node configuration.

7.3.1.2 Attack Detection Results

The average detection time for the different positions of the attacker where the threshold value is set to 8 (twice the maximum number of monitoring node neighbors) is depicted on Figure 7.7. No bar means that the corresponding node could not detect the attack. Error bars indicate standard deviation between the 3 run simulations. Table 7.4 summarizes the regular nodes monitored by the different monitoring nodes as it is showed in Figure 7.6. The table 7.5 shows for each location of the attacker which node was the targeted node.

Table 7.4: Neighborhood of the different monitoring nodes

Monitoring nodes	v'_1	v'_2	v'_8	v'_{11}
Monitored neighborhood	$\{v_3, v_4\}$	$\{v_3, v_5, v_6\}$	$\{v_5, v_6, v_7, v_9\}$	$\{v_3, v_4, v_{10}, v_{12}\}$

According to Figure 7.7 and Table 7.4 we can observe that the attack has been successfully detected in every case; which means that all monitoring nodes have detected the attacker when it was in their neighborhood. In general the detection time stands between 2 min 20 s and 2 min 25 s, since the attack has started after 2 min it means that it has been detected in less than 30 s. However when the monitoring node has been directly targeted (cf. Table 7.5), the detection time is shorter or equal, for example when the attacker is located at node v_3 and node v'_1 is targeted, the detection time is 2 min 21 s while it is 2 min 25 s for node v'_2 and 2 min 23 s for node v'_{11}. When a monitoring node is targeted the overhearing mode is useless because the attack packets are directly addressed to this node. A lesser or

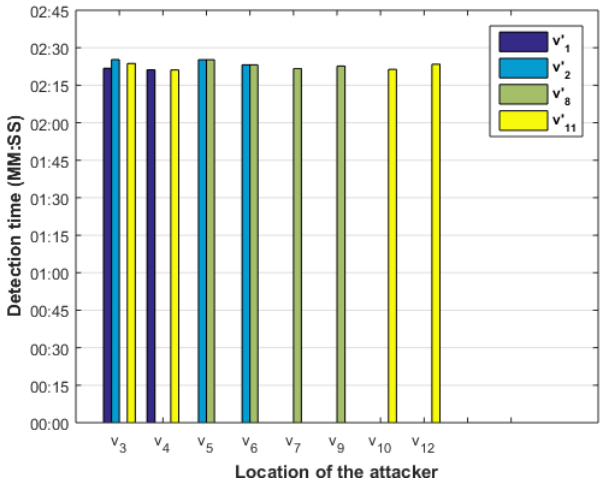


Figure 7.7: Detection time of the DAG inconsistency attack for different location of attacker.

a bigger threshold value results in detecting the anomaly quicker or slower. However, it is important to keep in mind that low value for threshold might also impact the repair of genuine loop conditions, this is why it is set to twice the maximum number of neighbors in our implementation. The aggressiveness of the attack also influences the detection time. Indeed, if the attacker has been less aggressive, the detection time would have been higher. While our previous work in Chapter 5 has been focused on mitigation solution deployed on each node, this study is focusing on the possibility to detect the same threat without implementing a detection algorithm on each regular node.

Table 7.5: Targeted node for the different location of the attacker

Attacker's position	v_3	v_4	v_5	v_6	v_7	v_9	v_{10}	v_{12}
Targeted node	v'_1	v'_1	v'_2	v'_2	v_5	v_6	v_3	v_4

7.3.1.3 Deployment of Mitigation Solution on Monitoring Nodes

Besides the detection module used to identify the DAG inconsistency attack for every node in the network, we also implement the dynamic mitigation solution presented in Section 5.2.3 on monitoring nodes in order to limit consequences of such an attack when monitoring nodes are targeted. The same simulation setup as in Section 7.3.1.1 has been used in this analysis. Each simulation has been repeated five times and lasted one hour.

Figure 7.8 presents the overall outgoing control message overhead experienced by the network per node. The left bar represents the overhead with the mitigation enabled and the right bar without mitigation. v_0 means a scenario without attacker

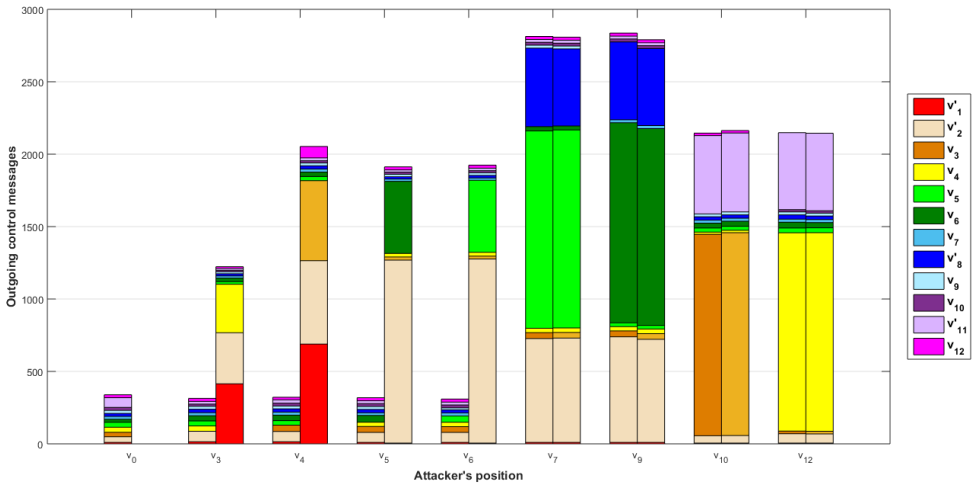


Figure 7.8: Per node outgoing packet overhead, for every location of the attacker when no mitigation and dynamic mitigation is deployed on monitoring nodes.

and is used as a baseline to compare other results with. It can be seen that the mitigation strategy is successful to limit the control message overhead when the attacker was at positions v_3, v_4, v_5 and v_6. This is explained by the fact that, in those cases, the targeted node was a monitoring node as detailed by Table 7.5. Since this mitigation is a node-level approach, it has no impact when the target is a regular node. This can be observed in Figure 7.8 when the attacker is located at v_7, v_9, v_{10} and v_{12}. Either the overhead is reduced by up to 600% when the target is a monitoring node or it is nearly unchanged when the target is a regular node. By exploiting results from Chapters 4 and 5, the benefit of the dynamic threshold approach for the same attack pattern (720 attack messages per hour or one attack message every 5 s) is 1100%. The difference can be explained by the fact that the topology used are different and the number of neighbors of the targeted node, which is a parameter of the dynamic mitigation, is not the same.

The DAG inconsistency attack scenario has showed that our monitoring architecture is efficient to detect such an attack in a RPL network. In this scenario, the detection module has been implemented on monitoring nodes for a neighborhood-level detection but the architecture can be exploited so the monitoring nodes exchange data to provide a collaborative detection. By collecting monitoring data to the sink, the architecture might be able to perform a global detection as we will see in the next section.

7.3.2 Version Number Attack

In Section 4.3 we have studied the version number attack. The pattern of this attack makes it hard to detect or mitigate locally since there is no possibility to identify precisely the malicious node. The Algorithms 5, 6 and 7 allow detecting and locating an attacker which performs the version number attack. They exploit the

collaboration among the monitoring nodes of our architecture.

7.3.2.1 Simulation Setup

We evaluate the performance of the Algorithms 5, 6 and 7 through experiments by implementing a Proof of Concept prototype. We have set up the same grid topology of 20 nodes such as in Section 4.3 as presented in Figure 7.9. The same implementation of the attacker has been used for these simulations. The Contiki 2.7 operating system has been used to implement the sink, regular nodes and monitoring nodes. Due to technical issues with the Contiki code, monitoring nodes have been configured to print required data for detection while Algorithms 6 and 7 have been implemented and run offline using Matlab scripts to produce the different results. The Cooja tool has been used to run the simulation with the compiled binaries of the different nodes. The radio model used was the DGRM model (Directed Graph Radio Medium) to emulate the links as showed in Figure 7.9: regular nodes can communicate with their neighbors horizontally and vertically, while the monitoring nodes can also listen diagonally since they are higher-order devices in our architecture. Across all experiments, node v_1' is the DODAG root and the sink to which all other nodes send messages every twenty seconds. The attacker is designed to constantly send incorrect version numbers, which are greater than the root's. Each simulation has lasted ten minutes which is enough to test our detection algorithms since only the first attack message is taken into account. The location of the attacker has been varied to one of each regular node.This entire set of simulations has been repeated three times for accuracy reasons. Attacks have been started after five minutes of simulation time, so that the network has enough time to settle and a stable RPL topology emerges.

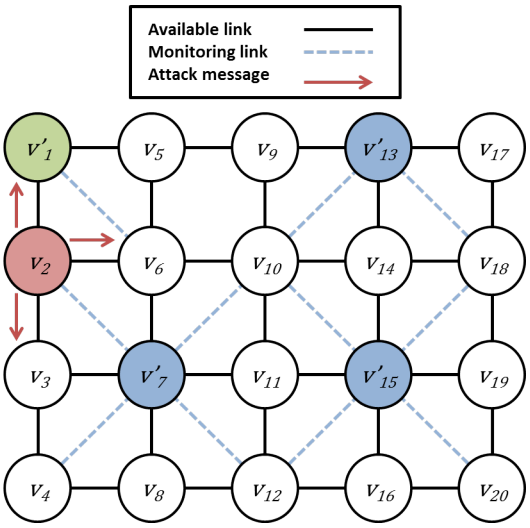


Figure 7.9: Grid topology of 20 nodes used to perform version number attacks.

Not only the location of the attacker has been varied but also the location and the number of monitoring nodes. Indeed, in the previous chapter, we have seen that

it was possible to encounter false positives results depending on the fact that a node is monitored by one or several monitoring nodes. The next section details how and why different monitoring node configurations were chosen.

7.3.2.2 Attack Detection Results

In this section, we first describe our monitoring node placement selection. We then detail the performance results of our detection module.

Monitoring Node Placement Selection. Since the version number attack detection module depends on the coverage of regular nodes by monitoring nodes, we defined the following metrics: (i) Cov_i representing the percentage of regular nodes covered by exactly i monitoring nodes ($i \in [1, M], M$ is the number of monitoring nodes); (ii) Ca_i representing the percentage of regular nodes covered by at least i monitoring nodes, e.g. $Ca_2 = Cov_2 + Cov_3 + Cov_4$ for $M = 4$. In all cases, we target Ca_1 equals to 100% because all regular nodes should be covered by at least one monitoring node since the architecture is able to monitor all regular nodes. Ca_2 is also an important parameter for selecting the configurations to be considered, because the number of false positives depends on the monitoring node neighborhood overlapping. Therefore, monitoring nodes configurations have been selected for different Ca_2 values in order to quantify the impact of this metric on the number of false positives.

Table 7.6: Number of possible configurations for 4 monitoring nodes ranked by increasing Ca_2.

Ca_2 (%)	Number of configurations
0	2
6.25	0
12.5	3
18.75%	5
25	2
31.25	6
37.5	3
43.75	3

Five different Ca_2 values have been chosen including the lowest and the highest possible values (corresponding to worst and best cases) for 4 and 5 monitoring nodes in the considered topology. The minimal number of required monitoring nodes is 4 so that Ca_1 equals to 100%. This value is given by the resolution of an Integer Linear Program (ILP) with our grid topology under the constraint that the sink, v'_1, is a monitoring node. The rest of the monitoring nodes are chosen among all the other nodes. For 4 monitoring nodes, the number of possible configurations so $Ca_1 = 100\%$ is 24 as described in Table 7.6. This result has been obtained from a program that

computes all correct configurations. A particular Ca_2 value corresponds to several combination of Cov_i. Table 7.7 shows the different configuration possibilities for selected values of Ca_2. One representative of each configuration possibility has been run. For instance, we can observe in Table 7.6 that for $Ca_2 = 12.5\%$ there are 3 possible monitoring node configurations, while, in Table 7.7 only one configuration with $Ca_2 = 12.5\%$ is available. This is because the three possible configurations for $Ca_2 = 12.5\%$ have the same Cov_i combination.

Table 7.7: Different configurations of Cov_i for 4 monitoring nodes ranked by increasing Ca_2.

$\boldsymbol{Ca_2}$ (%)	$\boldsymbol{Cov_2}$ (%)	$\boldsymbol{Cov_3}$ (%)	$\boldsymbol{Cov_4}$ (%)
0	0	0	0
12.5	12.5	0	0
25	25	0	0
25	18.75	6.25	0
37.5	37.5	0	0
43.75	37.5	6.25	0
43.75	31.25	12.5	0

We have also selected configurations with 5 monitoring nodes because the obtained Ca_2 values allow us to have zero false positive as illustrated by Figure 7.12. For 5 monitoring nodes, 427 configurations can be run. As for 4 monitoring nodes, we have selected 5 Ca_2 values including the lowest and the highest values (13.33%, 26.67%, 46.67%, 60% and 66.67%). Among the possible configurations for these Ca_2 values, we have simulated 26 configurations, each one being a representative of different Cov_i combinations as detailed in Table 7.8.

