T-TRAIL: Preventing Decreased Rank Attacks in RPL-based IoT Networks

W.E.P. Tolsma[1], **Chhagan Lal**[1], **Mauro Conti**[1]
[1]TU Delft

Abstract The Routing Protocol for Low-Power and Lossy Networks (RPL) has gained in popularity since the increased connectivity of everyday items to the Internet. One of the discovered attacks on RPL is the rank attack, which opens up possibilities for attackers to control traffic in the RPL network by spoofing their priority. Many solutions have been proposed to mitigate this attack over the past few years. There is no perfect solution yet, partly because the success of a mitigation is dependent on the network configuration in which it is implemented. In some network configurations, as this paper will show, common mitigation solutions are less effective. By selecting and analyzing four well-cited mitigation and detection solutions, the effectiveness of these proposals is reviewed when the network is configured to use nonlinear objective functions (NOFs). After this, a proposal is given to defend against a rank attack when using NOFs. TRAIL was proposed as a solution for preventing decreased rank attacks and uses a challenge-response mechanism to verify the path from a node to the root. This paper proposes T-TRAIL; an extended version of TRAIL that allows the measurement of downwards-trip-time to detect outliers in the network. By doing this, the rank attack can be prevented when the network uses a NOF. Finally, an estimation of the performance impact of T-TRAIL on the network is given based on the performance measurements of TRAIL.

***Index Terms*—RPL, 6LowPAN, IoT, LLN, Rank Attack, TRAIL, Mitigation**

I. Introduction

With the increasing amount of everyday objects being connected to the Internet, a need arises for efficient and secure communication for this new generation of devices. Many of these new devices are resource-constrained, meaning they have limited battery, processing, and networking capabilities.

A breach in the security of *Internet of Things* (IoT) could create serious threats in the physical world (e.g. a connected cardiac device malfunctioning), which is why it is important to study and improve the security of the protocols used by these networks. An attacker can target a network to disrupt, eavesdrop, or modify the data flowing through it, and depending on the network scenario that could have drastic consequences.

To allow resource-constrained devices to communicate over IPv6, 6LowPAN was created to reduce the minimum packet size by fragmenting IPv6 datagrams [1]. To better suit the routing requirements for *Low-Power and Lossy Networks* (LLNs), the *IPv6 Routing Protocol for Low-Power and Lossy Networks* (RPL) was proposed. RPL is a distance vector routing protocol within 6LowPAN and is specifically designed for multipoint-to-point (MP2P) traffic, but also supports point-to-point (P2P) and point-to-multipoint (P2MP) traffic [2]. RPL is more basic than standard routing protocols such as the network layer in the *Internet Protocol* (IP) stack. This works well for resource-constrained devices, but because of this and the varying availability of the nodes in an RPL network, security and privacy is not easy to achieve and remains an active area of research.

Since the RFC on RPL was released in 2012, much research has been done on the security and usability of this protocol. The paper by Mayzaud et al. (2016) [3] gives a good overview of possible attacks and proposes various ways to avoid and prevent them. The paper concludes that there is no end-all solution and that there will always be a trade-off between the level of security and the overhead added by countermeasures. It is thus relevant to look for countermeasures that add minimum overhead whilst still being effective against attacks. In a survey done by Kim et al. (2017), 97 studies on RPL security were evaluated [4]. One of the conclusions was that many optional functionalities in RPL were not supported or needed in common scenarios. Furthermore, RPL is still not widely adopted by the industry and this might have to do with the complexity, amount, and ambiguity of the optional features. Many *intrusion detection systems* (IDS) and mitigation solutions have been proposed, each with its advantages and drawbacks. In literature, some of the well-cited solutions are Secure-RPL [5], SVELTE [6], VeRA [7] and TRAIL [8].

The contribution of this paper is twofold; first, the four mitigation and detection proposals above are analyzed by giving the disadvantages and advantages each method has. In this analysis, a possible exploit is introduced by showing that a *decreased rank attack* is still possible when using *nonlinear objective functions (NOFs)*. Secondly, a modification to TRAIL is proposed that allows nodes to calculate the *downward-trip-time (DTT)* to nodes in the parent set to overcome this exploit.

The scope of this paper is limited to defining the core idea and motivation behind this without simulating the actual modification to TRAIL, which could be performed in further research. Besides DTT being useful for spotting incorrect usage of the NOFs, it could potentially be used in more advanced IDS that make use of link quality to detect intruders. The research question that this paper will try to answer is;

> *"What effect does the use of a nonlinear objective function have on existing mitigation solutions for the rank attack on RPL and how can possible exploits be defended against?"*

The structure of this paper is as follows. Section II gives an overview of how RPL currently operates. Section III explains the difference between nonlinear and linear objective functions (NOFs and LOFs). Section IV explains the rank attack and how current state-of-the-art mitigation solutions try to mitigate this by summarizing the advantages and disadvantages each solution has. In section V, an exploit on some of these mitigation solutions is introduced when NOFs are used. Part VI then proposes a method to prevent this exploit by measuring *downwards-trip-time* (DTT). Part VII looks at the performance drawback that T-TRAIL imposes. Finally, the work is concluded and a short discussion is given about the effectiveness of this paper.

