STIR: Preventing Routing Table Overload Attacks in RPL-based IoT Networks

Marin Duroyon[1] , **Mauro Conti**[1] , **Chhagan Lal**[1]
[1]Delft University of Technology (TU Delft)

Abstract

Routing Protocol for Low Power and Lossy Networks (RPL) is a routing protocol for Internet of Things (IoT) devices with limited resources. As IoT is becoming prevalent, it is important to secure the underlying protocols that compose it such as RPL. This paper sought to avoid an RPL-specific routing attack by modifying the protocol's functionalities. As a result, STIR is a novel method improving memory efficiency of routing tables in RPL's storing mode of operation to prevent *Routing Table Overload* attacks. STIR assigns IPv6 addresses in a specific way to form clusters of addresses, thus resulting in routing tables with sizes proportional to the node's number of sub-DODAGs. The proposed contribution increases memory efficiency in RPL's storing mode, therefore preventing routing table overload attacks. STIR necessitates few protocol modifications and additional control messages making it an ideal preventive method.

1 Introduction

The Internet of Things (IoT) is expanding at an unprecedented rate and is unlikely to slow down [1]. The increase in the appearance of 'smart' objects in modern life originates from their benefits. For example, the healthcare industry has a significant market opportunity in IoT as it helps insurance companies, patients, physicians and hospitals [2]. According to *Wipro*, the healthcare IoT infrastructure strongly reduces costs and improves the efficiency of treatment [2]. For instance, IoT devices allow the systematic monitoring of the evolution of patients' health conditions. With the increased use of such devices, more scrutiny is dedicated into ensuring privacy and security in IoT networks.

Different protocols allow communication within Wireless Sensor Networks (WSN), networks of sensor-based IoT devices. The utilized protocols correspond to different needs demanded by the infrastructure. This research paper will explore the routing protocol used in low power and lossy networks (LLN) appropriately named: *Routing Protocol for Low Power and Lossy Networks* (RPL). Such networks consist of so-called 'constrained nodes,' which have limited memory, processing power, or energy consumption [3]. Therefore, the mitigations to existing attacks must strike a careful balance between performance and security. Solutions must be complete enough to defend the attack, but over-engineering might hinder the performance of the RPL protocol.

Previous works have analyzed and identified solutions to existing RPL-specific vulnerabilities such as DODAG Inconsistency, Replay, or Version attacks. For example, *Verma et al.* offers a solution that "determines when to stop resetting the trickle timer to save the node's resources" [4, p. 7] thus preventing control message overheads caused by DIS flooding attacks. Moreover, *Le et al.* designed a *Specification-based Intrusion Detection System (IDS)* which is effective in detecting the topology attacks of RPL [5]. Topology vulnerabilities describe a large set of RPL-specific attacks such as DIS, Neighbor, Rank, Local Repair, and Sinkhole.

While RPL is at the forefront of routing protocols for LLNs and the center of research for WSN protocols, security vulnerabilities lack specific solutions and more research is required. According to *Raoof et al.*, there is an urgent need to research solutions to "DIS attack, neighbor attack, RPL storing mode attacks, DODAG inconsistency attacks, and replay attacks in dynamic networks or with mobile nodes" [6, p. 20]. There are research gaps in existing threats to RPL networks, therefore, it is crucial to analyze different attacks. The goal of this paper is to understand an RPL storing mode attack, the routing table overload attack, and to propose a prevention method, STIR, for such vulnerability. This attack has been mentioned as a resource exhausting issue [7], but no direct mitigations have been proposed for it at the time of writing. It is therefore important to cover such basis with the paper's contribution. The storing mode of operation, which will be discussed in the section 2.2, is an RPL-specific property simplifying the protocol but opening the door to various attacks.

This paper contributes to existing research by discussing a novel protocol modification that prevents the RPL-specific *Routing Table Overload* attack by simplifying routing table entries. The goal of STIR is to use specifically assigned IPv6 addresses to perform address coalescing. Following a search tree pattern, efficient storage and data packet routing avoids memory overflow caused by the routing table overload attack.

This report seeks to explore: *How can RPL protocol's functionalities be modified to avoid an RPL-specific routing attack*? This research question will be done by identifying an

RPL-specific vulnerability and designing a possible solution to mitigate or prevent it.

Firstly, this paper will describe background information on IoT and RPL in section 2. We will then discuss related works in the domain of RPL security vulnerabilities and the attack specific to the paper in section 3. In addition, section 4 explores the paper's contribution by providing details of the attack and solution with examples. Furthermore, a performance analysis section, subsection 4.4, will seek to analyze the impact of the prevention on RPL networks. Section 5 will review the methodology and discuss future possible research for STIR. Finally, section 6 will consider responsible research in the context of this project and section 7 will draw an overarching conclusion on the topic in question.

2 Background

2.1 The Internet of Things

The IoT infrastructure permits the collection of digital data on previously inanimate objects. For example, the GSM Association made a point on how 'smart' trash cans gathered data which turned around a famously "inefficient and resource-intensive industry" such as waste collection [8]. IoT permits the analysis of information coming from traditionally non-computerized sources. Additional digital data in urban infrastructures would further allow to predict and simulate entire cities [9], leading to efficient civil life and prosperous development. While IoT provides many benefits, it also faces several challenges including security as the networks can be the victim of cyberattacks. While the data can be inconsequential to a malicious attacker at times, the integrity of the network might bear larger responsibilities. Smart devices are as useful as the trust placed upon them. Hence, their reliability is of the utmost importance.

2.2 Routing Protocol for Low Power and Lossy Networks

The Routing Protocol for Low Power and Lossy Networks (RPL) was developed as a layer 3 (OSI model) routing protocol serving the needs of "resource-constrained devices in industrial, home, and urban environments" [10, p. 1]. It is built on top of the adaptation layer for IPv6 low-power wireless personal area networks (6LoWPAN). This protocol permits various types of intercommunication, such as point-to-point, multipoint-to-point, and point-to-multipoint, between nodes with unstable links called lossy networks.