For each simulated scenario, the false positive rate (FPR) has been calculated according to Equation 7.1, where FP and TN are respectively the number of false positives and the number of true negatives. A false positive is a node which has been incorrectly detected as malicious by our detection solution (the node is actually safe). A true negative is a node which has been properly considered as safe.

$$FPR = \frac{FP}{FP + TN} \tag{7.1}$$

Detection Results. Across all experiments the detection successfully identifies the attacker but other regular nodes were sometimes detected as malicious too. Figure 7.10 details false positive results for the topology presented in Figure 7.9 where the monitoring nodes are v'_1, v'_7, v'_{13} and v'_{15} and the Ca_2 is 43,75% (maximum for 4 monitoring nodes). We can observe on Figure 7.10 that the FPR is 0 for 13 positions of the attacker. Details about the detection results when the FPR is more than 0 are given in Table 7.9. When the attacker corresponds to node v_5, node v_9 is always detected as malicious too, because node v_9 is each time the direct relay of the attacker v_5 and is monitored by only one monitoring node (v'_{13}). No other

Table 7.8: Different configurations of Cov_i for 5 monitoring nodes ranked by increasing Ca_2.

Ca_2 (%)	Cov_2 (%)	Cov_3 (%)	Cov_4 (%)	Cov_5 (%)
13.33	13.33	0	0	0
13.33	0	13.33	0	0
26.67	26.67	0	0	0
26.67	20	6.67	0	0
26.67	13.33	13.34	0	0
26.67	13.33	6.67	6.67	0
26.67	6.67	20	0	0
46.67	46.67	0	0	0
46.67	40	6.67	0	0
46.67	33.33	13.34	0	0
46.67	33.33	6.67	6.67	0
46.67	26.67	20	0	0
46.67	20	26.67	0	0
60	60	0	0	0
60	53.33	6.67	0	0
60	53.33	0	6.67	0
60	46.67	13.33	0	0
60	46.67	6.67	6.66	0
60	40	20	0	0
60	40	13.33	6.67	0
60	40	6.67	13.33	0
60	33.33	26.67	0	0
66.67	60	6.67	0	0
66.67	60	0	6.67	0
66.67	53.33	13.34	0	0
66.67	53.33	6.67	6.67	0

monitoring nodes could have exonerated it. This is also the case for other positions of the attacker. However, attack relays were not considered as malicious each time. This can be explained by the fact that the attack relays can change depending on the timing for each simulation. For example, when the attacker is v_{18}, v_5 is considered as malicious only once, this is because monitoring node v'_1 receives only once the attack relay message from v_5. The other times, the relay node v_6 is also monitored by v'_7 which exonerates it. Similar results have been obtained for the other 32 configurations from Tables 7.7 and 7.8.

Figures 7.11 and 7.12 show average false positive rate for the different values of Ca_2 with varying location of the attacker. Error bars are calculated as standard error of the mean (SEM) of the different possible configurations (Cov_i combination)

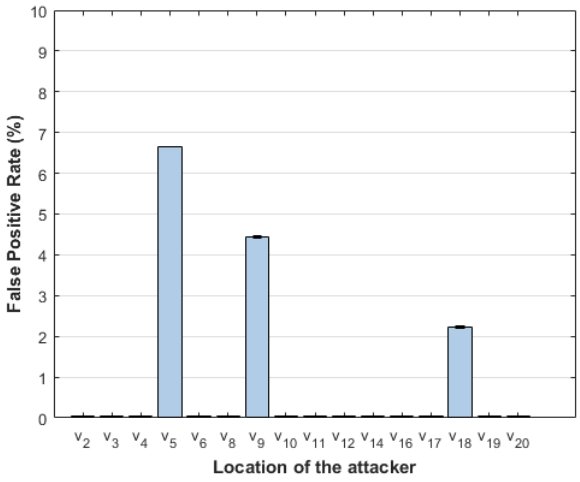


Figure 7.10: False positive rates for different location of attacker when configuration is the topology of Fig. 7.9.

Table 7.9: Detection result details when configuration is the topology of Fig. 7.9

Attacker's position	Series 1	Series 2	Series 3
v_5	v_5, v_9	v_5, v_9	v_5, v_9
v_9	v_5, v_9	v_5, v_9	v_9
v_{18}	v_5, v_{18}	v_{18}	v_{18}

for a particular Ca_2 value. We can see on both figures that the false positive rate decreased for increasing Ca_2 values, which means that the more nodes are covered by at least two nodes, the less is the number of false positives. When there are 4 monitoring nodes we can see in Figure 7.11 that the maximum value of the FPR is 20% which corresponds to the worst case (no node is covered by at least two monitors), and at best the FPR stands around 1%. We only obtain a false positive rate almost null when the Ca_2 is 66,67% (Figure 7.12) with 5 monitoring nodes.

According to these results we can conclude that monitoring nodes placement is strategic in order to obtain satisfying performance in detecting version number attacks. Nevertheless, while the DAG inconsistency attack detection only relies on neighborhood monitoring and such, only 100% coverage is enough (Cov_1 =100%), the version number attack detection requires monitoring node neighborhoods overlapping to perform well. This raises questions about monitoring nodes number and placement to support scaling.

7.4 Scalability Evaluation

We analyze the scalability of our solution in line with the considered monitoring node placement. We have showed previously that particular coverage can be required to

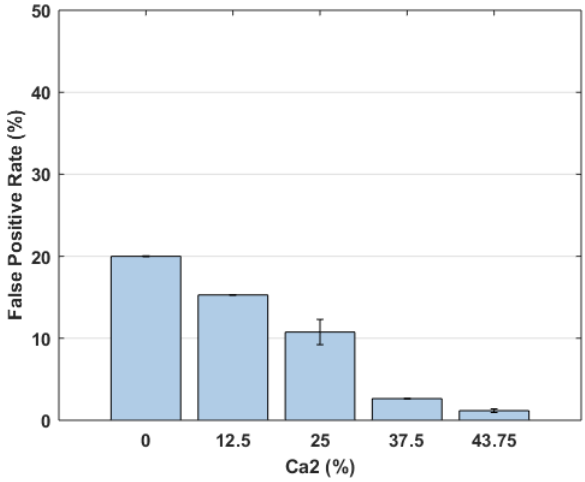


Figure 7.11: Average false positive rate for different Ca_2 values with 4 monitoring nodes.

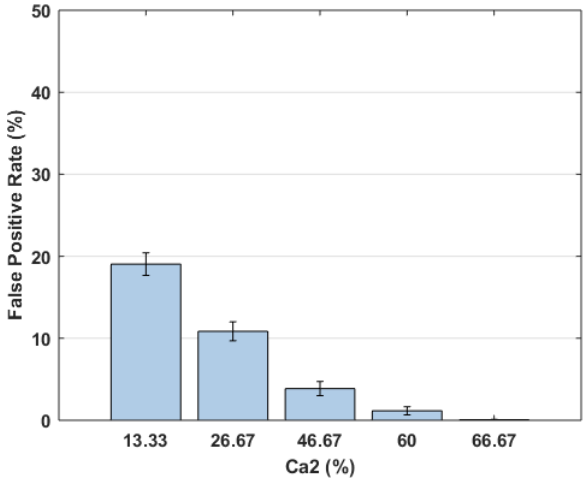


Figure 7.12: Average false positive rate for different Ca_2 values with 5 monitoring nodes.

efficiently detect threats in RPL networks according to the detection strategy such as the version number attack detection module. These coverage constraints can be modeled by optimization problems and, thus, resolved for a given topology.

In the same way that the problem of having all regular nodes covered by at least one monitoring node, i.e. $Ca_1 = 100\%$ (see Chapter 6); it is also possible to represent with an optimization model the problem of having at least $C\%$ of regular nodes covered by at least two monitoring nodes, i.e. $Ca_2 \geq C\%$ while $Ca_1 = 100\%$. This constraint is transformed into having at most (100 - C)% of regular nodes covered by exactly one monitoring node, i.e. $Cov_1 <$ (100-C)% while $Ca_1 = 100\%$. The problem

can be formulated as follows: for a given topology, a given connectivity matrix for all possible monitoring nodes placement in this topology, a given number of monitoring nodes and a given value C, find a configuration of monitoring nodes placement that minimizes the number of regular nodes covered by only one monitoring node so that at most (100 - C)% of regular nodes are covered by exactly one monitoring node under the constraint that all regular nodes are covered by at least one monitoring node.

Table 7.10: Required inputs for monitoring node placement.

Domain	Parameter	Description
$[\![1, N]\!]$	N	Number of nodes in the topology
$[\![1, N]\!] \times [\![1, N]\!]$	A	Connectivity matrix for monitoring nodes, $A_{i,j} = 1$ if node i covers node j
$[\![1, N]\!]$	M	Number of monitoring nodes
$[O, 1]$	C	Value indicating a percentage of nodes

Table 7.11: Considered variables for modeling.

Domain	Variable	Description
$[\![1, N]\!]$	Y	Binary variable indicating whether Y_i is a monitoring node (= 1) or not
$[\![1, N]\!] \times [\![1, N]\!]$	W	Binary variable indicating if node v_i is covered by a monitoring node v'_j
$[\![1, N]\!]$	Z	Binary variable indicating if v_i is monitored by exactly one monitoring node (= 1) or not

As input to solve this problem we need four parameters detailed in Table 7.10. The first parameter is the size of the topology. The second one is the connectivity matrix detailing the links of possible monitoring nodes with other nodes, $A_{i,j} = 1$ when node v_i can listen to node v_j. We set the diagonal of this matrix to 0, i.e. $\forall i, A_{i,i} = 0$ which means that we consider that monitoring node does not cover itself. The third parameter is the number of monitors. The last parameter is the percentage of regular nodes we want to be monitored by at least two monitoring nodes. The variables used are Y which represents whether node v_i is a monitoring node($Y_i = 1$) or not ($Y_i = 0$), W indicating whether node v_i is covered by monitoring node v_j ($W_{i,j} = 1$) or not ($W_{i,j} = 0$). The last variable Z specifies whether node v_i is covered by exactly one monitoring node ($Z_i = 1$) or not ($Z_i = 0$). The total number of variables for this problem is $N(N + 2)$.

The constraints are detailed in Equation 7.2 to 7.7:

$$Y_1 = 1 \tag{7.2}$$

$$\sum_{i=1}^{N} Y_i = N \tag{7.3}$$

$$\forall i \in [\![1, N]\!] : \sum_{j=1}^{N} (A_{i,j}.Y_j) + Y_i \geq 1 \tag{7.4}$$

$$\forall (i, j) \in [\![1, N]\!]^2 : W_{i,j} = A_{i,j}.Yj \tag{7.5}$$

$$\forall i \in [\![1, N]\!] : 2 - \sum_{j=1}^{N} (W_{i,j}) + 2.Y_i \leq Z_i \leq 1 - Y_i \tag{7.6}$$

$$\sum_{j=1}^{N} Z_i \leq (1 - C).(N - M) \tag{7.7}$$

As in the previous chapter, Equation 7.2 is used to set v'_1, the root, as a monitoring node, it is possible to set another particular node to be a monitoring node according to the topology specifics. Equation 7.3 indicates how many monitoring nodes we choose. The constraint of having all nodes covered by at least one monitor, is given by Equation 7.4. Equation 7.5 calculates W variable which is used in Equation 7.6 to compute Z. The right part of this equation forces $Z_i = 0$ if v_i is a monitoring node or else $Z_i = 1$. The left part is equal to 1 only if $\sum_{j=1}^{N}(W_{i,j})$ is equal to 1 and v_i is not a monitoring node, which means that the left part equals 1 when the node v_i is monitored by only one monitor. Equation 7.7 indicates the constraint that at most (1-C) % of regular nodes are covered by exactly one monitoring node.

The objective function f_{obj} is given by Equation 7.8:

$$f_{obj} = min \sum_{i=1}^{N} Z_i \tag{7.8}$$

The objective is to minimize the number of regular nodes covered by only one monitoring node i.e. to maximize the number of nodes covered by at least two monitoring nodes. This objective is necessary to compute the Z variable correctly. Indeed if v_i is not a monitoring node or a regular node only monitored by one monitoring node, Z_i can be equal to 0 or 1. Minimizing the sum on Z forces the default value to 0 in those cases.