II. Overview of RPL

This part introduces the core concepts of RPL that are needed to understand the behavior of an attack and its implications to the performance of the network. The full documentation on RPL can be found in RFC-6550 [2]. In this study, multiple figures illustrate the topology and node types of RPL networks. Fig. 1 explains the different elements that these figures use.

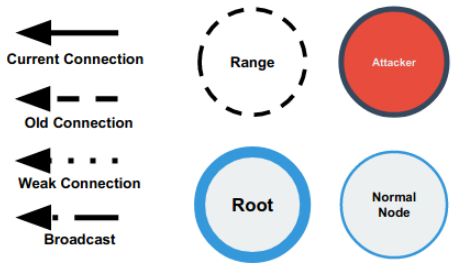

Figure 1. Overview of elements used in figures

Topology: RPL is a distance vector-based routing protocol that constructs a *directed acyclic graph* (DAG) to make routing decisions. Within a DAG, RPL constructs *destination-oriented acyclic graphs* (DODAG) with the root being one or more IPv6 Border Routers (6BR) that translates full IPv6 datagrams to and from fragmented 6LowPAN datagrams [4]. Fig. 2 illustrates the message flow for when a new node joins the RPL network. The new node starts by broadcasting a DIS message (A), waits for nodes to respond with DIO messages (B) after which the node selects a preferred parent. In (C), DAO messages are transmitted to make parents aware of the sub-DODAG. In the following paragraph, these different messages will be explained further.

Order in the DODAG: The *rank* of a node indicates the order in which nodes appear in the DODAG. The higher the rank, the further away it is to the root node, also called the monotonic increase of rank. A node learns information about its neighbors by reading DIO messages that can be requested by multicasting a *DODAG information solicitation* (DIS) message. When nodes in the network receive a multicast DIS message they, in turn, will also multicast a DIO message (see fig. 2 B). Another option is to unicast a DIS message, which will reduce traffic in some scenarios as receiving nodes will only unicast DIO messages to the sender. A node selects the parent set P based on all neighbors that have a rank lower than its rank. The *Objective Function* (OF) decides how the *preferred parent* (PP) should be chosen from the parent set P. When a node has to forward a message, it will first try to do so through the PP and if that does not work it will try the other members in PP.

Downwards routing: Once every node has established a path to the root node in the DODAG, nodes will transmit *destination advertisement object* (DAO) messages to their parents, which contain routing information for downwards routing. A parent will acknowledge a DAO message by sending a DAO-ACK message to the sending child. The DAO-Ack message contains a DAOSequence number that corresponds to the DAOSequence number in the DAO message. The *mode of operation* (MOP) indicates the routing mode that the RPL instance is in. In *storing* mode, a parent aggregates DAO messages from its children and sends that information to its parents. In this mode, nodes maintain their routing tables such that nodes can efficiently route messages directly to the target. In *non-storing* mode, parents do not store DAO information but insert their address and unicast to the DODAG root [3]. In this mode, every message is forwarded to the root before sent to the target.

Graph repair: RPL supports dynamically repairing the graph in case of a broken topology. A *global repair* is initiated when the root node increments the DODAG version number. This allows all nodes to reposition themselves in the network by recalculating their rank. Note that a global repair is quite an expensive operation. A *local repair* is less expensive and works by either routing through a close sibling of the same rank or selecting a different parent. It is initiated by either changing its rank to infinity and broadcasting this, or changing the DODAG ID value of the node [9]. Local repairs are needed for example when a loop is detected or when a node disappears from the network.

Security Modes: In the RFC [2] three security modes are defined. Note that RPLs goal is to remain lightweight and is not focused on security, hence why most implementations

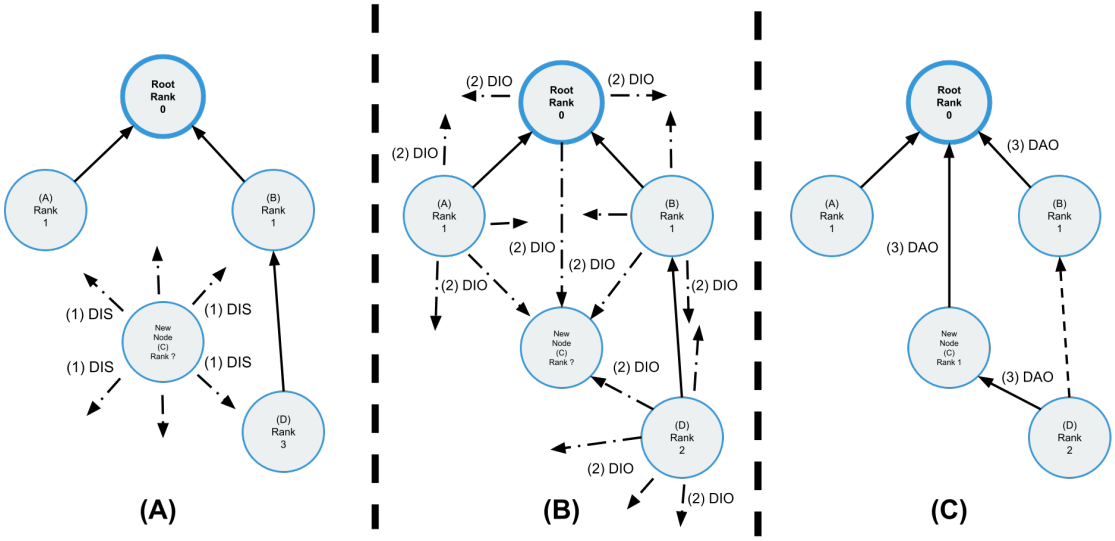

Figure 2. RPL Message Flow. A) New node joins the network and multicasts DIS. B) Neighbors respond by multicasting DIO. C) Nodes select preferred parent and unicast DAO on parent change.

only use unsecured mode and rely on other layers in the IP stack to provide confidentiality, integrity, and availability.