RPL Concepts

To adhere to its purpose, routing for Low Power and Lossy Networks (LLN), RPL is a distance vector routing protocol utilizing Destination Oriented Directed Acyclic Graphs (DODAG) as a topology [11]. DODAGs allow RPL to optimize traffic through a graph following a data sink format [3], the sink node is called the root. Multiple Directed Acyclic Graphs (DAG) can be merged together to form a DODAG which is the backbone of the routing protocol.

Multiple types of control messages are used in RPL. The DODAG Information Solicitation (DIS) message obtains information about nearby DODAGs from an RPL node [3]. The DODAG Information Object (DIO) carries data about the DODAG of the sender node. Moreover, the DIO control message is utilized as an answer to a DIS message. Finally, Destination Advertisement Object (DAO) and Destination Advertisement Object Acknowledgement (DAO-ACK) are used to share destination information and acknowledge DAO messages respectively [3]. DAO messages are sent upwards, to follow the DODAG topology, and announce nodes in their sub-DODAG [12]. These messages declare children nodes in the RPL instance from the source node.

RPL has two modes of operation: *storing* and *non-storing* mode. In the non-storing mode, the DODAG's root is the main traffic hub and routes all data packets [11]. On the contrary, the storing mode allows routing through any node. Furthermore, in storing mode, RPL data packets must travel upwards towards a common ancestor before taking a downward route towards a destination. The storing mode of operation will be the subject of interest in this paper.

DODAG Construction

This paper will present a preventive solution to the Routing Table Overload Attack through a modification in the RPL DODAG construction step. Therefore, it is important to understand the existing methods used to build and maintain DODAGs.

The initial step is a DIO message from the root sharing information regarding the network, thus allowing new nodes to join the RPL instance [7]. Obtaining a DIO message, the new node wishing to join the network can determine its preferred parent nodes and rank through the objective function (OF) of RPL [13]. The rank is an RPL-specific value representing the logical distance to the root node of the DODAG. Nodes then build an upward route towards their preferred parent and can have backup parents in case the default route fails [13]. The response to a DIS message or simply broadcasting DIO control packets allows to further transmit information for newer children nodes. The DAO message, sent from a node to its parent, subsequently permits to the build of downward routes, and, the parent then recursively sends DAO messages to warn higher up nodes of existing routes.

3 Related Works

This paper will analyze an RPL-specific attack, the routing table overload attack, and propose a solution as prevention. It is important to understand existing security vulnerabilities on the routing protocol for LLNs. We will explore the taxonomy of security attacks on RPL networks. Then in the subsection 3.2, we will focus on the routing table overload attack.

3.1 RPL-specific Attacks

According to *Mayzaud et al.*, RPL attacks can be separated into three categories: traffic, resources, and topology [7]. While these differences are exploited and impact the network differently, they all pose a risk to the network and should be identified and mitigated if possible. Traffic attacks encompass more traditional sniffing and misappropriation attacks [7], therefore they will not be discussed throughout this paper.

Resources Attacks

The objective of resources attacks is to encumber the resources of RPL nodes, such as memory, processing, or energy. Flooding attacks are directly instigated by a malicious node. Their goal is to overload the network with control messages leading to network suffocation. For example, the DIS attack uses the DIS control message, network information solicitation, to trigger a forced response from receiver nodes [4]. In consequence, this affects the network control packet overhead and results in increased power consumption in the affected network nodes. This attack has been mitigated in 2020 by *Verma et al.* [4].

The rank attack is an example of an indirect resource attack. As a reminder, rank is an attribute in each child node that indicates the logical distance to the root node [3]. Moreover, increasing rank values as the DODAG deepens avoids loops, thus keeping the DODAG's integrity. By modifying this property, nodes can become more or less attractive as possible parents to their neighbors. That means that if a malicious node increases its own rank, upwards traffic will be rerouted to that node thus disrupting the DODAG by generating loops [14]. Furthermore, this leads to inefficient routing [15], therefore increases resource usage by the affected nodes in the network. If nodes are constantly dealing with non-optimal routing their resources become exhausted.

While resource attacks have two facets, direct and indirect, these types of attacks drain resources on already constrained devices. As RPL is optimized for LLNs, it is becoming increasingly apparent that it is necessary to strike a balance between performance and security. Meaning that the protocol must have enough security measures to avoid rendering the network unusable while avoiding as many overhead computations and control messages. Attackers will use every opportunity to corrupt the nominal functioning of the routing protocol and it then becomes the goal of the research to weigh the pros and cons of solutions.

Topology Attacks

An attack capable of targeting the topology of an RPL attack belongs to the *Topology attack category*. Topology security vulnerabilities can be further distinguished into sub-optimization and isolation classes, according to *Mayzaud et al.* [7]. Sub-optimization attacks attempt to deteriorate the performance of the RPL protocol by corrupting the nominal optimal IoT network setup. For instance, the routing table falsification attack in the storing mode of operation attempts to modify routing table entries of parent nodes to create fake routes through the network. In consequence, the attacked nodes route their messages through non-optimized routes leading to impacts such as pack delays and drops, as well as network congestion [7].

Isolation attacks in the topology category of RPL-specific vulnerabilities attempts to isolate "a node or a subset of nodes in the RPL network" [7, p. 466]. In consequence, these nodes are unable to communicate with surrounding neighbors and lose their importance on the infrastructure. The most common example is the blackhole attack, which is performed by a malicious node with legitimate children nodes. Its goal is to drop data packets routed to the malicious node [16]. Leading to the isolation of children nodes. The DAO inconsistency attack in the storing mode of operation utilizes the RPL mechanism to repair obsolete downward routes to remove legitimate routing links, thus isolating certain nodes [7]. Having the ability to set apart sections of an RPL network empowers an attacker. This permits cutting off part of the topology potentially creating a considerable amount of damage.

The topology of the routing protocol for LLN is a fragile web of interconnections. Any attack on these links can impact the normal functioning and awareness of the infrastructure.

