Improving Security and Performance of the RPL Routing Protocol for Low Power and Lossy Networks

by

Ahmad Shabani Baghani

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Communications

Department of Electrical and Computer Engineering

University of Alberta

Abstract

The Internet of Things (IoT) is an emerging technology that is connecting billions of otherwise ordinary devices to the Internet. A key component of IoT is Low- power and Lossy Networks (LLNs), composed of various resource-constrained devices with limited energy, memory, and processing power. To communicate with each other, devices (referred to as nodes) in LLNs require an efficient routing protocol. ROLL (Routing Over Low power and Lossy networks), a working group of the Internet Engineering Task Force (IETF), designed RPL, the standard IPv6 Routing Protocol for Low-Power and Lossy Networks, to meet specific needs of LLNs. RPL generates low control plane traffic and offers a range of interesting features for LLNs. However, RPL has several deficiencies with regard to security and point-to-point communications. This thesis investigates and tackles some of these deficiencies.

Chapter 3 introduces and analyses the DAO induction attack, a new attack against RPL. In the DAO induction attack, a compromised node in the network periodically transmits a special control message. Each of these crafted control messages induces many nodes in the network to transmit in response. This significantly increases the power consumption of nodes, hence reducing the lifetime of battery-operated IoT devices. In addition, the attack severely impacts end-to-end latency and packet delivery ratio, two important network performance metrics. The chapter proposes a lightweight solution to counter the attack. The proposed solution imposes no overhead when the network is in its normal operation (i.e., it is not under attack) and can quickly detect the

attack even when the network experiences high packet loss rates.

Chapter 4 studies the sender's authentication problem in RPL and proposes a solution based on the Blom key pre-distribution scheme. The proposed solution has a significantly lower computation cost than the original Blom scheme, hence is more suitable for computationally constrained IoT devices.

Finally, Chapter 5 studies the quality of the RPL's Point-to-Point (P2P) paths. In particular, it analyzes how much RPL's P2P paths "stretch" compared to the shortest paths. It shows that the average stretch is a factor of at least two in any RPL network. Furthermore, it shows that RPL's stretch factor can be considerably higher than two in some network topologies including linear networks and grid networks. To improve the quality of RPL's P2P paths, the chapter proposes a solution that is simple to implement and fully compatible with RPL.

Preface

The results presented in Chapter 3 were published in International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2020) [1] and IEEE Internet of Things Journal [2].

To my beloved
parents, brother and my lovely wife
without whom I would not have survived

Acknowledgements

First of all, I would like to express my sincere appreciation to Dr. Majid Khabbazian for his great support and supervision during this research project. This research and dissertation would not have been possible without his continuous guidance. Besides my supervisor, I am also thankful to my thesis committee members, Dr. Masoud Ardakani and Dr. Hai Jiang, for their comments and suggestions to improve my thesis. I would like to thank Dr. Reihaneh Safavi-Naini for serving as the external examiner for my final Ph.D. examination. I should also thank Dr. Chintha Tellambura for offering his time to read my thesis and agreeing to be an examiner for my final Ph.D. examination. I appreciate the advice of Dr. Ehab Elmallah who was my candidacy examiner and I was able to use his knowledge and further enhance my research. Also, I must thank all those whose actions, either directly or indirectly, have helped me to reach this point in my life.

I am incredibly grateful to my parents and brother for their love and spiritual support. Last but not least, I must express my deepest gratitude to my lovely wife for her love, encouragement, and support. Thank her for helping me to follow my dreams and believe in myself.

Contents

List of Tables

List of Figures

Acronyms

AODV Ad-hoc On-demand Distance Vector

ASLR Address Space Layout Randomization

CC Consistency Check

CSM Chained Secure Mode

CSMA Carrier-Sense Multiple Access

DAG Directed Acyclic Graph

DAO Destination Advertisement Object

DIO DODAG Information Object

DIS DODAG Information Solicitation

DODAG Destination Oriented Directed Acyclic Graph

DTSN DAO Trigger Sequence Number

IETF Internet Engineering Task Force

IoT Internet of Things

IPv6 Internet Protocol version 6

LLNs Low Power and Lossy Networks

MMU Memory Management Unit

MP2P Multi Point to Point

MRHOF Minimum Rank with Hysteresis Objective Function

OF Objective Function

P2MP Point to Multi Point

P2P Point to Point

PDR Packet Delivery Ratio

RDC Radio Duty Cycle

ROLL Routing Over Low Power and Lossy Networks

RPL IPv6 Routing for Low Power and Lossy Networks

UDGM Unit Disk Graph Model

UDP User Datagram Protocol

Chapter 1

Introduction

RPL (Routing Protocol for Low-Power and Lossy Networks) is the standard routing protocol for Low power and Lossy Networks (LLNs). LLNs are a key component of IoT, and are typically composed of various resource-constrained devices with limited energy, memory, and processing power [3]. RPL generates low control plane traffic and offers a range of interesting features for LLNs. For instance, RPL can cope with low-speed links and high transmission error rates and can adjust its traffic with the network's dynamics. RPL, however, has deficiencies with regards to security and point-to-point communications. In this thesis, we study these deficiencies and propose easy to implement solutions.

1.1 Motivation

1.1.1 RPL's Security

Recent research has shown that RPL is vulnerable to a wide range of attacks, including Denial-of-Service (DoS) attacks which can considerably degrade the network's performance and shorten the lifetime of nodes operating in the network. Security is one of the main concerns of RPL networks, particularly when these networks are deployed in critical infrastructures such as smart grid. Due to the multicast nature of transmissions and lack of tamper-resistant equipment in IoT devices, providing security for these networks is a complex task. In particular, mitigating security attacks in LLNs is not trivial because implementing security solutions such as digital signature can greatly degrade the performance of resource-constrained nodes. In the absence of security solutions

such as digital signatures, an internal attacker (e.g., a compromised node) may alter, inject, replay, and generate data or control messages to impact the regular operation of RPL networks [4], [5]. For instance, in the version number attack [6], a compromised node can initiate a whole network repair by sending a single control message on behalf of the root. Similarly, in our proposed attack (Chapter 3), a compromised node can send a control message on behalf of the root and trigger many nodes in the network to send redundant messages. Such attacks, generate many redundant transmissions, hence can significantly degrade the performance of the network, and reduce the lifetime of IoT devices that run on batteries. Consequently, it is necessary to design lightweight and efficient solutions to counter such attacks.

1.1.2 Point-to-Point Routing in RPL

RPL supports three main communication patterns: Multipoint-to-Point (MP2P), Point-to-Multipoint (P2MP), and Point-to-Point (P2P). MP2P and P2MP enable communications between the network's root (i.e., the sink) and the remaining nodes in the network, while P2P enables communications between any two nodes in the network. The main focus in designing RPL was on MP2P and P2MP communications (from the root), rather than P2P communications. As a result, RPL is not optimized for P2P communications. For instance, in RPL's basic mode of operation (RPL's common mode of operation) a P2P packet has to travel from the source all the way up to the root and then travel all the way back down from the root to the destination. This is normally a much longer path than the shortest path between the source and destination, as we will show in this thesis. This is not desired as P2P communications is an essential component in many emerging IoT applications such as building automation and remote control applications [7] where the IoT networks aim to control actions rather than collect data.

To improve P2P communications, RPL provides two "upgrades" over its basic mode of operation. 1) Multicast Destination Advertisement Object (MDAO), and 2) storing mode. The former solution allows a source node to send its data packets directly to the destination if the destination is within

the transmission range of the source. The latter solution allows the common ancestor of the source and destination (instead of the root) to redirect the source packet down towards the destination. Therefore, in the storing mode, a packet may not need to go all the way up to the root and then get redirected towards the destination. This solution requires non-leaf nodes to store a routing table, which can add pressure on nodes with limited memories. This is more harsh for nodes near the root as they have more nodes as their children. Therefore, it is important to devise a lightweight and efficient method to improve RPL's P2P routing.

1.2 Thesis Contributions

To tackle the problems mentioned in the previous section, we present three research works in Chapter 3 to Chapter 5. The contributions of these chapters are summarized as follows:

1.2.1 The DAO Induction Attack: Analysis and Countermeasure

Chapter 3 introduces and studies the DAO induction attack, our proposed novel attack against RPL. In the DAO induction attack, a compromised node in the network periodically transmits a special control message. Each of these crafted control messages induces many nodes in the network to transmit in response. We show that transmitting these unnecessary messages can significantly increase the power consumption of nodes, hence reduce the lifetime of battery-operated IoT devices. In addition, we show that the attack severely impacts end-to-end latency and packet delivery ratio, two important network performance metrics. For instance, in a network with 50 nodes, our simulation results show that the attack increases the average end-to-end latency and packet loss ratio by 410% and 260%, respectively. To counter the attack, we propose a lightweight solution. We show that our solution imposes no overhead when the network is in its normal operation (i.e., it is not under attack) and can quickly detect the attack even when the network experiences high packet

loss rates.

1.2.2 LBAM: A Lightweight Authentication Mode for RPL

Chapter 4 studies the authentication problem of the RPL protocol and proposes a new authentication mode for RPL using the Blom key pre-distribution scheme [8]. The main contribution of this work is that we show that the computational cost of the Blom scheme can be significantly reduced at the cost of slight increase in memory requirement. This makes the Blom scheme more suitable for computationally-constrained IoT devices. We compare the computation overhead of original Blom scheme with the proposed lightweight Blom for MSP430 CPU family and show that the computation overhead of the proposed scheme is much lower than original Blom scheme. Moreover, we formulate the attacker's cost for capturing nodes and show that there is no better algorithm other than brute-force.

1.2.3 P2P paths in RPL

We study the quality of RPL's P2P paths in Chapter 5. In particular, we analyze how much RPL's P2P paths "stretch" compared to the shortest paths. We prove that the average stretch is a factor of at least two in any RPL network. That is, the RPL's P2P path between two randomly selected source and destination nodes in any network is expected to be at least twice as long as the shortest path between the two nodes. Furthermore, we show that RPL's stretch factor can be considerably higher than two in some network topologies including linear networks and grid networks. To improve the quality of RPL's P2P paths, we propose a solution which is simple to implement and fully compatible with RPL. Moreover, our solution does not require nodes to store any routing table, which is important as nodes in LLNs are typically highly resource-constrained. We evaluate our proposed solution using the Contiki-NG operating system and its built-in Cooja emulator, and show that our proposed solution can significantly improve the quality of RPL's P2P paths with a modest overhead.

1.3 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 provides the necessary background on Low Power and Lossy networks, RPL and key management techniques. The next two chapters deal with RPL's security, and propose new mitigations. In particular, Chapter 3 introduces DAO induction attack and proposed a lightweight mitigation technique, and Chapter 4 proposes a new authentication mode for RPL based on well known Blom key predistribution scheme. Chapter 5 analyzes P2P routing efficiency in RPL and provides a lightweight method to enhance P2P routing in RPL. Finally, Chapter 6 concludes the thesis and presents possible extensions to these works, and provides potential directions for future works.

Chapter 2

Background

2.1 Internet of Things and Wireless Sensor Networks

Internet of Things (IoT) and Wireless Sensor Networks (WSN) share certain characteristics with each other [9], [10]:

- In both networks, nodes are usually limited in terms of processing power, battery and memory.
- Both networks suffer from unstable connectivity. Devices may turn off due to limitation in power. Communication links between devices are lossy and unstable.

Despite these similarities, IoT and WSN are not the same and have several differences. First, the traffic pattern in WSN is mostly Multiple Point to Point (MP2P); WSN originally designed for data collection without any smartness at nodes. Furthermore, nodes send data to specific sink nodes and the data is processed outside of the network. IoT, however, introduces intelligence to the devices by delegating certain decisions to them. In addition, IoT devices may communicate directly with each other to coordinate actions.

The second major difference is that IoT devices use the IP protocol stack. This means that each device can directly communicate with any Internet end system. Also, routing protocols in IoT are based on IP addresses. This is not the case in WSN. In WSN, routing protocols usually use the location of nodes

and the way they are located with respect to each other to route a packet to the sink [9].

2.1.1 General Security Issues in Wireless Sensor and Ad-hoc Networks

Security is one of the main concerns in Ad-hoc and wireless sensor networks. These networks are susceptible to security attacks that impact the following security attributes:

- Availability: this feature guarantees network services despite the denial of service attacks. An attacker can perform a denial of service attack in various layers of the network to disrupt the service. For instance, an adversary may apply jamming to make interference for communication between nodes in the physical layer. In the network layer, a malicious node could disrupt the routing services and disconnect nodes from the network. A denial of service attack usually concerns battery exhaustion as devices are usually resourced constrained with limited power. For example, in a sleep deprivation attack, an attacker prevents a node from turning off its radio to save energy. As a result, nodes consume more energy, and their battery is exhausted faster.

- Confidentiality: this attribute keeps certain information private from unauthorized entities. Some network transmissions, depending on applications, contain sensitive information. If an attacker gets access to this information, it can use them to establish strong attacks against networks. For example, routing messages can provide valuable information to an attacker to identify and locate her targets.

- Integrity: this ensures that a message is delivered to the destination without any change. This is very important for routing protocols. An attacker may alter routing messages to downgrade the routing performance or even disconnect nodes from the network.

Table 2.1: Classes of constrained devices in Low Power and Lossy Networks (1 KiB = 1024 bytes) [12].

Device Categories	RAM (data size)	ROM (code size)
Class 0	$<$ 10 KiB	$<$ 100 KiB
Class 1	$\approx$ 10 KiB	$\approx$ 100 KiB
Class 2	$\approx$ 50 KiB	$\approx$ 250 KiB

- Authentication: this allows a node to verify the identity of other nodes in the network. In a network without authentication an attacker can easily use fake identities or steal legitimate identities of other nodes to gain unauthorized access to resources and information.

Ad-hoc and WSN networks use wireless links which are susceptible to attacks such as eavesdropping, message replay, and message distortion. These attacks can violate all the attributes mentioned above. Second, nodes in these networks usually are not tamper-resistant. Therefore, in addition to external security threats, we must take into account the attacks that launched from inside of the network. Finally, the topology of these networks can be changed. Therefore, the relation between nodes may change over time. This can cause problems for security solutions that only work within static networks.

2.2 Low Power and Lossy Networks

Low Power and Lossy Networks (LLNs) are networks in which nodes are highly resource constrained, and communication links are unstable with relatively high loss rates and low data rates. LLNs are a key component of IoT and have many applications including industrial monitoring, building automation (e.g. heating and lightning), asset tracking, smart agriculture, and eHealth [11]. Devices in LLNs can be placed into three categories as shown in Table 2.1.

2.2.1 Challenges of Routing in LLNs

LLNs are different from the traditional IP networks. Routers in LLNs are highly constrained in terms of battery, memory, and processing power. LLN

devices are connected through different communication mediums including wired and wireless. LLNs networks can compromise thousands of nodes [13], and must support different types of traffic patterns. High data rate traffic can easily congest the network and increases the packet loss and latency. A routing protocol for LLNs must handle and cope with all these limitations and challenges.

Routing Over Low Power and Lossy Networks (ROLL) is a working group of Internet Engineering Task Force (IETF) that works on the routing topics of Internet of Things. One of the main tasks of ROLL was to investigate and analyze the existing routing protocols and define fundamental routing requirements for LLNs. The group limited the scope of LLNs application into four major categories: urban network, building automation, industrial automation, and home automation. They chose these applications as a set of representative networks, and assumed a routing protocol that addresses the requirement of these applications is deemed to be a good choice for other networks. Considering the above applications, the ROLL group established four routing requirements for LLNs.

- Home automation routing requirements in LLNs [14].
- Industrial routing requirements in LLNs [15].
- Routing requirements for urban LLNS [16].
- Building automation routing requirements in LLNs [17].

The above documents did not make any assumption about link layer protocol; they only determined a list of requirements for the network layer of LLNs. Some of these requirements are as follows:

- A routing protocol must support unicast, anycast, and multicast communication to support three main traffic patterns. Multi Point-to-Point: traffic from several nodes to a single sink node, Point-to-Point: traffic between any pair of nodes, and Point-to-Multipoint: routing traffic from a single node to several nodes in the network.

- A routing protocol must support adaptive routing, that is a new path must be dynamically and automatically recomputed if the network condition changes. In addition, a routing protocol must support different metrics to find routes.

- A routing protocol must support constrained devices.

- Scalability: number of nodes in an LLN may vary from 250 nodes in home automation [14] to up to $10,000$ in urban applications [16]. A routing protocol for LLNs must support all such networks.

- Security is very important in many LLNs applications such as smart grid, building automation, and industrial automation. In particular, authentication is listed as a mandatory feature in all documents.

The IETF ROLL group investigated the current and the state-of-the-art routing protocols to find whether any existing routing protocols can satisfy LLNs requirements [18]. They considered several routing protocols including link state protocols: OSPF [19], IS-IS [20], OLSR [21] , OLSRv2 [22], TBRPF [23], and distance vector protocols: AODV [24], RIP [25], DSR [26], DYMO [27]. The ROLL group analyzed these routing protocols using the following metrics:

- **Node cost,** which is the ability of a protocol to integrate router properties into routing metrics and uses node features for constraint-based routing.

- **Control cost,** which indicates the efficiency of a routing protocol in terms of controlling traffic power consumption.

- **Link cost,** which evaluates the ability of a protocol in term of integrating link properties into routing metrics.

- **Routing state,** which shows whether a routing protocol scales reasonably with regards to memory.

Table 2.2: Protocol comparison.

Routing protocol	**Routing state**	**Loss response**	**Control cost**	**Link cost**	**Nose cost**
OSPF	fail	fail	fail	pass	fail
IS-IS	fail	fail	fail	pass	fail
AODV	pass	fail	pass	fail	fail
DSR	fail	pass	pass	fail	fail
RIP	pass	fail	pass	NA	fail
TBRPF	fail	pass	fail	pass	NA
DYMO	pass	NA	pass	NA	NA
OLSRv2	fail	NA	NA	pass	pass

- **Loss response,** which measures the performance of a routing protocol in terms of handling link failures and recomputing paths.

Table 2.2 shows the result of evaluation of these routing protocols [18]. The value 'NA' indicates that the protocol does not have that feature so the ROLL group could not conclude if the test was successful.

As shown in Table 2.2 none of the existing routing protocol could satisfy the requirements of LLNs. Therefore, ROLL proposed a new distance-vector routing protocol called RPL (IPv6 Routing Protocol for Low power and Lossy Networks), which is specified in the standards document RFC 6550 [13].

2.2.2 Link State vs Distance Vector Routing

There are two general routing algorithms: link-state and distance-vector. In link-state algorithms, each node maintains a global view of the network topology in addition to a cost for each link in the network. Each node finds the best path to other nodes using its own view of the topology. To maintain an up-to-date topology information, each node periodically sends link costs of its own links to other nodes. In link-state protocols each node must have $O(N^2)$ space to maintain topology information, where N is the number of nodes in the network.

In distance vector protocols, on the other hand, nodes do not have a global

view of the network. Instead, each node maintains the distance and the next hop towards each destination. In distance vector routing, information are advertised as vector of distance and direction (next hop). The memory requirement of distance-vector routing protocols is $O(N \times e)$, where e is the average degree of nodes (i.e., the average number of neighbours of nodes).

Although link-state routing protocols are more powerful in terms of finding the best path between any pair of nodes and updating the routing information in case of a change, they require a significant amount of resources such as memory and control messages to synchronize routing information. Because of limited resources, therefore, link-state routing protocols are not suitable for LLNs. Consequently, ROLL used the distance-vector approach in designing the RPL routing protocol for LLNs.

2.3 Overview of the RPL Protocol

RPL is a distance-vector routing protocol, which can operate on various link layer standards, including Bluetooth and IEEE 802.15.4 [28]. RPL was designed to be very adaptive to different network conditions and do not overreact to minor changes in the network. This section overviews the operation of RPL protocol [13]. Before delving into the RPL protocol operation, let us first cover some terminologies and introduce some of the RPL's main components.

2.3.1 RPL Terminology

DODAG

RPL builds its topology in one or more Destination Oriented Directed Acyclic Graph (DODAG), a loop-free network topology, as shown in Fig. 2.1.

DODAG Root

Each DODAG has a single root (a node with no outgoing edges), which acts as the data sink. The root controls several parameters over the network which are advertised over the network using control messages. These parameter include Trickle Timer options, Path control size, MinHopRankIncrease, and DODAG

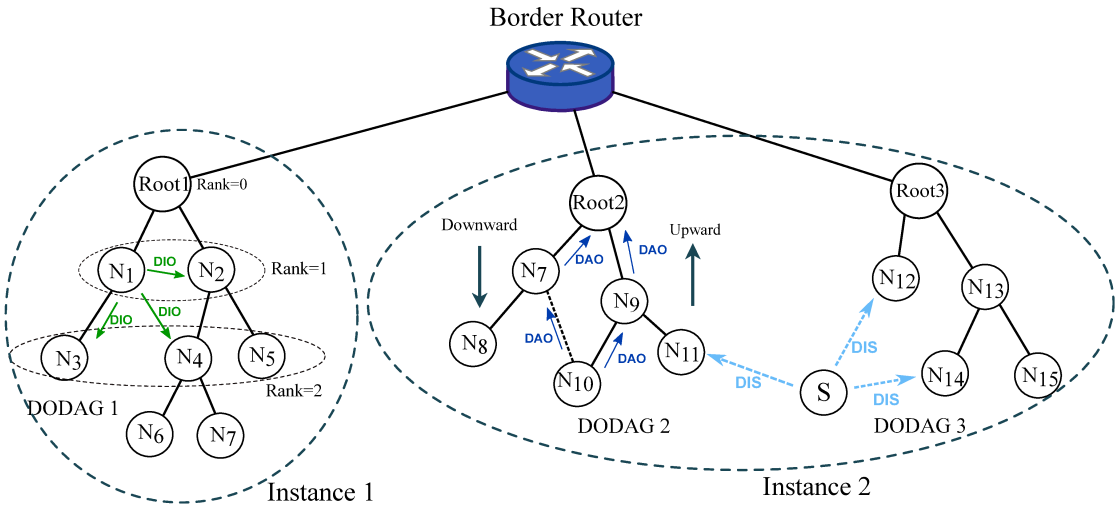

Figure 2.1: An example of RPL network with two instances and three DODAGs.

preference field. The root also assists the multicast packet by acting as a proxy point for the RPL nodes and connects them to non-RPL networks [13].

RPL Instance

A set of RPL DODAG together create a RPL instance and share a single RPL instance ID. Each RPL instance works independently of other instances and may have different objective functions. An RPL network may maintain several instances of RPL at the same time. A single RPL node may belong to multiple RPL instances. However, each node can belong to only one DODAG at the same time.

Rank

A node's rank is a number that determines the node's position with respect to the root. Nodes ranks increase as they go farther from the root. Nodes use rank to detect loops in DODAG, and select parents.

Version number

A DODAG version number shows the iteration of a specific DODAG. The DODAG root increments the version number counter to rebuild a new version

of DODAG. A DODAG Version is identified uniquely by the (RPLInstanceID, DODAGID, DODAGVersionNumber) tuple.

Parent

A parent of a node in a DODAG is the next node on a route towards the DODAG root. In a DODAG, a node's rank is always higher than the rank of its parent.

Sibling nodes

Two nodes are called siblings if they have the same rank and are withing the transmission range of each other. Note that sibling nodes do not necessarily share a common DODAG parent.

Upward and downward routes

In RPL, upward routes are from leaf nodes toward the DODAG root. Upward routes provide MP2P communication. In RPL, the upward routes are created via nodes' preferred parents: each RPL node has a set of neighbours in its one-hop distance. From this set, the node selects a group of nodes that have strictly smaller rank than its own rank. This group is called *candidate parents*. From this group, the node chooses one as a preferred parent to convey traffic toward the DODAG root.

In contrast, downward routes refer to the reverse direction. These routes are used for P2MP or P2P communications.

Modes of operation

RPL supports two modes of operation for downward traffic: storing mode and non-storing mode. In the storing mode, all nodes except the root and leaf nodes store routing table entries for destinations. In this mode, a P2P packet sent from a source node to a destination node can be directed by the common ancestor of two nodes. In the non-storing mode, only the DODAG root stores a routing table. In this mode, a P2P packet must travel upward all the way

to the DODAG root where it is directed downward towards the destination node.