We solved this problem with different sizes of grid topologies from 20 to 1000 nodes and with C=60% using the CPLEX solver [2] under the AMPL environment [1]. The C value has been chosen according to previous results from Section 7.3.2.2 because the false positive rate was very low. A script has been designed to establish the connectivity matrix of grids of corresponding sizes (4×5, 7×7, 10×10, 20×25, 25×40). The minimal number of monitoring nodes required to find a solution has been determined empirically by running the solver several times. For instance for a grid of 20 nodes (4×5), 5 monitoring nodes are required to obtain $Ca_2 = 100\%$, it is not possible with 4 monitoring nodes since the highest Ca_2 value is 43.75% as showed in the previous section. We have restrained the exploratory domain by solving the problem stated in Chapter 6 ($Ca_1 = 100\%$) for every size. The model has also been modified to find the minimal number of monitoring nodes so $Ca_2 = 100\%$ which means in this case that FPR should be 0. Figure 7.13 shows the minimal

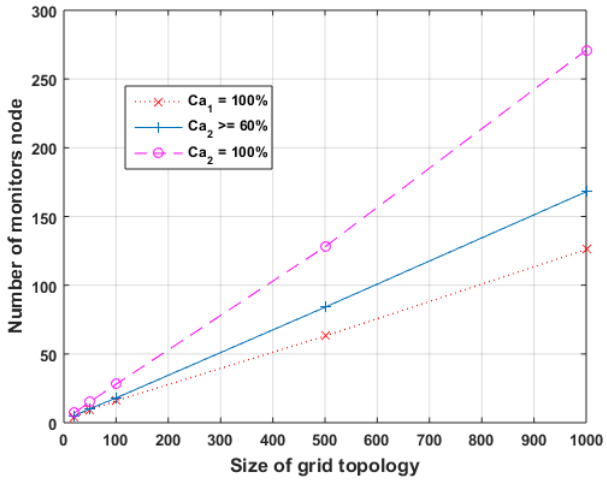


Figure 7.13: Number of monitoring nodes required to have $Ca_1 = 100\%$, $Ca_2 \geq 60\%$ and $Ca_2 = 100\%$ for different topology sizes

number of monitoring nodes required so Ca_2 is at least 60%. The value of C has been set to 60% because it ensures low false positive rate for the detection algorithm, as showed in the previous section. We can observe on Figure 7.13 that the number of monitoring nodes required to have the different Ca values is proportional to the number of nodes. These results show that our solution supports scalability.

7.5 Conclusions

In this chapter, we have evaluated our monitoring architecture through a set of experiments. In particular, we have quantified the performance and cost of the promiscuous mode in that context, considering different distances, sending frequencies and neighborhood sizes. Experimental results with the Cooja environment have showed the feasibility of the proposed monitoring approach with respect to traffic load and neighborhood size. The cost of the overhearing may seem heavy, this is why it is performed by higher-order devices able to carry out this load. Also it can be envisioned to dynamically adapt the overhearing mode based on network events.

We have then analyzed the detection performance of our dedicated algorithms for the DAG inconsistency attack. Thanks to the overhearing mode of monitoring nodes, we have showed that the detection time was less than 30 s in every configuration of the malicious node. Besides the DAG inconsistency attack detection module, we have deployed the dynamic threshold mitigation, seen in Chapter 5, on monitoring nodes to limit the impact of this attack when they were targeted. The obtained results were consistent with the previous study i.e. the control message overhead has been reduced for monitoring nodes when they were targeted by the attacker. Regarding the version number attack detection module, we have showed that the attack has been successfully detected for each case. Even if the attacker has been

identified, the detection module may produce false positive results by detecting safe nodes as malicious depending on monitoring node placement. However, we have also demonstrated the false positive rate of our solution can be reduced to 0 by a strategic monitoring node placement.

We have finally considered the scalability of our architecture by proposing an optimization problem which can be easily adapted to different topologies. By resolving this problem, we have quantified the number of required monitoring nodes to ensure an acceptable false positive rate in detecting the version number attack for a given size of topology.

This evaluation demonstrates that our security monitoring approach is successful in detecting attacks targeting the RPL-based Internet of Things while having a minimal impact on the target nodes.

Chapter 8

General Conclusions

The emergence of the Internet of Things results in the large scale deployment of Low power and Lossy Networks. In order to deal with the requirements of such networks, new communication protocols have been designed, and in particular the RPL protocol which provides a routing solution for these environments. While this paradigm enables new applications for everyday life and business, it also represents major security risks. Indeed, the lack of resources such networks suffer make them particularly vulnerable to security threats compromising their availability. In this thesis we have proposed to investigate a security monitoring strategy for addressing the trade-off between security and its induced cost in the RPL-based Internet of Things.

Our contributions are organized into three main axes. We have first assessed security threats targeting the RPL protocol through the identification and classification of attacks and proposed a dedicated taxonomy. We have analyzed the impact of two RPL specific attacks which are the DAG inconsistency and the version number attacks and showed the importance of addressing them. We have then presented a local strategy to detect and mitigate DAG inconsistency attacks in RPL networks and evaluated its performance and costs. We have designed a security-oriented monitoring architecture in order to complement our node-level approach and address more complex attacks. In a passive and distributed manner, this solution preserves constrained nodes resources by exploiting RPL mechanisms such as the multi-instance feature and by relying on higher-order devices which implement detection modules responsible for identifying the considered security attacks. We have evaluated this architecture through extensive series of experiments and discussed the placement of monitoring nodes in that context.

8.1 Achievements

After presenting the challenges for monitoring and security in the RPL-based Internet of Things, we have described the RPL protocol functioning and its mechanisms, and have provided a state of the art on existing monitoring approaches for IoT networks. Their comparison has pointed out their limits in the context of RPL-based Internet

of Things.

In order to quantify security threats in RPL networks, we have first proposed a taxonomy classifying the attacks against the RPL protocol in three categories: attacks that over-consume nodes resources, attacks targeting the topology and attacks against networks traffic. Using this classification, we have compared the properties of these attacks and discussed methods and techniques to avoid or prevent them. We have then assessed the impact of two major attacks targeting the RPL protocol: the DAG inconsistency and the version number attacks. The DAG inconsistency attack exploits an RPL loop avoidance mechanism to forge fake loops in the network forcing nodes to send unnecessary control messages. We have showed through our experiments that these attacks can increase considerably the control message overhead of targeted nodes and their descendants, reducing their lifetime. Meanwhile, the version number attack misuses the global repair mechanism of the RPL protocol. This attack leads to the propagation within the network of a malicious version number forcing a global rebuild of the DODAG. Quantifying the impact of such an attack has revealed that the control message overhead can increase significantly (by up to 18 times compared to a normal scenario). We have also observed the occurrence of many loops in the network, an important decrease of the overall delivery ratio and a strong correlation between location of the attacker and observed effects. Through this impact analysis, we have showed that such attacks can meaningfully consume node energy. It is therefore crucial to address these attacks through the design of appropriate security strategies.

To propose security solutions suitable for RPL environments, we have first designed a node-level approach in order to deal with DAG inconsistency attacks. Since the RPL protocol proposes a fixed threshold solution but gives no justification about the chosen value, we have designed an adaptive approach. This one mimics the fixed threshold in a steady state, however it takes into account network characteristics and it adjusts itself under varying attack patterns. Even if we have showed its performance compared to the fixed threshold, it relies on pre-configured parameters which can represent an inconvenience depending on network specifics. We have improved this solution to be fully dynamic by relying on auto-configured parameters. Our strategy has been evaluated through experiments which have showed its efficiency. In particular, it reduces the control message overhead and increases the delivery ratio. The impact of the considered parameters has also been discussed in an analysis to show their potential limitations. We have performed a cost evaluation of the proposed strategy in terms of computational overhead and energy consumption. This analysis has demonstrated the limited cost of our algorithms and their benefits regarding the energy savings (up to 50%). Such a node-level approach is efficient to detect attacks having only a local impact, however, it is not able to identify a malicious node in case the attack is spread across the whole network. Also in some cases, dedicated code cannot be implemented on nodes because they lack of memory or RAM or they cannot be flashed.

In that context, we have designed a security-oriented monitoring architecture which preserves nodes resources by passively listening to network traffic. This dis-

tributed architecture is based on higher order devices typically used in many IoT applications. It exploits RPL mechanisms and in particular the multi-instance feature to organize the monitoring nodes into an independent topology so the resources of regular nodes are preserved. This solution is also able to perform multi-level detection by implementing local algorithms but also gives the capability for the monitoring nodes to collaborate and to send information to the root in order to have a global view of the network. As such, our architecture meets the different requirements provided in Table 2.1 from Chapter 2. We have designed specific modules to address DAG inconsistency and version number attacks. The first module uses the overhearing capacity of monitoring nodes to track malicious data packets. The second module exploits interactions among monitoring nodes to identify malicious nodes. We have evaluated our monitoring architecture through extensive series of experiments. We have first quantified the performance and cost of the overhearing mode in Cooja environment, considering several parameters which allows us to show the feasibility of our solution. We have then analyzed the efficiency of our detection strategy. It was demonstrated that DAG inconsistency attacks can be detected in a limited time despite the limitations of the overhearing mode. We have also showed that the false positive rate for detecting the version number attack can be notably reduced by considering a strategic monitoring node placement. This evaluation permits to conclude on the efficiency of our strategy in detecting security threats. For addressing monitoring node placement, we have finally considered the scalability of our architecture by proposing and resolving optimization models.

8.2 Perspectives

The work achieved during this thesis opens several perspectives with respect to IoT security monitoring. We first discussed possible future work regarding complementary experiments in real testbeds and implementation considerations. We then examine the extension of our security monitoring strategy to new elaborate attacks. Finally, we consider risk management methods to enhance our solution.

Complementary Experiments in Real Testbeds

We are interested in performing complementary experiments in real testbeds. The Cooja simulator used in this thesis is a valuable environment for testing and debugging code in the Contiki OS with many features. It is also useful for research purpose since simulation results are close to real experiments as we have showed through a simulation validation. Authors of [69] have showed this tool could be used for other OS such as the RIOT OS [5]. However, they have identified issues about time inaccuracies which can biased fine results regarding synchronization and time related performance. As future work, it would as such make sense to deploy our proposed monitoring architecture in a real testbed. This permits to use several classes of devices in order to evaluate the costs of our proposed solution on monitoring nodes. Moreover performance of the overhearing mode could be finely evaluated

and detection algorithms could be adapted accordingly.

Also all of our experiments were run using the Contiki OS. At the beginning of this thesis, it was the most complete project that implemented the RPL protocol and was widely used in research area. However, this operating system presents some drawbacks, for instance, the lack of documentation and issues related to coding as pointed out in [67, 68] and that we have encountered regarding the multi-instance mechanism. Also, since then, new projects have been developed and are beginning to be really popular. This is especially the case for the RIOT OS [5] and the OpenWSN project [4]. These OS open new perspectives to deploy and test our security strategy for RPL networks.

Extension to New Attacks

We are also planning to enhance our solution by considering new attack scenarios. First, our detection strategy could be adapted to counter attacker coalition by refining the proposed algorithms since we have considered during our analysis that only one attacker was performing attacks at a time. An attacker coalition is characterized by several malicious nodes involved at the same time in the network. Also, the different implemented solutions could be optimized for better performance and for resource saving. For instance, the overhearing mode of monitoring nodes known to be resource consuming which we have used to detect the DAG inconsistency attack could be dynamically enabled by taking into account network events.

As future work, it can additionally be envisioned to extend our study to new attacks from our established taxonomy, not only those which target node resources but also topology and traffic related threats. We can note that more and more analysis regarding attacks targeting the RPL protocol are performed in the literature which can facilitate the extension of this work. Indeed, at the beginning of this thesis only few studies regarding security in RPL networks were available. Not only our work can be expanded by considering other attacks targeting the RPL protocol but it can also be enhanced by considering attacks targeting other protocols in the network stack. Several existing works can be exploited for this purpose such as [42] and [60] which focused on 6LoWPAN protocol security.

Risk Management Framework

Risk management offers new perspectives to dynamically activating or deactivating security mechanisms in RPL-based networks, in order to prevent and limit attacks while maintaining network performances. We therefore envisage to improve our framework by considering these methods. The risk management process is composed of two main activities: risk assessment and risk treatment. Risk assessment consists in quantifying the potentiality of attacks through detection methods. Risk assessment aims also at quantifying the consequences of successful attacks. The objective is to assess the relative importance of nodes in the RPL network, and to analyze how the attack against a given node may impact on the functioning of the overall network. The risk treatment activity consists then in selecting and applying the required se-

curity mechanisms while keeping in mind the cost of the solution. In this thesis, we can notice that we have performed some steps in this direction, especially regarding the risk assessment process since we have classified attacks in a taxonomy, quantified the impact of two attacks and designed a detection strategy relying on a monitoring architecture. We have also proposed countermeasures related to the risk treatment process. Risk management methods and algorithms could support the selection and activation of these different countermeasures within our monitoring architecture.

Publications

Publications

Journals

- Anthéa Mayzaud, Rémi Badonnel, Isabelle Chrisment. « A Taxonomy of Attacks in RPL-based Internet of Things ». In *International Journal of Network Security*, IJNS 2016, 18 (3), pp.459-473. 2016. Link.

- Anthéa Mayzaud, Anuj Sehgal, Rémi Badonnel, Isabelle Chrisment, Jürgen Schönwälder. « Mitigation of Topological Inconsistency Attacks in RPL based Low Power Lossy Networks ». In *International Journal of Network Management*, IJNM 2015, Wiley-Blackwell, 2015. Link.