3.2 The Routing Table Overload Attack and Related Works

The Routing Table Overload attack, as the name describes, is an overload of the routing table entries within parent nodes. This attack exploits all RPL's devices with limited memory. A malicious node attempts to encumber a parent through an overwhelming amount of data packets. This attack is particularly prevalent in the *Storing Mode of Operation* as the routing tables are established in every node. As RPL devices can be excessively constrained, the routing table overload attack has more of an impact with RPL in storing mode. Theoretically, this attack is present in the non-storing mode of operation, nonetheless, as the routing is performed through the root node, which often is a non-constrained device [11], it requires more effort to overload seemingly limitless memory. For the rest of the paper, we will consider the RPL network in the storing mode of operation.

The routing table overload attack is of the resources and topology types of attacks as it overflows memory on parent nodes [17] while at the same time blocking the creation of "legitimate optimized routes" [18, p. 6]. *Kamble et al.* describe the attack without mentioning existing solutions [17]. The vulnerability identification first appeared in *Mayzaud's et al.* taxonomy paper and was described with no available mitigation [7]. Furthermore, the routing table overload attack affects *Availability* and *Integrity* in the Confidentiality, Integrity, and Availability (CIA) model which impacts battery and memory normal functioning.

On the other hand, several solutions to limit memory overflow in the storing mode of operation have been erected in the world of RPL research. One of these solutions is called *D-RPL* and seeks to bypass memory limits in routing nodes by utilizing multicast techniques [19]. Meaning that when a parent node exceeds its routing table buffer and is unable to identify where to route an incoming packet, the parent node sends the data packet on the "D-RPL multicast channel" where other nodes with more memory will be able to correctly route the information [19]. As described by *Oh, Hwang, et al.*, D-RPL contains some limitations as its implementation requires increased network traffic due to the multicast messages [20]. Moreover, this solution only bypasses the memory limit issue and there is no active method to limit overloads in the future or to slowly regain routing table entries. Therefore, an RPL network with an overloaded parent node will stay overloaded and constantly use the multicast D-RPL technique for the rest of the IoT infrastructure, if the topology is static.

An interesting proposition, moving away from non-storing

and storing mode-specific downward route issues, attempts to hybridize the two RPL modes of operation to improve the performance of downward packet transmission in RPL [20]. By separating routing authorities for leaf and parent nodes this novel technique holds on to the benefits of each model. This paper proposes a new mode of operation but does not address specific memory overloads caused in the already implemented storing mode, like MERPL [21]. MERPL is a modification to the storing mode in RPL which improves memory efficiency, by re-ordering routing entries throughout the network, thus preventing routing table overload attacks. This method, however, generates packet processing overhead as a re-balancing of routing tables needs to be performed.

In this article, we propose STIR, a protocol modification to RPL's storing mode of operation that clusters addresses from sub-DODAGs together. Ultimately, routing nodes solely store intervals of addresses instead of every child in a sub-DODAG. This method enhances and addresses storing mode issues while its intention is to solve the routing table overload attack. STIR's proposed modification on the RPL protocol can help secure against routing table overload attacks.

4 Methodology

Throughout this section, we will explore the routing table overload attack in more detail by examining an example. Having understood the impact and ability of an attacker to perform the memory overflow attack, the paper will then present its contribution as a preventive solution against the overload attack in RPL's storing mode of operation. Eventually, section 4.4 will detail a performance analysis of the paper's contribution.

4.1 Routing Table Overload Attack in Action

As a reminder, the routing table overload attack is a vulnerability, prevalent in RPL's storing mode, where an attacker overflows the node's routing table's entries leading to the impossibility to add new routing table entries. In consequence, legitimate nodes joining the network will be inaccessible by the parent nodes as no path will be stored to reach them. RPL implementations should account for nodes with "no more than 128kB (host) or 256kB (parent) of memory" [10, p. 2]. With IPv6 addresses taking 128 bits or 16 bytes and assuming simple storage of only the IP address, the routing table can store 16000 addresses if the parent has 256kB of memory. Practically RPL uses volatile memory, RAM, to store its routing table entries. Consequently, with RPL open-source implementations and standard devices running the routing protocol, their RAM can only store 50 to 60 routing entries if the RAM is dedicated to the network stack [19] and if we remove software application memory requirements. It is realistic for a malicious non-constrained device to spoof messages leading to an overload in the routing tables of a network.

Example of a Routing Table Overload Attack

To perform a Routing Table Overload attack, a malicious node can spoof DAO packets and establish illegitimate downward nodes [7]. For example, in the Figure 1, the malicious node *N3* sends messages to the parent node *N2* asking to route packet, through DAO messages, to children in the sub-DODAG of *N3*. In accordance, *N2* realizes that children nodes of *N3* exist and therefore creates routing table entries for these faked nodes *S1* and *S2*. On a small scale this might seem inconsequential, however, with the limited memory of *N2* and more powerful devices, a malicious node can overflow the memory of parent nodes, such as *N2*. As a result, the routing table entries are completely full, thus blocking the creation of routes towards new legitimate nodes [18]. Accordingly, this leads to an impact on the efficiency of the network, negatively affecting the routing to certain nodes.

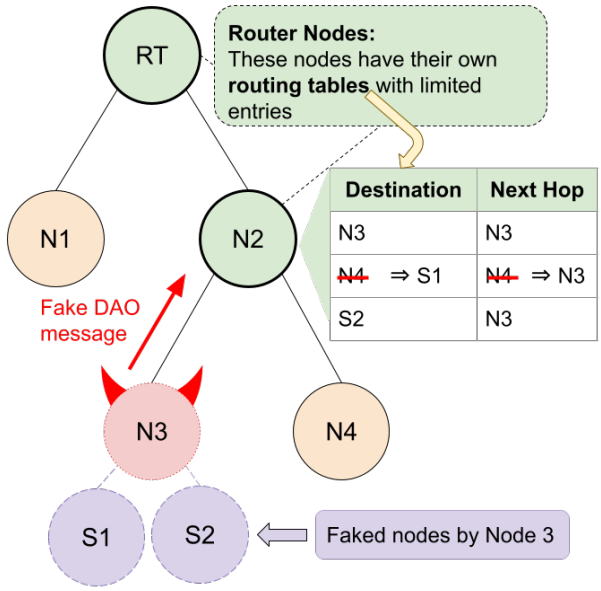

Figure 1: An example of a routing table overload attack performed by the malicious node N3

4.2 Our Contribution

This section will discuss the paper's contribution, STIR. RPL's storing mode of operation has a large memory flaw which results in the existence of the routing table overload attack. By increasing the routing table's memory opportunities can prevent the mentioned vulnerability and improve the versatility of the storing mode. This section will propose a method to efficiently store routing table entries.