Objective Function (OF)

An objective function specifies essential settings such as routing metrics, optimization objectives, rank calculation formula, and parent selection criteria in RPL DODAG [13]. RPL standard defines objective functions to accommodate different IoT applications for the following reasons:

- To choose the best DODAG to join.
- To compute the rank of each node in DODAG.
- To create a parent set and select a preferred parent.

Currently, RPL has two standard objective functions:

- Objective Function Zero (OF0) [29]: this OF uses hop count. Therefore, the rank of a node calculated based on OF0 is basically the hop-distance of the node from the root. OF0 is the default objective function to ease the inter-operation of different implementations of the RPL protocol [30].
- Minimum Rank with Hysteresis Objective Function (MRHOF) [31]: this OF uses hysteresis for choosing the path with smallest metric. The metric that MRHOF uses is determined by the metric container field of DIO messages. By default the MRHOF uses Expected Transmission Count (ETX) to compute Rank. ETX of the link is defined as the expected number of transmission required to successfully transfer a single packet over a link [29].

RPL is flexible with regards to OFs. Designers can modify the existing OFs or design new ones that better fit their application.

2.3.2 RPL Control Messages

RPL protocol has five main control messages. These messages are used to create, maintain, and propagate the network topology and routing information. RPL's control messages are:

1) DODAG Information Object (DIO)

DIO messages are sent by RPL nodes to advertise information about the DODAG and its characteristics. DIO messages are used to form the DODAG topology, find other nodes in the network, and maintain DODAG. DIO conveys several mandatory as well as optional information. The mandatory information of the DIO is: RPLInstanceID to identify the current RPL instance, DODAG's ID, DODAG's version number, RPL's modes of operation, the rank of sender, and DODAG configuration such as objective function. The base structure of the DIO message is shown in Fig. 2.2.

2) Destination Advertisement Object (DAO)

The DAO message is used to propagate downward routing information along the DODAG to provide support of P2MP and P2P traffic. A DAO message includes the following information:

- DAO sequence: an integer counter that is incremented by the sender whenever a new DAO is sent.
- RPLInstanceID: it shows the ID of the current RPL instance. It is learned from DIO message.
- Path information: it indicates addresses of nodes that are reachable via the sender of the DAO message. A node uses this information to create and update its routing table.

When RPL is in the non-storing modes, nodes send DAO messages to the root. In the storing mode of operation, a node sends DAO messages to its preferred parent instead of the root. An special case of DAO message is No-Path DAO. This message is used in the storing mode to remove a downward route from the routing tables of other nodes.

3) Destination Advertisement Object Acknowledgment (DAO-ACK)

The recipient of a DAO message may send a DAO-ACK. The DAO-ACK message is a unicast message that is sent in response to a unicast DAO message.

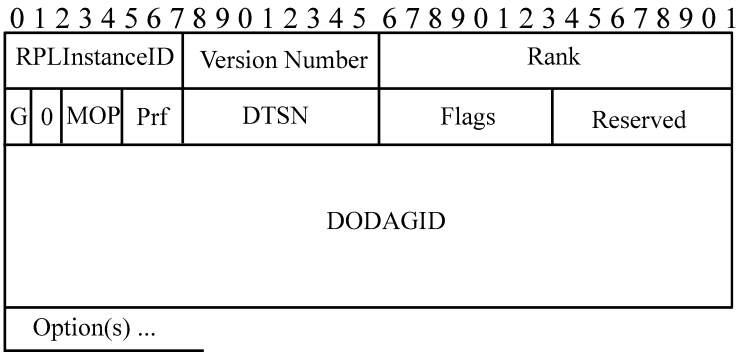

Figure 2.2: DIO base object.

4) DODAG Information Solicitation (DIS)

The DIS message is used to request information about the RPL DODAG from other RPL nodes. In RPL when a node wants to join the network it must receive a DIO message. If the node does not receive any DIO messages for a while it may request a DIO message by sending a DIS message. A node may also may use a DIS message to probe its neighbours in other DODAGs. A node can send a unicast or a multicast DIO message. The network reacts differently to unicast and multicast DIS messages. If a node receive a unicast DIS, it only replies to the sender of DIS with an updated DIO message. If a node receives a multicast DIS, however, it resets its trickle timer (see 2.3.3) and broadcast a DIO to all its neighbours.

5) Consistency Check (CC)

The CC message is used to verify counters of secure messages and perform a challenge response action. For example, if replay protection is enabled in RPL, then when a new node joins the DODAG other nodes can sends CC message to the recently joined node to synchronize its security counters with the network. By receiving a CC message, the new node updates its incoming security counters to a number greater than those in received messages. Also, by replying to CC messages, the requesting nodes update their outgoing counters.

2.3.3 Trickle Timer

RPL uses an algorithm called Trickle timer to decrease the overhead of its control messages in the network [32]. Following to the Trickle timer, nodes send control messages when there is an inconsistency in the received control messages. If a node receives a DIO message from one of its neighbours, and the message is consistent with the node's information, it increments a "redundancy counter" by one. If the value of redundancy counter is greater than a threshold in a specific listening interval, the node suppresses its own DIO message and does not broadcast it. Furthermore, the node doubles its listening interval. If a node receives an inconsistent DIO message, however, it resets its listening interval and redundancy counter to their initial values and broadcasts a DIO message. This way, the Trickle timer algorithm tries to avoid transmitting control messages when they are not necessary.

The Trickle timer algorithm has three main components:

1. The minimum interval size (I_{min}) which sets the minimum length for listening intervals.

2. The maximum interval size (I_{max}) which indicates the largest listening interval. Therefore, the maximum number of times a listening interval can be doubled is $\log_2(I_{max}/I_{min})$.

3. An integer that shows the redundancy threshold.

The Trickle timer's parameters can be changed based on network's application and condition.

2.3.4 DODAG Building Process

RPL builds a DODAG step by step. To this end, nodes that are already in the network (initially just the root) periodically transmit DIO messages according to Trickle timer. If a new node wants to join the network, it should receive a DIO message to obtain DODAG information. If the new node does not receive a DIO message, it can send a DIS message to request DODAG information.

When an existing node in the network receives a DIS message, it replies by transmitting a DIO message.

After receiving a DIO message from a lower rank node, the receiving node adds the sender address of the DIO message to its candidate parent set and calculates its rank with respect to the new candidate parent. The candidate parent that results in the lowest rank is selected as the node's preferred parent. At the end of this procedure, each node has an upward path towards the root (through its parents).

A node that wants to be reachable by the root advertises its address in a DAO message and sends it to one of its DAO parents. The course of action taken by the node's DAO parent depends on the RPL's mode of operation. In the storing mode, the receiving node first updates its routing table and then forwards the DAO message to its own parent. In the non-storing mode, the receiving node only forwards the DAO packet toward the root (in this mode, the receiving node does not have a routing table to update).

2.3.5 Loop detection and avoidance mechanisms

Routing loops can dismantle the network operation. The RPL protocol can detect loops and inconsistencies and repair the topology. In a DODAG, a loop may occur for various reasons. For example, a loop may occur if DIO messages are lost, or if a leaf node fails to inform its parent that a specific destination is not reachable anymore. Also, a loop can occur if an RPL node changes its position within the network.

Loop avoidance mechanism

RPL defines two rules to prevent loops from occurring: max-depth rule and anti-greedy loop. The max-depth rule states that a node must not select a node with a rank higher than the its rank + DAGMaxRankInrease as a parent. This rule decreases the chance of a node to attach to another node in its own sub-DODAG and create a loop. The max-depth rule cannot completely avoid loops from occurring, but it can confine loop sizes and ease the detection of loops.

The second rule, anti-greedy, does not allow a node to increase its rank and go deeper (i.e., farther from the root) to have more parent choices. If a node increases its rank in a greedy manner, it can create loops and instability in DODAG.

2.3.6 RPL Security Modes

RPL provides the following security modes:

- *unsecured mode:* In this mode, RPL does not provide a security measure. It, however, may utilize the link-layer security to protect its messages.
- *pre-installed mode:* In this mode, security keys are manually installed on nodes before they are deployed in the network. Nodes use these pre-installed keys to secure RPL messages.
- *authenticated mode:* similar to the previous mode, nodes have pre-installed keys, but these keys are used only for authenticating the nodes who want to join the network as a leaf. All nodes except the leaf nodes must obtain a second key from an authentication authority after joining the network.

2.4 RPL Security Concerns

Although RPL provides two security modes, it is prone to several security attacks. We can divide the RPL attacks into two major categories with respect to the state of the adversary in the network: internal attacks and external attacks. The latter refers to attacks where the adversary does not have a valid security key, hence cannot join the RPL DODAG. The former attacks, on the other hand, refer to those where the adversary has a valid security key and can participate in RPL. Recall that RPL relies on its pre-installed mode (where keys are pre-installed in nodes) for its security.

The attributes of LLNs, such as limited physical security and resource constraints, make it easy for an adversary to breach the security and join the network. As a result, many proposed RPL attacks are internal attacks. Some of these attacks such as Blackhole attacks, sinkhole attacks, wormholes, and

flooding attacks, are inherited from ad hoc and sensor networks. There are, however, many attacks that are exclusive to RPL. These attacks typically misuse RPL control messages and parameters to disrupt the routing services. Some of the well-known and most important attacks in this category are Version number attack [6], Rank attack [33], DAO insider attack [34], and DIO suppression attack [35].

The security solutions and mitigation mechanisms in RPL fall into two major classes. The first class includes those that use an intrusion detection system (IDS). IDS methods are designed to detect different attacks and mitigate their impacts on the network [33]. IDS solutions continuously monitor the network for attacks. If one found, they try to identify the attacker and limit the impact of the attack. We can classify the IDS in RPL into four classes based on their detection strategy:

- Signature based IDSs: this family of IDSs create a distinct pattern for each security attack based on the attack's characteristics and its impact on network behavior, such as control messages overhead, energy consumption, and delay [4]. The caveat of the methods in this class of IDSs is that they cannot detect an attack which has not been registered before.

- Anomaly based IDSs: the main detection strategy in this category of IDSs is to compare the network's behavior to its normal behavior. An anomaly based IDS creates a normal behavior profile for the network using the network's statistics. If the behavior of the network is different from this normal profile, it raises an alarm for a possible attack. This family of IDSs has typically a large false-positive and false-negative as defining an accurate normal profile for the network is not straightforward.

- Specification based IDSs: this type of IDSs are similar to the anomaly based IDSs. The distinguishing factor is that they define the normal profile manually and based on the network protocol parameters. An example of an IDS in this class is SVELTE [36] proposed by Raza et al..

- Hybrid IDSs: methods in this class combine two or more of previous techniques to improve the performance.

The second class of solutions for securing RPL are those who target a specific attack. These solutions typically introduce new control messages, parameters, thresholds, or cryptography solutions to either completely prevent the attack or limit the attack's impact. For example, authors in [37] proposed a hash-chain technique to prevent Version number attack in RPL. In Section 3.7, we explain the existing RPL attacks and the possible countermeasures/mitigations.

2.5 Key Management Schemes

Message encryption and authentication, hence security keys, are necessary to secure RPL networks. There are three main security key agreement methods in general: trusted-server, self-enforcing, and key pre-distribution [38]. Trusted-server methods use a trusted server to provide keys to nodes in the network. Although RPL standard proposes the authenticated mode to support this type of key management, this mode is currently reserved for future work. As of September 2021, there is no official document that explains the procedure of this mode. The RPL standard specifies that the authentication mode must not use symmetric cryptography. However, it does not mention how asymmetric cryptography could be deployed to provide node authentication and support a key retrieval by the node intending to operate as a router. The methods of the second type that is self-enforcing methods, are based on asymmetric cryptography such as Diffie-Hellman [39] and RSA [40]. Asymmetric methods are not desirable for RPL devices because they are computationally heavy. Finally, in key predistribution methods, security keys are distributed to nodes before they are deployed in the network. This kind of key management is completely compatible with the pre-installed mode of the RPL protocol and is suitable for LLNs' devices. There are three main sub-categories for key predistribution techniques: network shared key, full pairwise key, and random key.

Network-Wide Shared Key

As the name suggests, in this method all nodes in the network use a single shared key. This key is used by every node in the network to encrypt and decrypt messages. This simple method has certain advantages: 1) it requires minimum memory as each node needs to store only a single key, 2) it does not need any key agreement procedure to run between nodes in order to find common keys. On the negative side, the network-wide shared key method has minimum resiliency against node captures; if an attacker captures and compromises a single node, the security of the entire network is compromised. Furthermore, revoking the access of a compromised node is challenging since all nodes, except the compromised node, must receive a new key.

Fully Pairwise Scheme

In this method, each node has a distinct security key for every other node in the network. This method provides a full level of authentication and maximum resiliency against node compromises. If an attacker compromises a node, it cannot gain any information about keys between non-compromised nodes. The fully pairwise key scheme, however, needs a large memory because each node must store $N - 1$ keys in its memory, where N is the number of nodes in the network. In addition, when a new node joins the network, key sets of all the existing nodes must be updated, which is not an easy task.

Random Key Pre-distribution

In this scheme, each node is pre-loaded with a random subset of keys (key ring set) from a large key pool. After deployment in the network, any two nodes can use the collection of keys that they share (if there is any) to establish a pairwise key. The size of key pool and the key ring set are set such that two randomly selected nodes would have at least one shared key with a given probability. Random key pre-distribution schemes can be viewed as a trade-off between network-wide shared key and fully pairwise schemes: they require less memory than the fully pairwise key scheme, and are more resilient to node

capture than the shared key scheme.

Chapter 3

The DAO Induction Attack: Analysis and Countermeasure

3.1 Overview

In this chapter we present a new attack against IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), the emerging routing standard for the Internet of Things (IoT). In the proposed attack, a compromised node periodically transmits a special control message. Each of these crafted control messages induces many nodes in the network to transmit in response. We show that these redundant transmissions significantly increase the power consumption of nodes, hence reduce the lifetime of battery-operated IoT devices. In addition, we show that the attack severely impacts end-to-end latency and packet delivery ratio (PDR), two important network performance metrics. For instance, in a network with 50 nodes, our simulation results show that the attack increases the average end-to-end latency and packet loss ratio by 350% and 160%, respectively. To counter the attack, we propose a lightweight solution. We show that our solution imposes no overhead when the network is in its normal operation (i.e., it is not under attack) and can quickly detect the attack even when the network experiences high packet loss rates.

3.2 Adversary Model

As shown in Fig. 3.1, we consider an RPL multi-hop network with a single root. We assume that RPL uses a shared secret key (at the link-layer or by

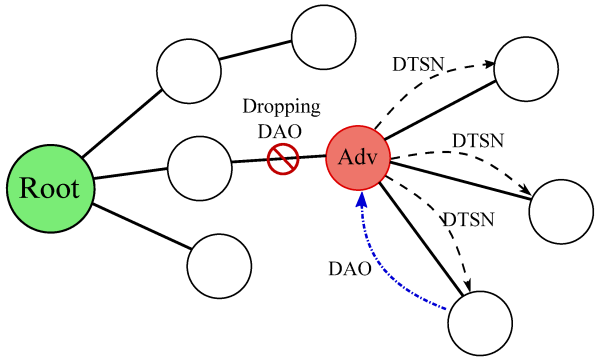

Figure 3.1: The compromised node (red) increases its DTSN number, which triggers its three descendants to transmit DAO updates. The compromised node can drop these DAO updates to prevent the root from receiving them.

itself in the pre-installed mode) to secure its messages. Since all nodes use the same secret key, they cannot authenticate the root's control messages. We assume that the adversary controls a single insider node (e.g. a compromised node). RPL networks are mainly composed of constrained devices such as sensors and actuators, which are low cost and non-tamper resistant. An attacker can, therefore, compromise an IoT device through either physical access, or remote access using hardware and/or software vulnerabilities such as week or default access credentials. Accessing and compromising IoT devices are easier than PCs as they are always online, have no antivirus installed, often use weak credentials [41], and are often too constrained to implement complicated security solutions such as public key cryptography, and memory protections such as memory management unit (MMU), and address space layout randomization (ASLR) [42]. After compromising an IoT device, the attacker can inject its malicious code (in our case the code to execute the DAO induction attack) and launch the attack.

We refer to the node controlled by the adversary as the malicious node. The malicious node can be any node in the network except the root. In this work, we limit the malicious node's misbehavior to 1) running the DAO induction attack (explained next), and 2) selectively dropping DAO packets to, for example, avoid detection by the root. Attacks combining the DAO induction attack with other existing ones can be more powerful and lie outside

the scope of this work. Fig. 3.1 illustrates the above adversary model.

3.3 The DAO induction Attack

In RPL, each node maintains a DAO Trigger Sequence Number (DTSN) and reports it in its DIO messages. If a node receives a DIO message from one of its DAO parents and realizes that the parent has incremented its DTSN, the node must schedule a DAO transmission [13]. In the non-storing mode, the node must also increment its own DTSN. Therefore, in the non-storing mode, a DTSN increment by a node will cause all its descendants to increment their DTSN in turn, triggering DAO transmissions from the entire sub-DODAG [13].

In the DAO induction attack, a malicious node repeatedly increases its DTSN to trigger DAO transmissions. This can cause many transmissions, particularly in the non-storing mode (a common mode of operation as many IoT devices are too constrained to operate in the storing mode [13]) as all descendants of the malicious node transmit each time the malicious node increments its DTSN.

For example, consider the network shown in Fig. 3.2. Assume that the network operates in the non-storing mode. In this mode, if node A increments its DTSN, every other 11 nodes in sub-DODAG1 will increment their DTSN in turn and send a DAO message to the root. Note that sending each of these DAO messages to the root requires several transmissions, as each node on the path to the root has to forward the message.

In Fig. 3.2, every node has only one DAO parent (bold lines show parent child connections). If some nodes decide to select more than one DAO parent, the DAO induction attack may trigger even more nodes to transmit a DAO message. For instance, suppose that node L chooses two nodes, P, and DP, as its preferred parent and DAO parent, respectively. Then, a DTSN increment by A would trigger not only all the nodes in sub-DODAG1 but also L and all its six descendants. The number of triggered nodes, in this case, accounts for about two-thirds of all the nodes in the network.

We remark that the DTSN counter is an 8-bit unsigned integer, so it has a

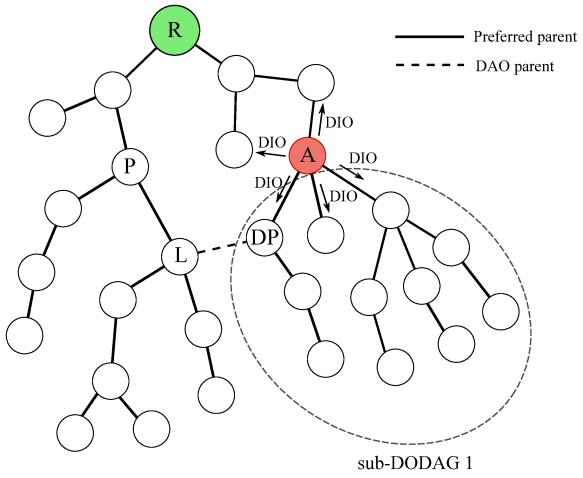

Figure 3.2: An example of a DODAG under the DAO induction attack by node A (red node). R (green node) is the root of the network. Node "P" is the preferred parent and node "DP" is the DAO parent of node "L", respectively.

limited range. However, this limitation does not restrict the number of times a malicious node can update DTSN in a DAO induction attack. This is because sequence numbers in RPL operate according to a "lollipop fashion" [13], where an increment of a sequence number with the maximum value will wrap the number back to zero. Therefore, the number of times an attacker can increment DTSN is practically unlimited.

Similar to the version number and DAO insider attacks, we can mitigate the DAO induction attack by enabling security mechanisms at the link-layer or at RPL itself. However, these mechanisms are ineffective when the attacker is an insider or a compromised node.

3.4 The number of triggered nodes

In the non-storing mode, the attacker can trigger all its descendants to transmit a DAO message. The number of descendants of a node depends on several factors, including the node's location in the network. For example, in Fig. 3.2, node A and L have 11 and 6 descendants, respectively. The number of descendants also depends on the number of parents that nodes select. In Fig. 3.2, if node L chooses node DP (in addition to node P) as a parent, the num-

ber of descendants of A increases from 11 to 18, as in this case L and all its descendants become descendants of node A.

To get an idea of how many nodes the attack may trigger, consider a rectangular network with a root placed at a fixed location. Suppose N nodes are distributed in the network uniformly at random. In addition, suppose that the malicious node is selected uniformly at random from one of the root's neighbours. Note that root's neighbours have the maximum number of descendants among non-root nodes. The following proposition establishes a lower bound on the expected number of nodes that the attack triggers.

Proposition 3.1 *Let δ denote the node's density, Δ be the portion of the root's transmission area that is within the network's rectangular area, and Γ denote the number of nodes triggered by the attack. Then, we have*

$$\mathbb{E}[\Gamma] \geq \left(\frac{\eta}{\Delta \cdot \delta}\right) \cdot N - \eta,$$

where η is the average number of node's ancestors among the root's neighbours, and the average is taken over all the nodes that are neither root or root's neighbour.

Proof. Let k denote the number of neighbours of the root, and m_i, $1 \leq i \leq k$, denote the number of descendants of root's ith neighbour. Let u be a node, and A_u denote the number of u's ancestors among the root's neighbours. Then, η is the average of A_u, where the average is taken over all the nodes that are neither root or root's neighbour. We have

$$\eta \geq 1,$$

because every node that is not the root or its neighbour is a descendant of at lease one of the root's neighbours. By the definition of η, we have

$$\sum_{i=1}^{k} m_i = \eta \cdot (N - k). \tag{3.1}$$

Since the malicious node is selected from the roots' neighbours uniformly at random (i.e, with probability $\frac{1}{k}$), the expected number of descendants of the

malicious node is

$$\begin{aligned}\mathbb{E}\left[\frac{\sum_{i=1}^{k} m_i}{k}\right] &= \mathbb{E}\left[\frac{\eta \cdot (N-k)}{k}\right] \\ &= \eta \cdot \mathbb{E}[\frac{1}{k}] \cdot N - \eta,\end{aligned} \tag{3.2}$$

where the first equality is by (3.1). By Jensen's inequality, if X is a random variable and ϕ is a convex function, then

$$\mathbb{E}[\phi(X)] \geq \phi(\mathbb{E}[X]).$$

Note that k is a random variable. Thus, setting $\phi(X) = \frac{1}{X}$ and $X = k$, we get

$$\mathbb{E}[\frac{1}{k}] \geq \frac{1}{\mathbb{E}[k]}. \tag{3.3}$$

The size of the root's transmission area πR^2, where R is the root's transmission range. Let Δ be portion of the root's transmission area that is within the network's rectangular area. We have

$$\Delta = \frac{\pi R^2}{\kappa},$$

where $\kappa \geq 1$ depends on the location of the root. For instance, if the root is located on the boundary of the network, then $\kappa \geq 2$ because at most half of the transmission area of the root falls inside the network. If the root is located at the corner of the network, we get $\kappa = 4$ as, in this case, one quarter of the root's transmission area would fall inside the network rectangular area. The expected number of neighbours of the root is equal to $\Delta * \delta$, that is

$$\mathbb{E}[k] = \Delta \cdot \delta.$$

Therefore, by (3.2) and (3.3), we get

$$\begin{aligned}\mathbb{E}\left[\frac{\sum_{i=1}^{k} m_i}{k}\right] &\geq \frac{\eta}{\Delta \cdot \delta} \cdot N - \eta \\ &= \left(\frac{\kappa \cdot \eta}{\pi R^2 \delta}\right) \cdot N - \eta\end{aligned}$$

□

Example 1 *Consider a network where 100 nodes with transmission range 25m are distributed uniformly at random in a* $100m \times 100m$ *square area. Suppose the root is located at the corner of the network, and each node chooses two DAO parents in addition to its preferred parent. Then, by Proposition 3.1, we get* $\mathbb{E}[\Gamma] \geq 61$*, that is, at least* 61% *of all nodes are expected to be triggered by the DAO induction attack.*

As mentioned in Proposition 3.1, the value of η is at least equal to one. In the particular case where each node selects precisely one parent, we get $\eta = 1$. In general, the more parent nodes select, the higher the value of η. One way to look at this is that when nodes select more parents, they are more likely to become a descendant of the malicious node. In Section 3.6, we confirm the result of Proposition 3.1 for various settings.