- Unsecured: RPL does not add any security to communication.
- Preinstalled: every node comes with a preinstalled key that is used to communicate over the network.
- Authenticated: every node comes with a preinstalled key, but now also needs a second key that can be obtained through a specified authentication authority. The RFC does not specify how this should be implemented.

III. Linear and Nonlinear Objective Functions

In this paper, a distinction is made between linear and nonlinear increasing rank sequences as it impacts the effectiveness of mitigation and detection solutions for the rank attack, described later on. The terms linear and nonlinear are based on the increased rank between each hop. Fig. 3 illustrates an example of the rank increase based on hop-count of a *nonlinear objective function* (NOF) and a *linear objective function* (LOF).

Linear Objective Functions

A well-documented LOF is defined in RFC6552 [10] called OF0 that calculates the rank of a node based on the number of hops to the root. In this paper, this OF is defined as linear since the rank sequence of a path from the root to any node is steadily monotonically increasing, whilst NOFs have more freedom in deciding the next rank of a node.

Nonlinear Objective Functions

A common NOF is the *Minimum Rank increase with Hysteresis OF (MRHOF)* [11], which bases the rank of a node on more complex metrics that indicate link quality to each parent in the parent set. One of these metrics is the *Expected Transmission Count (ETX)* which is given by $\frac{1}{D_f D_r}$, where $\{D_f, D_r\}$ denote the forwarding and receive success rate respectively. Other metrics included are hop-count and latency, but any metric can be configured as long it can be minimized. MRHOF uses hysteresis to prevent rapid switching between parents when the link metrics are jittering. According to the RFC, a node sets its rank to the maximum of the following three values:

- The rank of the preferred parent + the rank increase associated with the link to the PP.
- The rank of the parent R_p from the parent set with the highest rank rounded up based on the minimum rank increase R_{min}: $R = R_{min} * (1 + R_p/R_{min})$.
- The rank associated with the worst path through the parent set, minus maximum rank increase.

Choosing the Right Objective Function

In Pradeska et al. [12], OF0 and MRHOF are simulated in various scenarios; “From the simulation, we conclude that MRHOF gives better performance than OF0 in the aspect of network reliability while OF0 is faster in network convergence and consumes less power than MRHOF during the DODAG convergence time interval”. Capone et al. [13] claims that using hop count as a metric (meaning the use of LOF), compared to using MRHOF, leads to a stable topology but poor network lifetime, as nodes close to the root become a traffic bottleneck that consumes more power and thus run out of energy faster. Based on these two papers, MRHOF seems to be more energy-efficient once the initial DODAG is constructed, although it consumes more energy during the convergence phase. Even though using NOFs is not standard, the results from the study above tell us that NOFs have the potential to save on energy consumption in certain scenarios.

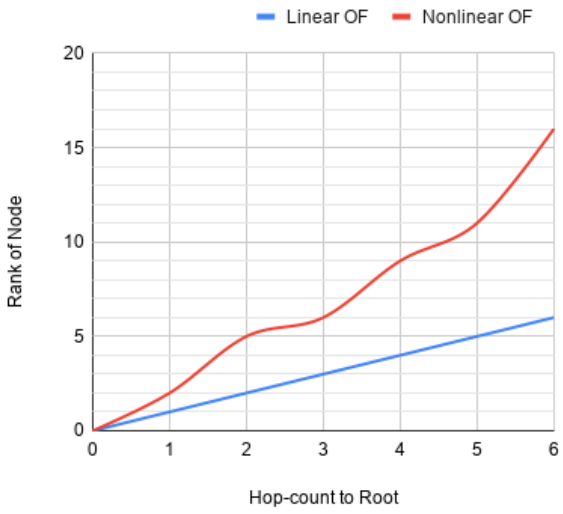

Figure 3. Linear versus nonlinear rank increase.

IV. The Rank Attack and Current Mitigation Solutions

Standard RPL is exposed to various kinds of attacks, which are properly introduced by Mayzoud et al. (2015) [3]. The rank attack is considered as being the strongest threat to routing performance and energy consumption of the network [14]. It also serves as a basis for many other attacks, since it attracts or repels traffic. Note how the rank of a node is essential for determining its place and priority in the network. Any node can change its rank at any moment, which opens up possibilities for exploits.

A node can advertise a false rank at any time by creating a false DIO message containing the desired rank. Besides falsifying its rank, a node can also pick a suboptimal parent. The three main types of rank attacks are as follows.

