

Detection and Mitigation of Version Number Attacks in RPL-Based IoT Networks:

Evaluating the Real-World Readiness of Existing Defenses

Fahad Alfaify

Student ID: 262000500

Research Methodology in Computer Science (AI800)

Lecturer: Dr. Mohammed Anbar

Final Research Project

Table of Contents

Abstract	1
Chapter 1: Introduction and Problem Definition	2
1.1 Introduction	2
1.2 Background	3
RPL and the DODAG.....	3
The Version Number Attack	3
Types of Existing Defenses	3
The Security Gap	3
1.3 Problem Statement	4
1.4 Research Objectives	4
1.5 Research Scope	5
Chapter 2: Literature Review.....	6
2.1 Overview	6
2.2 Review of Selected Studies.....	6
2.3 Summary of Reviewed Studies (Synthesis Matrix)	10
2.4 Connections, Conflicts, and Research Gap	11
Chapter 3: Research Methodology.....	13
Phase 1: Review and Classify the Defenses.....	13
Phase 2: Identify the Added Control Signals and Define the Attacker Model	13
Phase 3: Build the Test Scenarios in Cooja/Contiki	14
Phase 4: Run the Experiments and Measure Robustness.....	14
Phase 5: Validate one or more defense on real hardware	15
Phases 6: Fair Head-to-Head Comparison and Real-World-Readiness Verdict.....	15
Experimental Parameters	15
Expected Outcomes.....	16
Ethical Considerations	16
Limitations	17
Gantt Chart.....	17
References.....	19

Abstract

RPL (Routing Protocol for Low-Power and Lossy Networks) is the main routing protocol for IoT and 6LoWPAN networks. RPL is a tree-based protocol, with nodes in a tree known as a DODAG, rooted at a border router. The tree is rebuilt by all nodes, and the root raises a global version number in the DIO messages to repair the tree. This version number field is not authenticated in RFC 6550. A Version Number Attack (VNA) is an attack in which one insider node continually increases the version number and causes the entire network to rebuild repeatedly. This is a waste of energy, increases traffic control, decreases packet delivery ratio (PDR), and increases delays. Even though there are many defenses suggested, they are tested only in a few small, static Cooja simulations, they assume that the sink is trustworthy, and most of them assume that there is only one non-colluding attacker. Even worse: Every defense imposes its own unauthenticated control signals (alert tags, self-reported counts, majority votes) that re-open the same integrity gap that the attack exploits, but no study tests an attacker that forges these signals or colludes. No study validates real hardware, mobility and real scale. The goal of this research is to test the effectiveness of the existing defenses of VNAs in more difficult and realistic settings and to fairly compare them. The method involves a combination of simulation-based evaluation (Cooja/Contiki), emulated Zolertia Z1 motes, one validation on real IoT hardware, and a fair head-to-head comparison based on detection accuracy, PDR, control overhead, energy, and delay. The anticipated contributions are easily identified as evidence of the readiness of current VNA defenses and suggestions for more robust defenses in the real world.

The keywords are RPL, Version Number Attack, IoT security, DODAG and intrusion detection.

Chapter 1: Introduction and Problem Definition

1.1 Introduction

RPL (Routing Protocol for Low-Power and Lossy Networks) is the primary routing protocol for battery-powered, small devices, like battery-powered sensors in a 6LoWPAN network. RPL is a tree-based protocol where nodes connect to form a tree known as a DODAG with a root at a border router. This tree indicates to each node how to send data towards the root. For these devices, good routing is very crucial as they possess very little energy, small memory and weak radios (Sharma et al., 2023). When the routing tree continually breaks and rebuilds, energy is wasted on the nodes and data is lost. Hence, a stable and safe RPL is essential to maintain the stability of an IoT network for a longer time.