3.5 Our proposed Countermeasure

The DAO induction attack is significantly more severe in the non-storing mode than the storing mode (we will confirm this in Section 3.6). Therefore, it is more important to mitigate this attack in the non-storing mode. In this mode, similar to the version number attack, we can avoid the DAO induction attack by authenticating the root's messages. To support this authentication, however, we need to rely on either digital signatures or hash chains. As mentioned in Section 3.5, these two methods may not be practical and/or desired in networks with constrained devices. To address this, we propose a lightweight, reactive solution to detect and mitigate the DAO induction attack. An important feature of our solution is that it does not impose an overhead on the network when it is not under an attack.

Our proposed solution works in two phases: detection and identification. In the detection phase, we use the root to detect the onset of the DAO induction attack. Upon detecting the attack, our solution moves to the identification phase, where the root searches the network for the compromised node so it can be removed from the network. We remark that our solution relies on the

root and does not require any external monitoring nodes to be deployed in the network. In designing our solution, we consider the following objectives:

1. detect the DAO induction attack with high probability,
2. impose minimal or no extra overhead under the normal condition when there is no attack in the network, and
3. be compatible with the current RPL specification and require minimum changes to the standard.

3.5.1 Detecting the Attack

Without any provision, the root may not be able to detect the DAO induction attack. This is particularly the case when nodes select a single parent, the default setting in the Contiki operating system[43], and many IoT networks. This is because, in this setting, the malicious node can drop all the generated DAO messages, as all such messages are generated by the descendants of the malicious node; hence they must go through the malicious node to get to the root. Even if a DAO message reaches the root, the root may not suspect the DAO induction attack. This is due to two facts: first, the DTSN number is not echoed in DAO messages [13]. Therefore, the root cannot observe a DTSN update by examining the received DAO messages; second DAO messages can be generated because of other reasons such as a parent change [13].

Our detection mechanism. To enable the root to detect the DAO induction attack, we propose a minor adjustment in the RPL's DTSN processing. In this adjustment, we simply require nodes to accept DTSN updates from any of their neighbours rather than just their DAO parents. Note that this solution imposes no overhead on the nodes. It is because with or without this adjustment, a legitimate update DTSN in the non-storing mode will disseminate in the whole network (i.e., in both cases, each node transmits a single DAO update). With the adjustment, however, DAO updates triggered by the attacker will find their way to the root, while without the adjustment, they may not, as the attacker can drop DAO updates to hide the DAO induction attack from

the root. For instance, in Fig. 3.2, without applying our adjustment, a DTSN update by the attacker (node A) will only propagate in sub-DODAG 1; hence the update is not noticed by the root. Further, the attacker can drop all the DAO packets generated by the nodes in sub-DODAG1 because all such DAO packets have to go through the attacker (as the root of sub-DODAG 1) to reach the root. Therefore, the attacker can prevent the root from receiving DAO messages[1].

Our detection mechanism comes with two significant advantages. First, it enables the root to always detect the DAO induction attack. Second, it imposes no overhead when the network is under no attack. We formally state and prove these advantages in the following two propositions.

Proposition 3.2 *The proposed detection mechanism enables the root to detect any DAO induction attack launched by a single attacker*[2]*.*

Proof. Recall that in our detection mechanism, each node must accept a DTSN update from any of its neighbours. Let A and R denote the attacker and the root, respectively. Since the network is connected, there must be a path of nodes $u_1, u_2, ..., u_k$, where $u_1 = A$, $u_k = R$, and u_i and u_{i+1} are within the transmission range of each other for every $1 \leq i < k$. Let $2 \leq j \leq k$ be the largest integer such that u_j has received the DTSN update. If $j \neq k$, then we must have that node u_{j+1} has not received the DTSN update. This is a contradiction as u_{j+1} is within the transmission range of u_j, thus it will receive the DTSN update from u_j. Therefore, we must have $j = k$, which implies that the root will receive the DTSN update.

□

Proposition 3.3 *The proposed detection mechanism does not impose any overhead when the network is not under the DAO induction attack.*

[1]Even if the root receives a DAO message, it may not detect the attack. It is because DTSN is not included in DAO messages.

[2]In Proposition 3.4, we consider the case where there are more than one attackers.

Proof. When the network is not under attack, the only legitimate DTSN update is the one that is initiated by the root. In RPL, a DTSN update by the root will trigger every node in the network to transmit a single DIO message and unicast a single DAO message to the root. In the modified version of RPL, as in the original version, a legitimate DTSN update in the non-storing mode will trigger every node to transmit a single DIO message and unicast a single DAO message to the root. Therefore, our detection mechanism imposes no overhead when the network is not under the DAO induction attack. When the network is under attack, however, our mechanism ensures that the root will hear about the DTNS update, hence detect the attack.

□

Note that the overhead of our detection mechanism on the root is very small. All the root needs to do is to check whether or not the DTSN number has been updated in received DIO messages. If there is an updated DTSN number, some other node (i.e., the attacker) must have illegitimately incremented the DTSN number.

Thus far, we have assumed that there is only a single compromised node in the network. In practice, however, the attacker may compromise more than one node because, for instance, multiple nodes in the network may use the default or weak access credentials, or multiple nodes have the same software/hardware vulnerability, or because the attacker has physical access to multiple nodes prior to their installation in the network. If the attacker compromises multiple nodes, then the attacker can launch a stronger DAO induction attack by using a single compromised node to repeatedly increase the DTSN number, and using the remaining compromised nodes to drop the triggered control messages to prevent them from reaching the root. For this strong attack to be effective, however, the set of compromised nodes that drop messages must constitute a cut in the communication graph (Proposition 3.4).

Note that an attacker controlling a cut in the network is in a strong position as it can control the whole traffic going to/from one part of the network to the other part where the root is located. To be placed in this vital position,

the attacker has to carefully select nodes to compromise. In particular, compromising random nodes will not be helpful to the attacker unless the attacker compromises a large number of nodes (see Fig. 3.10 in Section 3.6.3). We remark that in the presence of such strong attacks, we need stronger measures (such as the deployment of monitoring nodes which regularly report to the root) to detect the DAO induction attack.

Proposition 3.4 *The attacker cannot prevent the root from detecting the DAO induction attack if the set of compromised nodes that drop messages does not constitute a cut in the communication graph.*

Proof. Let S be the set of nodes that are not compromised, $S_1 \subseteq S$ be the set of nodes that receive the DTSN update initiated by the attacker, and $S_2 = S \backslash S_1$ be the ones that do not receive the update. If the root is in S_1, then the attack is clearly detectable. Therefore, let us assume that the root is in S_2, which means that S_2 is not empty. Note that no node in S_2 is a neighbour of any node in S_1, as otherwise the node in S_2 will receive the DTSN update from its neighbour in S_1. This implies that the set of nodes that drop messages must be cutting the network into disconnected non-empty components S_1 and S_2, hence they constitute a cut in the network. □

3.5.2 Malicious Node Identification

The proposed detection mechanism enables the root to detect and report the DAO induction attack. After detecting the attack, the network administrator should examine and search the network for the malicious node. The root can help here by narrowing down the search space; it can provide the network administrator with a shortlist of potentially compromised nodes. In the following, we propose an *identification method* which enables the root to narrow down the search space to two nodes, guaranteeing that at least one of them is a compromised node.

Our identification method. We assume that each node has a pairwise secret key with the root. This single key is pre-installed at nodes (similar to

Algorithm 1: Performing a DTSN update with the new adjustments

Procedure: initialization
DTSNparent ← NULL
DTSN_update ← False
end procedure
Procedure: DIO processing
// process the received DIO messages from each node in the neighbours set ($\mathcal{N}$)
if *DIO received from* $n_i \in \mathcal{N}$ **then**
  // if the DIO reflects a DTSN update from n_i and the DTSN_update is False
  if *DTSN(DIO)* > *DTSN_*n_i & *DTSN_update* = *False* **then**
    DTSN_update ← True
    // update DTSN value of n_i and DTSNparent
    DTSN_n_i ← DTSN_n_i++
    DTSNparent ← n_i
    // update its own DTSN and broadcast DIO
    DTSN ← DTSN++
    // enable the detection timer
    Trigger_detection_timer($\mathcal{T}$)
    // schedule DAO messages to the root
    for *each DAO parent* $\in \mathcal{P}$ **do**
      Send(DAO)
    end
    Broadcast_DIO()
    // wait until the timer expires
    Wait_for_timer()
    // reset parameters
    DTSNparent ← NULL
    // reset DTSN_update
    DTSN_update ← False
  end
end
end procedure

Algorithm 2: Detection mechanism of the DAO induction attack at the root

```
Procedure: initialization
anomaly_detected ← False
probing_procedure ← False
end procedure
Procedure: DIO processing
// process the received DIO messages from each node in the
   neighbours set (𝒩)
if DIO received from n_i ∈ 𝒩 then
    // if the received DIO reflects an illegitimate DTSN update
    if DTSN(DIO) > DTSN_n_i & root_DTSN_update = False &
     anomaly_detected = False then
        // an attack has been detected
        anomaly_detected ← True
        potential_attackers ← {}
        // call the probing procedure (Algorithm 3)
        potential_attackers ← Probing_procedure (n_i)
        anomaly_detected ← False
        return potential_attackers
    end
end
end procedure
```

the secret network key) and is used by the root to authenticate queries and responses. Because messages between the root and legitimate nodes are authenticated, a malicious node cannot modify or generate legitimate query/response messages; at best, the malicious node can drop legitimate messages. The proposed identification method has a parameter $\mathcal{T}$, and requires each node to

1. temporarily store the ID of the node from which it has accepted the latest DTSN update (for simplicity we call this node DTSNparent). The ID can be deleted after $\mathcal{T}$ seconds;

2. accept at most one DSTN update every $\mathcal{T}$ seconds.

Algorithm 1 demonstrates how each node processes a DTSN update, and Algorithm 2 explains how the root detects a DAO induction attack. The parameter $\mathcal{T}$ ensures that a DTSN update cannot propagate within $\mathcal{T}$ seconds of a previous one. The value of $\mathcal{T}$ is set such that the root has enough time between

Algorithm 3: Probing procedure at the root to identify the attacker

Input: n_i
Output: list of potential attackers
Procedure: initialization
attacker_1 ← NULL
attacker_2 ← NULL
target_node ← n_i
visited_nodes ← {}
end procedure
Procedure: probing
`// search until finding potential attackers`
while *target_node* ! = *NULL and target_node not in visited_nodes* **do**
    `// update the list of visited nodes with target_node`
    Update (visited_nodes, target_node)
    `// update potential attackers`
    attacker_2 ← attacker_1
    attacker_1 ← target_node
    `// If no answer receives, the target node is NULL`
    target_node ← request_DTSNparent (target_node)
end
return attacker_1, attacker_2
end procedure

two consecutive updates to receive the first update and identify the malicious node through the following fast probing procedure.

Probing procedure. The probing procedure (Algorithm 3) works in steps as follows. In the first step, the root queries the node, say u_1, from which it received an illegitimate DTSN update (i.e., a DTSN update not initiated by the root) for the first time. In response to this query, u_1 reports the ID of the node, say u_2, from which it received the DTSN update for the first time. In Step $i \geq 2$, the root sends a query to u_i through the path $u_1, u_2, \ldots u_i$, asking u_i about the node from which it received the DTSN update for the first time. We remark that honest nodes only respond to authenticated queries from the root.

Since the number of nodes in the network is finite, the above probing procedure will inevitably terminate with one of the following two cases: 1) the root does not receive any response from u_j for some $j \geq 1$; or 2) for some $j > 1$, the root receives a response claiming that u_j has received the DTSN update

from an earlier node u_i, where $i < j$. In the first case, the root reports nodes u_j and u_{j-1} as potential compromised nodes[3]. In the second case, the root reports nodes u_j and u_i as potentially compromised nodes. In the following proposition, we prove that the compromised node is always one of the two nodes that the root reports.

Proposition 3.5 *The compromised node is always one of the two nodes in the root's report.*

Proof. The compromised node has only two options when it receives the root's query: 1) do not respond; 2) report a node. In the first case, by our probing procedure, the root will include the compromised node in its report. The second case can result in two different situations according to the attacker's response:

- The compromised node reports a node that has already been probed by the root: in this case, by the probing procedure, the root will put the compromised node in its report.

- The compromised node reports a node, say v, that has not been probed yet. By the procedure, in the next step, the root will send a query to v, asking v about the first node from which it has first received the DTSN update. If v is not within the communication range of the attacker, then the root's query will not reach v.[4] In this case, the root will not receive any response from v; hence the root will put both v and its previous node (the compromised node) in its report. Now, let us assume that v is within the transmission range of the compromised node. In this case, v will report the compromised node, since as its neighbour, v must have received the DTSN update from the compromised node first. If the root receives v's response, it will put the compromised node in its report. If the root does not receive v's response, it will also put the compromised node in its report. Therefore, in any case, the compromised node will

[3]If $j = 1$, then the root reports only u_1 as the compromised node.

[4]The attacker can change the route to deliver the packet to v. However, this change will be detected by v as the message will fail the integrity check at node v.

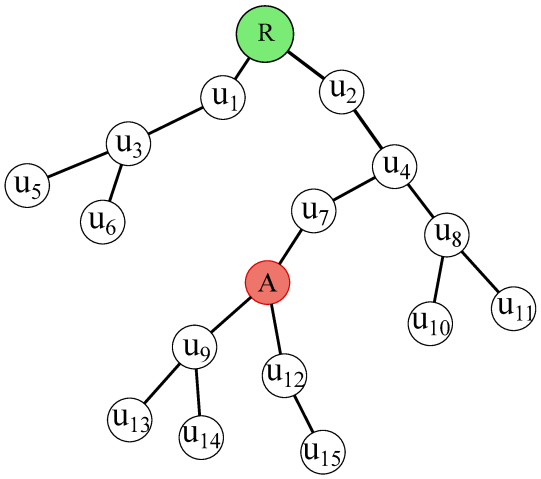

Figure 3.3: An example of RPL DODAG under the DAO induction attack.

be one of the nodes in the root's report. We remark that while the root is querying nodes, no other DTSN update can propagate. Therefore, all the root's queries correspond to exactly one DTSN update.

□

Example 2 (The probing procedure) *Consider the RPL DODAG shown in Fig. 3.3. Node R is the root, node A is the compromised node (the attacker), and the rest of the nodes are honest. The attacker initiates the DAO induction attack by changing the DTSN number and transmitting a DIO message. This generates a network-wide broadcast of DIO messages. At some point, the root will receive a DIO message from* u_2 *with the updated DTSN number. The root notices the illegitimate DTSN update, hence detects the attack. Next, the root starts the probing procedure, contacting* u_2 *first. In response to the root's query,* u_2 *will report* u_4*. In the next step, the root unicasts a query to* u_4 *through* u_2*, asking about the node from which* u_4 *received the DTSN update first. In this example,* u_4 *will report* u_7*. Next step* u_7 *will report A, hence the root eventually contacts A. If A does not respond, the root will include A in its report.*

If A reports one of its neighbours, say u_9*, then* u_9 *will respond to the root's query by reporting A as the node from which it received the DTSN update for*

the first time. Of course, A can drop u_9's response, but this will result in the root putting A in its report. Instead of node u_9, if A reports, say, u_{15}, then the root will contact u_{15} using the path

$$u_2, u_4, u_7, A, u_{15}$$

In this case, node A has to change the route, as otherwise, the query will not reach u_{15}. Such a change, however, will be detected by u_{15}; hence u_{15} will not respond at all. Since the root does not receive any response from u_{15}, it will put A in its report.

Impact of a mobile attacker. As shown earlier, the set of nodes that the attacker triggers depends on the location of the attacker. By moving in the network, therefore, the attacker can trigger different sets of nodes. If the objective of the attacker is to trigger the maximum number of nodes, however, a mobile attacker should find the optimal location in the network and stay there.

With regards to avoiding detection, a mobile attacker does not have an advantage over a stationary attacker. It is because, in our solution, a DTSN update is broadcast in the whole network. Such a broadcast update will always find its way to the root unless the network is partitioned. Also, our solution can find the ID of the mobile attacker, as each honest node stores and keeps the ID of the node from which it has received the first DTSN update. Even if the attacker moves, this information will remain in the network and will be used in the probing procedure to identify the attacker. Note that in the probing procedure, the root will identify the attacker if the attacker does not respond (e.g., because the attacker has moved).

3.6 Experimental Analysis

To evaluate the impact of the DAO induction attack on the network's performance, we performed a diverse set of simulations using the Contiki operating system [43], a lightweight and open-source operating system designed for IoT.

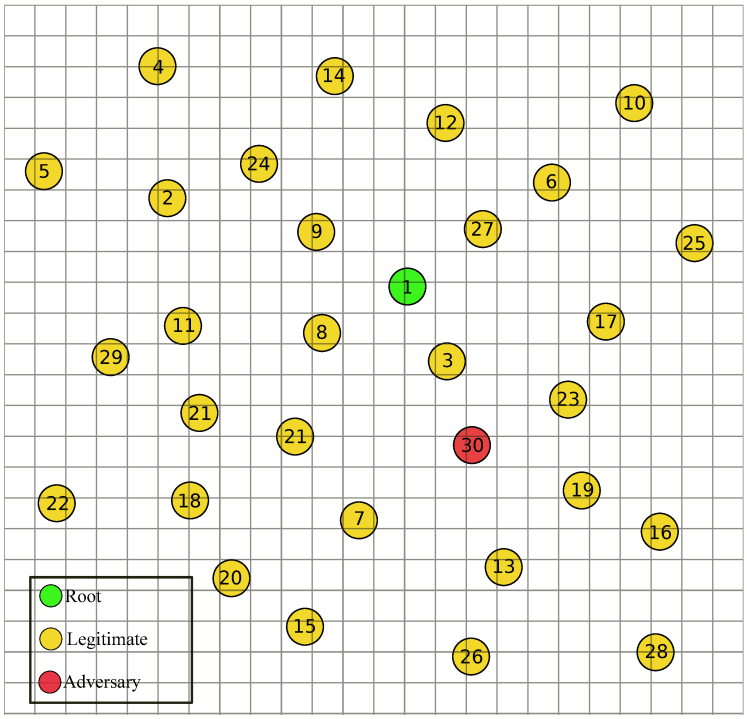

Figure 3.4: A sample network generated in our simulation. The green node is the root, the red one is the adversary and the yellow nodes are normal RPL nodes.

All the simulations were performed using Cooja [44], the default simulator of the Contiki OS, which emulates the operation of real IoT devices.

3.6.1 Evaluation Setup

We used the Zolertia Z1 mote [45], which are an MSP430-based board benefiting from a radio chip compatible with the IEEE 802.15.4 link layer protocol. To implement the DAO induction attack, we modified the RPL protocol stack of the Contiki OS on the malicious node. Similar to other nodes, the malicious node joins the network and actively participates in the creation and maintenance of the DODAG. The main difference between the malicious node and the others is that it is programmed to periodically increment its DTSN number and send it in a DIO message to its neighbours. To evaluate the maximum impact of the DAO induction attack in the non-storing mode, we selected the malicious node randomly from the neighbours of the root. Note that these nodes have the maximum number of descendants among all non-root nodes.

We considered a sample scenario in which nodes are distributed uniformly at random in a $150m \times 150m$ square area network. The network dimensions have been chosen based on a sample smart building IoT network such as a warehouse. Fig. 3.4 shows a sample network generated in our simulation.

Each node is static and transmits one data packet with a payload of 50 bytes to the root every 60 seconds. This data traffic transmission model tries to model a typical sensing IoT network with sensors which transmit their data to the sink of the network at predefined slots. To simulate link failure, we used the Unit Disk Graph Model (UDGM) with a unified transmission range of $40m$ for all nodes including the root. UDGM is the default setting of Contiki, and emulates distance loss for propagation model. RPL uses its default objective function which is the Minimum Rank with Hysteresis Objective Function (MRHOF) [46].We used CSMA/CA for the link layer and ContikiMAC [47] for the radio duty cycling (RDC) protocol. In ContikiMAC, which is the default setting of RDC protocol in Contiki OS, the radio is maintained off when there is no incoming or outgoing packet. ContikiMAC is power efficient but imposes larger end-to-end latency. For the transport layer, we used the User Datagram Protocol (UDP) [48]. The simulation parameters are summarized in Table 3.1.

3.6.2 Impact of the DAO Induction Attack

We evaluated the impact of the DAO induction attack on the following metrics.

- *DAO overhead:* the total number of DAO transmissions, including transmissions of original DAO messages as well as transmissions for forwarding DAO messages towards the root.

- *Average power consumption:* the average consumed power by each node in the network.

- *Average packet loss ratio:* the packet loss ratio averaged over all the nodes in the network. The packet loss ratio of a node is one minus the ratio of the number of received packets by the DODAG root from that node over the total number of packets sent by the node.

Table 3.1: List of Simulation Parameters

Simulation parameters	**Value**
Simulation time	$1800s$
Radio medium	UDGM: Distance Loss
Topology dimension	$150m \times 150m$
Number of nodes	20, 30, 40, and 50
Modes of operation	Storing and Non-storing
Objective Function	MRHOF
Transmission/ Interference range	$40m/80m$
Traffic rate per node	1 packet per minute
Sensor node type	Zolertia Z1
Number of simulations	10 per each topology
link layer protocol	IEEE 802.15.4-CSMA/CA
Node positioning	Uniform distribution

- *Average latency:* the average end-to-end latency of all packets successfully received by the root.

Fig. 3.5 shows the total number of DAO transmissions (i.e., the DAO overhead) for both RPL modes of operation. As shown, the DAO induction attack significantly increases the DAO overhead in both storing and non-storing modes. In more extensive networks, this overhead is higher. When there is no attack, the DAO overhead increases slowly with the number of nodes. Under the DAO induction attack, however, the DAO overhead grows at a significantly higher rate. Note that, the impact of the DAO induction attack is higher in the non-storing mode than the storing mode. This is expected because, in the non-storing mode, a DTSN increment triggers all the nodes in the attacker's sub-DODAG to transmit DAO messages.

Fig. 3.6 shows the average power consumption of nodes when the network is under the DAO induction attack. To calculate the average power consumption, we used the collect-view feature available in Contiki [49]. As shown, the power consumption increase because of the DAO induction attack is more noticeable in the non-storing mode than in the storing mode. This is expected because, in

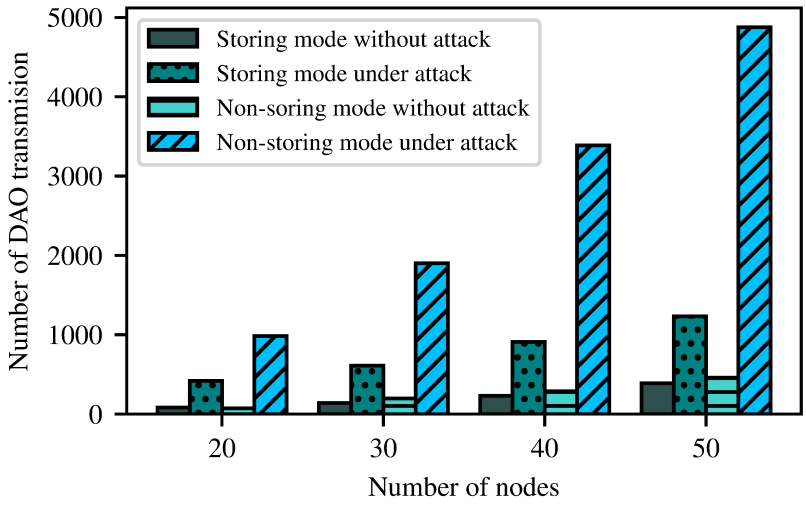

Figure 3.5: The impact of the DAO induction attack on the number of DAO transmissions in the storing and non-storing modes.

the non-storing, the attack engages more nodes and generates more overhead.