- The *decreased rank attack* decreases a node's rank such that its neighbors will pick it as their preferred parent, which allows the node to start other attacks.
- The *increased rank attack* increases a nodes' rank such that its children will have to start looking for a different parent, which will consume resources and could create loops in the network.
- The *worst-parent attack* is performed by letting the malicious node pick the worst parent, which will negatively impact the end-to-end delay and ETX.

The adversary must be able to control a node inside the network that preferably has a large neighbor set since it will try to trick as many nodes as possible into believing it has a specific rank. The goal of the adversary is to manipulate traffic flow by either promoting itself as a low-rank node or degrading itself to disrupt the topology. Once the adversary has placed itself in the network, it can initiate other attacks such as the Blackhole attack.

The decreased rank attack (Fig. 4) can have the largest impact, since this is the only type of rank attack that attracts more traffic than it should, thereby opening up possibilities to manipulate, inspect or drop most traffic. Many *intrusion detection systems (IDS)* and mitigation solutions have been proposed against this type of attack. In this paper, SVELTE [6], Secure-RPL [5], VeRA [7] and TRAIL [8] are expanded on as they all offer a unique approach to preventing rank attacks. VeRA is a well-cited mitigation solution against the rank attack and TRAIL was written as a response to it, as it exposes security risks that VeRA still has and proposes a solution to this problem.

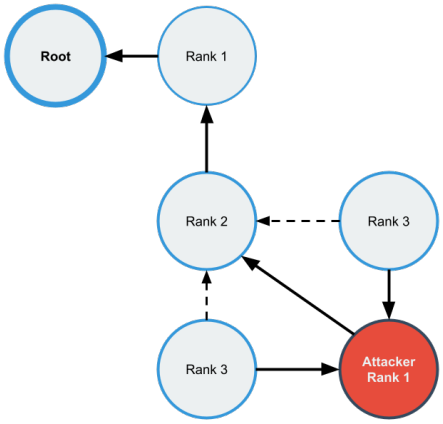

Figure 4. Decreased rank attack. Dotted connections are replaced by solid connections.

Secure-RPL

Secure-RPL was introduced in 2017 and has a unique take on preventing rank attacks. In essence, the writers argue that the closer a node is to the root, the more power it has to control or drop traffic from its sub-DODAG. Furthermore, a node with a large parent set could have a big impact on the topology if he were to falsely lower his rank. Also, a node with a large descendant set could have a large impact in the same way by falsely increasing its rank. Because of this, the authors propose the use of a threshold D and I, where D gets smaller as the rank gets lower and the size of the parent set increases. I gets smaller as the size of the descendant set increases. When a node wants to change its rank, the new parent checks if the rank change is within the limits of D and I. Secure-RPL also makes use of *hash-chains* to validate a node's proximity to a lower rank node, similar to VeRA (will be explained in-depth later). The simulation results from Secure-RPL are promising, but it is unclear what effect this proposal would have in mobile scenarios. Secure-RPL works best in large, dense networks with a big DODAG since the thresholds D and I have more effect in these situations.

Advantages: Secure-RPL has a unique approach using thresholds that could work for NOF as well. Secure-RPL seems to be conservative with resources based on simulation results. Glissa et al. (2016) [5] shows simulation results from Cooja [15] that indicate the effectiveness of Secure-RPL.

Disadvantages: Does not protect against slow rank attacks, where an attacker slowly decreases or increases its rank. Restricts freedom of node movement as nodes will have to adhere to the maximum rank difference D and I. Because of the hash chain verification and limit calculation, Secure-RPL could be computationally complex for resource-constrained devices.

SVELTE

SVELTE is an IDS that was introduced in 2013 and is made up out of three systems. The first system is hosted at the root node and reconstructs the entire DODAG in memory by sending mapping requests to every node in the DODAG. Each node then appends its rank, parent ID, and all neighbor ranks with ID's. The second system makes use of the reconstructed DODAG and tries to detect anomalies, such as the rank attack. The third module is a distributed firewall that defends against unwanted traffic from the Internet. SVELTE detects decreased rank attacks in similarly to VeRA and TRAIL, namely by checking if the rank is decreasing towards the root. Since SVELTE is an IDS, it only raises an alarm when an attack is discovered but does not talk about mitigation techniques for rank attacks.

Advantages: SVELTE provides a framework that is easy to extend, as the actual detection of attacks is not mathematically restricted (as is the case with VeRA for example). It can be used to possibly detect new attacks and be implemented in most network scenarios.

Disadvantages: SVELTE raises alarms but does not take action against rank attacks, which could be implemented in a future version. Due to the three systems, SVELTE can be complex to implement in an existing network. The root sends mapping requests to every node in the network, which will increase traffic based on the number of nodes in the network.

VeRA

Dvir et al. (2011) [7] propose a unique method called *Version number and Rank Authentication* (VeRA) that uses hash chains from the root to verify the validity of the version number and rank of a parent. VeRA enforces the requirement from [2] that the rank of all nodes in the path from a leaf to root should be monotonically decreasing, as malicious nodes will often advertise a lower rank than their real parent, illustrated in Fig. 4.