Version Number Attack (VNA) is a serious attack on RPL. In response to a broken tree, the root can increment a global “version number” in the DIO messages it sends and all nodes will rebuild the DODAG. The problem is that there is no authentication for this version-number field in RFC6550 (Alsukayti & Singh, 2022). A VNA has one node which continues to increment the version number in its DIO messages. This makes the whole network to build the tree again and again. The attack consumes a lot of energy, introduces a lot of additional control traffic, decreases the packet delivery ratio (PDR) and increases delay (Mayzaud et al., 2017). This attack is a major concern for IoT as one weak node can impact the entire network.

Although there are many defenses against VNA, they are tested in only small and static Cooja simulations and most of the defenses assume that there is only one attacker who does not collude (Osman et al., 2021; Nandhini et al., 2021). Many of these defenses also rely on the sink, and introduce their own additional control signals, like alert tags, self-reported counts or majority votes. However, these new signals are not themselves authenticated and can be forged, and no study has been conducted on an attacker who can forge these and/or collaborate with other attackers (Sharma et al., 2023; Rouissat et al., 2023). Furthermore, these defenses have not been evaluated on real hardware, node mobility or on a larger scale. For this reason, the effectiveness of existing defenses in the real world remains to be tested. Existing VNA defenses will be tested for their robustness and resilience against more challenging and realistic scenarios (such as colluding attackers, an untrusted sink, node mobility, and larger networks), and they will be fairly compared to assess their readiness for actual deployment.

1.2 Background

RPL and the DODAG

RPL is the main routing protocol for low-power IoT and 6LoWPAN networks. The nodes connect together to create a tree, which is referred to as a DODAG, with a border router (the sink) at the root of the tree. Data can be passed up to the root by each node selecting its parent by using control messages, called DIO. The first time the tree cracks, the root can will begin repair to repair the paths.

The Version Number Attack

If the entire tree is to be repaired, the root increases a global version number in the DIOs it sends out, and all the nodes in the tree rebuild the DODAG. The issue is that the version-number field isn't authenticated in RFC 6550. In a Version Number Attack (VNA), one insider node continuously increases this version number, causing the network to rebuild over and over again, and resulting in the following energy, control-overhead, PDR and delay costs as described in Section 1.1.

Types of Existing Defenses

Currently there are three families of VNA defenses. The first is extra monitoring, which involves having additional monitor nodes that monitor the traffic to discover and identify the attacker (Mayzaud et al., 2017). The second one is in-protocol cooperative checks, in which normal nodes cooperate within RPL to validate the version number and possibly to reduce the attack (Nandhini et al., 2021; Alsukayti & Singh, 2022; Sharma et al., 2023; Rouissat et al., 2023). The third one is machine learning detection in which a trained model is used to determine whether the traffic is normal or attack (Osman et al., 2021).

The Security Gap

The new defense adds its own additional control signals (alert tags, self-reported counts, majority votes, etc.) but these signals are not authenticated (Alsukayti & Singh, 2022; Sharma et al., 2023; Rouissat et al., 2023). This again exposes the same integrity problem that is exploited in the attack, but no research validates an attacker who is able to fake these signals or an attacker who collaborates with others. Most defenses are only evaluated in small static Cooja simulations, they rely on the sink and they assume of one attacker (Mayzaud et al., 2017; Nandhini et al., 2021). For that reason, it is too early to tell how well current VNA defenses will be able to withstand real-world attacks.

1.3 Problem Statement

The DIO version-number field is not authenticated by RFC 6550, so any node in the DODAG can increment the version-number and cause all nodes to repeat global repair – Version Number Attack (VNA). The whole network can therefore be degraded by a single, inexpensive 'insider', leading to increased energy consumption, control overhead, and delays, as packet delivery fails. Detecting and blocking VNA is a critical and realistic challenge for the deployment of IoT devices, which are weak and deployed in critical locations.