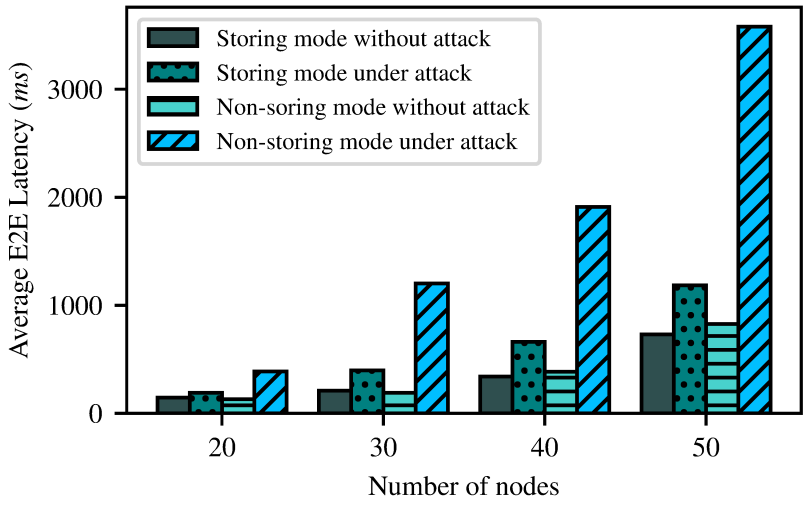

Figure 3.7: The impact of the DAO induction attack on the average end-to-end latency in the storing and non-storing modes.

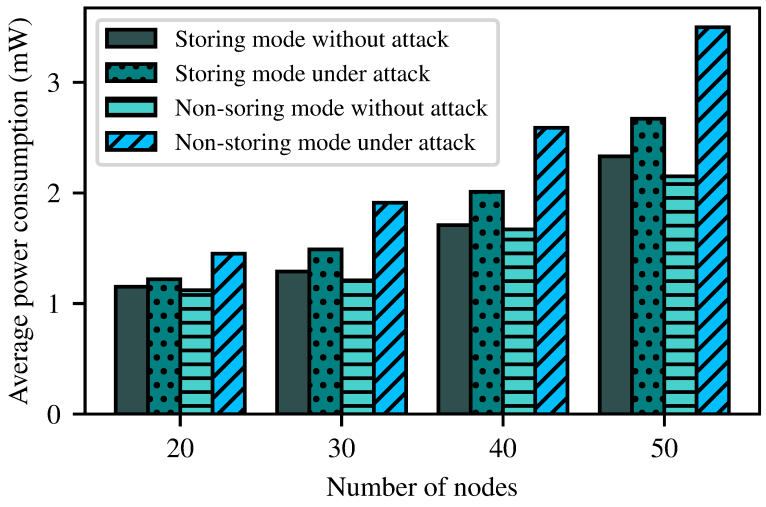

Figure 3.6: The impact of the DAO induction attack on the average power consumption in the storing and non-storing modes.

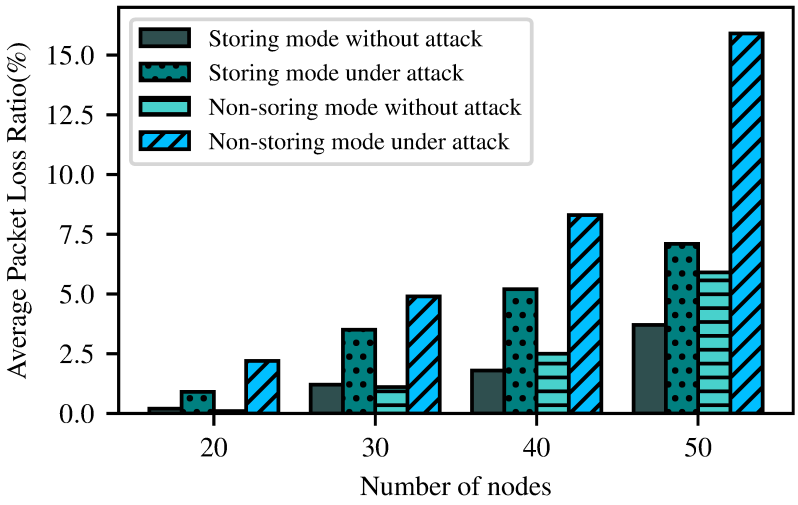

Figure 3.8: The impact of the DAO induction attack on the packet loss ratio in the storing and non-storing modes.

Fig. 3.7 shows the impact of the DAO induction attack on the average end-to-end latency. As shown in the figure, the DAO induction attack significantly increases the average end-to-end latency in the network. This increase is considerably higher in the non-storing mode than the storing mode. Again, the underlying reason is that the DAO induction attack engages more nodes and creates more overhead in the non-storing mode.

The impact of the DAO induction attack on the packet loss ratio is shown in Fig. 3.8. This impact is insignificant in small networks, particularly in the storing mode. The impact is, however, considerable in networks with about 40 and more nodes. As in the previous cases, the DAO induction attack is more severe in the non-storing mode than in the storing mode.

3.6.3 Detecting the DAO Induction Attack

In Proposition 3.2, we proved that using our method, the root can always detect any illegitimate DTSN update. In the presence of packet loss, however, the root may not hear about a DTSN update immediately. This is because the DTSN update may not reach the root if the network experiences high packet loss. An interesting question then is how many times an attacker can update DTSN before it gets detected by the root.

To evaluate the impact of packet loss on the detection capability of the root, in each run of simulation we distributed 50 nodes uniformly at random in a network of size $150m \times 150m$ and randomly selected one node as the attacker. We varied the packet loss rate from 0 to 60% and measured the expected number of times the attacker increments its DTSN before the root detects the attack. Fig. 3.9 shows the result of this simulation. As shown, when the packet loss rate is small the root is able to detect the DAO induction attack immediately.

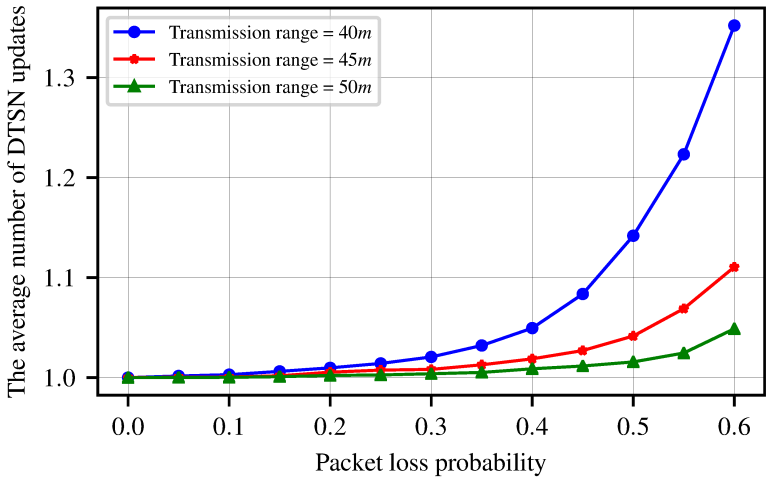

Figure 3.9: The average number of DTSN updates by the attacker before the root detects the attack.

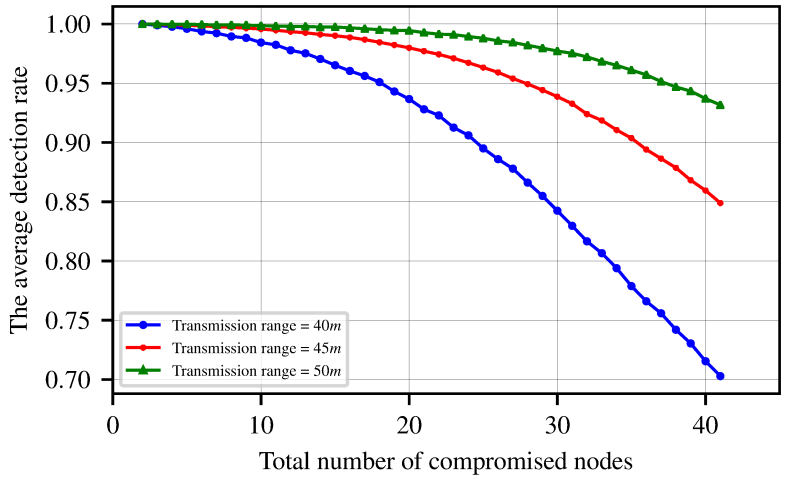

Figure 3.10: The probability of detecting the attack when there are multiple randomly-selected compromised nodes.

As the packet loss rate increases, the average number of times the attacker can increase its DTSN number increases. This number is, however, less than two (in fact, closer to one than two). The reason our method is so robust against packet loss is that the root can detect the attack as long as there is any packet-loss free path from the attacker to the root. As the transmission range increases, the network becomes more connected, and as a result, the number of potential paths from the attacker to the root increases. This, in turn, improves the detection as shown in Fig. 3.9.

Next, we evaluated the number of random nodes the attacker has to compromise in order to cut the network (in an attempt to hide the attack from the root). In Fig. 3.10, the x-axis shows the number of compromised nodes, and the y-axis shows the probability the network is cut. As shown in this figure, the attacker needs to compromise a considerable number of nodes to cut the network, particularly as the transmission range increases. For example, when the transmission range of nodes is 40m, the detection rate is above 90% even when the attacker compromises nearly half of the total number of nodes in the network.

3.7 Related Work

3.7.1 Existing security attacks against RPL

Recent research has shown that RPL is vulnerable to a wide range of security attacks [36], [50], [51], [5], [52]. In the following we briefly overview the existing related attacks against RPL and their countermeasures.

- **DIO suppression attack:** the goal of this attack is to intercept or slow down the transmission of DIO messages in the network. To this end, a malicious node replays previously heard DIO messages. The effect of this attack is a general degradation of the routes quality or a partition of the network [53]. This attack can also create loops in the DODAG. The RPL's replay protection mechanism [13] can prevent this attack at the cost of an increase in the signaling overhead (hence an increase in the energy consumption).

- **DIS flooding attack:** when a new node wants to join an RPL network, it can send a DIS messages to receive DIO messages from its adjacent nodes. In the DIS attack [54], the attacker repeatedly sends DIS messages, which causes all the nodes in the vicinity of the attacker to repeatedly transmit DIS messages in response. As shown in [55], this attack can significantly increase the number of control message transmissions.

Authors in [51] claimed that their proposed intrusion detection system can mitigate the DIS attack.

- **DAO insider attack:** in this attack a compromised node periodically transmits DAO control messages to its parent. Each of these DAO transmissions triggers a sequence of transmissions by the nodes on the path from the compromised node to the root of the network. The authors in [34] introduced this attack and investigated its side effect on the performance of the network. They also proposed a solution called SecRPL to alleviate the impact of this attack by restricting the number of times a parent can forward DAOs.

- **Routing table falsification:** in this attack, the attacker advertises false routing information to other nodes. Consequently, neighbours choose wrong route for sending information [5].

- **Worst parent attack:** in RPL, each node chooses a node with the lowest rank from its potential parent list as its preferred parent. In the worst parent attack, a malicious node is programmed to choose the worst node in its parent list as its preferred parent and create a non-optimized route. Consequently, all the traffic routed through the malicious node experience a delay. The malicious node may add extra delay by forwarding DAO messages of its children with delay. The impact of this attack was investigated in [56]. As proposed in [36], one possible solution to detect this attack is to draw a global view of the network graph.

- **Neighbour attack:** in this attack the attacker forwards DIO messages without updating them (i.e., without adding its own information to the message) [57]. As a consequence, a node may choose a non-neighbour node as a parent. Since the non-neighbour parent is far away, the victim node has to consume more transmission power to communicate with its parent. This attack can also create loops and inefficient routing paths.

- **Blackhole attack:** in the blackhole attack, an attacker drops all the packets that it is supposed to forward. This attack can target the routing

control packets in an attempt to disrupt the network traffic [58], [59]. Authors in [60] evaluated the impact of the blackhole attack in RPL networks.

- **DAG inconsistency attack:** a malicious node modifies or adds fake information about DODAG in the header of RPL control messages. This attack can be used to reset the trickle timer of a victim node, create a black hole, or even partition the network [5]. The RPL local and global repair mechanisms can fix the problems caused by this attack. However, they cannot prevent the DAG inconsistency attack in the first place.

- **Version number attack:** an attacker increases the version number in DIO messages; this forces nodes to rebuild the whole DODAG. As shown in [6], this attack can cause significant data loss through creating many loops in the topology. To prevent this attack, [37][61] proposed using a hash chain to authenticate root's requests to change the version number. Authors in [62] proposed a distributed monitoring approach to detect this attack.

- **Energy depletion attack:** a malicious node deliberately generates and transmits a large number of packets to legitimate destination nodes to deplete the energy resource of nodes on the routing path to the destination nodes. Authors in [63], proposed a threshold detection scheme called "MAD" against the energy depletion attack. In MAD, each RPL node keeps the number of received packets from its child node during a window of time, and compare it with a threshold which is calculated dynamically to discover compromised nodes. If the number of received messages from a node exceeds the threshold, that node is considered as a potential attacker.

- **Sinkhole attack:** this attack runs in two steps: first, a malicious node attracts traffic from its neighbours by advertising fake control messages. Second, after receiving the traffic, it drops or modifies selected packets. In an RPL network, the malicious node can attract traffic by changing its

rank. This attack has been evaluated in [33] and [55]. Also, the authors in [64] proposed two countermeasure against this attack: parent-fail-over and rank verification.

- **Wormhole attack:** a wormhole is an out of band connection between two nodes. Many packets can be delivered faster through the wormhole than via normal paths. A wormhole in itself is not necessarily a security breach. For example, a wormhole can be used to forward mission critical messages where high throughput is important. However, when the wormhole is controlled by attackers, they can use it to launch attacks such as sinkhole or traffic analysis [65].

- **Selective forwarding attack (gray hole attack)** In the selective forwarding attack, the attacker refuses to forward selected packets with the objective of disrupting communications. An instance of this attack is to selectively drop RPL control messages and forward the remaining traffic. Heartbeat protocol [33] can be used mitigate this attack.

- **Rank attack:** a malicious node increases or decreases its rank intentionally in order to change the structure of the network. By increasing rank, the attacker can create loops and inconsistency in the network. By decreasing its rank, on the other hand, the attacker attracts other nodes to select the attacker as their parent [5]. Authors in [67] and [55] studied this attack and evaluated its impact using the NS2 network simulator and Contiki OS.

- **Routing table overload attack:** in this attack, a malicious node advertises many fake routing information to overload the routing table of a victim node. When the routing table is saturated, the victim node is unable to add any new route to the table.

- **Sybil attack:** in a Sybil attack, the attacker uses many identities to, for example, increase its influence in the network. A general technique to prevent the attack is identity validation. A solution against this attack in IoT is proposed in [68].

Security attacks	Prerequisites	Effect	Countermeasure
DIO suppression attack	-	DODAG inconsistency and degrading routing services	Replay protection mechanism
DIS flooding attack	-	Resource consumption	None
DAO insider attack	-	degrading routing and increasing power consumption	SecRPL [34]
Routing table falsification	Storing mode	Sub DODAG isolation and increasing latency	None
Worst parent attack	-	Loops and inconsistencies	None
DAG inconsistency attack	Optional header	Power and link availability	RPL loop detection mechanism
Version number attack	-	Power consumption and making loop	VeRA and TRAIL
Energy depletion attack	-	Power consumption	MAD
Wormhole attack	at least two malicious nodes	Sub DODAG isolation	Geographical data [33] and Merkel trees [66]
Sinkhole attack	-	Degrading routing services and delay	SVELTE, Rank verification [37], and parent failover [64].
Blackhole attack	-	disrupting routing protocol	SVELTE and Monitoring of counters [60]
Selective forwarding attack	-	sub DODAG isolation	Monitoring of counters
Rank attack	-	Route disruption	VeRA and TRAIL
Neighbour attack	-	False routes and resource consumption	RPL loop detection mechanism
Routing table overload attack	storing mode	Increasing delay	None
Sybil attack	-	Compromised Route or Broken Network	Authentication
Replay attack	-	DODAG inconsistencies and loops	Replay protection mechanism

Table 3.2: Summary of attacks on RPL protocol

- **Replay attack:** an attacker captures a packet, and resend it at a later time. One kind of this attack in the context of RPL network is Routing Information Replay attack which can change the routing information with outdated routing paths [69], [70], [5]

3.8 Conclusion

In this chapter, we introduced the DAO induction attack, a novel attack against the RPL protocol in which a malicious insider node increments its DTSN number periodically to trigger many redundant transmissions in the network. Through various simulations, we showed that the attack adversely impacts network performance and power consumption, particularly when the network operates in the non-storing mode. To mitigate, we proposed a light-weight solution to detect the attack. The proposed solution can be implemented on common IoT platforms with minimum changes to the RPL protocol. Furthermore, it imposes no overhead on IoT devices, requires no external monitoring nodes, and can quickly detect the attack even when the network experiences high packet data loss rates. To discover the attacker, we proposed an identification method that returns two nodes as potential attackers. We proved that one of the returned nodes is always the attacker. Finally, we remark that our proposed solution can mitigate similar security attacks, such as the version number attack.

Chapter 4

A Lightweight Authentication Mode for RPL

In this chapter, we tackle the sender's authentication problem of the RPL protocol. We propose a new authentication mode for RPL based on the Blom key pre-distribution scheme [8]. We introduce a lightweight Blom scheme and show that the computational cost of the Blom scheme can be significantly reduced at the cost of a slight increase in memory requirement. We compare the computational overhead of our proposed scheme with that of the original Blom scheme and show that our proposed solution needs much lower resources.

4.1 Problem Overview

Currently, there is no mechanism in RPL to allow nodes authenticate the sender of a message [71]. This has led to a range of identity attacks where an attacker generates fake identities or steals legitimate identities of other nodes to masquerade itself. The existing identity attacks in RPL can be placed into three different classes: Sybil attack, ID spoofing attack, and Clone ID attack. In Sybil attack a compromised node takes multiple identities on a single physical node that operates in DODAG [5], [4], [33]. We refer to these identities as Sybil IDs. In Clone ID attack a malicious node uses a single valid identity on several cloned physical nodes [57]. In ID spoofing attack a malicious node replaces the source address of a packet with a fake or a valid ID to mislead the recipient of the packet [57].

In an RPL network, a malicious node can use an identity attack to damage the network while hiding its true identity. For example, a malicious node can periodically send DIS messages with different fake IDs to force its adjacent nodes to reply with DIO messages and reset their Trickle timers [72]. This will impose congestion and increase power consumption of nodes in the attacker's vicinity. In another group of attacks, an attacker can send RPL control messages using the ID of other nodes to perform illegitimate actions in the network without putting its identity in danger. For example, an attacker can spoof the ID of a parent node and send a DIO message with infinite rank to its children to detach them from the DODAG. In another possible attack, a malicious node increases DTSN or Version number and sends DIO messages using stolen identities to perform the DAO induction attack [73] or the version number attack [74] without compromising its identity. Or, the attacker can send DAO messages to the root or parents with fake IDs or stolen IDs to install downward paths in the network. In general, an attacker can use an identity attack in combination with other attacks to protect its true identity from being detected while damaging the network.

To mitigate identity attacks, nodes should be able to verify identity of the sender for each received message. Supporting authentication in RPL networks is, however, challenging because

1. Impracticality of asymmetric cryptography: the limited computational power of RPL devices makes it hard to use asymmetric algorithms such as RSA signature [40].

2. Limited memory: RPL constrained devices can maintain a limited number of security keys in their memory. In large networks, its is therefore not possible for nodes to keep a distinct key for every other node in the network.

3. Node compromise: RPL devices are low cost and are not tamper resistant [5]. Furthermore, in many applications RPL nodes are deployed in public areas, which exposes devices to physical attacks by an adversary.

An adversary is able to take control of the captured nodes and extract their security keys.

4. Lack of a prior knowledge of traffic pattern: in many applications the topology of the network is not known before deployment. Furthermore, the topology changes in dynamic networks. This makes it difficult to determine nodes' positions in the network.

A practical solution for RPL authentication problem must satisfy the following requirements:

- Each node must be able to authenticate any other node in the network.
- The solution should work without requiring a third authority or verifier.
- The solution must require a minimum overhead for adding new nodes to the network.
- The solution must be resilient to node capture. In particular, an attacker who has captured and compromised security keys of a small number of nodes in the network should not be able to impersonate as an uncompromised node.
- The solution must impose low computational overhead.
- The solution must be independent of the network's topology.

Among the existing solutions, Blom key pre-distribution scheme satisfies all the above features with the exception that its computational overhead is on the high side. In this work, we show how we can modify the Blom scheme to significantly reduce its computational overhead at the cost of a slight increase in memory requirement.

4.2 Blom Key Pre-distribution Scheme

Consider a network with N nodes. In the bootstrapping phase of the Blom key pre-distribution scheme, a server generates a public $(\lambda + 1) \times N$ matrix

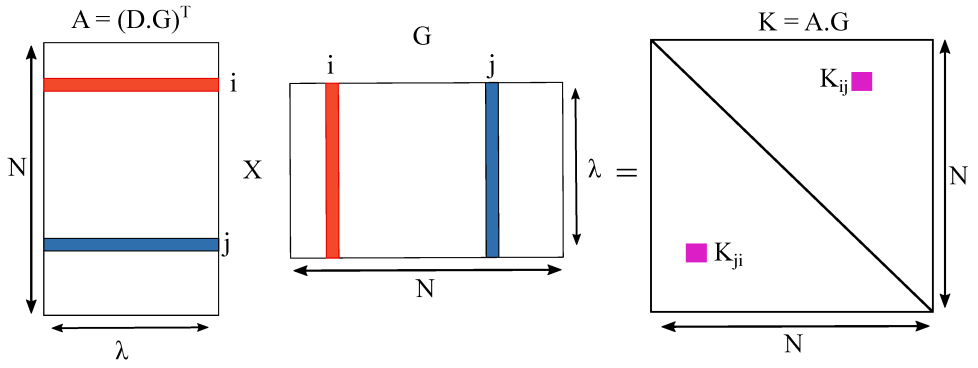

Figure 4.1: In the Blom's scheme, node n_i keeps column j of matrix G as public information, and row i of matrix A as private information. Nodes n_i and n_j exchange their public column vectors and generate K_{ij} and K_{ji}, respectively.

G and a private $(\lambda+1) \times (\lambda+1)$ matrix D over a finite field F_q, where λ is a security parameter and $q > N$. The matrix D is a symmetric matrix whose elements are chosen uniformly at random from F_q. The server then computes $A = (DG)^T$, where $(DG)^T$ is the transpose of DG, and pre-loads node u_i, $1 \leq i \leq N$, with the ith row of matrix A; $A_{i,:}$, and ith column of matrix G; $G_{:,i}$. Let $K = AG$. Since D is symmetric we have

$$K^T = (AG)^T = G^T(DG) = G^T D^T G = AG = K, \tag{4.1}$$

thus K is a symmetric matrix, i.e., $K_{i,j} = K_{j,i}$. In the Blom scheme, $K_{i,j}$ $(K_{j,i})$ is used as the pairwise key between u_i and u_j. In the key establishment stage, nodes u_i and u_j exchange their public information, i.e., their columns of G, and compute the pairwise key as

$$K_{i,j} = \langle A_{i,:}, G_{:,j} \rangle = \langle A_{j,:}, G_{:,i} \rangle \tag{4.2}$$

where $\langle A_{i,:}, G_{:,j} \rangle$ denotes the inner product. If every $\lambda+1$ columns of the generator matrix G are linearly independent, it can be proven that no information about the pairwise keys between uncaptured nodes is revealed if the number of captured nodes is at most λ [8], [38]. Constructing such a generator matrix G is not difficult. For instance, it can be shown that when $q > N$, every $\lambda+1$ columns of the following Vandermonde matrix (with $g \in F_q$ being a primitive

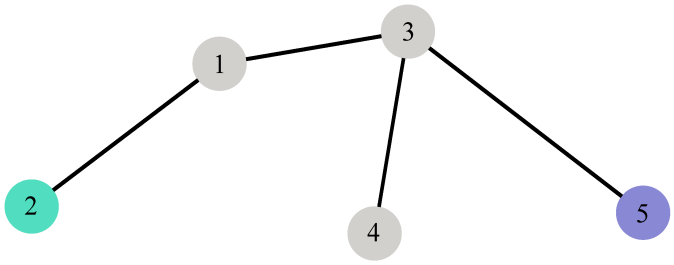

Figure 4.2: A simple network with 5 nodes

element) are linearly independent [38].

$$G = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ g & g^2 & g^3 & \cdots & g^N \\ g^2 & (g^2)^2 & (g^2)^3 & \cdots & (g^2)^N \\ & & & \vdots & \\ g^\lambda & (g^\lambda)^2 & (g^\lambda)^3 & \cdots & (g^\lambda)^N \end{bmatrix} \tag{4.3}$$

An advantage of using a Vandermonde matrix as the generator matrix is that node u_i only needs to store g^i, instead of the ith column of $G(G:, i)$, since $G:, i$ can be constructed given g^i.