International Conferences

- Anthéa Mayzaud, Rémi Badonnel, Isabelle Chrisment. « Detecting Version Number Attacks in RPL-based Networks using a Distributed Monitoring Architecture ». In *Proceeding of IEEE/IFIP Conference on Network and Service Management*, CNSM 2016, Montreal, Quebec, Canada. Oct-Nov 2016.

- Anthéa Mayzaud, Anuj Sehgal, Rémi Badonnel, Isabelle Chrisment, Jürgen Schönwälder. « Using the RPL Protocol for Supporting Passive Monitoring in the Internet of Things ». In *Proceeding of IEEE/IFIP Network Operations and Management Symposium*, NOMS 2016, Istanbul, Turkey. Apr 2016. Link.

- Anuj Sehgal, Anthéa Mayzaud, Rémi Badonnel, Isabelle Chrisment, Jürgen Schönwälder. « Addressing DODAG Inconsistency Attacks in RPL Networks ». In *Proceedings of the Global Information Infrastructure and Networking Symposium*, GIIS 2014, pp.1 - 8, Montreal, QC, Canada. Sep 2014. Link.

- Anthéa Mayzaud, Anuj Sehgal, Rémi Badonnel, Isabelle Chrisment, Jürgen Schönwälder. « A Study of RPL DODAG Version Attacks ». In *Proceedings of the 8th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security*, AIMS 2014, pp.92 - 104, Brno, Czech Republic. Jun 2014. (**Best Paper Award**) Link.

- Anthéa Mayzaud, Rémi Badonnel, Isabelle Chrisment (2013).« Monitoring and Security for the Internet of Things ». In *Proceedings of the 7th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security* (PhD Symposium), AIMS 2013, pp.37-40, Barcelona, Spain. Jun 2013. Link.

National Conferences

- Anthéa Mayzaud, Anuj Sehgal, Rémi Badonnel, Isabelle Chrisment. « Gestion de risques appliquée aux réseaux RPL ». In *9ème Conférence sur la Sécurité des Architectures Réseaux et des Systèmes d'Information*, SARSSI 2014, Saint-Germain-Au-Mont-d'Or, France. May 2014. Link.

List of Figures

1.1 Extension of the Internet to everyday objects. 2
1.2 Example of IoT applications (based on [7]). 3
1.3 Road map of the contributions. 7

2.1 Overview of the IoT protocol stack. 10
2.2 Example of a RPL network composed of two instances and three DODAGs. 11
2.3 Classification of monitoring architectures. 15
2.4 Architecture used in DAMON. 16
2.5 Poller-pollee architecture used in [39]. 17
2.6 Architecture used in SVELTE. 18
2.7 Architecture used in the specification-based IDS from [41]. 18
2.8 Architecture used in LiveNet. 19
2.9 Architecture used in EPMOSt. 21

3.1 Taxonomy of attacks against RPL networks. 26
3.2 Rank increased attack in a RPL network. 28
3.3 A wormhole attack in a RPL network. 33
3.4 Illustration of a DAO inconsistency attack. 35
3.5 Illustration of a decreased rank attack. 39

4.1 DAG inconsistency attack scenarios. 44
4.2 Total control message overhead experienced by network presented in Figure 4.1(a) under DAG inconsistency attacks. 47
4.3 Example of new DODAG iteration. 48
4.4 Grid topology used for performing experimental evaluation of the version number attack. 50
4.5 Incoming and outgoing control message overhead for every location of the attacker. 51
4.6 Per node outgoing packet overhead for every location of the attacker. 52
4.7 Total delivery ratio and end-to-end delay for every location of the attacker. 52
4.8 Total number of loops and inconsistencies in the network for every location of the attacker. 53

4.9 Total number of loops detected per node for every location of the attacker. 54
4.10 Total number of inconsistencies detected per node for every location of the attacker. 55

5.1 Topologies for mitigation approaches evaluation (same as Figure 4.1). 63
5.2 Comparison of per node outgoing packet overhead for node 2 in the topology from Fig. 5.1(a). 64
5.3 Total control message overhead per node when no mitigation strategy and default mitigation strategy are used. 65
5.4 Total control message overhead per node when default mitigation strategy and adaptive threshold $\gamma = 20$ and $\gamma = 25$ are used. 66
5.5 Total control message overhead per node when default mitigation strategy and dynamic threshold are used. 66
5.6 Time-lines of outgoing packet overhead of node 2 in topology of Fig. 5.1(a). 67
5.7 Total control message overhead per node with fixed threshold, adaptive threshold ($\gamma = 25$) and dynamic threshold (Fig.5.1(b)) 68
5.8 Effect of γ parameter on total control packet overhead experienced by node 2 in topology of Fig. 5.1(a). 70
5.9 DAG inconsistency attack scenario used to study the effect of neighborhood size on the dynamic threshold. 71
5.10 Outgoing packet overhead experienced by node 2 in the topology of Fig. 5.9 with varying neighborhood sizes. 71
5.11 Outgoing packet overhead experienced by node 2 in the topology of Fig. 5.1(b) with varying sending frequencies and neighborhood sizes. 72
5.12 Global delivery ratio for different neighborhood sizes of node 2 (Fig.5.1(b)) with dynamic threshold. 72
5.13 Energy required for adaptive (γ=25) and dynamic thresholds computation under different attack patterns. 75
5.14 Energy consumption caused by control message overhead and thresholds computation under different attack patterns. 76

6.1 Typical AMI network [19]. 80
6.2 Example of our passive monitoring architecture exploiting the RPL multi-instance feature. 81
6.3 DAG inconsistency attack illustration where node 6 is the malicious node. 88
6.4 Version number attack illustrations. 92

7.1 Monitoring architecture and detection strategy evaluation. 96
7.2 Average success ratio while varying sending frequency. 97
7.3 Average number of listened packets in one hour while varying sending frequency. 98
7.4 Energy consumed by the overhearing mode for a monitoring node while varying sending frequency (2 neighbors). 100

7.5 Energy consumed by the overhearing mode while varying neighborhood size for one hour. 101
7.6 Scenario used to perform the DAG inconsistency attack. 102
7.7 Detection time of the DAG inconsistency attack for different location of attacker. 103
7.8 Per node outgoing packet overhead, for every location of the attacker when no mitigation and dynamic mitigation is deployed on monitoring nodes. 104
7.9 Grid topology of 20 nodes used to perform version number attacks. . 105
7.10 False positive rates for different location of attacker when configuration is the topology of Fig. 7.9. 109
7.11 Average false positive rate for different Ca_2 values with 4 monitoring nodes. 110
7.12 Average false positive rate for different Ca_2 values with 5 monitoring nodes. 110
7.13 Number of monitoring nodes required to have $Ca_1 = 100\%$, $Ca_2 \geq 60\%$ and $Ca_2 = 100\%$ for different topology sizes 113

1 Taxonomie des attaques contre les réseaux RPL 132
2 Nombre de messages de contrôle envoyé par nœud sous différents scenarios d'attaque 133
3 Nombre total de boucles par nœud pour chaque position de l'attaquant. 134
4 Nombre de messages de contrôle total envoyés par le réseau quand la mitigation par seuil fixe et celle par seuil adaptatif sont déployées. . 135
5 Nombre de messages de contrôle total envoyés par le réseau quand la mitigation par seuil fixe et celle par seuil dynamique sont déployées. 136
6 Temps de détection par nœud de surveillance pour différentes positions de l'attaquant. 137
7 Taux de faux positifs moyen pour différentes valeurs de Ca_2 avec 5 nœuds de supervision. 138
8 Nombre de nœuds de supervision nécessaires pour obtenir différentes valeurs de Ca pour plusieurs tailles de topologie. 139

List of Tables

1.1 Classes of constrained devices used in the Internet of Things (IoT) [11]. 3
1.2 Protocol comparison results [47]. 4

2.1 Comparison of monitoring approaches. 22

3.1 Summary of attacks on resources. 30
3.2 Summary of attacks on topology. 36
3.3 Summary of attacks on traffic. 40

5.1 Average computation time (ms) to calculate adaptive and dynamic thresholds for different attack patterns. 73
5.2 Memory occupancy of the different thresholds. 75
5.3 Energy model for the CC2420 radio and MSP430F1611 microcontroller operating at 1 MHz on the TelosB platform. 75

6.1 Summary of considered notations. 83
6.2 Required inputs for monitoring node placement. 85
6.3 Considered variables for modeling. 86
6.4 Potential attacker list and neighbors list obtained by the root after messages aggregation. 92
6.5 States of attacker list and safe list during the localization procedure. 93

7.1 Performance while varying the distance. 97
7.2 Performance while varying the neighborhood size. 99
7.3 Energy model for the CC2420 radio and MSP430F1611 microcontroller operating at 1 MHz on TelosB. 99
7.4 Neighborhood of the different monitoring nodes 102
7.5 Targeted node for the different location of the attacker 103
7.6 Number of possible configurations for 4 monitoring nodes ranked by increasing Ca_2. 106
7.7 Different configurations of Cov_i for 4 monitoring nodes ranked by increasing Ca_2. 107
7.8 Different configurations of Cov_i for 5 monitoring nodes ranked by increasing Ca_2. 108
7.9 Detection result details when configuration is the topology of Fig. 7.9 109

7.10 Required inputs for monitoring node placement. 111
7.11 Considered variables for modeling. 111

Résumé de la thèse en français

Supervision et Sécurité pour l'Internet des Objets utilisant le protocole de routage RPL

1 Introduction

L'Internet des Objets (IdO) se traduit par le déploiement de réseaux avec pertes et à faible puissance appelés réseaux LLN[15]. Ces réseaux permettent à de nombreux équipements embarqués comme des sondes ou des capteurs de pouvoir communiquer entre eux. Un protocole de routage appelé RPL[16] a été spécialement conçu par l'IETF pour répondre aux contraintes spécifiques qu'impose ce type de réseaux. Cependant, ce protocole reste exposé à de nombreuses attaques de sécurité. Si des mécanismes de protection existent, leur mise en œuvre est coûteuse c'est pourquoi notre objectif dans cette thèse est de proposer des stratégies de supervision de la sécurité efficaces qui consomment peu de ressources. Ces solutions doivent être capable de détecter des comportements malveillants de nœuds dans le réseau afin d'en limiter les effets.

Dans un premier temps, nous nous intéresserons à l'évaluation de la sécurité dans les réseaux RPL en identifiant et caractérisant les différentes menaces auxquelles sont soumis ces types de réseaux. Nous classerons également ces menaces selon plusieurs critères afin de mettre en avant celles à traiter en priorité. Grâce à cette évaluation, nous allons dans un second temps concevoir des solutions de supervision de sécurité qui minimisent la consommation des nœuds du réseau pour les attaques identifiées. Ces solutions se doivent d'exploiter les fonctionnalités du protocole RPL et les particularités de l'IdO comme l'hétérogénéité des nœuds pour préserver au mieux des ressources déjà fortement contraintes. Nous évaluerons également l'efficacité et le coût des solutions proposées.

Les travaux de cette thèse sont organisés comme suit. La section 2 présente succinctement le protocole de routage RPL. Nous proposons une taxonomie des attaques visant ce protocole dans la section 3. La section 4 analyse en détail différentes attaques identifiées dans la section précédente à savoir l'attaque d'incohérence DAG et l'attaque sur le numéro de version. La section 5 montre plusieurs mécanismes de

[15] Low power and Lossy Networks
[16] Routing Protocol for LLNs

mitigation locaux pour l'attaque d'incohérence DAG. Nous présentons ensuite une architecture de supervision orientée sécurité dans la section 6 ainsi que des algorithmes de détection conçus pour les deux attaques précédemment étudiées. Nous évaluons cette architecture dans la section 7 avant de conclure cette thèse en section 8.

2 Protocole de routage RPL

Le protocole RPL est un protocole de routage à vecteur de distance utilisant IPv6, spécialement conçu par l'IETF pour répondre aux besoins des réseaux LLN. Cette section présente le fonctionnement de ce protocole et les mécanismes de protection existants.

Topologie, instance et fonction objectif. Les nœuds RPL s'interconnectent en formant une topologie spécifique appelée DODAG[17], c'est-à -dire un graphe acyclique orienté dirigé vers une destination qui est la racine du réseau. Un réseau RPL contient au moins une instance RPL qui elle-même se compose d'un ou plusieurs DODAGs. Chaque instance RPL est associée à une fonction objectif (OF) qui permet d'optimiser la topologie en fonction d'un ensemble de contraintes et/ou de métriques comme la préservation de l'énergie, le chemin le plus court ou la qualité des liens. Un nœud peut faire partie d'un seul DODAG par instance, mais peut participer à plusieurs instances simultanément.