STIR in Detail

STIR is a novel method utilizing search tree patterns and intervals in order to efficiently store and route data packets to the destination. By changing the routing methodology, IPv6 addresses can be coalesced, within RPL, to effectively route packets to clusters rather than individual nodes. Indeed, STIR seeks to let parent nodes store several children nodes into one routing table entry rather than keeping a one-to-one addressing. The proposed solution modifies two elements of the RPL protocol: the way a node joins the network and the routing methodology. However, there are several sub-tasks that must be performed for a node to correctly be part of the infrastructure. In STIR, a joining node is assigned an IPv6 address by the parent node it is connecting to. As we are discussing

the storing mode of operation, any node will be or become a parent node and store a routing table.

The previously described steps allow for the setup of the RPL DODAG and provide an infrastructure to perform routing operations. STIR also modifies the way routing is performed. Instead of storing one address to one network hop, similar to Figure 2, STIR allows a parent to store one interval of addresses to one network hop, as seen in Figure 3. The interval contains the minimum and maximum IPv6 addresses able to be routed through the node.

Destination (IPv6 address)	Next Hop (Node value)
baa5:95bf:3b50:8e82:a0fe:3324:936b:0ff2	Node 3
23b1:d95f:e824:b309:1624:5229:4f99:1893	Node 4
9be8:7c89:701b:ccf1:5734:3664:0df9:e9e7	Node 3
0684:7a08:5f1e:6d30:c5da:290f:f0cd:8171	Node 4

Figure 2: An example of routing table entries before STIR - Four stored entries

Destination (IPv6 address)	Next Hop (Node value)
baa5:95bf:3b50:8e82:a0fe:3324:936b:0ff2 - 9be8:7c89:701b:ccf1:5734:3664:0df9:e9e7	Node 3
23b1:d95f:e824:b309:1624:5229:4f99:1893 - 0684:7a08:5f1e:6d30:c5da:290f:f0cd:8171	Node 4

Figure 3: An example of routing table entries improvement due to STIR - Two stored entries

A parent node stores in memory the global interval it can deliver to, this is obtained when first joining the network from its own parent. The node can then use the general steps described in subsection 4.2 to divide its global interval into subintervals for children nodes. The DODAG is then built with decreasing interval sizes following the same pattern as described above. Indeed, every parent node simplifies their routing table entries, thus reducing memory usage while setting up structures for future joining nodes. To route, whenever a data packet reaches the node, the parent identifies the destination IP address and checks the interval it belongs to. Accordingly, the next network hop can be determined and the information packet is forwarded to that link. If the IP address is out of its own global interval, it redirects the data packet to its parent.

Initializing and Implementing STIR

STIR relies on a specific network arrangement to benefit from the efficient routing and storage provided by the intervals. In order to initialize the network to comply to STIR's properties, an algorithm or list of steps should take place. This paper will provide the essential features for STIR, as a lack of time prevents a detailed implementation. Various ways exist to develop STIR, for instance, one could use RPL's options such as *Prefix Information Option* and *RPL Target*. These could facilitate the assignment of prefixes (IPv6 addresses following a specific order) and the routing to clusters of nodes [3]. Nonetheless, the following section will discuss a global approach to the contribution in order to better grasp an understanding of STIR. Furthermore, following standard steps allows the reader to better reflect on the solution's place in the RPL infrastructure.

With STIR, RPL network nodes must store an interval, a minimum (*min*), and a maximum (*max*) of IPv6 addresses in their own sub-DODAG. The interval is the stored property by the parent allowing to identify coalesced addresses. While, the minimum and maximum IPv6 values allows to define tight boundaries of the utilized allocated space in an interval. In addition, the *min* and *max* values highlight the interim which cannot be redefined, informing the parents of the latter. For instance, when joining the network a node is assigned to the interval [100-200] and contains a few nodes in its sub-DODAG. The node's interval is then [100-200] but its *min* value could be 100 and its *max* value 135. In consequence, the parent knows the minimum space required for that node.

A node joining the network, directly requests an IPv6 address to the parent it wishes to connect too. The parent, acting as a SLAAC server, responds with an address and an interval for that node. To calculate these properties, the parent utilizes knowledge of allocated space in its own interval. If possible, if the interval allows it, the parent subdivides its interval with the new joining node. However, if no space allows a new node to join, the parent requests its own parent for more available space to allocate the new child that is trying to join. Considering these steps are done at every level, they can recursively free up space for a new node at a higher rank. Every time a joining node has a lower or higher IPv6 address than the parent, the *min* or *max* value must be updated. Furthermore, this property update should recursively take upward routes until a higher or lower value is found. Thus, lower ranked nodes are informed of *min* and *max* values in the sub-DODAGs.

A new node tries to join the network, the parent acts accordingly:

1. Is the parent able to allocate enough interval space to add the new node to my routing table, keeping in mind *min* and *max* of each sub-DODAG stored in the routing table entries?
    - If yes, send the IPv6 address and interval to the child.
    - If no, ask the parent's parent for more interval space and transmit it to the child along with an IPv6 address. This method is recursive and will go upwards until the necessary interval space is returned.
2. Finally, if the assigned IPv6 address is smaller than the parent's *min* or larger than the parent's *max*, update the node's *min* or *max* accordingly.