A Toy Example

Consider a network with $N = 5$ nodes. Let $q = 19$, $\lambda = 3$, and:

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 4 & 8 & 16 & 13 \\ 4 & 16 & 7 & 9 & 17 \\ 8 & 7 & 18 & 11 & 12 \end{bmatrix} \qquad D = \begin{bmatrix} 6 & 15 & 14 & 7 \\ 15 & 1 & 8 & 16 \\ 14 & 8 & 3 & 9 \\ 7 & 16 & 9 & 5 \end{bmatrix}$$

Matrix D is symmetric and every four columns of matrix G are linearly independent. The matrix A can be calculated as:

$$A = (DG)^T = \begin{bmatrix} 15 & 6 & 0 & 1 \\ 16 & 12 & 5 & 3 \\ 8 & 6 & 14 & 3 \\ 12 & 13 & 2 & 0 \\ 10 & 14 & 11 & 10 \end{bmatrix}$$

Before been deployed in the network, node i is pre-loaded with the ith column of G and ith row of A as its public and private information, respectively.

Assume nodes 2 and 5 want to communicate securely with each other after they are deployed in the network. To establish a pairwise key node 2 calculate the following:

$$K_{2,5} = \langle A_2, G_5 \rangle = \begin{bmatrix} 16 & 12 & 5 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 13 \\ 17 \\ 12 \end{bmatrix} = 8$$

Similarly, node 5 computes $K_{5,2}$ as

$$K_{5,2} = \langle A_5, G_2 \rangle = \begin{bmatrix} 10 & 14 & 11 & 10 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 4 \\ 16 \\ 7 \end{bmatrix} = 8$$

In this example, the secret pairwise key between nodes 2 and 5 is 8. Note that, to provide enough security, the order of finite fields used in practice is much larger than 19 (the order is typically at least 2^{80}).

4.3 Enhancing the Blom scheme for RPL networks

In this section, we show how to significantly reduce the computational cost of the Blom's KPD scheme at the cost of a slight increase in memory requirement. The main idea is to use random (binary) linear matrices instead of Vandermonde matrices over for matrix G.

4.3.1 Revisiting the Generator Matrix G

Recall that nodes i and j need to compute an inner product (4.2) to generate a pairwise secret key. Computing this inner product requires $\lambda + 1$ field multiplications and λ field additions. In general, computing a field multiplication is considerably harder than computing a field addition. Therefore, the main computation cost of generating a pairwise key is $\lambda + 1$ field multiplications.

Suppose the elements of the generator matrix G are either zero or one (i.e., G is a binary matrix in F_q). In this case, computing the inner product does not require any multiplications because multiplication by one or by zero

are trivial. Therefore, if the generator matrix is binary, a pairwise key can be generated much faster than when the generator is a Vandermonde matrix. The problem with using a binary generator matrix is that a binary matrix may have a set of $\lambda + 1$ columns that are linearly dependent. As a result, an attacker may compromise a node by capturing less than $\lambda + 1$ other nodes, while in the original Blom scheme (with a Vandermonde generator matrix) the attacker needs to capture $\lambda + 1$ nodes to compromise any other node.

The above problem can be mitigated in two ways. First, suppose that every $\lambda - k$ columns (as opposed to every $\lambda + 1$ columns) of the binary matrix G are linearly independent, where k is a non-negative integer. Then, the attacker needs to capture at least $\lambda - k$ nodes (as opposed to $\lambda + 1$ nodes) to comprise any other node. If k is small, one may prefer to use the binary matrix G (instead of the Vandermonde matrix) to significantly reduce the computation at the cost of a slight decrease in the resilience of the network against node captures. Alternatively, if network resilience against λ node captures is a must, one can increase the value of λ by k. This approach essentially trades memory for computation as increasing the value of λ by k requires nodes to store k extra field elements in their memory.

The second mitigation is based on the idea that a binary matrix G is practically as good as a Vandermonde matrix in terms of resilience to node captures if an attacker is unable to find a set of less than $\lambda + 1$ linearly dependent columns in the binary matrix G. As will be discussed later, there are indications that, in general, finding a set of linearly independent columns in matrix G is hard.

The idea of using a binary generator matrix can be extended to using a generator matrix with elements from a set $S \subset F_q$. For instance, the set S can consist of 0, 1, as well as -1, that is $S = \{0, 1, -1\}$. Without loss of generality, assume that nodes $u_1, u_2, \ldots, u_x, x \geq 1$ have been captured. Clearly, the pairwise key between two nodes u_i and u_j is compromised if either $G_{:,i}$ or $G_{:,j}$ is a linear combination of $G_{:,1}, G_{:,2}, \ldots, G_{:,x}$. Also, capture of nodes $u_1, u_2, \ldots, u_x$ reveals no information about the pairwise key between u_i and u_j if neither $G_{:,i}$ nor $G_{:,j}$ is a linear combination of $G_{:,1}, G_{:,2}, \ldots, G_{:,x}$.

Theorem 4.1 *Let u_i be an uncaptured node (i.e., $x < i \leq N$). The probability that $G_{:,i}$ is a linear combination of $G_{:,1}, G_{:,2}, \ldots, G_{:,x}$ is at most $(\frac{1}{|s|})^{\lambda+1-x}$, where $x \leq \lambda + 1$ and $|\mathcal{S}|$ denotes the cardinality of the set $\mathcal{S}$.*

Proof. Without loss of generality assume that $G_{:,1}, G_{:,2}, \ldots, G_{:,x}$ are independent. By performing elementary column operations on the $(\lambda + 1) \times x$ matrix $[G_{:,1}, \ldots, G_{:,x}]$ we can obtain a $(\lambda + 1) \times x$ matrix $G' = [G'_{:,1}, \ldots, G'_{:,x}]$ such that for some $1 \leq t_1, \ldots, t_x \leq \lambda + 1$

$$[(G'_{t_1,:})^T (G'_{t_2,:})^T \ldots (G'_{t_x,:})^T] = I_{x,x}, \tag{4.4}$$

where $I_{x,x}$ denotes the identity matrix of size $x \times x$. Since $G' = [G'_{:,1}, \ldots, G'_{:,x}]$ is constructed by performing elementary column operations, $G_{:,i}$ is a linear combination of $G_{:,1}, G_{:,2}, \ldots, G_{:,x}$, i.e.

$$G_{:,i} = a_1 G_{:,1} + a_2 G_{:,2} + \ldots + a_x G_{:,x}, \tag{4.5}$$

if and only if it is a linear combination of $G'_{:,1}, G'_{:,2} \ldots, G'_{:,x}$, i.e.

$$G_{:,i} = b_1 G'_{:,1} + b_2 G'_{:,2} + \ldots + b_x G'_{:,x}. \tag{4.6}$$

where $b_1, b_2, \ldots, b_x \in \mathcal{S}$, is at most $|\mathcal{S}|^x$.[1] The vector $G_{:,i}$ is a random vector from a vector set of size $|\mathcal{S}|^{\lambda+1}$. Consequently, the probability that $G_{:,i}$ is a linear combination of $G'_{:,1}, G'_{:,2}, \ldots, G'_{:,x}$ (and hence a linear combination of $G_{:,1}, G_{:,2}, \ldots, G_{:,x}$) is at most

$$\frac{|\mathcal{S}|^x}{|\mathcal{S}|^{\lambda+1}} = \frac{1}{|\mathcal{S}|^{\lambda+1-x}}. \tag{4.7}$$

□

By Theorem 5.1, the probability that the pairwise key between two uncaptured nodes u_i and u_j is compromised is at most

$$1 - (1 - (\frac{1}{\mathcal{S}})^{\lambda+1-x})^2, \tag{4.8}$$

[1] Note that not all such vectors have all elements in $\mathcal{S}$

when $x < \lambda + 1$ and is one when $x \geq \lambda + 1$. For example, when $|\mathcal{S}| = 8$ and $\lambda + 1 = 200$ a pairwise key between two uncaptured node is secure with probability at least 0.999 if $x < 197$. Therefore, for relatively small values of $|\mathcal{S}|$, the modified Blom scheme provides nearly the same resiliency as the original scheme.

We can carefully choose the finite field F_q and $\mathcal{S}$ so the inner product is still computationally light. One approach is to use F_p, where p is a Mersenne prime (a prime which is one less than a power of two) and choose

$$\mathcal{S} \subseteq \{0, 2^0, 2^1, 2^2, \ldots, 2^{\lfloor \log_2(p) \rfloor}\}. \tag{4.9}$$

In this case, computing the inner product only requires (circular) shifts instead of modular multiplications [75]. Note that there are only four Mersenne primes whose size is less than 128 bits. These primes are

$$2^2 - 1, 2^7 - 1, 2^{31} - 1, \text{and } 2^{127} - 1. \tag{4.10}$$

If the security level is, for example, 64 bits then we can choose $p = 2^7 - 1$. In this case, to achieve 64 bits security, we can generate a single generator matrix G and 10 symmetric matrices $D_1, \ldots, D_{10}$. Then, every node u_i will be loaded with $(D_1 G)^T_{:,i}, \ldots, (D_{10} G)^T_{:,i}$ and $G_{i,:}$. To compute a pairwise key, a node has to compute 10 inner products and combine the results. Note that, the computational cost of 10 inner products when the vector elements are 7 bits is approximately the same as the computational cost of one inner product when the vector elements are 70 bits.

4.4 LBAM: a lightweith Blom based Authentication Mode for RPL

In this section, we propose a lightweight Blom-based authentication mode (LBAM) for RPL based on the lightweight Blom scheme introduced in previous section. The proposed authentication mode can work beside other internal and external security mechanisms such as intrusion detection systems and trust management schemes. LBAM is compatible with mobile RPL networks and

does not need any re-initializing mechanism when a node restarts. Another advantage of LBAM is that new nodes can be added to the network at any time without requiring any updates/changes in the existing nodes. The LBAM mode can completely mitigate sender's authentication problem i.e. no node (internal or external) can claim the identity of another node unless the attacker physically compromises the victim node or compromises $\lambda + 1$ different nodes.

To run a network with LBAM, the root first generates matrices G and D. After generating the public and private matrices, in the bootstrapping phase each node is loaded with a public information, $G_{:,i} = \text{RPL\_LBAM\_PB}_i$, and a private secret, $A_{:,i} = \text{RPL\_LBAM\_PR}_i$, before deployment in the network. Each node must preserve its private secret and do not reveal it to any other node under any circumstances. After deployment and during the neighbour discovery phase nodes exchange their own public information, RPL_LBAM_PB, with their neighbours. Each node should maintain the public information of its neighbours to use it in case of sending a packet. To send an authenticated packet to a destination node N_d, the sender node N_s must perform the following steps (Figure 4.3):

1. Prepare the message according to the pre-installed mode, i.e. encrypt the message with the network wide shared key, if any.

2. Add its RPL_LBSM_PB to the option part of the message.

3. Calculate the common pairwise key $K_{s,d}$ using $(\text{RPL\_LBSM\_PR})_s$ and $(\text{RPL-LBSM-PB})_d$ between itself and the destination node according to the Blom scheme. To accelerate the procedure of key generation a node may save this common pairwise keys for its neighbours for future usages.

4. Set the code field in the IP header of ICMPv6 message to reflect that LBAM is enabled.

5. Generate a HMAC of entire message, $HMAC = Hash(K_{s,d}, message)$, using proper lightweight MAC algorithm such as light-MAC [76] and concatenate the MAC to the message.

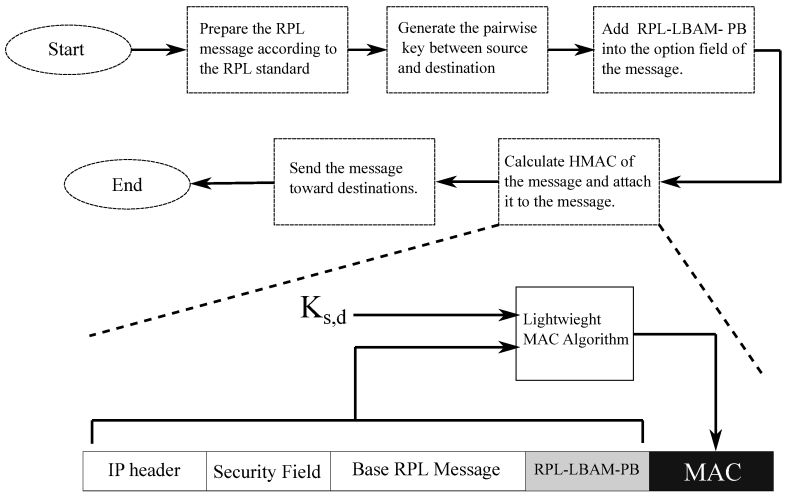

Figure 4.3: Sending an authenticated packet in RPL LBAM mode

6. Send the packet to the destination according to the RPL standard.

At the receiver, the destination node first checks the header of a received message. If LBAM mode is enabled, the receiver extracts the public information of the sender and calculates the pairwise key, $K_{d,s} = K_{s,d}$ using the public information of the sender and its own RPL_LBSM_PR. The receiver generates the HMAC of the received message to validate the authenticity of the sender using the same MAC algorithm. If validation terminates successfully, it delivers the packet. Otherwise, it drops the packet without any further processing. In case of multicast message the sender must calculate the HMAC for each of its neighbours using corresponding keys and attach all of these MACs to the message. At the destination, the receiver only needs to check its corresponding MAC which is generated by the pairwise key between source and destination. If one of the MAC passes the validation procedure, the receiver delivers the packet according to RPL.

4.5 Numerical Evaluation

We evaluated the resiliency of our proposed lightweight Blom scheme against node captures. To this end, we set $\lambda = 29$, $N = 1000$ and select a prime field

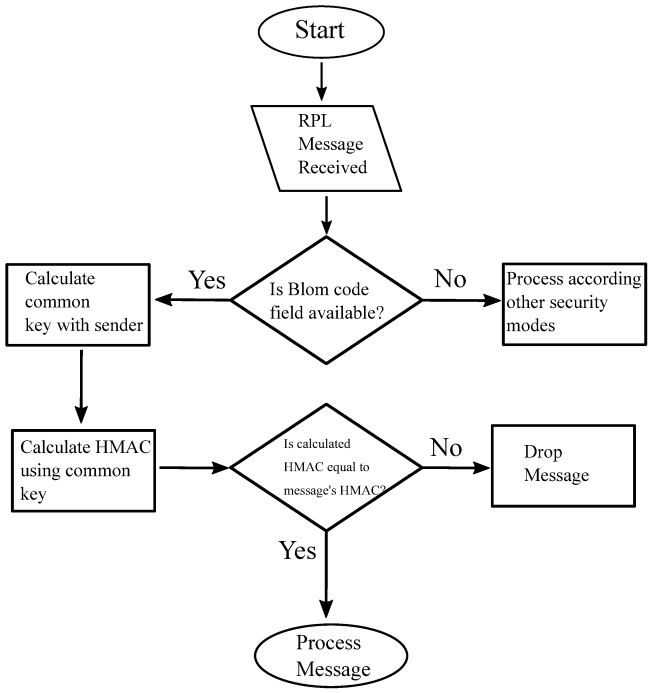

Figure 4.4: Receiving an authenticated packet

with $q = 160$ bits. We generate a single random matrix G whose elements are chosen from $\{-1, 0, 1\}$ uniformly at random. We simulate the capturing attack by selecting $\lambda = 29$ columns of G uniformly at random and checking if the capture of 29 nodes (corresponding to the selected columns) would compromise the key of any uncaputured node. We repeat the above process $100,000$ times[2]. The result shows that the attacker fails to compromise any key except those that belong to the captured nodes. This shows that the chance of the attacker to compromise the key of an uncaptured node is extremely low even if the attacker captures 29 nodes.

In the above simulation, the attacker selects its victim nodes randomly. Alternatively, the attacker can carefully pick its victim nodes with the objective of compromising as many keys as possible. Clearly, the keys of victim nodes are compromised. The question is whether the attacker can select its victim nodes in such a way that the key of at least one uncapured node is

[2]The process took more than 240 hours on a Microsoft Azure sever with 64 virtual CPUs and 512 GB RAM.

compromised. For instance, we may ask whether there are any 20 nodes in the network whose keys will reveal the key of at least one other node. If so, by capturing those 20 nodes, the attacker can compromise 21 nodes instead of 20 nodes. The above question is equivalent to asking whether the minimum distance of the linear code with parity-check matrix G is at most 21. As proven in [77] and[78], finding or even approximating the minimum distance of a code is a hard problem. This is an indication that the attacker practically does not gain from carefully selecting its victim nodes over selecting them at random.

4.6 Overhead analysis

In this section, we compare the computation cost of the proposed lightweight Blom scheme with that that of the original Blom scheme. To this end, we consider the 16 bits MSP430 microprocessor family as a reference. The processors in this family are commonly used in IoT devices such as Tmote Sky and Zolertia Z1.

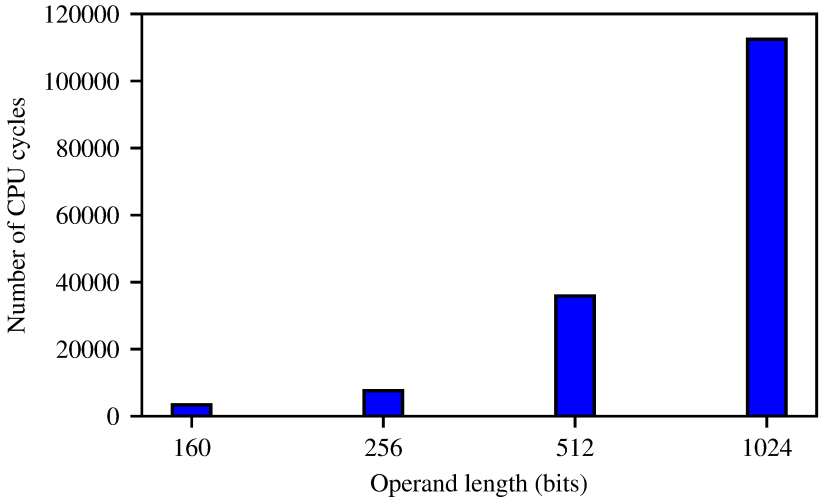

Figure 4.5: The average number of CPU cycles needed to compute a single field multiplication in MSP430 processor.

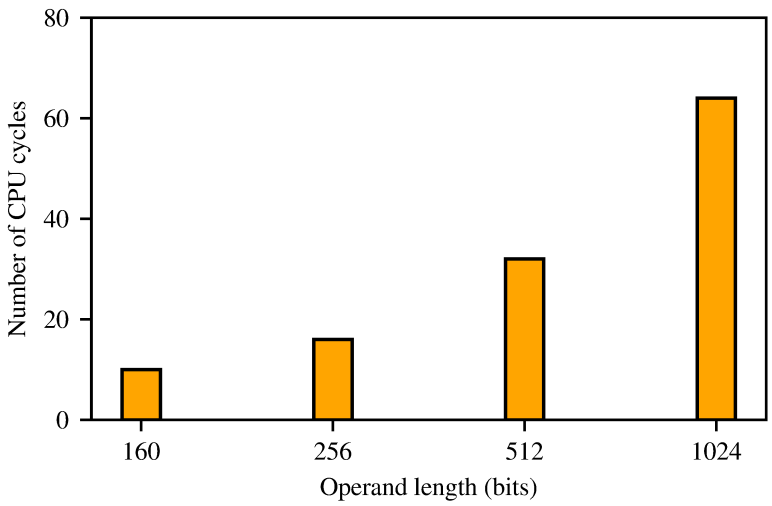

Figure 4.6: The average number of CPU cycles needed to compute a single field addition in MSP430 processor.

Figures 4.5 and 4.6 show the average number of CPU clock cycles needed to compute a field multiplication and a field addition, respectively. Computing a pairwise key in the Blom scheme requires about the same number of field multiplications and field additions. Therefore, Figures 4.5 and 4.6 give a good estimate of the speedup of the lightweight Blom scheme over the original Blom scheme. For instance, assuming that the order of the field is 256 bits, the original Blom scheme requires about $\frac{8752}{16} = 547$ times more CPU cycles than the proposed lightweight Blom scheme.

4.7 Related Work

In the following, we summarize the existing work that tried to address the authentication problem in RPL networks.

Raoof et al. [71] proposed a new secure mode for RPL called Chained Secure Mode (CSM). In CSM, each node encodes its unicast and multicast messages using a randomly generated code. The CSM method is based on the assumption that the first message of each node is authenticated and all nodes have the same code for the very first message. While CSM can protect the network against external attacker, it is still vulnerable to internal attacks

because each node knows the next code of its neighbours.

In another work, Airehrour et al. [79] proposed SecTrust-RPL, a new security framework for RPL. The authors proposed to install additional hardware on IoT devices to calculate a trust factor for each node. They showed that SecTrust-RPL can mitigate the impact of Rank attack and Sybil attack significantly. To protect the network agaist Sybil attack, the author proposed to bind the physical location of each node to its identity. Although this method can provide some degree of mitigation to the sender's authentication problem, a node can still take another node's ID and send packets using the other node's ID.

Authors in [80] studied a Sybil attack in which the main goal of the adversary is to disrupt the routing process of the RPL protocol by sending wrong routing information to the root or other nodes in the network. To detect the attack, the authors proposed a detection algorithm based on highest rank common ancestor (HRCA). In their approach each non-leaf node analyzes the incoming packets periodically and extracts the previous node ID for each received packet. If two incoming packets with the same source address are received via two different previous nodes, the common ancestor reports a Sybil attack alarm to the root of the network. In the non-storing mode, this approach can only detect the existence of the attack but not the attacker itself. Also, this method can have a high false positive rate.

Other works that study identity-based attacks in RPL are [81], [82], [72], [80]. These works only considered a limited version of the attack or assumed a special case of Sybil attack. In term of mitigation, there is not an effective and lightweight countermeasure for Sybil attack to completely prevent the attack in RPL. Current solutions only offer a limited mitigation and, in the best case, can detect the attack but not the attacker. Currently, there is no solution that can prevent Sybil attack from taking place in the network, or discover the attackers.

Authors in [81] evaluated the impact of Sybil attack against RPL when the attacker is a mobile node. They considered an adversary that moves in the network and broadcasts DIS messages with multiple fake identities. They

analyzed the impact of the attack on the performance of RPL and showed that the attack can severely impact packet delivery and control message overhead. The authors did not propose a mitigation method for this special Sybil attack.

In a recent work, Pu evaluated the DIS flooding Sybil attack in RPL networks using OMNeT++ [72]. In this attack, similar to the one studied in [81], an attacker broadcasts many DIS messages to overwhelm legitimate nodes. To mitigate the attack, the author proposed a solution based on the Gini index. This solution assumes that valid identities in the network have a specific distribution that the attacker is not aware of.

In [82], Murali et al. proposed a lightweight intrusion detection for Sybil attacks in mobile RPL networks. They also modeled the attack with artificial bee colony and evaluated the attack using Contiki operation system. To detect the Sybil attack they add three new features to DIO messages; None, timestamp, and control message counter. The None ID is assigned to each node upon joining the network. Each node uses its Nonce ID to broadcast messages to other nodes. If the Nonce ID of the received message is not the same as previous records, there is a possibility of a Sybil attack. This method cannot immune the network against internal Sybil attacker as all nodes are aware of the Nonce ID of other nodes. A malicious internal node can, therefore, send a message using another node's identity.

4.8 Conclusion

In this Chapter, we proposed *LBAM*, a lightweight authentication mode for RPL to provide message authentication in RPL networks. LBAM employs our proposed lightweight Blom key pre-distribution scheme. The lightweight Blom scheme uses binary matrices and require significantly less computation than the original Blom scheme, hence is more suitable for RPL networks. We sowed that the lightweight Blom scheme can be practically as good as the original Blom scheme with regards to resilience against node captures.

Chapter 5

Analyzing RPL Point-to-Point Communications in the Internet of Things

5.1 Overview

In this chapter, we study the quality of paths used in RPL for P2P communications. In particular, we analyze how much RPL's P2P paths "stretch" compared to the shortest paths. We prove that the average stretch is a factor of at least two in any network. We also show that, in some networks, the stretch factor grows with the number of devices in the network. We verify our analytical results using the Contiki-NG operating system and its built-in Cooja simulator. In addition, we evaluate the quality of RPL's P2P paths in terms of end-to-end delay. Our results show that the average end-to-end delay of RPL P2P paths is up to a factor of 2.3 higher than that of the shortest paths.