There were a number of ideas, but all of them had some issues. Mayzaud et al. (2017) are able to detect the attacker but are not able to block it and require additional monitor nodes. The ML-LGBM model is a machine learning model with 99.6% accuracy, but the model is 348KB which is too large to fit the motes' 8KB of RAM. Nandhini et al. (2021) employ a Claim algorithm without providing any results for detection, and it does not work well in case of multiple bad nodes. CDRPL proposed by Alsukayti and Singh (2022) detects and prevents the attack and it is tested in simulation and assumes the trust of the sink. Sharma et al. (2023) suggest the use of QSec-RPL and test the mobility, but the accuracy is 0.75 and the reported counts can be faked. Rouissat et al. (2023) employ parent supervision, but test only a small 16 node network and are not using alert tags that are protected.

The big difference is evident. Most solutions are tested in a small and fixed simulation, they rely on the sink, and they only employ one attacker. Additionally, new control signals (e.g., alert tags and self-reported counts) are introduced in each solution, which are not protected and can be spoofed by an attacker (Alsukayti & Singh, 2022; Sharma et al., 2023; Rouissat et al., 2023). No study tests real hardware, node mobility, fake signals and/or attackers who cooperate. So we have no idea whether these solutions are effective in the real world and there is a need for more research.

1.4 Research Objectives

This research aims to verify the effectiveness of the existing VNA solutions in more challenging and realistic scenarios (e.g. attackers coordinating, an untrusted sink, node mobility, larger networks, and faked signals) and to fairly compare them. To achieve this objective, the following objectives are:

1. To compare and review the six core defenses of VNAs based on their detection, mitigation, memory cost and attacker model.
2. To discover the control signals which each defense introduces and to specify an attacker which can create those control signals or collaborate with other attackers.

3. To create test cases using node mobility, larger networks and an untrusted sink.
4. To test the accuracy, PDR, control overhead, energy and delay of the defenses against this attacker.
5. To fairly compare the defenses under the same conditions, and display which defenses are ready for the real world.

1.5 Research Scope

Version Number Attacks (VNA) in RPL-based IoT networks are the main focus of this study.

The study reviews six core VNA defenses in depth: Mayzaud et al. (2017), Osman et al. (2021), Nandhini et al. (2021), Alsukayti and Singh (2022), Sharma et al. (2023), and Rouissat et al. (2023). These six constitute the corpus which is compared to each other. Moreover, four additional studies are reviewed in Chapter 2 (Almusaylim et al., 2020; Verma & Ranga, 2019; Yavuz et al., 2018; Alzubi et al., 2020) to broaden the scope of security context. The study by Almusaylim et al. (2020) is reviewed but is intentionally not included in the head-to-head comparison since it protects against both the rank attack and the VNA, and separating its VNA-specific behaviour from the rank attack would be confounded by the rank attack logic.

Most of the testing is performed in the Cooja/Contiki simulator with emulated Zolertia Z1 motes. At least one defense is also validated on real IoT hardware to verify the readiness beyond simulation. The study gradually increases the difficulty of the tests, such as a forging and colluding attacker, an untrusted sink, mobility of nodes, and larger networks. Detection accuracy, Packet delivery ratio (PDR), Control overhead, Energy consumption and Delay are used to measure each defense.

- Out of scope: Other RPL attacks (e.g., rank and blackhole attacks) are not covered in this study.
- Out of scope: It is not intended to be a complete new routing protocol, but it is a study of and a comparison of existing defenses.

Chapter 2: Literature Review

2.1 Overview

This chapter summarizes 10 studies related to the security in RPL-based IoT networks. The first six studies are defenses that are based on the Version Number Attack (VNA). The last four studies are included to provide a broader perspective: They address related RPL routing-attack detection and general machine-learning intrusion detection. Each study includes an aim, method, key results and the main limitations cited in the text. At the end of this booklet there is a summary table and a short synthesis.