The root precalculates a *version number hash chain* of size n based on a random number r. Since the root is the only node that knows the entire hash chain, it can prove that it is the root by giving a prior version hash V_{i-1}, after which a node can verify that $h(V_{i-1}) = V_i$. This results in the entire DODAG being able to agree on the latest version number. For every $V_i \in V$ a *Rank hash chain* of size l is calculated by the root with a unique random number x_i for every V_i. For every rank hash chain, the maximum rank hash value (R_{mrh}) is defined as the final element in the chain. DIO messages must now contain R_{sender} which is the rank chain hash element belonging to the rank of the sender. Using R_{sender} and R_{mrh}, the rank of the sender can be verified by calculating $R_{check} = h^{l-Rank_{sender}}(R_{sender})$ which should be equal to R_{mrh}. The receiving node then calculates its own $R_{sender} = h^{Rank_{increase}}(R_{sender})$ and uses this when broadcasting its own DIO message. For a more in-depth explanation of the exact method and encryption schemes used, see [7].

Using VeRA, it can be proven that a parent of rank R_i is in range of a node that has a rank R_{i-1} such that $R_i > R_{i-1}$. This, however, is not enough to prevent all decreased rank attacks. Landsmann et al. (2014) describe the two types of rank attacks that are still possible when using VeRA [16]:

1) If a malicious node can prevent the propagation of a version number update, it can forge a new hash chain after receiving a second version number update. Because the attacker now has all knowledge of the hash chain, it can choose to advertise any rank it wants.
2) By copying the R_{sender} from their parents, malicious nodes can decrease their rank to the rank of their parents.

Advantages: Nodes can prove the rank of their parent using verifiable hash chains. VeRA is clearly described, making it easy to implement. VeRA protects against the version attack as well, which is another infamous attack on RPL.

Disadvantages: Computationally complex for resource-constrained devices. It is still possible to decrease rank in the two situations described above. In the original paper, VeRA was not implemented in a real system or simulation, so numbers about the effectiveness and efficiency are lacking.

TRAIL

TRAIL (Perrey et al., 2016) was published as an alternative to VeRA and exposes the same security risks as mentioned above. Landsmann et al. (2014) proposed a challenge-response mechanism to prevent the second attack, which inspired the invention of TRAIL.

TRAIL also uses a challenge-response mechanism (see Fig. 5) to verify the parents' rank from a node by sending a nonce upwards. The parent of the challenger then adds their rank to the message, which is then compared by the grandparent to the advertised rank of the parent. If an inconsistency is detected by the grandparent, it will not forward the message and thus the challenging node will not receive a response, thereby marking the parent as suspicious. If the rank is valid, the message will be forwarded to the root. Based on the assumption that the root can be trusted (called the *trust anchor*), it then signs the nonce and sends it back down to the challenging node. Note that the parent cannot include a different rank than advertised, as the grandparent will be aware of the advertised rank as well. This basic idea does not scale very well; if each node in the DODAG initiates its separate challenge, the network will be clogged with control messages, which depletes the resource-constrained nodes quicker.

To overcome this large increase in traffic, a scalable version of TRAIL is also explained in the same paper. In essence, intermediate nodes aggregate nonces from all children and efficiently forward this data to their parent in

a single message using Bloom filters, thereby reducing the number of messages to $2n$ whilst adding some complexity and length to the messaging. The root receives the full Bloom filter array in which the nonces from every node in the DODAG exist. After adding the version number, the root signs it and multicasts this data to all children. Upon receiving this message, a node can check if the nonce from their preferred parent is present in the Bloom filter. Furthermore, the node also checks if the entire Bloom filter array it sent upwards is present in the received array. To prevent malicious cooperating nodes from inserting their nonce at a different spot, nodes must also verify that their parents nonce is only present in one subarray.

Using TRAIL, nodes can be sure that the rank sequence in the path from the root is monotonically increasing. However, this does not fully prevent a decreased rank attack, as will be explained in the next section.

Advantages: Protects against more rank attacks than VeRA. The challenge-response mechanism validates the round-trip message flow. Constructing and validating the bloom filter is less complex for resource-constrained devices compared to Secure-RPL and VeRA.

Disadvantages: Because a node must wait on all children to send nonces, there is an increased routing convergence time. When NOFs are used, decreased rank attacks are still possible in specific network scenarios.

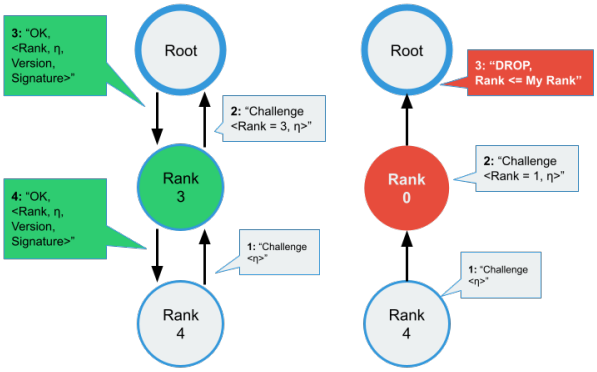

Figure 5. TRAIL message flow. Left: middle node correctly verifies his rank. Right: malicious node is not able to convince the root to sign the message.