Messages de contrôle et construction du DODAG. La construction et la maintenance des DODAGs sont réalisées grâce à des messages de contrôle ICMPv6. Plus particulièrement, trois nouveaux messages sont définis: (1) *DODAG Information Solicitation* (DIS), (2) *DODAG Information Object* (DIO) et (3) *Destination Advertisement Object* (DAO). Un nouveau nœud peut rejoindre un réseau déjà formé en diffusant un message DIS pour solliciter en réponse un message DIO qui contient des informations sur le DODAG comme le numéro de version et l'identifiant du DODAG, l'identifiant de l'instance et l'OF utilisée. Un nœud peut également attendre de recevoir un message DIO diffusé périodiquement par ses voisins. La fréquence d'envoi des messages DIO est déterminée par un temporisateur fondé sur l'algorithme Trickle (appelé également temporisateur Trickle). À la moindre anomalie dans le réseau, le temporisateur Trickle est réinitialisé pour permettre à la topologie de reconverger plus rapidement.

Après avoir reçu un message DIO, le nœud calcule son rang en utilisant l'OF spécifiée dans ce message. Le rang d'un nœud correspond à son emplacement dans le graphe par rapport à la racine. La valeur du rang augmente toujours en descendant dans le graphe. C'est donc la racine qui a le rang le plus petit dans le graphe. Si un nœud reçoit des DIOs de voisins différents, l'émetteur avec le meilleur rang (le plus petit donc) est choisi comme le parent préféré vers lequel seront envoyés tous les messages à destination de la racine. À la fin de ce processus seulement

[17] Destination Oriented Directed Acyclic Graph

les routes ascendantes (i.e. vers la racine) sont construites. Pour établir les routes descendantes, un nœud doit envoyer un message DAO à son parent contenant le préfixe des nœuds situés dans son sous-DODAG. Lorsque le message se propage vers le haut, les préfixes sont agrégés et les routes descendantes deviennent disponibles pour les parents.

Mécanismes de protection existants. RPL intègre différents mécanismes afin d'éviter les boucles, détecter les incohérences et réparer le graphe. Le rang joue un rôle important pour construire une topologie sans boucle. En effet, un nœud ne peut choisir qu'un parent dont le rang est inférieur au sien, autrement dit tous les nœuds se trouvant dans le sous-DODAG d'un nœud ont un rang supérieur à ce nœud. Si un nœud ne respecte pas cette propriété du rang, le graphe n'est plus acyclique. De plus, afin d'éviter les boucles, si un nœud doit changer son rang, il doit utiliser un mécanisme de *poisoning* (en annonçant un rang infini) ou de déconnexion (en formant un DODAG temporaire).

Dans les cas où des boucles apparaissent dans le graphe, le protocole RPL fournit une fonctionnalité appelée validation du chemin de données [18]. Des informations de contrôle sont transportées dans les paquets de données via des *flags* placés dans l'en-tête d'extension IPv6 Hop-By-Hop:

• Le *flag* 'O' indique la direction attendue du paquet, i.e., vers le haut ou le bas. Si un nœud place ce *flag* à 1 le paquet est destiné à un descendant, sinon le paquet est supposé être envoyé à un parent avec un rang inférieur, vers la racine du DODAG.

• Le *flag* 'R' indique si une erreur de rang a été détectée par un nœud transférant le paquet. Ce *flag* est mis à 1 lorsqu'un nœud observe une incohérence entre la direction supposée du paquet indiquée par le *flag* 'O' et le rang du nœud qui vient de le transférer. Le *flag* 'R' est utilisé pour réparer ce type d'anomalie appelée incohérence DODAG. Concrètement, à la première incohérence détectée, le nœud place ce *flag* à 1 et transfère le message. Lors de la réception d'un autre paquet avec le *flag* positionné à 1 et l'incohérence à nouveau détectée, le paquet est supprimé et le temporisateur Trickle est réinitialisé de sorte que les messages de contrôle sont envoyés plus rapidement, afin de refaire converger la topologie et réparer la boucle.

Deux principaux mécanismes de réparation sont utilisés dans les réseaux RPL en cas d'incohérences ou de pannes: la réparation locale et globale. La réparation locale consiste à trouver un chemin alternatif pour router les paquets. Par exemple, lorsque que la communication avec le parent préféré est rompue, un nœud peut choisir un autre parent pour transférer ses paquets. Si aucun autre parent n'est disponible, il peut aussi envoyer les paquet à un frère, i.e., un nœud de même rang. Si les réparations locales ne suffisent pas, la racine peut initier une réparation globale en incrémentant le numéro de version du DODAG. Ceci a pour résultat la reconstruction complète du graphe.

[18] data path validation

3 Taxonomie des attaques contre le protocole RPL

Les différentes attaques visant le protocole RPL ont été répertoriées ainsi que les contre-mesures existantes dans cette section. Elles ont de plus été classifiées selon qu'elles menaçaient en priorité les ressources des nœuds (surconsommation d'énergie par exemple), la topologie du réseau et enfin le trafic comme le montre la Figure 1. Les attaques de la première catégorie ont pour but de consommer l'énergie, la mémoire ou le temps de calcul des nœuds. On distingue dans cette catégorie les attaques dites directes et indirectes. Dans le cas des attaques directes un nœud malveillant génère directement la surcharge; pour la seconde sous-catégorie, le nœuds malveillant fait en sorte que les cibles génèrent la surcharge. Les attaques de la seconde catégorie visent la topologie du réseau. Plus précisément, ces attaques peuvent engendrer une sous-optimisation du réseau (les routes sont plus longues que ce qu'elles devraient être par exemple) ou l'isolation de nœuds dans le réseau. Enfin la dernière catégorie concerne les attaques ciblant le trafic que nous avons sous-diviser en deux: l'écoute du trafic afin d'accéder au contenu des données et à leur confidentialité et le détournement du trafic qui vise à usurper certaines propriétés comme l'identité d'autres nœuds.

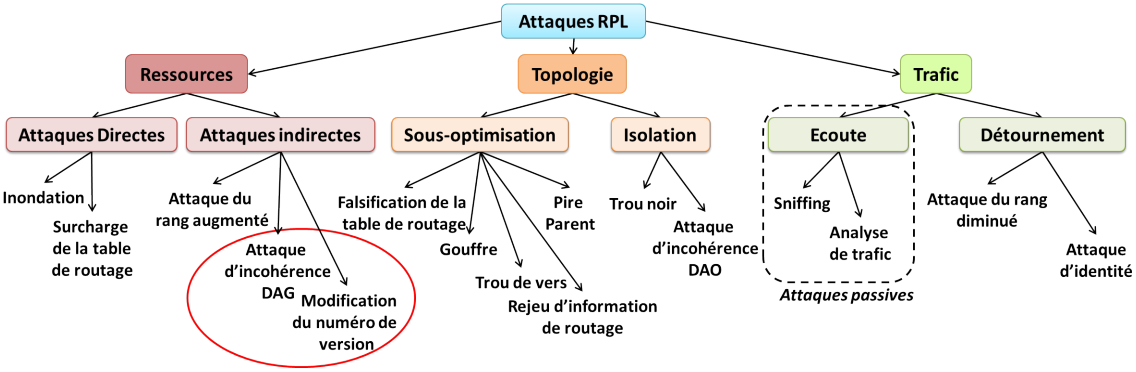


Figure 1: Taxonomie des attaques contre les réseaux RPL

4 Analyse d'attaques visant le protocole RPL

Nous avons étudié deux attaques particulières visant les ressources des nœuds du réseau dans le cadre de cette thèse, à savoir l'attaque d'incohérence DAG et la modification du numéro de version. Ces attaques ont été choisies car elles visent les ressources des nœuds et sont spécifiques au protocole RPL.

4.1 Attaque d'incohérence DAG

En théorie, le mécanisme de validation du chemin de données, présenté dans la Section 2.2, a pour objectif d'améliorer la fiabilité générale du réseau. Cependant, il est possible de détourner cette fonctionnalité pour attaquer le réseau. En effet, un

attaquant peut faire croire à des nœuds qu'il y a des boucles alors que la topologie est stable.

Deux approches peuvent être adoptées par l'attaquant: (1) forger des paquets directement avec le *flag* 'R' et la mauvaise direction positionnés, (2) modifier les paquets qui transitent par lui en positionnant les *flags* contenus dans l'en-tête d'extension. Dans les deux cas, la conséquence immédiate de cette attaque est l'inondation du réseau en messages de contrôle, puisque tous les nœuds victimes ayant reçus les paquets malveillants ainsi que leur voisinage réinitialiseront leur temporisateur Trickle. Ceci réduira à terme la durée de vie du réseau. La figure 2 montre cette augmentation du nombre de messages de contrôle envoyés par les différents nœuds du réseau pour des fréquences d'attaques variées. Dans le second cas plus particulièrement, les paquets modifiés seront supprimés par le parent de l'attaquant ce qui engendrera un *blackhole* déporté sur ce parent où les paquets de données seront perdus.

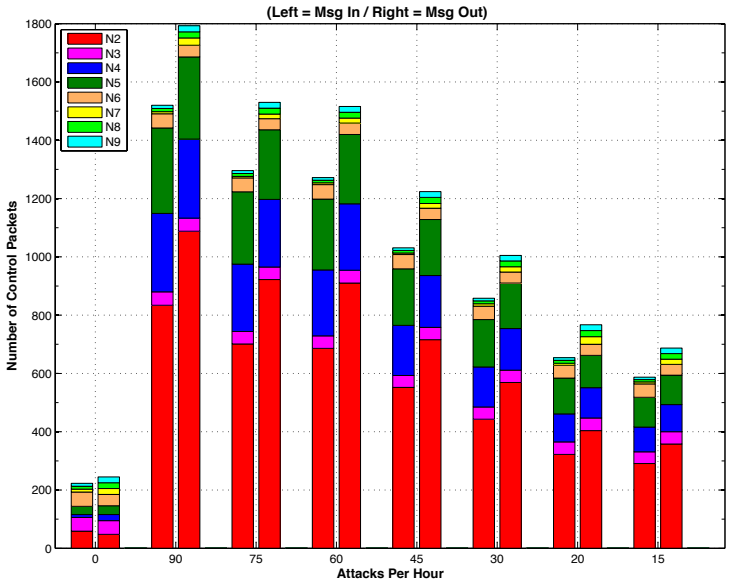


Figure 2: Nombre de messages de contrôle envoyé par nœud sous différents scenarios d'attaque

4.2 Attaque sur le numéro de version

Le numéro de version est un champ important dans les messages DIO. Il doit être propagé sans être modifié le long du DODAG. Seule la racine peut l'incrémenter afin de créer une nouvelle version du DODAG pour revalider l'intégrité du réseau et permettre une réparation globale. Si un nœud annonce une version plus ancienne, cela signifie qu'il n'a pas migré vers la nouvelle version et qu'il ne doit pas être choisi en tant que parent. Un attaquant peut changer la version du DODAG en incrémentant illégitimement le champ correspondant dans ses messages DIO avant de les transmettre à ses voisins. Ceci aura pour conséquence la génération de boucles dans le

graphe et la reconstruction entière du DODAG impliquant une consommation de la batterie des nœuds. Nous avons étudié les conséquences de cette attaque en faisant varier la position de l'attaquant dans le réseau, celui-ci incrémentant régulièrement le numéro de version du graphe. La figure 3 montre le nombre de boucles détectées par nœud pour différentes positions de l'attaquant dans le réseau. On peut voir que cette attaque génère un grand nombre de boucles dans le réseau ce qui amène le réseau à envoyer plus messages et donc réduit sa durée de vie.

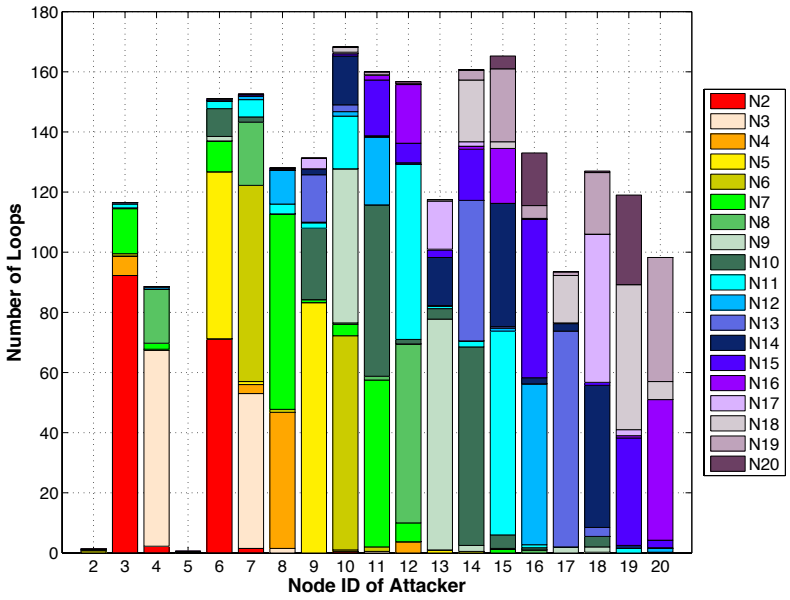


Figure 3: Nombre total de boucles par nœud pour chaque position de l'attaquant.