5.2 RPL's Stretch Factor

The stretch factor (also referred to as dilation or distortion) is a metric which can be used to compare length of paths selected by a routing protocol to length of the shortest paths.

Let $G = (V, E)$ be the communication graph, where V is the set of nodes in the network. There is an edge $(s, d) \in E$ between any two distinct nodes s and

d if and only if s and d are within the transmission range of each other[1]. Let $d_{s,d}$ denote the minimum hop distance between two nodes s and d in G, and $d_{s,d}^{(RPL)}$ denote the length of the path that RPL uses[2] for P2P communications between s and d.

Definition 1 (Stretch Factor of a Single Path) *Given a routing protocol $\mathcal{P}$, we denote the stretch factor of a path between s and d as $St_{s,d}^{(\mathcal{P})}$ and define it as*

$$St_{s,d}^{(\mathcal{P})} = \frac{d_{s,d}^{(\mathcal{P})}}{d_{s,d}}, \tag{5.1}$$

where $d_{s,d}^{(\mathcal{P})}$ denotes the length of the P2P path between s and d in the protocol $\mathcal{P}$.

Remark 1 *When RPL works in the non-storing mode, we have*

$$d_{s,d}^{(RPL)} = d_{s,r} + d_{r,d}, \tag{5.2}$$

where r is the root of the network. It is because in the non-storing mode, a P2P packet has to travel up to the root and then travel down to the destination.

Example 3 *Consider the network shown in Fig. 5.1. In this network, the shortest distance between nodes s and d is two (i.e. $d_{s,d} = 2$), because node s is connected to d via Node 10. If RPL works in the non-storing mode, then we will have $d_{s,d}^{(RPL)} = 6$, because a packet from s to d has to travel all the way up to the root and then travel all the way down to the destination d. If RPL works in the storing mode, however, we get $d_{s,d}^{(RPL)} = 4$, because a packet from s to d can be redirected at Node 2, which is the common ancestor of nodes s and d. Therefore, in this example, we get that $St_{s,d}^{(RPL)} = \frac{6}{2} = 3$ if RPL is in the non-storing mode, and $St_{s,d}^{(RPL)} = \frac{4}{2} = 2$ if RPL is in the storing mode.*

[1]Note that the DODAG generated by RPL is a spanning subgraph of G.

[2]If the RPL's path from s to d is different than the RPL's path from d to s, then $d_{sd}^{(RPL)}$ is defined as the length of the longer one.

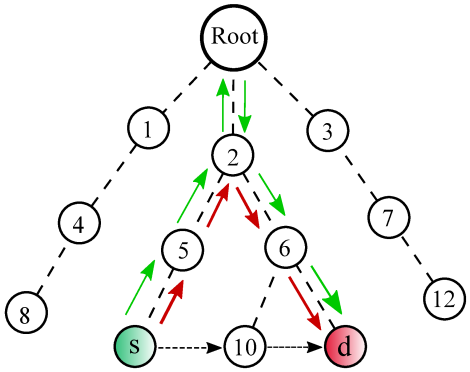

Figure 5.1: The path a P2P packet travels in an RPL DODAG. In the non-storing mode, the packet from s to d has to travel all the way up to the root first. In the storing mode, however, the packet can be redirected by Node 2, which is the common ancestor of the source and destination. Note that the links between Node 10 and nodes s and d do not exist in DODAG.

Definition 2 (Stretch Factor of a Routing Protocol) *We define the stretch factor of a routing protocol $\mathcal{P}$ as*

$$St^{(\mathcal{P})} = \frac{2}{(|V|-1)(|V|-2)} \sum_{s \neq d \in V} St^{(\mathcal{P})}_{s,d}, \tag{5.3}$$

where $\frac{(|V|-1)(|V|-2)}{2}$ is the number of pairs of source and destinations among non-root nodes.

Our first analytical result shows that the stretch factor of RPL's non-storing mode is at least two in any networks. In other words, if we select the root, and a pair of nodes s and d in the network uniformly at random, then the length of the path selected by RPL is expected to be at least twice the length of the shortest path between s and d. We remark that this result holds for any network, that is it holds for any distribution of nodes and any number of nodes $N \geq 3$.

Theorem 5.1 *Suppose that RPL works in the non-storing mode. Then, for any communication graph G, the stretch factor of RPL is at least two.*

Proof. Note that $St^{(RPL)}$ is essentially equal to $\mathbb{E}[St^{(RPL)}_{s,d}]$, when the source node s, the destination node d, and the root are all selected uniformly at

random. Therefore, by (5.2) we get

$$St^{(RPL)} = \mathbb{E}[St_{s,d}^{(RPL)}] = \mathbb{E}\left[\frac{d_{s,r} + d_{r,d}}{d_{s,d}}\right],$$

Notice that

$$\mathbb{E}\left[\frac{d_{s,r} + d_{r,d}}{d_{s,d}}\right] = \mathbb{E}\left[\frac{d_{s,d} + d_{d,r}}{d_{s,r}}\right] = \mathbb{E}\left[\frac{d_{r,s} + d_{s,d}}{d_{r,d}}\right],$$

because the source, the destination and the root are all selected uniformly at random. Therefore

$$\begin{aligned}
St(G) &= \frac{1}{3} \cdot \left(\mathbb{E}\left[\frac{d_{s,r} + d_{r,d}}{d_{s,d}}\right] + \mathbb{E}\left[\frac{d_{s,d} + d_{d,r}}{d_{s,r}}\right] \right. \\
&\qquad \left. + \mathbb{E}\left[\frac{d_{r,s} + d_{s,d}}{d_{r,d}}\right] \right) \\
&= \frac{1}{3} \cdot \mathbb{E}\left[\frac{d_{s,r}}{d_{s,d}} + \frac{d_{r,d}}{d_{s,d}} + \frac{d_{s,d}}{d_{s,r}} + \frac{d_{d,r}}{d_{s,r}} + \frac{d_{r,s}}{d_{r,d}} + \frac{d_{s,d}}{d_{r,d}}\right] \\
&= \frac{1}{3} \cdot \mathbb{E}\left[\left(\frac{d_{s,r}}{d_{s,d}} + \frac{d_{s,d}}{d_{s,r}}\right) + \left(\frac{d_{s,d}}{d_{r,d}} + \frac{d_{r,d}}{d_{s,d}}\right) \right. \\
&\qquad \left. + \left(\frac{d_{s,r}}{d_{r,d}} + \frac{d_{r,d}}{d_{s,r}}\right)\right] \\
&\geq \frac{1}{3} \cdot (2 + 2 + 2) \\
&= 2,
\end{aligned}$$

where the inequality is by the fact that, for every positive real number r, we have $r + \frac{1}{r} \geq 2$, because

$$(r-1)^2 \geq 0 \quad \Longrightarrow \quad r^2 + 1 \geq 2r \quad \Longrightarrow \quad r + \frac{1}{r} \geq 2.$$

□

The stretch factor of RPL can be considerably higher than two in a network. In fact, the RPL's stretch factor can grow with the number of nodes in the network. For instance, in linear networks, RPL's stretch factor grows logarithmically with the number of nodes as stated in the next theorem.

Theorem 5.2 *Suppose that RPL works in the non-storing mode in a linear network. Then, the RPL's stretch factor is $\theta(\log N)$, where N is the number of nodes in the network.*

Proof. Let N_l and N_r denote the number of nodes that are located on the left and right side of the root, respectively. We have

$$\begin{aligned} St^{(RPL)} =& \frac{2}{(N_l+N_r)(N_l+N_r-1)} \times \\ & \left(N_l \cdot N_r + \sum_{1 \le i < j \le N_l} \frac{i+j}{j-i} + \sum_{1 \le i < j \le N_r} \frac{i+j}{j-i} \right). \end{aligned} \tag{5.4}$$

Using the fact that the nth harmonic number is $\theta(\log n)$, we get that

$$\left(\sum_{1 \le i < j \le n} \frac{i+j}{j-i} \right) \in \theta(n^2 \log n). \tag{5.5}$$

Note that either $N_l \ge \frac{N}{2}$ or $N_r \ge \frac{N}{2}$, where $N = N_l + N_r$. Therefore, by (5.4) and (5.5), we get that $St^{(RPL)} = \theta(\log N)$. □

Theorem 5.2 proves that RPL's stretch factor in linear networks grows with the number of nodes. This may not happen in other networks, particularly in 2-D networks. For instance, in the following we show that RPL's stretch factor in 2-D grid networks is bounded by a constant. To prove this, we first consider a general family of networks called *ζ-uniform networks*.

Definition 3 (ζ-uniform Network) *Let $N_i(u)$ denote the number of nodes that are at hop distance i from node u in the communication graph. We call a network ζ-uniform if*

$$\forall i \ge 1, u: \quad N_i(u) \le \left(\frac{\zeta \cdot i}{\Delta^2} \right) \cdot N,$$

where N is the total number of nodes, and Δ is the network's diameter, i.e.

$$\Delta = \max_{u,v \in V} d_{u,v}.$$

Lemma 5.3 *Any 2-D grid is a 16-uniform network.*

Proof. Let G_g be a $n \times n$ grid network with $N = n^2$ nodes. Without loss of generality, assume that nodes are located at coordinates (i, j), $0 \le i, j \le n-1$. Note that in 2-D grid networks, the shortest distance between two nodes is the Manhattan distance between them. Therefore, for any node u, we get

$$N_i(u) \le 4i.$$

For instance, $N_1(u) \leq 4$, because there are at most 4 nodes at distance one from u in the grid network. Moreover, we have $\Delta = 2n$. Thus

$$\begin{aligned} N_i(u) &\leq 4i \\ &= \left(\frac{16i}{(2n)^2}\right).n^2 \\ &= \left(\frac{16i}{\Delta^2}\right).N \end{aligned}$$

□

Theorem 5.4 *In any ζ-uniform network, the stretch factor of RPL's non-storing is at most $\Gamma \approx \frac{2}{3} \cdot \zeta^2$.*

Proof. The stretch factor is

$$\begin{aligned} &\frac{1}{N(N-1)(N-2)} \cdot \sum_{r \neq s \neq d} \frac{d_{s,r} + d_{d,r}}{d_{s,d}} \\ =&\frac{2}{N(N-1)(N-2)} \cdot \sum_{r \neq s \neq d} \frac{d_{s,r}}{d_{s,d}} \\ =&\frac{2}{N(N-1)(N-2)} \cdot \sum_{s} \sum_{1 \leq i,j \leq \Delta} \frac{i}{j} \cdot N_i(s) N_j^{(i)}(s) \end{aligned} \tag{5.6}$$

where

$$N_j^{(i)}(s) = \begin{cases} N_j(s) & \text{if } j \neq i \\ N_j(s) - 1 & \text{if } j = i \end{cases}$$

Since the network is ζ-uniform, for every integer i, and node s, we have

$$N_i(s) \leq \left(\frac{\zeta \cdot i}{\Delta^2}\right) \cdot N$$

Thus, by (5.6), the stretch factor is

$$\begin{aligned}
&\frac{2}{N(N-1)(N-2)} \cdot \sum_{s} \sum_{1 \le i,j \le \Delta} \frac{i}{j} \cdot N_i(s) N_j^{(i)}(s) \\
=&\frac{2}{(N-1)(N-2)} \cdot \sum_{1 \le i,j \le \Delta} \frac{i}{j} \cdot N_i(s) N_j^{(i)}(s) \\
\le&\frac{2}{(N-1)(N-2)} \cdot \sum_{1 \le i,j \le \Delta} \frac{i}{j} \cdot \left(\frac{\zeta \cdot i}{\Delta^2}\right) \cdot N \cdot \left(\frac{\zeta \cdot j}{\Delta^2}\right) \cdot N \\
=&\frac{2N^2 \cdot \zeta^2}{(N-1)(N-2)} \cdot \frac{1}{\Delta^4} \cdot \sum_{1 \le i,j \le \Delta} i^2 \\
=&\frac{2N^2 \cdot \zeta^2}{(N-1)(N-2)} \cdot \frac{1}{\Delta^3} \cdot \sum_{1 \le i \le \Delta} i^2 \\
=&\frac{2N^2 \cdot \zeta^2}{(N-1)(N-2)} \cdot \frac{1}{\Delta^3} \cdot \frac{\Delta(\Delta+1)(2\Delta+1)}{6} \\
\approx&\frac{2}{3} \cdot \zeta^2.
\end{aligned}$$

□

The following corollary is a direct result of Lemma 5.3 and Theorem 5.4.

Corollary 1 *For any 2-D grid network G_g, we get*

$$St^{(RPL)} \in \mathcal{O}(1),$$

that is the RPL's stretch factor is constant with the number of nodes in the network.

As mentioned earlier, RPL has two mechanisms to reduce its stretch factor: 1) supporting direct communication between neighbouring nodes, and 2) storing mode. The former mechanism allows neighbouring nodes to communication directly without involving the root. While this is a good solution, it has limited impact on the RPL's stretch factor because majority of nodes are non-neighbours. The latter mechanism can reduce the stretch factor of non-neighbouring nodes at the cost of increasing communication overhead, and requiring every node to store a routing table.

In the next section, we propose a solution that does not require nodes to store routing tables, yet it achieves a lower stretch factor than the RPL's

storing mode. To achieve this, in our solution, nodes randomly select and report the ID of a small number of their low rank neighbours to the root. We show that reporting even a single neighbour can greatly help the root in discovering shorter P2P paths between nodes, hence significantly reducing the stretch factor. A feature of our solution is that it is fully compatible with RPL as the RPL standard allows nodes to select and report more than one parent (i.e. more than one neighbour with lower rank than themselves) to the root in their DAO messages.

5.3 Proposed Solution

Suppose a source node s wants to communicate with a destination node d. In the non-storing mode of RPL, a packet from s to d first travels up all the way to the root, and then travels all the way down to the destination d. If the root knows the shortest path between s and d, it can inform the source about this shortest path when it receive the first packet form s. This way, the subsequent packets from s to d can go through the shortest path instead of the long path through the root.

The main challenge with the above approach is that the root does not know the whole network topology in RPL networks. In fact, the root only knows the DODAG, which is a subgraph of the communication graph. To provide the root with the whole network topology (i.e., the communication graph), one approach is to have every node report all of its neighbours to the root using DAO messages [83]. This approach is, however, costly for the following reasons:

- It imposes a large communication overhead as each node has to report all its neighbours to the root. The amount of this overhead is particularly large on nodes near the root, since these nodes need to forward all such messages to the root.

- It is sensitive to network changes: if any neighbour of a node turns off or goes out of the node's range, the node must report the change to the

root by sending a new DAO message.

- It is resource consuming for nodes, particularly those with several neighbours, because nodes have to constantly check the connection to their neighbours (so they can update the root if any changes occur).

Our RPL-Compatible Solution. We propose that each node randomly selects and reports a small number (e.g., up to three) of its DAO parents to the root. We remark that the RPL standard allows each node to select more than one parent and report them to the root in their DAO messages. Therefore, our proposal is fully compatible with the RPL standard. Since each node reports only a small number of its neighbours (instead of all its neighbours), the root will have only partial information about the communication graph, hence may not be able to find the shortest path between two nodes. However, in the next section we show that even with this partial information, the P2P paths that the root can find are considerably shorter than those used in the RPL's storing mode. In fact, our simulation results show that, if nodes report only a single extra DAO parent, the P2P paths that the root discovers are up to 25% shorter than those in RPL's storing mode. Furthermore, if nodes report as few as three extra DAO parents, the P2P paths that the root discovers are nearly as short as the shortest paths.

Reporting Random Neighbours. In our solution, nodes randomly select a small number of nodes from their parent set and report them to the root in DAO messages. Notice that the solution restricts nodes to select from their parent set. A generalization of the solution is to allow nodes to randomly select nodes from their whole set of neighbours rather than just their parent set. This clearly gives more flexibility to nodes, and can be beneficial when there is not enough nodes in the parent set. This generalization is, however, not fully compatible with RPL as the standard has no provision for reporting neighbours (other than those from the parent set). This incompatibility can make the above generalization hard to implement in practice.

Selection Criteria. A node may have several options to choose nodes from its parent set. For instance, suppose the parent set has four nodes, and

the node needs to select only two nodes for the set. In our solution, the two nodes are selected uniformly at random. Our solution can be generalized to allow nodes to select their parents based on a criteria such as the parent's maximum residual energy or its stability. For example, a node may decide to select the two parents that have been in its range for the longest period of time in an attempt to minimize the number DAO messages it has to send due to a change in its neighbourhood. We remark that this generalization, unlike the previous one, is fully compatible with RPL, as RPL does not enforce how nodes should select parents from their parent set.

5.4 Experimental Analysis

In this section, we evaluate the performance of RPL P2P communication and our proposed method using Contiki-NG (release v4.5) [84] a lightweight and open source operating system specially designed for low power and lossy networks. Contiki-NG supports many standard protocols including IPv6, 6LoWPAN, and RPL. We performed a diverse range of simulations using Cooja (Contiki's default emulator) which emulates the real behaviour of IoT devises. Contiki-NG routing stack provides two RPL implementations: RPL-lite and RPL-classic [84]. The former only supports the non-storing mode with minimal functionalities of RPL, while the latter supports both the storing and non-storing modes. To get the full support for both modes, therefore, we used the RPL-classic.

5.4.1 Setup

We distributed a set of $N \in \{100, 200, 300, 400, 500\}$ nodes, uniformly at random in a square region of size $100m \times 100m$. Fig. 5.2 shows a sample network topology in our simulation with $N = 30$ nodes, and a randomly selected root. The nodes are static and operate in both storing and non-storing modes in a single DODAG. To simulate link failure, we used the Unit DISK Graph Model (UDGM) the default setting of Contiki which emulates distance loss for propagation model. We set the transmission range to $15m$ for all nodes including

the root. We used the default objective function of the RPL, which is the Minimum Rank with Hysteresis Objective Function (MRHOF). For the link layer and radio duty cycling (RDC) protocol we used CSMA/CA and ContikiMAC respectively. In ContikiMAC, radio is kept off when there is no incoming or outgoing packet. For the transport layer, we used the User Datagram Protocol (UDP) [48]. To evaluate the stretch factor, we used Cooja mote which provides flexibility to simulate large RPL networks. We also deployed Zolertia Z1 mote [45], which is an MSP430-based board benefiting from a radio chip compatible with the IEEE 802.15.4 link layer protocol to evaluate the average end-to-end delay.

In order to measure the end-to-end delay of the shortest path routes, we enhanced the Contiki NG RPL-classic implementation to extend the source routing support to all nodes[3]. To avoid extra communication overhead, the shortest path routing information (computed using the Floyd-Warshall algorithm [85]) was loaded into all nodes prior to the evaluation of the average end-to-end delay. To consider the impact of congestion on the end-to-end delay, we set all the nodes to periodically send 50 Bytes of data packets to the root, while measuring the end-to-end delay between two randomly selected nodes as source and destination. The 50 Byte size of data packets were selected to prevent packet fragmentation in the network (IEEE 802.15.4 maximum transmit unit is 127 bytes [28]). We further adjusted the transmission rate proportional to the number of nodes in the network to keep all networks at a similar congestion level.

The source sends few data packets to the destination according to RPL, and through the shortest path graph. To evaluate the impact of the root's location, we placed the root at three different positions in the network: corner, center, and random. We performed 50 rounds of emulations for each setting and reported the average in each case. In our simulations, we evaluated the performance of the RPL protocol for the following metrics.

- *Stretch factor* as defined in Definition 2;

[3]The original implementation supports source routing only for the root.

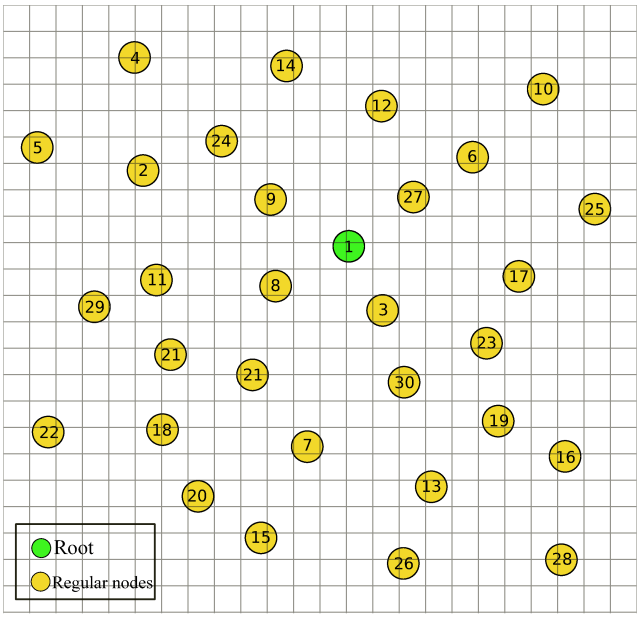

Figure 5.2: A sample network with $N = 30$ nodes randomly placed in a square area. The root (in green) was selected randomly from the nodes.

- *Average end-to-end delay:* the average end-to-end delay of all P2P packets successfully received at the destination;
- *Maximum required memory:* the maximum number of entries in the routing table of non-root nodes.

5.4.2 Evaluation Results

We first compare the stretch factor and end-to-end delay of storing and non-storing modes. We observe that storing mode performs considerably better than the non-storing mode, particularly in low-density networks. The storing mode's improvement, however, comes at the cost of memory to store routing tables. We evaluate this cost by measuring the size of routing tables as the number of nodes increases.

Next, we compare the stretch factor of our proposed solution with that of the storing mode. We observe that our solution considerably outperforms the storing mode even when nodes report only one extra parent. We also compare our solution with NG-RPL, which has the optimum stretch factor.

The results show that our solution can achieve a similar stretch factor as NP-RPL at significantly lower overhead.

Finally, to confirm the result of Theorem 5.4 and Corollary 1, we evaluate the stretch factor of RPL in 2-D grids. The results show that stretch factor in these networks grows with the number of nodes, but remains under five even in extremely large networks. This is in line with our theoretical results that the stretch factor of RPL's non-storing mode is bounded in 2-D grid networks.

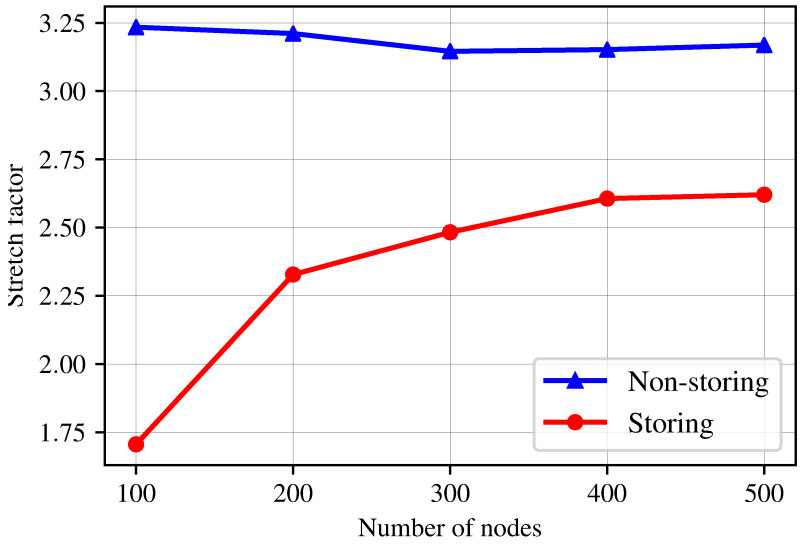

Figure 5.3: The stretch factor of RPL as the network density increases when the root is located at a corner.

Storing v.s. Non-Storing. Fig. 5.3, Fig. 5.4, and Fig. 5.5 show the stretch factor of RPL's modes versus the number of nodes for three different cases concerning the position of the root. Note that when the root is selected randomly (Fig.5.4), the RPL's stretch factor in the non-storing mode is always greater than two. This is consistent with the result of Theorem 5.1.