V. Security Risks when using NOFs

The characteristics of NOFs compared to LOFs seem problematic for most mitigation and detection solutions such as SVELTE, VeRA, and TRAIL, as these solutions only check if the rank is decreasing towards the root. With NOFs, malicious nodes will still be able to decrease their rank without seeming suspicious, by performing the *NOF Rank Attack* that is explained in Fig. 6. This only works when the attacker is still in range of the parent but lies about the link quality by advertising a better rank than it should. This situation was not considered by Landsmann et al. (2014)

TABLE I
LIST OF DEFINITIONS

Symbol	Description
p_k	Parent k in P_i
P_i	Parent set of node n_i
t_{root}, t_{curr}, t_k	Timestamp of root node, current node and DTT of node k
r_k	Rank ratio of parent p_k
$\bar{r_i}$	Mean rank ratio of P_i
SD_i	Standard deviation of rank ratio in P_i

[16], and can be seen as a third possible attack on VeRA given that a NOF is used.

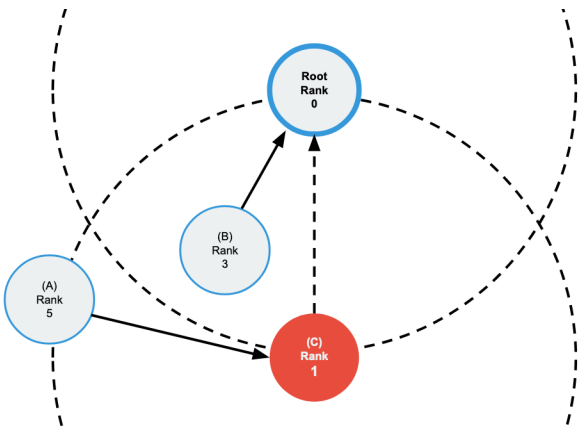

Figure 6. TRAIL exploit when using a NOF: Optimal path is A-B-Root, but A-C-Root is chosen. Dotted arrow indicates weak link. Dotted circle indicates range of center node.

Using LOFs would make sure that this exploit would not be possible, as nodes can easily verify that their child should have their rank plus a constant. With *downward-trip-time* (DTT), the time between a message from the root to a node in the network is indicated. This metric could provide insight into link speed and consequently be used to detect a decreased rank attack. Here, an adjustment to TRAIL, called *Time-TRAIL* (T-TRAIL) is proposed that allows all nodes in the network to measure their downward-trip-time from the root without adding any significant overhead or control messages. Note that T-TRAIL is currently only effective in situations where a NOF is used.

VI. T-TRAIL: A Novel Method to Detect the NOF Rank Attack

Assumptions

In this method, the following assumptions are made;

- All nodes in the network are homogeneous, meaning that they have the same amount of resources and transmitting/receiving power.
- The link quality between two nodes is the same for forwarding and receiving.

- The DTT is directly correlated with the metrics used by the OF. Meaning the higher the rank, the higher the DTT should be.

Method

TRAIL introduces a method to validate a rank sequence to the root, which can also be used to measure downward-trip-time. For this, the scaled version of TRAIL is used with slight modifications (see Tab. I for an overview of symbols used). Note that TRAIL does not specify when or how the challenge sequence should be initiated, which can be done in different ways. Here, it is suggested that nodes broadcast a special challenge initiation message containing the version number, to which children should reply and broadcast the same message. After this, all nodes in the DODAG should be aware of a new challenge cycle and send their nonces upwards. See Alg. 1 for when a node receives a response message. This is merely a suggestion and it is out of the scope of this paper to further specify and simulate this behavior. The modifications to TRAIL are as follows:

- The root includes a timestamp t_{root} in the set next to the version v_i, bloom filter b_i and signature $sign$.
- Instead of multicasting the signed message down, nodes broadcast this message. Nodes $n_i \in N$ should listen for any signed message sent by a $p_k \in P_i$ where P_i denotes the parent set of n_i that contains all lower-rank nodes in range of n_i
- Upon receiving a signed message, a node can calculate the downward-trip-time $t_k = t_{curr} - t_{root}$ belonging to parent p_k from the parent set P.

Further changes to prevent new exploits:

- When a node n_i wants to solicit to a new parent, it must publicly announce its interest by broadcasting a special soliciting message. This lets the sub-DODAG of n_i know that the path has changed and a new challenge must be transmitted.
- Whenever a parent switch is observed, a child should resend an individual challenge-response message and remeasure the DTT.

Proof: In this section, it will be proven that the measurement of DTT is resistant against spoofing in the long term, meaning that nodes can temporarily be convinced of a DTT but will detect an inconsistency quickly. This is done by showing how malicious behavior is mitigated in the two edge cases C1 and C2.

C1 Say a malicious node m rebroadcasts a challenge-response that does not come from his preferred parent (PP). In theory, this would allow it to advertise a different DTT than is true, implying that the PP has not yet sent his challenge-response. Because m broadcasts the challenge-response earlier than PP, PP will know that m is malicious and blacklist this node. This will isolate m and his entire sub-DODAG from the network, which will be easily detected by the sub-DODAG by the lack of traffic.

C2 Say a malicious node m changes parent after the challenging phase. As defined above, the new PP will only allow m if he publicly announces his change by broadcasting. This will notify the children of m that a parent switch has occurred, causing a new series of challenge-response messages and a remeasurement of the DTT.