5 Détection locale d'attaques d'incohérence DAG

Après avoir étudié les attaques précédemment décrites, nous nous sommes intéressés à des mesures de mitigation pour limiter leurs effets. Notre étude s'est porté dans un premier sur la mitigation des attaques d'incohérence DAG. Dans le standard proposé par l'IETF, un seuil fixe était proposé comme mesure pour limiter cette attaque bien qu'aucune explication quant à sa valeur n'était fournie. Nous avons donc proposé une nouvelle mesure de mitigation fondée sur un seuil adaptatif comprenant différents paramètres fixés. La figure 4 présente une partie des résultats obtenus. Elle montre le nombre de messages de contrôle reçus pour les différents nœuds du réseau en fonction du nombre d'attaques (paquets de données malveillants) envoyées par heure. Sur le graphique la barre de gauche montre les résultats quand le seuil fixe est utilisé, les deux autres barres correspondent à notre approche pour différentes valeurs de paramètres. D'après cette figure, on peut voir que notre approche réduit avec succès

le nombre de messages envoyés dus à l'attaque, même dans le cas d'attaque très agressive. Les résultats du seuil adaptatif sont entre 8% et 13% meilleurs que le seuil fixe, même dans les pires cas.

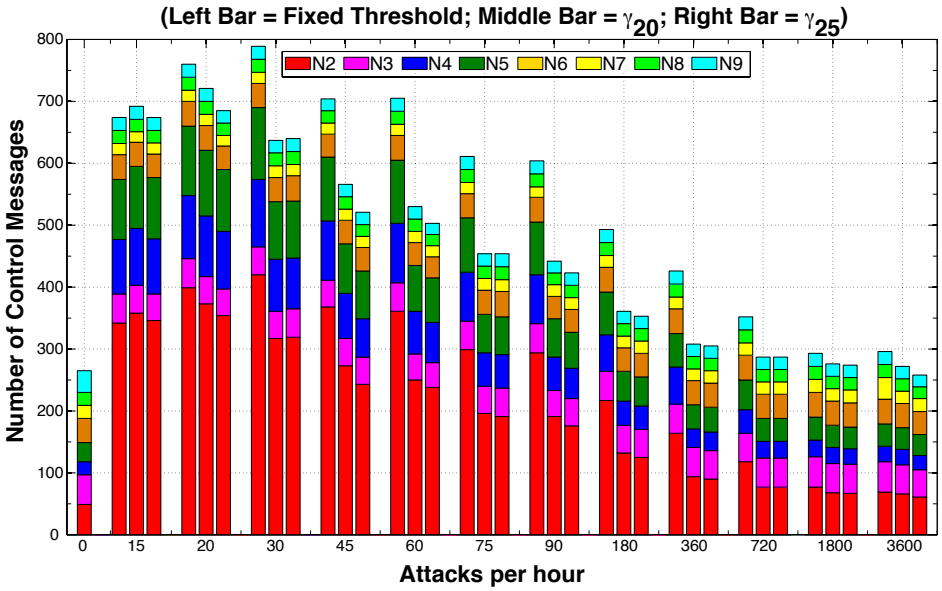


Figure 4: Nombre de messages de contrôle total envoyés par le réseau quand la mitigation par seuil fixe et celle par seuil adaptatif sont déployées.

Ces travaux ont été étendus et améliorés grâce un seuil dynamique prenant en compte les paramètres du réseau propre à chaque nœud et ne nécessitant pas de paramètres a priori fixés. La figure 5 montre que cette approche dynamique permet d'obtenir de meilleurs résultats que le seuil fixe. De plus en comparant les deux figures entre elles, on peut voir que l'approche dynamique est plus performante pour les attaques "lentes" et des performances similaires pour les attaques agressives. Nous avons également évalué le coût de ses solutions et avons montré que ces algorithmes consommaient peu par rapport aux bénéfices obtenus.

6 Architecture de supervision distribuée pour la sécurité

Afin de pouvoir détecter les attaques ne pouvant être gérées localement comme l'attaque du numéro de version dont les conséquences se propagent à tout le réseau, nous avons conçu une architecture de supervision et réalisé un POC (*Proof Of Concept*). L'architecture a pour visée de détecter les attaques dans un réseau RPL. Celle-ci se compose de deux types de nœuds: les nœuds réguliers qui sont des nœuds classiques fortement contraints de l'Internet des Objets et les nœuds de surveillance capables de collecter des informations sur les nœuds réguliers à portée. Les nœds de supervision forment un réseau indépendant appelé réseau de supervision, cependant ceux-ci participent aussi au réseau dit régulier afin de ne pas perdre de ressources.

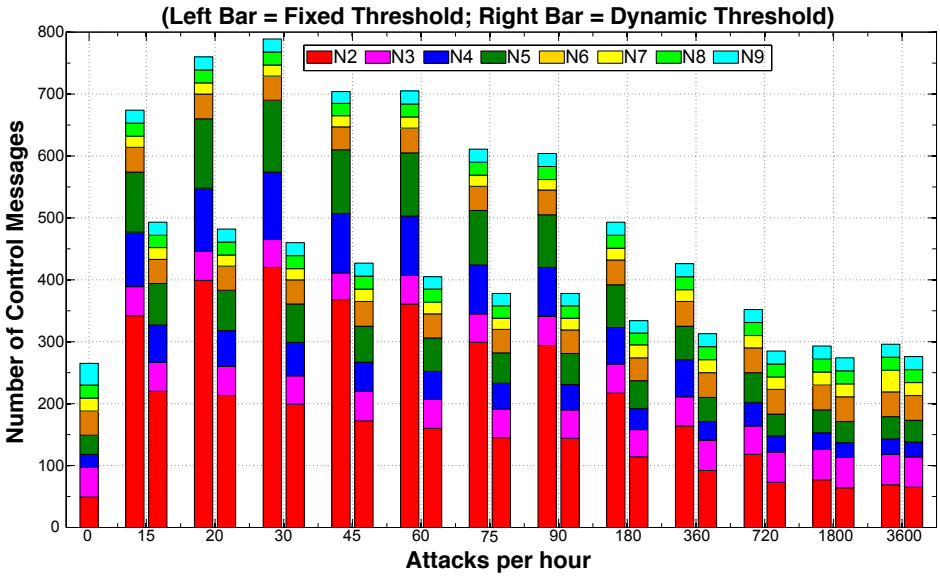


Figure 5: Nombre de messages de contrôle total envoyés par le réseau quand la mitigation par seuil fixe et celle par seuil dynamique sont déployées.

L'existence de ces deux réseaux est rendu possible grâce au mécanisme de multi-instance du protocole RPL.

Des algorithmes de détection sont implémentés sur les nœuds de supervision en utilisant les données récoltées. Nous avons développé deux modules permettant de détecter l'attaque d'incohérence DAG et l'attaque du numéro de version. Concernant le premier module chaque nœud de supervision maintient des compteurs pour chacun des voisins à portée afin de détecter si l'un d'eux exécute une attaque d'incohérence DAG. Le second module est quant à lui chargé de détecter et de localiser le nœud malveillant réalisant l'attaque du numéro de version. Ce module est composé de trois algorithmes. Le premier est déployé sur chaque nœud de supervision non racine. Ces derniers s'occupent de rapporter à la racine si un numéro de version incrémenté a été propagé ainsi que l'émetteur du numéro de version et la liste des voisins qu'ils surveillent. La racine grâce à un premier algorithme récolte toutes les informations envoyés par les nœuds de supervision et si une attaque est détecté, celle-ci déclenche la procédure de localisation de l'attaquant. Grâce aux informations précédemment récoltées la racine compare les émetteurs du numéro de version incrémenté et les voisins de chaque nœud de supervision. En procédant par élimination on obtient une liste d'attaquants potentiels. Bien que l'algorithme garantisse que le nœud malveillant soit établi comme attaquant, il se peut aussi qu'un nœud sain soit considéré comme attaquant en fonction de la configuration des nœuds de supervision. Ce point est étudié lors de l'évaluation de l'architecture.

7 Évaluation de l'architecture

Nous avons évalué la faisabilité et les performances de notre architecture à travers plusieurs séries d'expériences. Dans un premier temps nous avons évalué les performances et le coût du mode "promiscuité" utilisé par les nœuds de supervision lors de la collecte d'information. Ces expériences ont montré que les paramètres considérés pour l'étude n'influençaient pas le taux de succès moyen dans l'environnement de simulation. L'analyse du coût a montré que celui-ci était proportionnel au trafic réseau et au nombre de voisin.

Nous avons ensuite étudié les performances des deux modules de détection. Concernant l'attaque d'incohérence DAG, nous avons montré que notre module était capable de détecter l'attaque pour tous les scénarios considérés comme présenté par la figure 6.

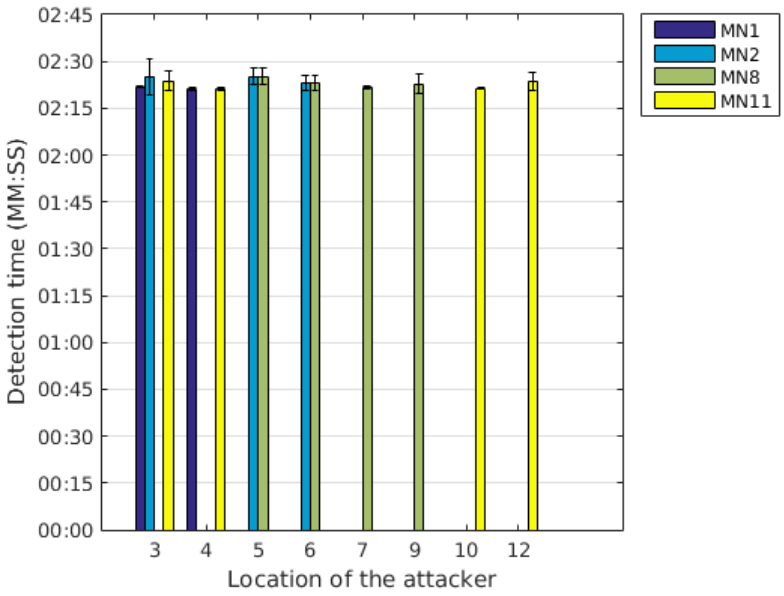


Figure 6: Temps de détection par nœud de surveillance pour différentes positions de l'attaquant.

Nos expériences ont montré pour le second module de détection que le nœud malveillant était toujours détecté. Cependant des nœuds sains pouvait également être classés comme attaquant c'est pourquoi nous avons étudié le taux de faux positifs dans nos expériences qui représente le nombre de nœuds sains considérés comme attaquant par notre module. Nous avons sélectionné différentes configurations de nœuds de supervision pour étudier l'influence de ce paramètre sur le taux de faux positifs. Les configurations ont été sélectionnées de sorte que le nombre de nœuds réguliers couverts par au moins deux nœuds de supervision varie (appelé Ca_2). Les résultats ont montré qu'un placement stratégique des nœuds de supervision permettait d'avoir un taux de faux positifs presque nul comme le montre la figure 7.

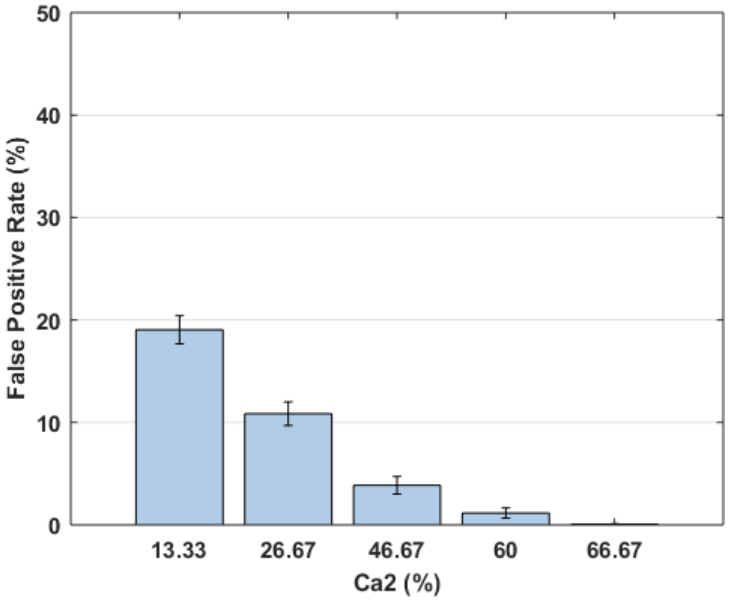


Figure 7: Taux de faux positifs moyen pour différentes valeurs de Ca_2 avec 5 nœuds de supervision.

Nous avons finalement étudié le passage à l'échelle de notre architecture et en particulier du second module de détection en considérant le problème du placement des nœuds de supervision dans le réseau. Nous avons modélisé ce problème à l'aide de l'optimisation linéaire en nombre entiers. Ce problème d'optimisation a ensuite été résolu grâce un *solver* appelé CPLEX. La figure 8 montre que le nombre de nœuds de supervision nécessaires pour différentes conditions en fonction de la taille de la topologie est linéaire ce qui montre que notre solution peut passer à l'échelle.