The majority of the algorithm is performed by the parent node, occasionally with lower ranked nodes, in order to set up the STIR-RPL system. The implementation is left to the vendors to identify allocatable interval space and how to choose the IPv6 address. For instance, the IPv6 address could be assigned by choosing the minimum of the transmitted interval to the child. Through these general instructions, there are few

control messages that emerge. Control messages are needed to identify interval space if the direct parent cannot subdivide its own interval. Furthermore, updating the sub-DODAG's *min* and *max* values is transmitted information. This subsection explored the general instructions needed to initialize and implement STIR. A deep dive into the instructions and their repercussion on RPL's performance will be analyzed in section 4.4.

Point of Views

To understand STIR in more detail, it is interesting to explore the point of views of the new and parent/parent node during the joining of the network as this preventive solution seeks to only change the initial steps of the RPL protocol.

From the joining node's perspective:

1. Identify the preferred parent through a similar method as RPL's Objective Function (OF).
2. Request DODAG information and IP with interval using a DIS and SLAAC control messages.
3. Receive a DIO control message. Assign IPv6 address and store the provided interval.
4. **Routing steps:** If it is a leaf node, routing is done through upward routes. If the node is now a parent, use the parent node's perspective, see the list 4.2 below.

Following these steps, the new node is now part of the RPL network. Moreover, it can now act as a parent node following the storing mode of operation if a node decides to join its sub-DODAG. To clarify the OF, RPL has a standard technique to select routes and preferred parents described in RFC 6552. For instance, OF0 or Objective Function Zero has the goal to select potential parents for upward routes [13]. Furthermore, the RFC focuses on the connectivity above everything else to choose a parent (6 out of 11 ordered criterion [13]); meaning, no information of the DODAG is previously required and the node can determine its connection in isolation. Therefore, by using these standards and replicating a similar method before joining the network it is feasible for STIR to occur.

The steps required from the parent node's perspective will now be explored in more detail:

1. From a node trying to join the network, the parent receives a SLAAC message.
2. In response to the new node, the parent should send an IP address and an interval that was determined.
3. The parent then appends the interval to the routing table using the new node as the next hop.
4. **Routing steps:** If the node is not in the parent's global interval, reroute through upward routes. If the node is in the parent's global interval, identify in which interval of the routing table buffer the IPv6 destination address corresponds to and route accordingly.

The previous steps are a general overview of what a parent node experiences when a new node seeks to join its sub-DODAG.

Why is STIR useful?

The novel protocol modification of RPL was designed with the goal of preventing the routing table overload attack. It successfully achieves this objective by greatly reducing the size of routing table entries, thus permitting more nodes to join the network without adding new entries for every node. In the traditional storing mode of operation, RPL nodes must store routes to every node in their sub-DODAG, therefore, a malicious attacker could falsify the existence of nodes in the sub-DODAG forcing parent nodes to store additional illegitimate routing entries. STIR-RPL prevents this attack by clustering sub-DODAGs with previously discussed intervals in every routing table entry. A malicious node could fake as many children nodes as it wishes but the parent node will never add a routing entry. Therefore, it will not overflow in memory due to the routing table buffer. The only way for new entries to be appended to the routing table entry is if a new node is joining the parent node directly. This issue will be further discussed in section 5, but a simple response is that it will require a large amount of colluding nodes to be realistic. STIR is intended to prevent the routing table overload attack, but it is also an improvement on RPL's storing mode. It permits for a more memory-efficient routing protocol and this aspect can be taken advantage of to ameliorate the current routing protocol.

STIRing with an example DODAG

This subsection will demonstrate STIR's routing methodology with the goal of showing an example. Throughout this paragraph, we will use a small DODAG with simplified intervals and IPv6 addresses. Taking for example routing from Node 3 (*N3*) with IPv6 address *2* to *N5* with IPv6 address *62* in Figure 4.

1. *N3* is trying to send a packet to IPv6: *62*. *N3*'s interval is *[2, 29]* and *62* is not in *N3*'s interval, therefore, *N3* routes upwards to *N1*.
2. *N1*'s interval is *[1, 60]* and *62* is not in *N1*'s interval, therefore, *N1* routes upwards towards the root.
3. Root's interval is *[0, 100]* so *62* is in the interval. The root node now iterates through its routing table to find the interval which contains *62*. *62* is contained in the interval *[61, 100]*, therefore the next hop is *N2* with IPv6 address *61*.
4. *N2*'s interval is *[61, 100]* containing *62*, therefore, *N2* routes downwards towards the next hop that contains *62* in its interval.
5. *N5*'s interval is *[62, 100]* and *N5*'s IPv6 address is *62*. The packet has arrived at its destination.

4.3 Miscellaneous Information for STIR

Assigning IPv6 addresses

STIR requires assigning IPv6 addresses using a specific algorithm. This can be done by utilizing a SLAAC or DHCPv6 server within every parent node a child is attempting to connect to. Therefore, this allows the parent node to immediately calculate the IPv6 address, using the ideas of subsection 4.2, and to distribute the IP address as a traditional SLAAC

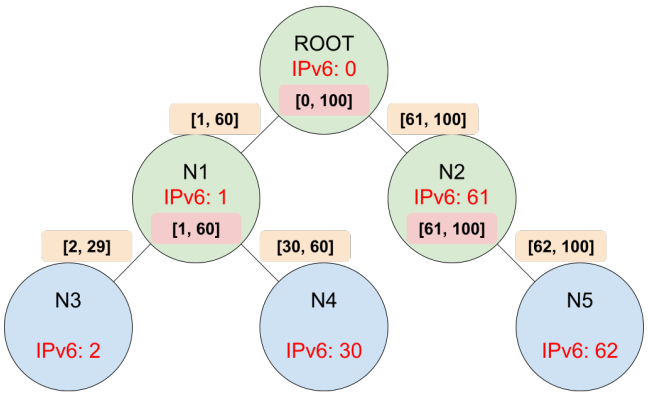

Figure 4: An example of a DODAG following the STIR method

server. IPv6 addresses are assigned through IPv6 Stateless Address Autoconfiguration (SLAAC), DHCPv6, or other out-of-scope techniques [22]. SLAAC dynamically calculates an IPv6 address using information from Router Advertisement (RA) messages and the MAC of the connecting device. The first 64 bits (out of 128) of the address identifies the network while the remaining 64 are usually a combination between the MAC address and hexadecimal value *ff:fe*, called the EUI-64 formation [23]. STIR seeks to exploit the remaining 64 bits to uniquely identify every device on the network. Instead of using more or less random final 64 bits, STIR will assign these values using a specific algorithm implementation discussed in subsection 4.2. Accordingly, it is realistic for STIR to take place in an IPv6-based environment such as RPL, moreover, this would only target a distinct modification in the already existing IPv6 addressing server.