Algorithm 1: Calculating t_k of all $p_k \in P_i$

```
Result: DTT for every p_k ∈ P_i
myRank;
myParent;
myChildren;
dttMap;
forwarded = False;
for every signed response m do
    t_curr = now();
    if rank(m) < myRank then
        Verify signature;
        dttMap[m] = t_curr − m.t_root;
        if m.sender == myParent then
            Broadcast m ;
            forwarded = True;
        end
    else if m.sender ∈ myChildren and not
     forwarded then
        Block m;
end
```

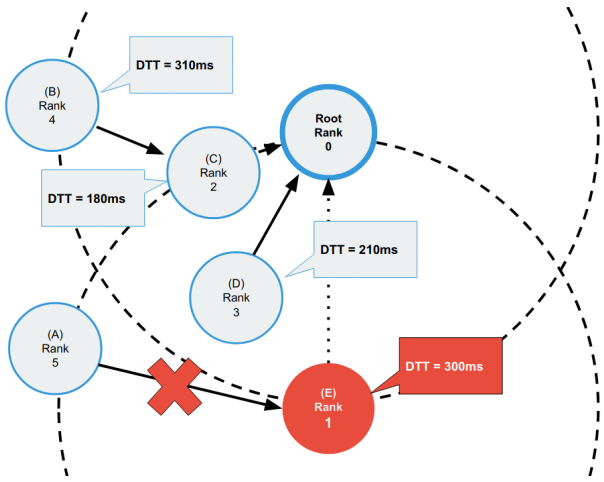

Figure 7. Detecting outliers using DTT: node E advertises rank 1 whilst having a DTT compared to a rank 4 node.

Attack Detection Using Downward-Trip-Time Measurement

Now that nodes $n_i \in N$ have an overview of the DTT from all parents $p_k \in P_i$, it can detect outliers by calculating the DTT / Rank ratio using equation 1.

$$r_k = \frac{t_k}{Rank(k)} \tag{1}$$

The bigger the size of P, the better it can be assumed that an outlier in this factor is a malicious node. An outlier means

that a node has promoted a rank that is too low compared to his measured DTT (see Fig. 7). An outlier can be detected by calculating the standard deviation of all the $r_k \in R_i$ (see equation 2 and 3), where R_i denotes the set of ratios belonging to parent set P_i

$$\bar{r_i} = \frac{\sum_{r \in R_i} r}{|R_i|} \tag{2}$$

$$SD_i = \sqrt{\frac{\sum_{r \in R_i} |r - \bar{r_i}|^2}{|R_i|}} \tag{3}$$

Next, a threshold should be specified as a multiple of SD_i to detect an outlier. Further research must be done in determining the right threshold value. Note that this would work particularly well for decreased rank attacks depicted in fig. 6 and 7, but would in general be a good way to enforce correct usage of OF.

VII. Performance Impact of T-TRAIL

For any addition to standard RPL, it is important to know the performance drawback imposed by it as security is only one of the many factors that determine the usefulness of a solution. Most mitigation solutions add complexity, traffic, and energy consumption which must be kept to a minimum in LLNs. As T-TRAIL builds upon TRAIL, this section will give an overview of the added overhead caused by measuring DTT. As simulating is out of the scope of this paper, metrics are selected that are based on theory and reasoning only. As it turns out, T-TRAIL has the worst effect on computational overhead and potentially adds a significant increase to the convergence time of the network.

- **Message count** Perrey et al. (2015) [8] explains that two messages per node are sent for the challenge-response protocol. Each node in the network will send and receive one message for the challenge-response. T-TRAIL does not increase this message count, as the timestamp will be piggybacked on the response message.
- **Message size** As nodes closer to the root will build increasingly larger Bloom filter arrays, the message size grows towards the root. The size of these messages depends on the parameters used for the Bloom filter. T-TRAIL adds a constant to the message size as a timestamp would be added in the response, which is usually 8 bytes.
- **Computational overhead** The largest performance drawback of T-TRAIL compared to TRAIL is that in normal TRAIL, a node only listens for the PP to respond, compared to a node now having to listen to all $p_k \in P_i$. This means each node will have to listen to and process messages from any node in its parent set
- **Convergence time** The convergence time of TRAIL was measured in a small experimental setup. The leaf node furthest away suffered the maximum overhead caused by TRAIL but remained under 20% in most cases. Without simulations or real-world experiments, there is no definite answer to what T-TRAIL will add to the convergence time, but it is expected to be significant as nodes will have to process information from all parents in the parent set.

Picking the Right SD_i Threshold

If the network is sparse (small parent sets) then nodes will have a less accurate SD_i, which makes T-TRAIL less functional. Because of this, parse networks are potentially better for resource efficiency but sub-optimal for effectiveness. Further research is needed to discover this true relationship. In this paper, no recommendation is given for the SD_i threshold that marks an outlier as malicious. By experimenting with this value, a threshold should be looked for that minimizes false positives and false negatives.