8 Conclusions

Dans cette thèse nous nous sommes intéressés à des stratégies de supervision orientées sécurité dans l'Internet des Objets utilisant le protocole RPL afin de proposer un compromis entre sécurité et le coût induit. Dans un premier temps nous avons évalué les menaces pesant sur le protocole RPL à travers l'identification et la classification d'attaques à l'aide d'une taxonomie. Nous avons analysé l'impact de deux attaques spécifiques qui sont l'attaque d'incohérence DAG et l'attaque du numéro de version et avons montré à quel point celles-ci pouvaient être nuisible. Nous avons ensuite présenté une stratégie locale pour détecter et limiter les attaques d'incohérence DAG et avons évalué son coût et ses performances. Afin de compléter notre approche locale et gérer des attaques plus complexes, nous avons conçu une architecture supervision orientée sécurité. En s'appuyant sur des nœuds distribués, cette architecture préserve les ressources des nœuds fortement contraints. Dans le même objectif, celle-ci ex-

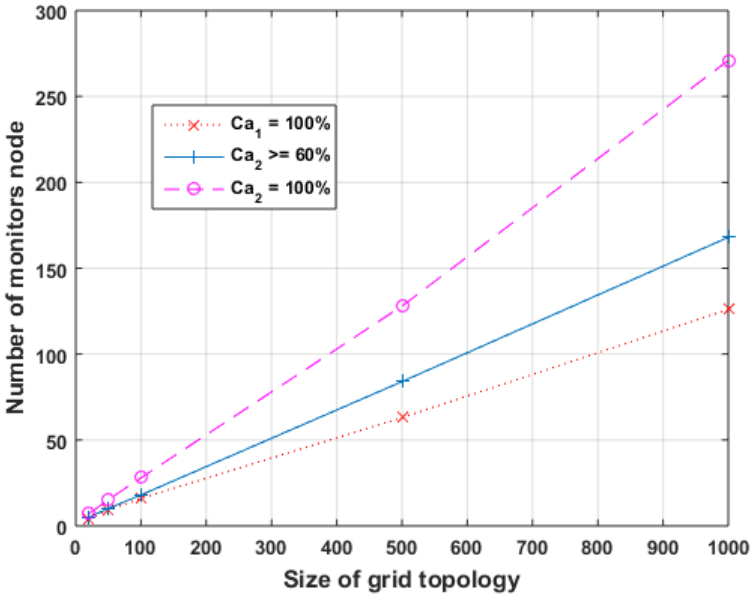


Figure 8: Nombre de nœuds de supervision nécessaires pour obtenir différentes valeurs de Ca pour plusieurs tailles de topologie.

ploite des mécanismes du protocole RPL comme le multi-instance. Les nœuds de supervision distribués implémentent des modules de détection chargés d'identifier les attaques considérées. Nous avons évalué notre architecture grâce à de nombreuses expériences et avons également discuté du placement des nœuds de supervision.

Le travail réalisé durant cette thèse ouvre de nombreuses perspectives dans le cadre de la supervision de sécurité dans l'Internet des Objets. Premièrement, nous nous intéressons à poursuivre nos expériences dans des environnements réels et aussi à porter l'implémentation de notre stratégie à d'autres OS qui se sont développés au cours de cette thèse comme RIOT ou OpenWSN. Nous voulons aussi étendre notre étude à de nouveaux scenarios d'attaques, notamment le cas de coalition d'attaquants mais aussi considérer d'autres attaques abordées dans notre taxonomie et même des attaques visant d'autres protocoles de la pile réseau. À long terme, nous envisageons d'intégrer notre solution au sein d'un framework de gestion de risque. La gestion de risque offre de nouvelles perspectives pour activer ou désactiver dynamiquement des mécanismes de sécurité dans les réseaux RPL afin de prévenir et limiter des attaques tout en préservant les ressources du réseau.

Bibliography

[1] A Mathematical Programming Language (AMPL).

[2] IBM ILOG CPLEX Optimization Studio.

[3] Internet of Things Global Standards Initiative.

[4] OpenWSN project.

[5] RIOT: The friendly Operating System for the Internet of Things.

[6] OSI IS-IS Intra-domain Routing Protocol. RFC 1142, feb 2014.

[7] L. Atzori, A. Iera, and G. Morabito. The Internet of Things: A survey. *Elsevier Journal Computer Networks*, 54(15):2787–2805, Oct. 2010.

[8] A. Awad, R. Nebel, R. German, and F. Dressler. On the Need for Passive Monitoring in Sensor Networks. In *Digital System Design Architectures, Methods and Tools, 2008. DSD '08. 11th EUROMICRO Conference on*, pages 693–699, Sept 2008.

[9] E. Baccelli, R. Cragie, P. V. der Stok, and A. Brandt. Applicability Statement: The Use of the Routing Protocol for Low-Power and Lossy Networks (RPL) Protocol Suite in Home Automation and Building Control. RFC 7733, feb 2016.

[10] A. Barbir, S. Murphy, and Y. Yang. Generic Threats to Routing Protocols. RFC 4593 (Informational), Oct. 2006.

[11] C. Bormann, M. Ersue, and A. Keranen. Terminology for Constrained-Node Networks. *IETF RFC 7228*, May 2014.

[12] C. Bormann and P. Hoffman. Concise Binary Object Representation (CBOR). RFC 7049 (Proposed Standard), oct 2013.

[13] T. Bray. The JavaScript Object Notation (JSON) Data Interchange Format. RFC 7159 (Proposed Standard), mar 2014.

[14] J. Case, M. Fedor, M. Schoffstall, and J. Davin. Simple Network Management Protocol (SNMP). RFC 1157 (Historic), May 1990.

[15] I. Chakeres and C. Perkins. Dynamic manet on-demand (dymo) routing. *draft-ietf-manet-dymo-19. txt (work in progress)*, 2010.

[16] B.-r. Chen, G. Peterson, G. Mainland, and M. Welsh. LiveNet: Using Passive Monitoring to Reconstruct Sensor Network Dynamics. In S. Nikoletseas, B. Chlebus, D. Johnson, and B. Krishnamachari, editors, *Distributed Computing in Sensor Systems*, volume 5067 of *Lecture Notes in Computer Science*, pages 79–98. Springer Berlin Heidelberg, 2008.

[17] Chipcon AS. CC2420 2.4 GHz IEEE 802.15.4/ZigBee-ready RF Transceiver. *Oslo, Norway*, 2004.

[18] K. Chugh, L. Aboubaker, and J. Loo. Case Study of a Black Hole Attack on LoWPAN-RPL. In *Proc. of the Sixth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE)*, Rome, Italy, August 2012.

[19] Cisco Systems. Routing in The Internet of Things – M2M Networks. *BRKSPG-1661*, 2013.

[20] T. Clausen, C. Dearlove, P. Jacquet, and U. Herberg. RFC7181: The Optimized Link State Routing Protocol Version 2. IETF - Proposed Standard RFC 7681, 2014.

[21] T. Clausen and U. Herberg. A Comparative Performance Study of the Routing Protocols LOAD and RPL with Bi-Directional Traffic in Low-power and Lossy Networks (LLN) . Master's thesis, Ecole Polytechnique, Centre de recherche INRIA Saclay, Orsay, France, 2011.

[22] S. R. Das, C. E. Perkins, and E. M. Belding-Royer. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, mar 2013.

[23] S. Dawans, S. Duquennoy, and O. Bonaventure. On Link Estimation in Dense RPL Deployments. In *7th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, Clearwater, FL, Oct 2012.

[24] P. V. der Stok and A. Bierman. CoAP Management Interface. Internet-Draft draft-vanderstok-core-comi-09, Internet Engineering Task Force, mar 2016. Work in Progress.

[25] D. Dong, X. Liao, Y. Liu, C. Shen, and X. Wang. Edge self-monitoring for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 22(3):514–527, March 2011.

[26] J. R. Douceur. The Sybil Attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, UK, 2002. Springer-Verlag.

[27] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. In *29th Annual IEEE International Conference on Local Computer Networks (LCN)*, pages 455–462, Tampa, FL, USA, November 2004.

[28] A. Dvir, T. Holczer, and L. Buttyan. VeRA - Version Number and Rank Authentication in RPL. In *8th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 709–714, Hangzhou, China, October 2011.

[29] S. Elyengui, R. Bouhouchi, and T. Ezzedine. Loadng routing protocol evaluation for bidirectional data flow in AMI mesh networks. *CoRR*, abs/1506.06357, 2015.

[30] R. Enns, M. Bjorklund, A. Bierman, and J. Schönwälder. Network Configuration Protocol (NETCONF). RFC 6241, Oct. 2015.

[31] F. P. Garcia, R. M. C. Andrade, C. T. Oliveira, and J. N. de Souza. EPMOSt: An Energy-Efficient Passive Monitoring System for Wireless Sensor Networks. *Sensors*, 14(6):10804, 2014.

[32] J. Hui and J. Vasseur. The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams. RFC 6553 (Proposed Standard), mar 2012.

[33] P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626, mar 2013.

[34] F. Khan, T. Shon, T. Lee, and K. Kim. Wormhole attack prevention mechanism for RPL based LLN network. In *Ubiquitous and Future Networks (ICUFN), 2013 Fifth International Conference on*, pages 149–154, July 2013.

[35] M. M. H. Khan, L. Luo, C. Huang, and T. Abdelzaher. SNTS: Sensor Network Troubleshooting Suite. In *Proceedings of the 3rd IEEE International Conference on Distributed Computing in Sensor Systems*, DCOSS'07, pages 142–157, Berlin, Heidelberg, 2007. Springer-Verlag.

[36] J. Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, and A. Terzis. Evaluating the Performance of RPL and 6LoWPAN in TinyOS. In *Workshop on Extending the Internet to Low Power and Lossy Networks (IP+SN)*, Chicago, IL, USA, April 2011.

[37] K. D. Korte, A. Sehgal, and J. Schönwälder. A Study of the RPL Repair Process Using ContikiRPL. In *AIMS*, pages 50–61, 2012.

[38] S. Kuryla and J. Schönwälder. Evaluation of the Resource Requirements of SNMP Agents on Constrained Devices. In *5th Conference on Autonomous Infrastructure, Management and Security (AIMS 2011), Springer LNCS 6734*, Nancy, France, June 2011.

[39] A. Lahmadi, A. Boeglin, and O. Festor. Efficient Distributed Monitoring in 6LoWPAN Networks. In *9th International Conference on Network and Service Management (CNSM)*, Zürich, Switzerland, October 2013.

[40] M. Landsmann, H. Perrey, O. Ugus, M. Wählisch, and T. C. Schmidt. Topology Authentication in RPL. In *INFOCOM. Poster*, 2013.

[41] A. Le, J. Loo, K. K. Chai, and M. Aiash. A Specification-Based IDS for Detecting Attacks on RPL-Based Network Topology. *Information*, 7(2), 2016.

[42] A. Le, J. Loo, A. Lasebae, M. Aiash, and Y. Luo. 6LoWPAN: a study on QoS security threats and countermeasures using intrusion detection system approach. *Int. J. Communication Systems*, 25(9):1189–1212, 2012.

[43] A. Le, J. Loo, A. Lasebae, A. Vinel, Y. Chen, and M. Chai. The Impact of Rank Attack on Network Topology of Routing Protocol for Low-Power and Lossy Networks. *IEEE Sensors Journal*, 13(10):3685–3692, 2013.

[44] A. Le, J. Loo, Y. Luo, and A. Lasebae. Specification-based IDS for Securing RPL from Topology Attacks. In *IFIP Wireless Days (WD)*, pages 1–3, Niagara Falls, Canada, October 2011.

[45] A. Le, J. Loo, Y. Luo, and A. Lasebae. The Impacts of Internal Threats towards Routing Protocol for Low power and lossy Network Performance. In *ISCC*, pages 789–794, 2013.

[46] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko. The trickle algorithm. RFC 6206 (Proposed Standard), mar 2011.

[47] P. Levis, A. Tavakoli, and S. Dawson-Haggerty. Overview of Existing Routing Protocols for Low Power and Lossy Networks, IETF Internet Draft: draft-ietf-roll-protocols-survey-07, April 2009.

[48] C. Liu and G. Cao. Distributed Monitoring and Aggregation in Wireless Sensor Networks. In *30th IEEE International Conference on Computer Communications (INFOCOM)*, San Diego, CA, USA, March 2010.

[49] T. Lys, C. Lavenu, H. Satoh, J. Dean, T. H. Clausen, A. C. Verdiere, J. Yi, A. Niktash, Y. Igarashi, and U. Herberg. The Lightweight On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng). Internet-Draft draft-clausen-lln-loadng-14, Internet Engineering Task Force, jan 2016. Work in Progress.

[50] G. S. Malkin. RIP Version 2. RFC 2453, mar 2013.