Interval Space

This paragraph seeks to demonstrate the validity of using intervals and how in practice not many interval re-balancing should happen. For simplicity, we will show how intervals will grow with a full DODAG. In STIR, we are dealing with intervals of IPv6 address which are 128 bits long and usually only modifying the *Interface ID* which is 64 bits, thus the root interval size is 2^{64}. The Interface ID, the end section of the IPv6 address, is the only modifiable part as it can be done through the SLAAC or DHCPv6 system. Assuming a full tree with 64 sub-DODAGs at each level, as we already discussed an RPL device on the network can store 50-60 devices is the maximum amount of routing entries that can be stored in RAM [19]. The root node (layer 0) has an interval size of:

$$2^{64}$$

Now we assume 64 or 2^8 nodes (layer 1) join the root node, their interval size will be:

$$2^{64}/2^8 = 2^{56}$$

There will be $1 + 2^8$ nodes in the DODAG. Now 64 more nodes join: interval size per node on layer 2.

$$2^{56}/2^8 = 2^{48}$$

There will be $1 + 2^8 + 2^{16}$ nodes in the DODAG. These steps can be extended until there are 8 layers in the DODAG, meaning at the last layer the interval size will be 1. By adding the totality of nodes per layer we get:

$$1 + 2^8 + 2^{16} + ... + 2^{64} = 18519084246547628289$$

Therefore, using STIR, a full DODAG can efficiently route to sufficiently enough nodes. Moreover, using different re-balancing techniques, DODAG's can be balanced with specific weights leading to a variety of depth in the network.

4.4 STIR's Performance Analysis

The research question of the paper attempts to find a solution to an existing RPL-specific attack. Having proposed STIR as a solution to the routing table overload attack it is of interest to analyze the performance of the solution. This allows to understand if STIR is a viable solution for more research. This subsection will present a performance analysis based on *Control Packet Overhead* and *Routing Table Size*. These metrics were chosen, out of a list, as they were proposed in the standard RFC 6687 [24]. The three metrics that are to be ignored in this paper are *Delay Bound for P2P Routing*, *Path Quality*, and *Loss of Connectivity*. *Delay in P2P routing* is disregarded as STIR does not affect RPL protocol once the topology initialized. *Loss of Connectivity* discusses RPL in a non-static organization and should be the subject of further research. *Path Quality* is a metric analyzing the chosen optimum path by RPL, a metric feasible to observe with a simulation not in the scope of this paper. Moreover, STIR routes in non-variable way, using intervals, therefore, the path quality metric would be irrelevant for this research. A summary of the analyzed performance metrics and their context in STIR can be observed in Figure 5.

	Impacted by STIR	In STIR's context
Path Quality	✖	STIR does not impact this metric as it routes in a singular fashion.
Routing Table Size	✚	STIR's goal is to reduce routing table size which it does by definition.
Delay Bound for P2P Routing	✖	This performance metric is not modified by using STIR as the solution does not modify RPL protocol once the network initialized.
Control Packet Overhead	✚	STIR adds a few additional control messages thus negatively impacting the overall control packet overhead.
Loss of Connectivity	▬	Loss of connectivity involves disconnecting nodes and DODAGs which is needed further research for this paper.

Figure 5: STIR's discussed performance based on RPL's performance metrics from RFC 6687

First of all, RFC 6687 clarifies that routing table size is determined by the number of entries for each node [24]. In addition, "each entry has the next-hop node and path cost associated with the destination node" [24, p. 8]. The *Routing Table Size* metric is appropriate to analyze STIR as the preventive

solution is reducing the routing table size. In traditional storing mode RPL, the nodes closer to the DODAG root will often store more routing table entries [24], this is due to the fact that the node must store routes to all children nodes. By using address coalescence with intervals, STIR permits to only store as many routing table entries as there are sub-DODAGs connected to a parent node. In consequence, rather than storing a complete list of children in every sub-DODAG connected to a node [11], STIR greatly optimizes routing table entries. Moreover, lower ranked nodes do not require larger memory to store important routing table entries. Due to the nature of the proposed solution, STIR has a great benefit on the *Routing Table Size* metric.

An important metric to discuss the feasibility of STIR-RPL is the control packet's overhead. In comparison to traditional storing mode RPL, STIR potentially adds extra packets whenever a new node tries to join the topology. Following subsection 4.2, new messages are added to the protocol to answer to STIR's properties. The IPv6 Router Advertisement (RA) packets used to obtain an IPv6 address and an interval will be the same then any other topology. However, this differs when a parent node needs to transmit upward messages in order to have more interval space or to update the *min* and *max* values. It is complicated to quantify how many additional control packets will be needed to answer to these STIR needs, especially since these messages can recursively attain the root node. Updating *min* and *max* values will be necessary, leading to additional data packets. Nonetheless, it is important to note that following paragraph 4.3, some of these messages will be extremely rare in large networks. In realistic RPL topologies, with more or less 60 nodes, the interval space control packets will never be utilized. Theoretically, STIR adds additional control messages for that could appear to be a heavy toll on STIR's feasibility, but, with the realistic interval space the occurrence of this is very unlikely.

STIR is a protocol modification to RPL which only modifies the method of a node joining the network. Accordingly, the affected performance metrics are contained to the routing table size and the control overhead messages. On one hand, STIR's purpose is to diminish the size of the routing tables which is a benefit to this performance metric. On the other hand, STIR increases the control message overheads through interval space request and an update of node properties. The data packet for more interval space is improbable and only happens when a node joins the network. The impact on RPL's performance with STIR is minimal but must be taken into account.