VIII. Discussion and Further Research

Every mitigation solution discussed in this paper has its advantages and disadvantages. There is no perfect solution yet, so choosing which one is right is up to the engineer to decide. The goal of this paper is to inspire further research on the use of DTT in OFs and mitigation/detection solutions. T-TRAIL is proposed as an improvement to TRAIL when NOFs are used. The effects of the decreased rank attack on NOFs is currently purely theoretical and needs to be quantified, for example by simulating it in Cooja [15]. If the simulation turns out to be successful, the next step would be to build a real-world experimental RPL network that uses T-TRAIL to detect the discussed decreased rank attack. Here, DTT is used only to detect this attack, but could potentially be used for more advanced mitigation solutions. Furthermore, a new OF could be designed that makes use of the DTT to select the best path based on this challenge-response mechanism.

Ethical considerations

This study bases its findings on the literature found through various online sources. The four mitigation solutions that are discussed in this paper were selected because they all offer a different approach to preventing and mitigating the rank attack. To the best of our knowledge, measuring and using DTT for mitigation purposes has not been discussed in previous papers.

As simulating and testing T-TRAIL is out of the scope of this paper, there is nothing to be reproduced yet. We hope that in further research, reproducible test cases will be developed that demonstrate T-TRAILs effectiveness.

This paper potentially exposes a new way to perform a decreased rank attack, with no intent to promote or assist illegal activities to be performed. By writing about RPL security, we hope to stimulate further research on this topic such that protocols for IoT can become safer, stable, and more scalable. This paper tries to refrain from making any claims regarding the usability and efficiency of the proposal, as there is more work to be done on simulating and testing before any results and claims can be presented.

IX. Conclusion

In this paper, the rank attack was considered to be a critical attack on RPL since it allows an attacker to take a strategic position in the network. Moreover, standard RPL is ill-equipped against this type of attack and thus mitigation is required in networks where security is critical. The research question that this paper tried to answer is: "What effect does the use of a nonlinear objective function have on existing mitigation solutions for the rank attack on RPL and how can possible exploits be defended against?". Four approaches to mitigating the rank attack were analyzed. Secure-RPL has a unique approach of limiting rank increase and decrease to prevent rank attacks but thereby limits the freedom of movement a node has. SVELTE is an IDS that can detect a broad range of attacks, but only raises an alarm if an attack is detected. VeRA adds complexity and does not fully prevent the rank attack, which is why TRAIL was proposed. TRAIL uses a challenge-response mechanism to validate a path from any node to the root. In this paper, it is shown that even with this mechanism in place, a decreased rank attack is still possible in the case where NOFs are used. Here, an addition to TRAIL is proposed that allows nodes to know the DTT to each parent in the parent set. By measuring the DTT, outliers can hopefully be detected by comparing the rank increase with the link quality. More research must be done on simulating and validating this proposal.

X. Acknowledgment

This research is part of an assortment of five papers written by my fellow undergraduates on RPL security. I would like to thank my peers for reviewing my work and for the weekly update sessions. I would like to thank Ruben Stenhuis specifically for the extended collaboration during the literature research.

References

[1] P. Pongle and G. Chavan, "A survey: Attacks on rpl and 6lowpan in iot." Institute of Electrical and Electronics Engineers Inc., 4 2015, this is a test.

[2] T. Winter, P. Thubert, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "Rfc 6550 - rpl: Ipv6 routing protocol for low-power and lossy networks," 2012. [Online]. Available: http://www.rfc-editor.org/info/rfc6550.

[3] 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.

[4] 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.

[5] G. Glissa, A. Rachedi, and A. Meddeb, "A secure routing protocol based on rpl for internet of things." Institute of Electrical and Electronics Engineers Inc., 2016.

[6] S. Raza, L. Wallgren, and T. Voigt, "Svelte: Real-time intrusion detection in the internet of things," *Ad Hoc Networks*, vol. 11, pp. 2661–2674, 11 2013.

[7] A. Dvir, T. Holczer, and L. Buttyan, "Vera - version number and rank authentication in rpl," 2011, pp. 709–714.

[8] H. Perrey, M. Landsmann, O. Ugus, M. Wählisch, and T. C. Schmidt, "Trail: Topology authentication in rpl," in *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, ser. EWSN '16. USA: Junction Publishing, 2016, p. 59–64.

[9] A. Le, J. Loo, Y. Luo, and A. Lasebae, "Specification-based ids for securing rpl from topology attacks," vol. 1, 2011.

[10] 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

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

[12] N. Pradeska, Widyawan, W. Najib, and S. S. Kusumawardani, "Performance analysis of objective function mrhof and of0 in routing protocol rpl ipv6 over low power wireless personal area networks (6lowpan)." Institute of Electrical and Electronics Engineers Inc., 2 2017.

[13] S. Capone, R. Brama, N. Accettura, D. Striccoli, and G. Boggia, "An energy efficient and reliable composite metric for rpl organized networks." Institute of Electrical and Electronics Engineers Inc., 11 2014, pp. 178–184.

[14] M. A. Boudouaia, A. Ali-Pacha, A. Abouaissa, and P. Lorenz, "Security against rank attack in rpl protocol," *IEEE Network*, vol. 34, pp. 133–139, 7 2020.

[15] T. Mehmood, "COOJA network simulator: Exploring the infinite possible ways to compute the performance metrics of IOT based smart devices to understand the working of IOT based compression & routing protocols," Dec. 2017.

[16] M. Landsmann, M. Wahlisch, and T. Schmidt, "Topology authentication in rpl." Institute of Electrical and Electronics Engineers (IEEE), 12 2014, pp. 73–74.