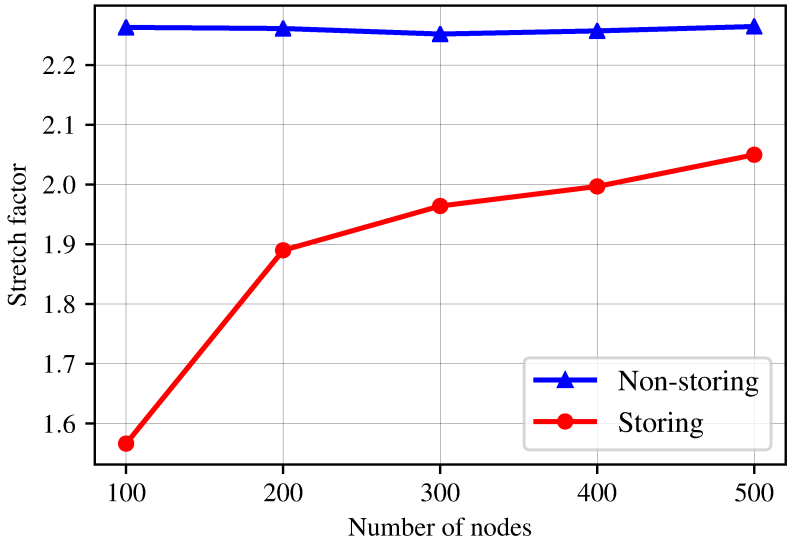

Figure 5.4: The stretch factor of RPL as the network density increases when the root is located at random.

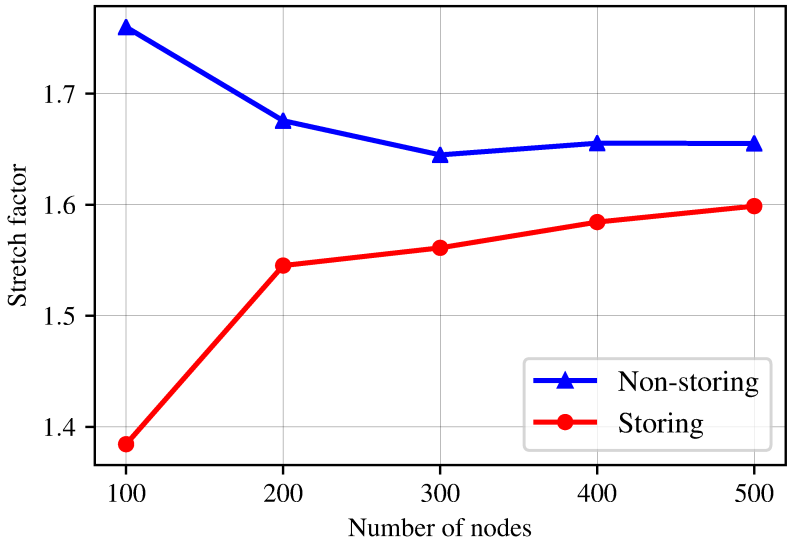

Figure 5.5: The stretch factor of RPL as the network density increases when the root is located at the center.

When the root is located at the center, however, the RPL's stretch factor can be smaller than two (Fig. 5.5). When the root is located in the corner (Fig. 5.3), on the other hand, the stretch factor can be considerably higher than two.

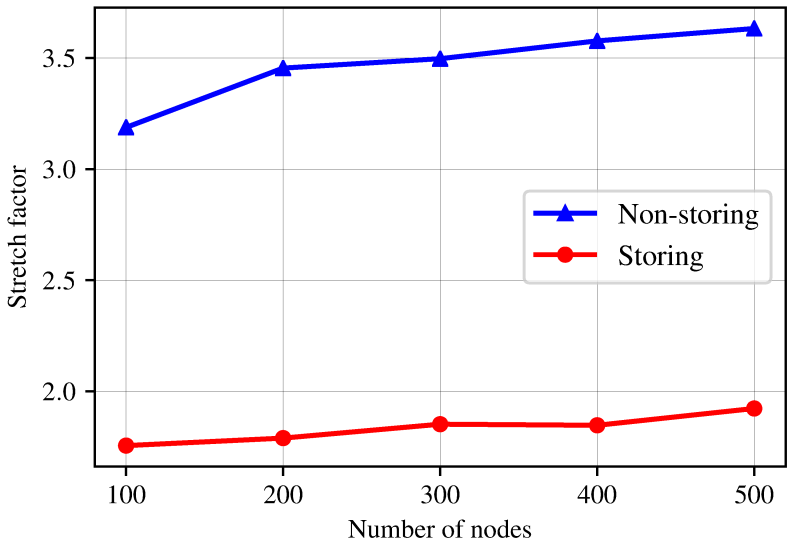

Figure 5.6: The stretch factor of RPL in networks with fixed node density when the root is located at a corner.

Another interesting observation from Fig. 5.3 is that as node density increases, the gap between the stretch factor of the non-storing mode and the storing mode decreases. This indicates that the storing mode becomes less effective as the network becomes denser. This is because as the density increases, the probability that the source and destination have a nearby common ancestor decreases.

Fig. 5.6 compares the RPL's stretch factor in the storing and non-storing modes in networks with fixed density. To fix the density, we increased the network area proportional to the number of nodes. As expected from Theorem 5.1, the stretch factor of RPL's non-storing mode is always greater than two when the root is selected randomly (Fig.5.4 and Fig. 5.7). The stretch factor increases with the number of nodes for both storing and non-storing modes (this increase is slower in the storing mode). This is expected, at least for the non-storing mode, because the average distance of the nodes to the root increases as the network area increases.

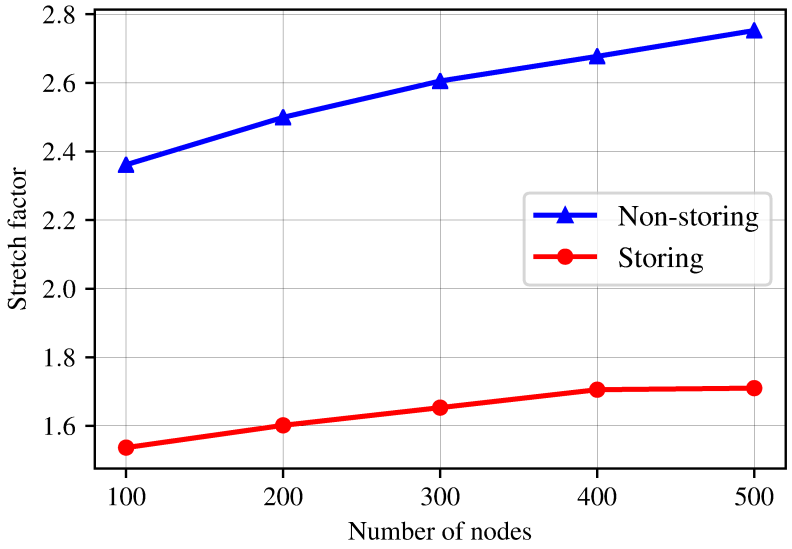

Figure 5.7: The stretch factor of RPL in networks with fixed node density when the root is located at random.

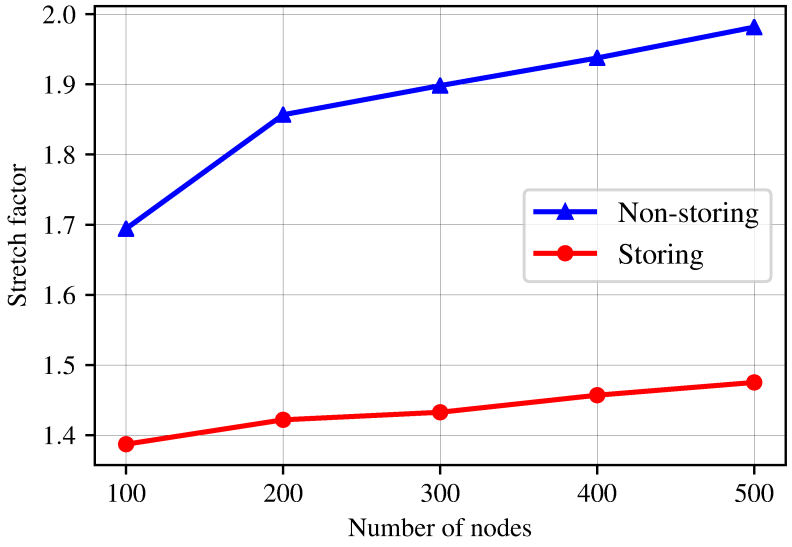

Figure 5.8: The stretch factor of RPL in networks with fixed node density when the root is located at the center.

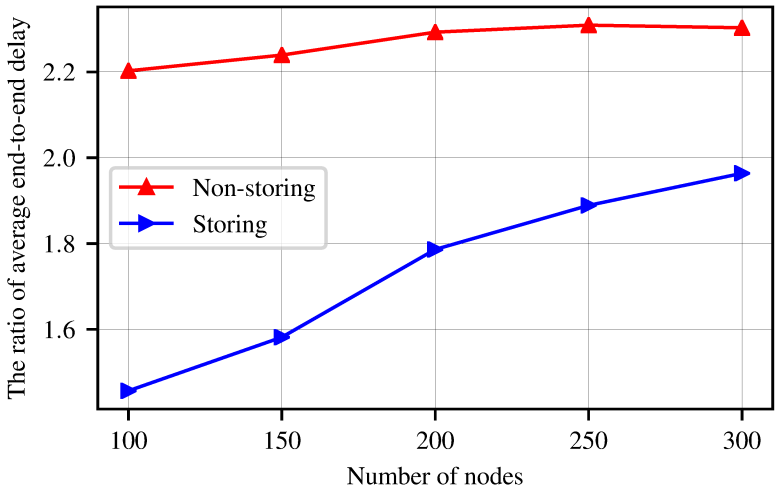

Figure 5.9: The average ratio of the end-to-end delay of RPL paths to that of the shortest paths in dense networks.

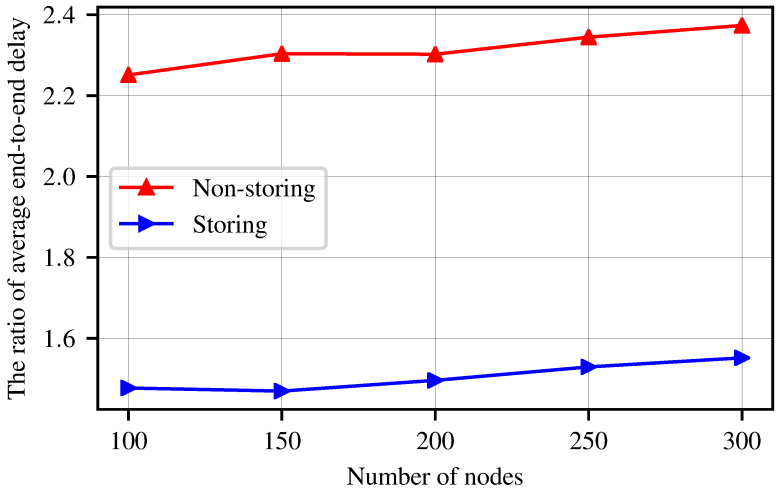

Figure 5.10: The average ratio of the end-to-end delay of RPL paths to that of the shortest paths in networks with fixed density.

Fig. 5.9 and Fig. 5.10 compare the end-to-end delay of RPL paths (in both storing and non-storing modes) to that of the shortest paths. As expected, the average end-to-end delay of non-storing mode is higher than that of the storing mode. It is because RPL paths in the non-storing mode are longer than paths in the storing mode. In addition, all the non-storing mode paths have to go through the congested area near the root, while paths in the storing mode may avoid this congested area depending on the locations of the source and destination. The results plotted in Fig. 5.9 and Fig. 5.10 indicate that the

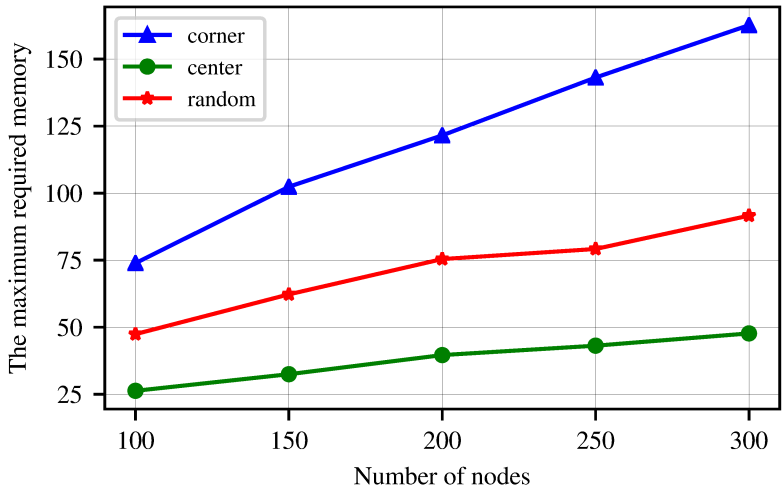

Figure 5.11: The maximum required memory among non-root nodes in dense networks. The three curves correspond to three root's positions: corner, center and random.

end-to-end delay of RPL paths (both in the storing and the non-storing modes) is considerably (up to a factor of 2.3) higher than that of the shortest paths. This is the case in dense networks, as well as networks with fixed density.

Memory Requirement of the Storing Mode. Fig. 5.11 and Fig. 5.12 show the maximum required memory among non-root nodes when the RPL works in the storing mode. The required memory in this figure is equal to the number of entries in the routing table. As shown in the figure, the maximum required memory increases as the number of nodes increases. In addition, the maximum memory requirement is higher when the root is located at a corner of the network compared to the cases where the root is at the center or at a random location in the network.

Our Solution v.s. Storing Mode. Fig. 5.13 compares the stretch factor of our solution to that of RPL's storing mode. In this figure, $\lambda \in \{1, 2, 3\}$ is the maximum number of nodes that our solution selects from the parent set. Our solution chooses the whole parent set if a node does not have enough parents in its parent set. For example, if $\lambda = 3$ and a node has only two parents, the solution selects two parents.

As shown in Fig. 5.13, when each node selects only a single parent (i.e., when $\lambda = 1$), our solution improves the stretch factor of the storing mode by

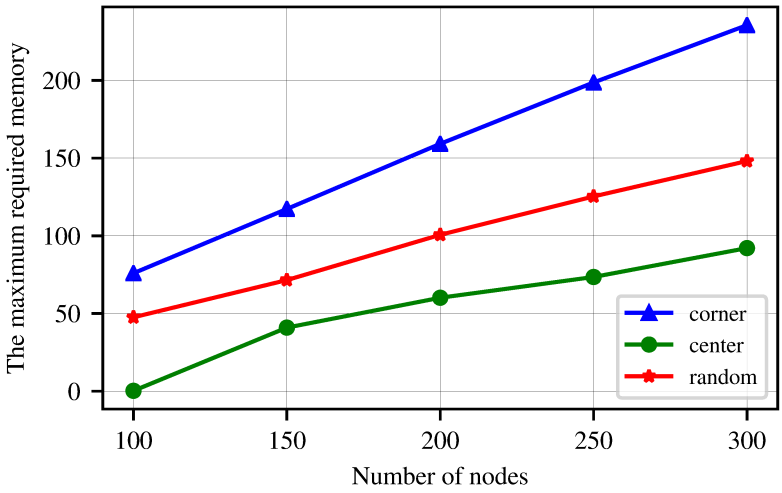

Figure 5.12: The maximum required memory among non-root nodes in networks with fixed density. The three curves correspond to three root's positions: corner, center and random.

up to 31%. When $\lambda = 3$, our solution improves the stretch factor of the storing mode by up to 42%. For instance, in a network with 200 nodes, the stretch factor of our solution for $\lambda = 3$ is about 1.08 while the stretch factor of the RPL's storing mode is about 1.88.

Our Solution v.s. NG-RPL. NG-RPL [83] can achieve the optimum stretch factor of one at the cost of high communication overhead. As mentioned earlier, in NG-RPL each node has to report all of its neighbours to the root, whereas in our solution nodes report only a small number of their DAO parents. As a result, our solution requires to transfer less information than NG-RPL. Another advantage of our solution is that it requires each node to monitor only a small number of its neighbours, whereas NG-RPL requires each node to monitor all its neighbours. In particular, if a node becomes unavailable (e.g. it turns off, or moves out of the transmission range), in NG-RPL all the neighbours of the node must report to the root, while in our solution only a small number of nodes who have selected the unavailable node as a parent need to report the change to the root. To compare the overhead of NG-RPL with that of our solution, we evaluated the ratio of the average number of nodes NG-RPL has to monitor/report to the average number of nodes our solution has to monitor/report. As sown Fig. 5.14, this ratio ranges form 1.6 to 12.4,

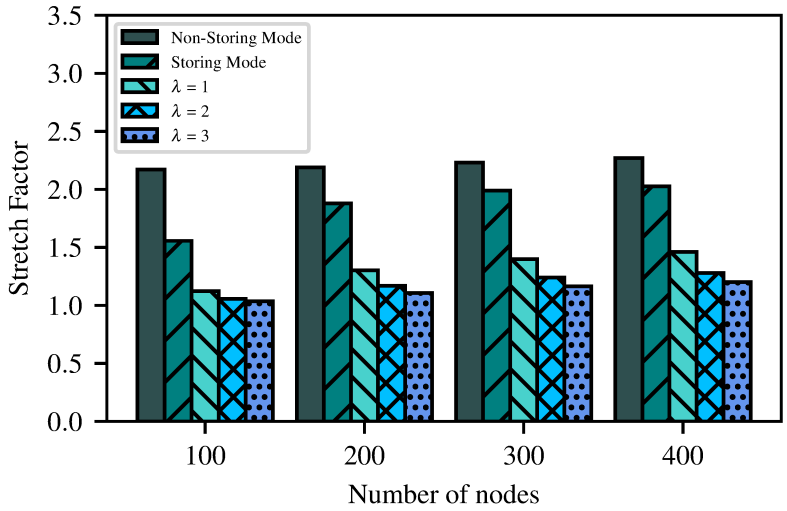

Figure 5.13: The stretch factor of our solution for $\lambda \in \{1, 2, 3\}$, v.s. the stretch factor of RPL's storing and non-storing modes.

depending on the size of the network and the parameter λ in our solution. For instance, when $\lambda = 2$ and $N = 200$, the overhead of NG-RPL is about 420% of that of our solution. For this setting (i.e., $\lambda = 2$ and $N = 200$) the stretch factor of our solution is about 1.16, which is only 16% higher than the optimum stretch factor of one. We remark that, in this setting, the stretch factor of the storing mode is 1.88, which is about 61% higher than the stretch factor of our solution.

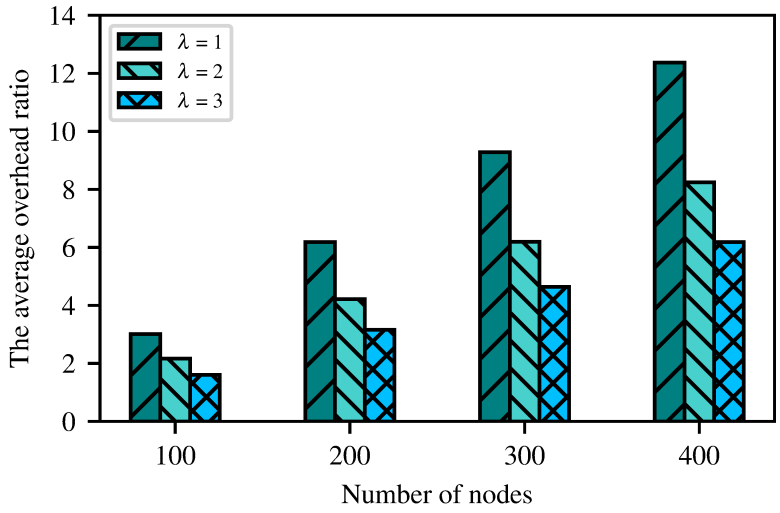

Figure 5.14: The average ratio of NG-RPL's overhead to that of our solution.

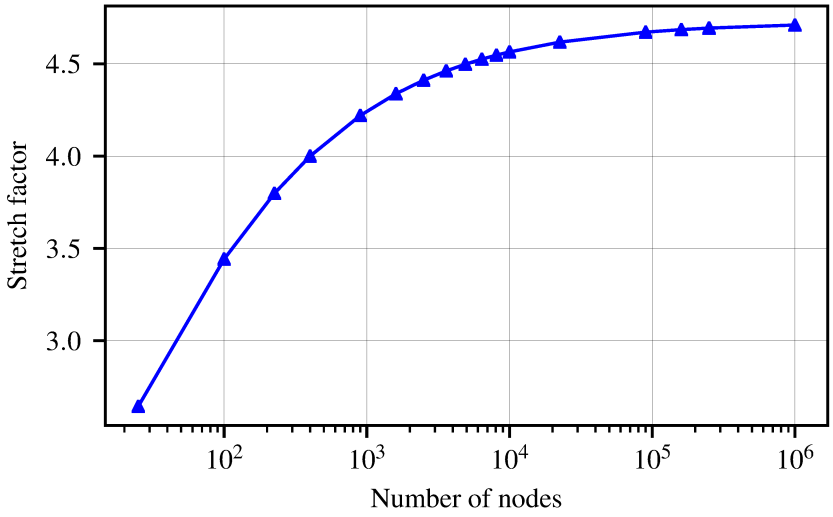

Figure 5.15: The stretch factor of RPL's non-storing mode in 2-D grid networks.

RPL's Stretch Factor in 2-D Grids. In Section 5.2, we proved that the stretch factor of RPL is bounded in 2-D grid networks. To verify this analytical result, we ran a simulation[4] to compute the stretch factor of RPL in 2-D grid networks with a wider range of nodes. We considered extremely large networks in this simulation to observe the growth of stretch factor.

[4]Since Contiki-NG cannot handle extremely large networks, we used a Python program instead of Contiki-NG to compute the stretch factor of RPL in 2-D grid networks.

As shown in Fig. 5.15, stretch factor of RPL increases with the number of nodes in the network. However, the stretch factor remains under five even in extremely large networks. This is in line with the result of Corollary 1, which states that the stretch factor of RPL is bounded in 2-D grid networks.

5.5 Related Work

Several works evaluated different aspects of the RPL protocol. Iova *et al.* [86] presented an analysis of the RPL protocol in terms of reliability and robustness, mobility, and scalability. They concluded that RPL needs improvement in different segments particularly in P2P as emerging IoT applications rely on this type of communication more than before. In another work, Tripathi *et al.* [87] evaluated the RPL protocol in both small and large scale outdoor networks and showed that RPL's P2P communication is not efficient. This evaluation was limited to the RPL's storing mode of operation, and two topologies, where the root is located at the center of the network. Majority of RPL networks, however, work in the non-storing mode. Furthermore, as we show in our analysis, the location of the root impacts the performance of P2P communication.

Since the inception of the RPL standard, several works tried to enhance its P2P routing by, for example, modifying the protocol, or proposing a new method. In the following, we briefly review the related research that aimed to improve the P2P routing support of RPL. We place these works into three categories.

RPL Modification. Kim *et al.* [83] recently proposed NG-RPL to improve the efficiency of RPL P2P routing. NG-RPL assumes that the root has a full view of the network topology. Whenever a P2P packet passes through the root, the root informs both the sender and destination nodes about the shortest path between them. To equip the root with the full network topology, however, NG-RPL requires every node to report all of its neighbours to the root. This imposes a considerable communication overhead. In addition, nodes must constantly monitor all their neighbours, so they can report

topological changes to the root in time. This is a considerable burden on resource-constrained nodes in RPL networks.

In another recent work, Rojas *et al.* proposed a hierarchical routing mechanism called IoTorii [88]. IoTorii assigns a set of meaningful MAC addresses to the nodes in the network to allow each node to have several path options to route its packets. Although this method can provide shorter paths for P2P communication, it requires leaf nodes to store large routing tables. This is not desirable because leaf nodes in RPL networks are typically highly resource constrained

Another work in this category is Enhanced RPL protocol (ERPL) [89]. ERPL provides the same functionality of the RPL's MDAO with a lower control message overhead. The idea of ERPL is to use DIO messages as a proxy to obtain the address of neighbours. In ERPL, when a node receives a DIO message, it extracts the sender's address from the message and then updates/adds an entry in its forwarding table. This allows discovery of one hop neighbours without relying on broadcasting MDAO messages.