5 Discussion

Throughout this paper, a novel technique has been presented to prevent the routing table overload attack in RPL networks. As the vulnerability stems from a lack of storage in routing table buffers of the storing mode of operation, the prevention opted to solve the issue through a change in the routing methodology. By utilizing address coalescence, individual parent nodes can store entire sub-DODAGs in a single routing entry. This method requires specific node addressing, entirely feasible with a modification to the RPL protocol, named STIR. While STIR has a goal of preventing the routing table overload attack, its implementation proposes an improvement to the already existing storing mode in RPL. Therefore, it alternatively proposes protection against accidental memory overflow within networks implementing RPL in storing mode. In the following sections, the conclusions from the methodology using an advantages and disadvantages approach will be explored. While the ultimate subsection 5.2, future research that could benefit STIR and what is left to be explored will be discussed.

5.1 Conclusions from the Methodology

Following solution propositions, a balance between benefits and disadvantages can be drawn on the STIR solution. The first step is to understand the performance analysis done in subsection 4.4, which presents two affected performance metrics. From the previous analysis, it is already demonstrated that STIR benefits the routing table size but has an eventual slight increase in terms of control packet overhead. However, as could be determined, the total number of additional control messages would greatly be limited in realistic RPL topologies.

Advantages

STIR is an RPL protocol modification leading to an increase in memory efficiency for RPL parent nodes in the storing mode of operation. This novel approach to the routing methodology stores clusters of addresses in a single routing table entry. Identifying the next network hop is simplified to comparing intervals in the routing table. In STIR-RPL, any node storing routes will only have as many entries as the node has children. STIR coalesces addresses from the same sub-DODAG, therefore, the stored intervals will store the entire sub-DODAGs of every children node. This property allows routing table overload attacks to be prevented no malicious child node can overflow a parent's memory by the simulation of illegitimate devices. All IPv6 addresses from the malicious node should be in the interval range of that node, and if they are not, the parent will simply discard the packet. Moreover, a request for more interval space will not add an entry but simply re-balance the intervals accordingly. To restate the practicality of STIR, this technique can also be used in RPL storing mode without an attacker attempting to overflow parent nodes' RAM.

Disadvantages

STIR presents issues which does not discredit the solution but demonstrates more research is needed to fill in the gaps. Indeed, an initial problem that faces STIR-RPL, as well as traditional RPL, is network mobility. Parent nodes moving around on the DODAG might cause issues as routing routes are completely corrupted. However, this issue is also present in traditional RPL in storing mode. A consequence of approaching simple protocol modification is that issues present in the original protocol exist in STIR-RPL. Although, nodes in RPL might occasionally have backup feasible parents allowing them to reroute their data packets [13]. This functionality is missing in STIR, nonetheless, more research in dealing with mobile nodes could elucidate this issue.

In terms of security vulnerabilities which the STIR method faces, the technique is not impervious to collusion attacks. As STIR simplifies routing table entries for sub-DODAGs, it cannot regulate on its own the number of nodes attempting to connect to a single parent node. There are as many routing entries as there are children, unfortunately or with enough coordination, the memory can be overflowed with enough malicious collusion. In addition, the communication to find interval space and update node properties from paragraph 4.2 must be secured a little more thoroughly. A system of exponential back-off linked to sufficient interval allocation could be a feasible solution, however, more research is needed.

STIR's use of intervals opens the door to discussion on any potential disadvantage of using intervals in RPL's routing entries. While current RPL implementations store a single IPv6 address for one hop, STIR requires the storage space of two addresses per routing entry to bound the interval. This slight disadvantage is necessary to permit a more memory-efficient network. Moreover, the proposed contribution will most likely need to access and modify routing entries more often than traditional RPL. The increased strains will be placed at the moment a node joins the network, therefore, this can be accounted for as startup overhead.

5.2 Future Research Directions

1. Research the mobility of RPL nodes in the storing mode of operation.
2. Research collusion between malicious node with the same rank.
3. Secure update control messages in STIR.
4. Perform a COOJA simulation to verify the implementation steps and practical performance metrics.

6 Responsible Research

RPL is a protocol used in many IoT infrastructures and has an important responsibility in domains such as healthcare, critical infrastructure, and transportation. A security issue or wrong implementation can have a disastrous effect on a technology with a market share of $381.30 billion [25]. Therefore, it is important to realize the ethical impact borne by the proposition of a protocol modification. An implemented mishap can negatively affect patients in the healthcare field. The routing table overload attack can have grave consequences on IoT networks and should never be exploited without proper approval. Moreover, an issue with STIR can open the door to even more cyberattacks, as steps ease or allow malicious attackers to find backdoors into networks. For this reason, this paper is the result of many weeks of research, and it then was reviewed by supervisors in the field of cybersecurity and the internet of things. A great deal of thought were spent improving and verifying the feasibility of the contribution. Furthermore, directions and ideas sought to impede future malicious actions on RPL networks. An important emphasis is placed on the discussion of future work as more research and peer reviews are needed to fully form the idea of STIR. Vendors should make sure to fully prove the efficiency of STIR before implementing it in the wild.

With the contribution brought to light by this paper, it is important to discuss the reproducibility of STIR. This article does not present any specific data in regards to STIR, but, it does provide information on the potential performance brought by the novel protocol modification. As the paper does not provide a specific implementation, in the end the performance metrics could vary between vendors using STIR. The general performance metrics follow the described ideas of STIR and are therefore reproducible. For instance, the reduction of the number of routing table size is proportional to the number of sub-DODAGs of the parent. This property follows the STIR methodology and can be observed and reproduced to derive performance metrics of the routing table size. The authors of this paper welcomes more research into the performance of STIR through simulations and other tests.