[51] D. A. Maltz and D. C. Johnson. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728, mar 2013.

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

[53] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, and J. Schönwälder. A Study of RPL DODAG Version Attacks. In *Proc. of AIMS conference*, 2014.

[54] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, and J. Schönwälder. Mitigation of topological inconsistency attacks in RPL-based low-power lossy networks. *International Journal of Network Management*, 2015.

[55] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, and J. Schönwälder. Using the RPL protocol for supporting passive monitoring in the internet of things. In *2016 IEEE/IFIP Network Operations and Management Symposium, NOMS 2016, Istanbul, Turkey, April 25-29, 2016*, pages 366–374, 2016.

[56] G. Meyer and S. Sherry. OSPF Version 2. RFC 2091, jan 1997.

[57] J. T. Moy. OSPF Version 2. RFC 2328, mar 2013.

[58] National Intelligence Council, Dirsuptive Civil Technologies. Six Technologies with Potential Impacts on US Interests Out to 2025. Conference Report CR 2008-07, 2008.

[59] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-Level Sensor Network Simulation with COOJA. In *31st IEEE Conference on Local Computer Networks (LCN)*, pages 641–648, Tampa, FL, USA, November 2006.

[60] P. Pongle and G. Chavan. A survey: Attacks on RPL and 6LoWPAN in IoT. In *Pervasive Computing (ICPC), 2015 International Conference on*, pages 1–6, Jan 2015.

[61] D. Popa, N. Cam-Winget, and J. Hui. Applicability Statement for the Routing Protocol for Low Power and Lossy Networks (RPL) in AMI Networks. Internet-Draft draft-ietf-roll-applicability-ami-13, Internet Engineering Task Force, may 2016. Work in Progress.

[62] K. N. Ramach, E. M. Belding-royer, and K. C. Almeroth. DAMON: A Distributed Architecture for Monitoring Multi-hop Mobile Networks. In *IEEE SECON*, Santa Clara, CA, USA, October 2004.

[63] S. Raza, L. Wallgren, and T. Voigt. SVELTE: Real-time intrusion detection in the Internet of Things. *Ad Hoc Networks*, 11(8):2661–2674, 2013.

[64] E. Rescorla and N. Modadugu. Datagram Transport Layer Security. RFC 4347 (Proposed Standard), Apr. 2006. Updated by RFC 5746.

[65] E. Rescorla and N. Modadugu. Datagram Transport Layer Security Version 1.2. RFC 6347 (Proposed Standard), oct 2015.

[66] A. RGHIOUI, A. KHANNOUS, and M. BOUHORMA. Denial-of-Service attacks on 6LoWPAN-RPL networks: Issues and practical solutions. *Journal of Advanced Computer Science & Technology*, 3(2):143–153, 2014.

[67] K. Roussel and Y.-Q. Song. A critical analysis of Contiki's network stack for integrating new MAC protocols. Research Report RR-8776, INRIA Nancy, Dec 2013.

[68] K. Roussel, Y.-Q. Song, and O. Zendra. Lessons Learned through Implementation and Performance Comparison of Two MAC/RDC Protocols on Different WSN OS. Research Report RR-8777, INRIA Nancy, Mar 2015.

[69] K. Roussel, Y.-Q. Song, and O. Zendra. Using Cooja for WSN Simulations: Some New Uses and Limits. In K. Roemer, editor, *EWSN 2016 - NextMote workshop*, EWSN 2016 - NextMote workshop, pages 319–324, Graz, Austria, Feb 2016. ACM, Junction Publishing.

[70] S. Seeber, A. Sehgal, B. Stelte, G. D. Rodosek, and J. Schönwälder. Towards A Trust Computing Architecture for RPL in Cyber Physical Systems. In *IFIP/IEEE International Conference on Network and Service Management (CNSM)*, Zürich, Switzerland, October 2013.

[71] A. Sehgal, A. Mayzaud, R. Badonnel, I. Chrisment, and J. Schönwälder. Addressing DODAG Inconsistency Attacks in RPL Networks. In *Proc. of GIIS conference*, 2014.

[72] A. Sehgal, V. Perelman, S. Kuryla, and J. Schönwälder. Management of Resource Constrained Devices in the Internet of Things. *IEEE Communications Magazine*, 50(12):144–149, 2012.

[73] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). RFC 7252 (Proposed Standard), June 2014.

[74] F. L. Templin, R. Ogier, and M. S. Lewis. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). RFC 3684, mar 2013.

[75] Texas Instruments. MSP430F1611 Mixed Signal Controler Datasheet, 2006.

[76] T. Tsao, R. Alexander, M. Dohler, V. Daza, A. Lozano, and M. Richardson. A Security Threat Analysis for Routing Protocol for Low-power and Lossy Networks (RPLs),. RFC 7416, IETF, 2015.

[77] M. Vucinic, B. Tourancheau, and A. Duda. Performance comparison of the RPL and loadng routing protocols in a home automation scenario. In *2013 IEEE Wireless Communications and Networking Conference (WCNC), Shanghai, Shanghai, China, April 7-10, 2013*, pages 1974–1979, 2013.

[78] L. Wallgren, S. Raza, and T. Voigt. Routing Attacks and Countermeasures in the RPL-Based Internet of Things. *International Journal of Distributed Sensor Networks*, 13(794326), 2013.

[79] K. Weekly and K. Pister. Evaluating Sinkhole Defense Techniques in RPL Networks. In *20th IEEE International Conference on Network Protocols (ICNP)*, pages 1–6, Austin, TX, USA, November 2012.

[80] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, IETF, 2012.

[81] W. Xie, M. Goyal, H. Hosseini, J. Martocci, Y. Bashir, E. Baccelli, and A. Durresi. Routing Loops in DAG-Based Low Power and Lossy Networks. In *Proc. of the 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 888–895, Washington, USA, 2010. IEEE Computer Society.

[82] X. Xu, J. Wan, W. Zhang, C. Tong, and C. Wu. PMSW: a passive monitoring system in wireless sensor networks. *International Journal of Network Management*, 21(4):300–325, 2011.

Glossary

6LoWPAN : IPv6 Low power Wireless Personal Area Network.
AMI : Advanced Measurement Infrastructure.
AMPL : A Mathematical Programming Language.
AT : Adaptive Threshold.
AODV : Ad hoc On-demand Distance Vector protocol.
CBOR : Concise Binary Object Representation.
CIA : Confidentiality, Integrity, Availability.
CoAP : Constrained Application Protocol.
CoMI : CoAP Management Interface.
CPU : Central Processing Unit.
DAG : Directed Acyclic Graph.
DAMON : Distributed Architecture for MONitoring mobile network.
DAO : Destination Advertisement Object.
DAO-Ack : Destination Advertisement Object Acknowledgement.
DGRM : Directed Graph Radio Medium.
DIO : DODAG Information Object.
DIS : DODAG Information Solicitation.
DODAG : Destination Oriented Directed Acyclic Graph.
DSR : Dynamic Source Routing protocol.
DT : Dynamic Threshold.
DYMO : Dynamic MANET On-demand protocol.
EPMOSt : Energy-efficient Passive MOnitoring System.
FP : False Positive.
FPR : False Positive Rate.
IEEE : Institute of Electrical and Electronics Engineers.
IETF : Internet Engineering Task Force.
IDS : Intrusion Detection System.
ILP : Integer Linear Programming.
IoT : Internet of Things.
IoT-GSI : Global Standards Initiative on Internet of Things.
IS-IS : Intermediate System to Intermediate System protocol.
IT : Information Technology.
JSON : JavaScript Object Notation.
LLN : Low-power and Lossy Network.

LOADng : Lightweight Ad hoc On-Demand - Next Generation distance vector routing protocol.
MANET : Mobile Ad hoc NETwork.
MIB : Management Information Base.
MOP : Mode Of Operation.
MP2P : Multipoint-to-Point.
NAN : Neighborhood Area Network.
NETCONF : NETwork CONFiguration protocol.
NIC : National Intelligence Agency.
OLSR : Optimized Link State Routing protocol.
OS : Operating System.
OSPF : Open Shortest Path First protocol.
P2MP : Point-to-Multipoint.
P2P : Point-to-Point.
PMSW : Passive Monitoring System for WSN.
RAM : Random-Access Memory.
RIP : Routing Information Protocol.
RoLL : Routing Over Low-power and Lossy networks.
RPL : Routing Protocol for Low-power and lossy networks.
SEM : Standard Error of the Mean.
SNMP : Simple Network Management Protocol.
SNTS : Sensor Networks Troubleshooting Suite.
TBRPF : Topology Broadcast Based on Reverse-Path Forwarding protocol.
TN : True Negative.
UDGM : Unit Disk Graph Medium.
WAN : Wide Area Network.
WSN : Wireless Sensor Network.
XML : Extensible Markup Language.

Résumé / Abstract

L'intérêt grandissant pour l'Internet des Objets s'est traduit par le déploiement à grande échelle de réseaux dits à basse puissance et avec pertes (LLN). Ces réseaux sont fortement contraints en matière de ressources (mémoire, CPU, batterie) et communiquent via des liens instables, à bas débit avec de forts taux d'erreur. Dans ce contexte, les protocoles de routages existants pour les réseaux filaires et pour les réseaux ad-hoc ne sont pas adaptés pour ces caractéristiques. Le groupe de travail RoLL à l'IETF a proposé un nouveau protocole de routage appelé RPL fondé sur IPv6 et spécifiquement conçu pour ces environnements. Cependant, le protocole RPL est exposé à de nombreuses attaques internes et/ou externes comme les attaques consommant les ressources ou les attaques d'interception de trafic. La mise en place de mécanismes de sécurité peut aussi représenter un coût considérable. C'est pourquoi, les réseaux LLN introduisent de nouveaux enjeux quant à leur supervision et leur sécurité. Dans le cadre de cette thèse, nous proposons d'étudier une approche de supervision pour la sécurité de l'Internet des Objets afin de répondre au compromis entre sécurité et coût dans l'Internet des Objets. Nous évaluons tout d'abord les menaces auxquelles sont soumis les réseaux RPL. En particulier, nous identifions et classifions les attaques visant le protocole RPL au travers d'une taxonomie. Nous quantifions également les conséquences de deux attaques appelées l'attaque d'incohérence DAG et l'attaque du numéro de version qui provoquent la surconsommation des ressources des nœuds du réseau. Les résultats obtenus montrent l'importance de gérer ces attaques pour préserver les infrastructures de l'Internet des Objets. Nous nous concentrons ensuite sur les solutions pour la sécurité dans les réseaux RPL. Nous proposons une stratégie locale qui détecte et limite les attaques d'incohérences DAG. Dans le but de détecter des attaques complexes comme les attaques sur le numéro de version et de compléter notre approche locale, nous présentons une architecture de supervision distribuée orientée sécurité pour les réseaux RPL. Cette solution nous permet de préserver l'énergie des nœuds contraints en effectuant les activités de surveillance et de détection sur des nœuds dédiés. Nous montrons la faisabilité de notre approche en implantant une preuve de concept capable de détecter les attaques d'incohérence DAG et les attaques sur le numéro de version. Nous quantifions ensuite les performances de cette architecture ainsi que la stratégie de détection proposée.

Mots-clés: Internet des Objets, LLN, RPL, Sécurité, Supervision

---

The growing interest for the Internet of Things (IoT) has resulted in the large scale deployment of Low power and Lossy Networks (LLN) such as home automation systems. These networks are strongly constrained in terms of resources (memory, power and processing) and communicate using unstable links with high error rates and low throughputs. In this context, existing routing protocols for wired networks and for ad-hoc networks do not cope with all these constraints. The IETF RoLL working group has proposed a new routing protocol called RPL based on IPv6 and specifically designed for these environments. The RPL protocol is however exposed to a large variety of internal and/or external attacks such as resource consuming attacks, traffic interception or loops building attacks. The deployment of security mechanisms may also be quite expensive in terms of resources. Therefore, LLN networks present new challenges in terms of monitoring and security. In this thesis we propose to investigate a security-oriented monitoring approach for addressing the trade-off between security and cost in the Internet of Things. In a first stage, we assess security threats faced by these networks. In particular, we identify and classify attacks targeting RPL networks through a dedicated taxonomy. We also quantify the consequences of two major attacks called DAG inconsistency attacks and version number attacks causing over-consumption of node resources. The obtained results show the importance of addressing them to preserve RPL-based infrastructures. We then focus our work on security solutions for RPL-based Internet of Things. We propose a local strategy for addressing DAG inconsistency attacks and evaluate it through experiments. In order to detect complex attacks such as version number attacks and to complement our node-level approach, we design a security-oriented distributed monitoring architecture for RPL networks. This solution allows us to preserve constrained nodes energy by performing monitoring and detection activities on dedicated nodes. We show the feasibility of our approach by implementing a prototype able to detect both DAG inconsistency and version number attacks. We quantify the performance and the cost of this architecture and the detection modules.

Keywords: Internet of Things, LLN, RPL, Security, Monitoring