Reactive Protocols. Many reactive routing protocols based on the Adhoc On-demand Distance Vector (AODV) were proposed to improve the RPL's P2P routing in LLNs. NST-AODV [90] and LOAD-ng [91] are two examples of such protocols. These protocols try to reduce the overhead of the AODV protocol to make it more suitable for LLNs. To discover P2P paths, however, these protocols require each node to flood the network with request/response messages. This imposes a large overhead specially in large and dense networks.

Another protocol in this category is P2P-RPL [92]. P2P-RPL is not based on AODV, yet it finds P2P routes to other nodes on demand by creating a temporary routing tree rooted at the source node. This process requires all nodes to take part in the formation of temporary routing tree in the discovery stage, hence generates a significant number of control messages, and increases the energy consumption of nodes [7].

Location-Based Methods. Zhao *et al.* [93] presented the Energy-Efficient Region-based Routing Protocol for LLNs (ER-RPL). ER-RPL takes advantage of the existing DODAG structure of RPL for P2P route discovery [93]. In par-

ticular, ER-RPL utilizes the region feature of static networks to discover P2P routes between a pair of source and destination with engaging only a small portion of nodes in the network. Efficiency of discovered P2P routes in ER-RPL highly depends on the accuracy of estimated location, which has a direct relation with the network configurations.

A Geographic location-based Dynamic Opportunistic routing protocol (GDO-RPL) for point-to-point communication was proposed by Chakraborty *et al.* [94] to reduce the memory overhead and mitigate mobility issues in P2P communication. In their method, a pair of nodes find a direct route between each other using geographical location information. Jung *et al.* proposed the location aware P2P-RPL (LA P2P-RPL) to improve the efficiency and reliability of P2P routes along the network [95]. As the previous method, their method depends on the location information of the nodes. Such location information may be inaccurate or impossible to obtain for some IoT devices.

5.6 Conclusion

In this chapter, we compared the length and end-to-end delay of RPL's P2P paths with those of their corresponding shortest paths. We proved that RPL's P2P paths in the non-storing mode (a common mode of operation for RPL) can be considerably longer than their corresponding shortest paths. In particular, we proved that the stretch of RPL's non-storing mode is at least two in any network, and can be considerably higher in some topologies such as linear networks and 2-D grid networks. Our simulation results showed that the RPL's storing mode can, to some extent, provide shorter P2P paths (depending on the network's density, size, and location of the root) than the non-storing mode at the expense of memory and communication overhead to store and maintain routing tables. To improve the P2P routing in RPL, we then proposed a simple and fully compatible solution. We showed that our solution can provide significantly shorter P2P paths than the RPL's storing mode, without requiring nodes to store and maintain a routing table. In addition, we showed that our solution imposes significantly less overhead than

NG-RPL, while achieving comparable stretch factors.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Low-power and Lossy Networks (LLNs), a key component of IoT networks, are composed of various resource-constrained devices with limited energy, memory, and processing power. RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) is the standard routing protocol for LLNs, which meets many requirements of these networks. RPL, however, has several shortcomings in terms of security and P2P routing efficiency. In this thesis, we presented three research projects related to RPL's security and P2P routing.

In Chapter 3, we introduced the DAO induction attack, a novel routing attack against the RPL protocol. In the DAO induction attack an internal attacker increments its DTSN number periodically to trigger redundant DAO transmissions in the network. Through extensive simulations, we showed that the attack negatively affects network performance and power consumption, in particular when the network operates in the non-storing mode. To alleviate, we proposed a two-fold lightweight solution to detect the attack and identify the attacker. The proposed solution can be implemented on common IoT platforms with minimum changes to the RPL protocol. Moreover, it imposes no overhead on IoT devices, requires no external monitoring nodes, and can quickly detect the attack even when the network experiences high packet data loss rates. We proved that unless the attacker cuts the network into two disjoint parts, the proposed mitigation always detects the attack and find the adversary.

In Chapter 4, we addressed the sender's authentication problem of RPL and proposed a lightweight authentication method for RPL based on the well known Blom key pre-distribution. We build a lightweight Blom scheme by using random binary matrices. We showed that random binary matrices can significantly decrease the computation cost of Blom for constrained devices. We compared the overhead of our proposed scheme with original Blom scheme using MSP430 processor family. The results showed that the computation of proposed scheme is much less than original Blom. We showed that our proposed Blom scheme preserve a good resiliency against capturing attack, using extensive simulations for large RPL settings. Finally, we proved that the adversary does not have any better strategy than a brute force to find the minimum set of nodes with dependent private keys.

In Chapter 5, we studied the performance of the P2P routing in RPL. We systematically formulated the average stretch factor criteria for RPL. We proved that the stretch of RPL's non-storing mode is at least two in any network, and can be considerably higher in some topologies such as linear networks and 2-D grid networks. Using extensive simulations, we showed that the RPL's storing mode can provide shorter P2P paths (depending on the network's density, size, and location of the root) than the non-storing mode at the expense of memory and communication overhead to store and maintain routing tables. To enhance the P2P routing in RPL, we proposed a simple and fully compatible solution. We showed that the proposed solution can achieve considerably shorter P2P paths than RPL's storing mode, without requiring nodes to maintain a routing table. In addition, we showed that our solution imposes significantly less overhead than NG-RPL, while achieving comparable stretch factors.

6.2 Further Extensions and Future Research

A valuable future work is to implement and evaluate the DAO induction attack (and similar attacks in the literature) in real-world multi-hop networks. Another interesting direction for future research is to combine the DAO in-

duction attack with other attacks such as various Sybil attack. Such hybrid attacks can be more severe and harder to mitigate. In terms of mitigation, a distributed external monitoring system can be studied to provide a strong countermeasure against the severe versions of the DAO induction attack, i.e., when many nodes are compromised or when the DAO induction attack is combined with other attacks.

The proposed lightweight Blom-based authentication mode in Chapter 3 can provide authentication for unicast messages and local multicast packets. It is not, however, suitable for source authenticating when it comes to network-wide broadcasts. To authenticate the source in a network-wide broadcast, we think key predistribution schemes can be still useful if designed carefully. For example, consider a network with at most $\binom{10}{2} = 45$ nodes, and suppose that we want to enable nodes to authenticate root's broadcast messages. To achieve this, the root can generate ten secret keys, and provide each node with a unique pair of these keys. To broadcast a message to the whole network, the root creates ten message authentication codes (MACs), and add them to the message. When a node receives the message, it verifies two of the MACs (corresponding to the keys it has), and accepts the message if both verifications pass. Notice that in the above scheme, no single node can convince another node that it is the root. This is because each node has a unique pair of secret keys. However, if two or more nodes collide, together they may create a message and convince another node that the message is from the root. This shows that more work is needed to make the above idea resilient against collusion attacks, or cases where multiple nodes are compromised.

In our proposed solution in Chapter 5, all the P2P packets, except the first packet in a P2P communication, travel a much shorter path than the path they travel in the RPL's storing mode. An interesting research direction is to find a way for the first packet to travel a short path too. To achieve this, a source node can proactively contact the root to inquire about P2P paths to its desired destinations. This approach, however, can increase the network overhead if many source nodes regularly contact the root for P2P path information.

References

[1] A. S. Baghani, S. Rahimpour, and M. Khabbazian, "The dao induction attack against the rpl-based internet of things," in *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2020, pp. 1–5.

[2] A. S. Baghani, S. Rahimpour, and M. Khabbazian, "The dao induction attack: Analysis and countermeasure," *IEEE Internet of Things Journal*, pp. 1–14, 2021.

[3] J. Vasseur, *Terms Used in Routing for Low-Power and Lossy Networks*, RFC 7102, Jan. 2014. DOI: 10.17487/RFC7102. [Online]. Available: https://rfc-editor.org/rfc/rfc7102.txt.

[4] A. Raoof, A. Matrawy, and C.-H. Lung, "Routing attacks and mitigation methods for RPL-based internet of things," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 2, pp. 1582–1606, 2019.

[5] A. Mayzaud, R. Badonnel, and I. Chrisment, "A taxonomy of attacks in RPL-based internet of things," *International Journal of Network Security*, vol. 18, no. 3, pp. 459–473, 2016.

[6] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, and J. Schönwälder, "A study of RPL DODAG version attacks," in *IFIP international conference on autonomous infrastructure, management and security*, Springer, 2014, pp. 92–104.

[7] M. Zhao, A. Kumar, P. H. J. Chong, and R. Lu, "A comprehensive study of RPL and P2P-RPL routing protocols: Implementation, challenges and opportunities," *Peer-to-Peer Networking and Applications*, vol. 10, no. 5, pp. 1232–1256, 2017.

[8] R. Blom, "An optimal class of symmetric key generation systems," in *Workshop on the Theory and Application of of Cryptographic Techniques*, Springer, 1984, pp. 335–338.

[9] H. Singh and D. Singh, "Taxonomy of routing protocols in wireless sensor networks: A survey," in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, IEEE, 2016, pp. 822–830.

[10] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[11] S. Li, L. Da Xu, and S. Zhao, "The internet of things: A survey," *Information Systems Frontiers*, vol. 17, no. 2, pp. 243–259, 2015.

[12] C. Bormann, M. Ersue, and A. Keränen, *Terminology for Constrained-Node Networks*, RFC 7228, May 2014. DOI: 10.17487/RFC7228. [Online]. Available: https://rfc-editor.org/rfc/rfc7228.txt.

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

[14] G. Porcu, J. Buron, and A. Brandt, *Home automation routing requirements in low-power and lossy networks*, RFC 5826, Apr. 2010. DOI: 10.17487/RFC5826. [Online]. Available: https://rfc-editor.org/rfc/rfc5826.txt.

[15] K. Pister, P. Thubert, S. Dwars, and T. Phinney, "Industrial routing requirements in low-power and lossy networks," 2009.

[16] T. Watteyne, T. Winter, D. Barthel, and M. Dohler, *Routing Requirements for Urban Low-Power and Lossy Networks*, RFC 5548, May 2009. DOI: 10.17487/RFC5548. [Online]. Available: https://rfc-editor.org/rfc/rfc5548.txt.

[17] J. Martocci, P. Mil, N. Riou, and W. Vermeylen, *Building Automation Routing Requirements in Low-Power and Lossy Networks*, RFC 5867, Jun. 2010. DOI: 10.17487/RFC5867. [Online]. Available: https://rfc-editor.org/rfc/rfc5867.txt.

[18] A. Tavakoli and S. Dawson-Haggerty, "Overview of Existing Routing Protocols for Low Power and Lossy Networks," Internet Engineering Task Force, Internet-Draft draft-ietf-roll-protocols-survey-07, Apr. 2009, Work in Progress, 26 pp. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-roll-protocols-survey-07.

[19] R. Coltun, D. Ferguson, J. Moy, and A. Lindem, "OSPF for IPv6," A, Tech. Rep., 2008.

[20] D. Oran, "OSI IS-IS intra-domain routing protocol," Tech. Rep., 1990.

[21] T. H. Clausen and P. Jacquet, *Optimized Link State Routing Protocol (OLSR)*, RFC 3626, Oct. 2003. DOI: 10.17487/RFC3626. [Online]. Available: https://rfc-editor.org/rfc/rfc3626.txt.

[22] T. H. Clausen, C. Dearlove, P. Jacquet, and U. Herberg, *The Optimized Link State Routing Protocol Version 2*, RFC 7181, Apr. 2014. DOI: 10.17487/RFC7181. [Online]. Available: https://rfc-editor.org/rfc/rfc7181.txt.

[23] F. L. Templin, R. Ogier, and M. S. Lewis, *Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)*, RFC 3684, Feb. 2004. DOI: 10.17487/RFC3684. [Online]. Available: https://rfc-editor.org/rfc/rfc3684.txt.

[24] S. R. Das, C. E. Perkins, and E. M. Belding-Royer, *Ad hoc On-Demand Distance Vector (AODV) Routing*, RFC 3561, Jul. 2003. DOI: 10.17487/RFC3561. [Online]. Available: https://rfc-editor.org/rfc/rfc3561.txt.

[25] G. S. Malkin, *RIP Version 2*, RFC 2453, Nov. 1998. DOI: 10.17487/RFC2453. [Online]. Available: https://rfc-editor.org/rfc/rfc2453.txt.

[26] Y.-C. Hu, D. A. Maltz, and D. B. Johnson, *The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4*, RFC 4728, Feb. 2007. DOI: 10.17487/RFC4728. [Online]. Available: https://rfc-editor.org/rfc/rfc4728.txt.

[27] I. D. Chakeres and C. E. Perkins, "Dynamic manet on-demand routing protocol," *ETF Internet Draft, draft-ietf-manet-dymo-12. txt*, 2008.

[28] IEEE, "IEEE std. 802.15.4: Wireless medium access control and physical layer specifications for low rate wireless personal area networks (LR-WPANs)," 2015.

[29] D. Barthel, J. Vasseur, K. Pister, M. Kim, and N. Dejean, *Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks*, RFC 6551, Mar. 2012. DOI: 10.17487/RFC6551. [Online]. Available: https://rfc-editor.org/rfc/rfc6551.txt.

[30] O. Gaddour, A. Koubâa, N. Baccour, and M. Abid, "OF-FL: QoS-aware fuzzy logic objective function for the RPL routing protocol," in *12th International symposium on modeling and optimization in mobile, ad hoc, and wireless networks (WiOpt)*, IEEE, 2014, pp. 365–372.

[31] O. Gnawali and P. Levis, *The Minimum Rank with Hysteresis Objective Function*, RFC 6719, Sep. 2012. DOI: 10.17487/RFC6719. [Online]. Available: https://rfc-editor.org/rfc/rfc6719.txt.

[32] P. Levis and T. H. Clausen, "The trickle algorithm," 2011.

[33] L. Wallgren, S. Raza, and T. Voigt, "Routing attacks and countermeasures in the RPL-based internet of things," *International Journal of Distributed Sensor Networks*, vol. 9, no. 8, pp. 314–326, 2013.

[34] B. Ghaleb, A. Al-Dubai, E. Ekonomou, M. Qasem, I. Romdhani, and L. Mackenzie, "Addressing the DAO insider attack in RPL's internet of things networks," *IEEE Communications Letters*, vol. 23, no. 1, pp. 68–71, 2019.

[35] P. Perazzo, C. Vallati, G. Anastasi, and G. Dini, "Dio suppression attack against routing in the internet of things," *IEEE Communications Letters*, vol. 21, no. 11, pp. 2524–2527, 2017. DOI: 10.1109/LCOMM.2017.2738629.

[36] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time intrusion detection in the internet of things," *Ad hoc networks*, vol. 11, no. 8, pp. 2661–2674, 2013.

[37] A. Dvir, L. Buttyan, *et al.*, "VeRA-version number and rank authentication in RPL," in *Mobile Adhoc and Sensor Systems (MASS), IEEE 8th International Conference on*, 2011, pp. 709–714.

[38] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks," *ACM Transactions on Information and System Security (TISSEC)*, vol. 8, no. 2, pp. 228–258, 2005.

[39] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[40] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[41] D. Kumar, K. Shen, B. Case, D. Garg, G. Alperovich, D. Kuznetsov, R. Gupta, and Z. Durumeric, "All things considered: An analysis of IoT devices on home networks," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1169–1185.

[42] W. Zhou, Y. Jia, A. Peng, Y. Zhang, and P. Liu, "The effect of IoT new features on security and privacy: New threats, existing solutions, and challenges yet to be solved," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1606–1616, 2018.

[43] 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*, 2004, pp. 455–462.

[44] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*, 2006.

[45] *Zolertia Z1 Mote Module*, [Online]. Available: https://zolertia.io.

[46] O. Gnawali and P. Levis, "The minimum rank with hysteresis objective function," *RFC 6719*, 2012.

[47] A. Dunkels, *The ContikiMAC radio duty cycling protocol*, Swedish Institute of Computer Science, Technical Report, Dec. 2011.

[48] *User Datagram Protocol*, RFC 768, Aug. 1980. DOI: 10.17487/RFC0768. [Online]. Available: https://rfc-editor.org/rfc/rfc768.txt.

[49] L. Sitanayah, C. J. Sreenan, and S. Fedor, "A cooja-based tool for maintaining sensor network coverage requirements in a building," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, 2013, pp. 1–2.

[50] 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," *International Journal of Communication Systems*, vol. 25, no. 9, pp. 1189–1212, 2012.

[51] A. Le, J. Loo, K. Chai, and M. Aiash, "A specification-based IDS for detecting attacks on RPL-based network topology," *Information*, vol. 7, no. 2, p. 25, 2016.

[52] P. O. Kamgueu, E. Nataf, and T. D. Ndie, "Survey on RPL enhancements: A focus on topology, security and mobility," *Computer Communications*, vol. 120, pp. 10–21, 2018.

[53] P. Perazzo, C. Vallati, G. Anastasi, and G. Dini, "DIO suppression attack against routing in the internet of things," *IEEE Communications Letters*, vol. 21, no. 11, pp. 2524–2527, 2017.

[54] C. Pu, "Spam DIS attack against routing protocol in the internet of things," in *International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, February 18-21*, 2019, pp. 73–77.

[55] 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 *Computers and Communications (ISCC), IEEE Symposium on*, 2013, pp. 789–794.

[56] 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*, vol. 13, no. 10, pp. 3685–3692, 2013.

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

[58] Z. Karakehayov, "Using reward to detect team black-hole attacks in wireless sensor networks," *Wksp. Real-World Wireless Sensor Networks*, pp. 20–21, 2005.

[59] B. Revathi and D. Geetha, "A survey of cooperative black and gray hole attack in MANET," *International Journal of Computer Science and Management Research*, vol. 1, no. 2, pp. 205–208, 2012.

[60] 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*, 2012, pp. 157–162.

[61] H. Perrey, M. Landsmann, O. Ugus, T. C. Schmidt, and M. Wählisch, "TRAIL: Topology authentication in RPL," *arXiv:1312.0984*, 2013.

[62] A. Mayzaud, R. Badonnel, and I. Chrisment, "A distributed monitoring strategy for detecting version number attacks in RPL-based networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 472–486, 2017.

[63] C. Pu, "Energy depletion attack against routing protocol in the internet of things," in *16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2019, pp. 1–4.

[64] K. Weekly and K. Pister, "Evaluating sinkhole defense techniques in RPL networks," in *Network Protocols (ICNP), 20th IEEE International Conference on*, 2012, pp. 1–6.

[65] O. Garcia-Morchon, S. Kumar, S. Keoh, R. Hummen, and R. Struik, "Security considerations in the IP-based internet of things," *Internet Engineering Task Force*, 2013.

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

[67] 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 *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, 2010, pp. 888–895.

[68] K. Zhang, X. Liang, R. Lu, and X. Shen, "Sybil attacks and their defenses in the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 372–383, 2014.

[69] A. Rghioui, A. Khannous, and M. Bouhorma, "Denial-of-service attacks on 6LoWPAN-RPL networks: Threats and an intrusion detection system proposition," *Journal of Advanced Computer Science & Technology*, vol. 3, no. 2, p. 143, 2014.

[70] P. Varga, S. Plosz, G. Soos, and C. Hegedus, "Security threats and issues in automation IoT," in *Factory Communication Systems (WFCS), 2017 IEEE 13th International Workshop on*, 2017, pp. 1–6.

[71] A. Raoof, C.-H. Lung, and A. Matrawy, "Securing RPL using network coding: The chained secure mode (CSM)," *IEEE Internet of Things Journal*, vol. Early Access, pp. 1–11, 2021.

[72] C. Pu, "Sybil attack in RPL-based internet of things: Analysis and defenses," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4937–4949, 2020.

[73] A. S. Baghani, S. Rahimpour, and M. Khabbazian, "The DAO induction attack against the RPL-based Internet of things," in *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2020, pp. 1–5.

[74] F. Ahmed and Y.-B. Ko, "Mitigation of blackhole attacks in routing protocol for low power and lossy networks," *Security and Communication Networks*, vol. 9, no. 18, pp. 5143–5154, 2016.

[75] J.-C. Bajard, L. Imbert, C. Negre, and T. Plantard, "Efficient multiplication in GF(p^k) for elliptic curve cryptography," in *Proceedings 2003 16th IEEE Symposium on Computer Arithmetic*, IEEE, 2003, pp. 181–187.

[76] A. Luykx, B. Preneel, E. Tischhauser, and K. Yasuda, "A MAC mode for lightweight block ciphers," in *International Conference on Fast Software Encryption*, Springer, 2016, pp. 43–59.

[77] I. Dumer, D. Micciancio, and M. Sudan, "Hardness of approximating the minimum distance of a linear code," *IEEE Transactions on Information Theory*, vol. 49, no. 1, pp. 22–37, 2003.

[78] A. Vardy, "Algorithmic complexity in coding theory and the minimum distance problem," in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, ser. STOC '97, El Paso, Texas, USA: Association for Computing Machinery, 1997, pp. 92–109.

[79] D. Airehrour, J. A. Gutiérrez, and S. K. Ray, "*SecTrust*-rpl: A secure trust-aware RPL routing protocol for internet of things," *Future Gener. Comput. Syst.*, vol. 93, pp. 860–876, 2019.

[80] P. Kaliyar, W. B. Jaballah, M. Conti, and C. Lal, "LiDL: Localization with early detection of sybil and wormhole attacks in IoT Networks," *Comput. Secur.*, vol. 94, pp. 101 849–101 862, 2020.

[81] F. Medjek, D. Tandjaoui, M. R. Abdmeziem, and N. Djedjig, "Analytical evaluation of the impacts of Sybil attacks against RPL under mobility," in *2015 12th International Symposium on Programming and Systems (ISPS)*, IEEE, 2015, pp. 1–9.

[82] S. Murali and A. Jamalipour, "A lightweight intrusion detection for sybil attack under mobile RPL in the internet of things," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 379–388, 2020.

[83] Y. Kim and J. Paek, "NG-RPL for efficient P2P routing in low-power multihop wireless networks," *IEEE Access*, vol. 8, pp. 182 591–182 599, 2020.

[84] A. Kurniawan, *Practical Contiki-NG: Programming for wireless sensor networks*, 1st. USA: Apress, 2018, ISBN: 1484234073.

[85] R. W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, vol. 5, no. 6, pp. 345–345, Jun. 1962.

[86] O. Iova, P. Picco, T. Istomin, and C. Kiraly, "RPL: The routing standard for the internet of things... or is it?" *IEEE Communications Magazine*, vol. 54, no. 12, pp. 16–22, 2016.

[87] J. Tripathi, J. C. de Oliveira, and J. P. Vasseur, "A performance evaluation study of RPL: Routing protocol for low power and lossy networks," in *2010 44th Annual Conference on Information Sciences and Systems (CISS)*, 2010, pp. 1–6.

[88] E. Rojas, H. Hosseini, C. Gomez, D. Carrascal, and J. R. Cotrim, "Outperforming RPL with scalable routing based on meaningful mac addressing," *Ad Hoc Networks*, vol. 114, pp. 102 433–102 446, 2021.

[89] M. O. Farooq and D. Pesch, "Reduced overhead routing in short-range low-power and lossy wireless networks," *Sensors*, vol. 19, no. 5, pp. 1240–1259, 2019.

[90] C. Gomez, P. Salvatella, O. Alonso, and J. Paradells, "Adapting AODV for IEEE 802.15.4 mesh sensor networks: Theoretical discussion and performance evaluation in a real environment," in *2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2006, pp. 159–170.

[91] T. Clausen, J. Yi, and U. Herberg, "Lightweight on-demand ad hoc distance-vector routing-next generation (LOADng): Protocol, extension, and applicability," *Computer Networks*, vol. 126, pp. 125–140, 2017.

[92] M. Goyal, E. Baccelli, M. Philipp, A. Brandt, and J. Martocci, *Reactive Discovery of Point-to-Point Routes in Low-Power and Lossy Networks*, RFC 6997, Aug. 2015.

[93] M. Zhao, I. W.-H. Ho, and P. H. J. Chong, "An energy-efficient region-based RPL routing protocol for low-power and lossy networks," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1319–1333, 2016.

[94] M. Chakraborty, A. Spano, and A. Cortesi, "Geographic location based dynamic and opportunistic RPL for distributed networks," in *IFIP International Conference on Computer Information Systems and Industrial Management*, Springer, 2019, pp. 120–131.

[95] J. Jung, Y. Choi, and Y. Kwon, "Location-aware point-to-point RPL in indoor IR-UWB networks," *Electronics*, vol. 9, no. 5, p. 861, 2020.