7 Conclusion

This paper proposes STIR, an RPL protocol modification that simplifies routing table entries through address coalescing. The preventive method originates from an answer to the research question: *how can RPL protocol's functionalities be modified to avoid an RPL-specific routing attack*? This solution prevents routing table overload attacks by permitting parent nodes to solely store as many entries as it has children or sub-DODAGs. As a result, no malicious child node can create new routing table entries based on fake nodes in their own sub-DODAGs. The strong advantage of STIR is that it provides a relevant modification to prevent storing mode RPL issues. Storing mode is limited by the constrained nodes acting as routers. Providing a more efficient way to store and route data packets allows larger RPL networks in storing mode. STIR improves on the routing table size metric, nonetheless, the protocol modification slightly reduces performance by adding additional control messages in order to inform higher nodes of STIR properties.

While this paper seeks to present a complete solution to the routing table overload attack, the preventive method, STIR, could benefit from additional research to tie the ends together. As an example, more research into the mobility of RPL networks could help implement mobile nodes in STIR. Moreover, a detailed algorithm to follow the re-balancing of STIR interval trees would benefit future implementations of STIR. Finally, with a detailed implementation a simulation in COOJA could verify the validity and efficiency of the paper's contribution.

References

[1] 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 and Tutorials*, vol. 17, pp. 2347–2376, 10 2015.

[2] R. Karjagi and M. Jindal, "Iot in healthcare industry — iot applications in healthcare - wipro." [Online]. Available: https://www.wipro.com/business-process/what-can-iot-do-for-healthcare-/

[3] A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and R. Alexander, *RPL: IPv6 Routing*

Protocol for Low-Power and Lossy Networks. IETF, Mar 2012. [Online]. Available: http://dx.doi.org/10.17487/RFC6550

[4] A. Verma and V. Ranga, "Mitigation of dis flooding attacks in rpl-based 6lowpan networks," *Transactions on Emerging Telecommunications Technologies*, vol. 31, 2 2020.

[5] A. Le, J. Loo, K. K. Chai, and M. Aiash, "A specification-based ids for detecting attacks on rpl-based network topology," *Information (Switzerland)*, vol. 7, 5 2016.

[6] 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, pp. 1582–1606, 4 2019.

[7] A. Mayzaud, R. Badonnel, and I. Chrisment, "A taxonomy of attacks in rpl-based internet of things," *International Journal of Network Security*, vol. 18, pp. 459–473, 2016. [Online]. Available: https://hal.inria.fr/hal-01207859

[8] G. Association, "Smart trash cans are quietly supporting the rise of the smart city," Nov 2019. [Online]. Available: https://www.gsma.com/iot/news/smart-trash-cans-telebelly/

[9] M. Akkurt and K. Küçük, "Simulation of smart city applications based on iot technologies with cupcarbon," in *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, 2018, pp. 179–184.

[10] H. S. Kim, J. Ko, D. E. Culler, and J. Paek, "Challenging the ipv6 routing protocol for low-power and lossy networks (rpl): A survey," *IEEE Communications Surveys and Tutorials*, vol. 19, pp. 2502–2525, 10 2017.

[11] M. Richardson and I. Robles, *RPL- Routing over Low Power and Lossy Networks*. IETF 94, 2015, pp. 1–66. [Online]. Available: https://www.ietf.org/proceedings/94/slides/slides-94-rtgarea-2.pdf

[12] 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.

[13] P. Thubert, "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)," RFC 6552, Mar. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6552.txt

[14] D. S. S. A. Professor, I. M. S. A. Professor, and S. Jain, "Impact factor: 4.295 a detailed classification of routing attacks against rpl in internet of things," *International Journal of Advance Research*, 2017. [Online]. Available: www.ijariit.com

[15] K. K. Rai and K. Asawa, "Impact analysis of rank attack with spoofed ip on routing in 6lowpan network," in *2017 Tenth International Conference on Contemporary Computing (IC3)*, 2017, pp. 1–5.

[16] K. Chugh, A. Lasebae, and J. Loo, "Case study of a black hole attack on 6lowpan-rpl," in *SECURWARE 2012 : The Sixth International Conference on Emerging Security Information, Systems and Technologies*, 07 2012.

[17] A. Kamble, V. S. Malemath, and D. Patil, "Security attacks and secure routing protocols in rpl-based internet of things: Survey," in *2017 International Conference on Emerging Trends Innovation in ICT (ICEI)*, 2017, pp. 33–39.

[18] A. Verma and V. Ranga, "Security of rpl based 6lowpan networks in the internet of things: A review," *IEEE Sensors Journal*, vol. 20, pp. 5666–5690, 6 2020.

[19] C. Kiraly, T. Istomin, O. Iova, and G. P. Picco, "D-rpl: Overcoming memory limitations in rpl point-to-multipoint routing," in *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, 2015, pp. 157–160.

[20] S. Oh, D. Hwang, K. Kim, and K.-H. Kim, "A hybrid mode to enhance the downward route performance in routing protocol for low power and lossy networks," *International Journal of Distributed Sensor Networks*, vol. 14, no. 4, p. 1550147718772533, 2018. [Online]. Available: https://doi.org/10.1177/1550147718772533

[21] W. Gan, Z. Shi, C. Zhang, L. Sun, and D. Ionescu, "Merpl: A more memory-efficient storing mode in rpl," in *2013 19th IEEE International Conference on Networks (ICON)*, 2013, pp. 1–5.

[22] C. Press, "Mastering ipv6 slaac concepts and configuration," Dec 2013. [Online]. Available: https://www.ciscopress.com/articles/article.asp?p=2154680

[23] T. Coffeen, "Slaac-to-basics (part 1 of 2)," Sep 2017. [Online]. Available: https://blogs.infoblox.com/ipv6-coe/slaac-to-basics-part-1-of-2/

[24] J. Tripathi, J. C. de Oliveira, and J. Vasseur, "Performance Evaluation of the Routing Protocol for Low-Power and Lossy Networks (RPL)," RFC 6687, Oct. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6687.txt

[25] F. B. Insights, "Internet of things (iot) market size, share & covid-19 impact analysis, by component (platform, solution & services), by end use industry (bfsi, retail, government, healthcare, manufacturing, agriculture, sustainable energy, transportation, it & telecom, others), and regional forecast, 2021-2028," May 2021. [Online]. Available: https://www.fortunebusinessinsights.com/industry-articles/internet-of-things-iot-market-100307