2.2 Review of Selected Studies

Mayzaud et al. (2017) suggest a distributed monitoring approach for detecting version number attacks in RPL networks. Their approach is to not make the detection work on the small nodes, so they create a second RPL instance consisting of additional, more capable “monitor” nodes which listen in silence to the DIO traffic. If the monitor receives a version number that is greater than its own, it notifies the root of the sender and its neighbours, and the root will locate the real attacker; they also model the placement of the monitors as an optimisation problem that is solved using an ILP solver. The method was able to always identify the attacker on a 20-node grid in Cooja, and the false-positive rate was reduced to near zero with two monitors watching a node, and was projected up to 1000 nodes (Mayzaud et al., 2017). Its main drawbacks are that it only detects and locates the attacker and does not prevent it, requires additional hardware (the monitor nodes), assumes the presence of a single attacker, and is tested only on a small static grid; besides, no results of energy, delay or real hardware are provided (Mayzaud et al., 2017).

In this paper, Osman et al. (2021) restructure version number attack detection as a machine-learning problem and create a model called ML-LGBM. They extract RPL and 6LoWPAN control traffic from Cooja simulations, and create a large dataset of over one million records, cleaning and reducing the features from 113 to 17, and using ten-fold cross-validation. The model has a very low false-positive rate and reaches 99.6% accuracy, a model size of approximately 348 KB, and a training time of approximately 140 seconds (Osman et al., 2021). The main disadvantage is that this “lightweight” model is 348 KB, which is too large to fit the 8 KB RAM of the Zolertia Z1 motes it targets; it also runs offline on a normal computer, so it is not really ready for the device; it only detects the attack and does not stop it; the data are imbalanced, which can make the accuracy appear higher than it is (Osman et al., 2021).

Nandhini et al. (2021) suggest a “Claim algorithm” that does not require cryptography to detect version number attacks. If a node receives a DIO with a higher version number, it does not

accept it immediately, but it requests a trusted list of its neighbours from the root node and then polls them, and a majority vote determines whether the increase is real or not (Nandhini et al., 2021). They also provide an overview of the previous defenses, including VeRA and TRAIL. The main limitation of the paper is that it does not report any result of the detection (i.e. accuracy, precision, recall) of its own method, the experiments only show that the attack increases energy and hop count. The basic configuration parameters like number of nodes and run time are not provided, so the work cannot be reproduced, and a majority vote may not be possible if multiple malicious neighbours collaborate (Nandhini et al., 2021).

Alsukayti and Singh (2022) suggest a lightweight in-protocol scheme called CDRPL that not only detects but also prevents version number attacks. The main idea is that a real version increase can only come from the root, so before joining the repair, a node checks if the change is a real change from the sink; it uses two small tagged messages (S-DIO_{nv} and S-DAO_{nv}) which share existing header fields, and when a node discovers an attacker it notifies the sink to blacklist the attacker. CDRPL achieved an accuracy of more than 99.3% within approximately 1 second, reduced energy consumption by more than 60%, reduced control overhead by at least 75%, and even worked with multiple attackers simultaneously (Alsukayti & Singh, 2022). The major drawbacks include the fact that it is only tested in simulation, it assumes the trustworthiness of the sink, and its new control flags are claimed to be safe but never tested with an attacker that forges or drops these flags, and its convergence time is also increased by around 8 seconds even without attack (Alsukayti & Singh, 2022).

Sharma et al. (2023) suggest the only reviewed scheme which tests node mobility is QSec-RPL. Every node keeps a count of the DIO messages associated with version changes, assigns a Q-Learning penalty to them, and sends a small 16-bit count to the border router, and every 120 seconds the border router runs a statistical outlier test (Tukey's IQR) to identify attackers. They first measure the damage (a static attacker raises DIO traffic by about 583%) and then test 10 to 70 nodes with 10–30% attackers, both in static and 50%-mobile settings, and demonstrate that the scheme fits into the 8 KB of RAM that is available in a real Z1 mote. (Sharma et al., 2023) The good is that it suffers from mobility and reports its own poor results: accuracy 0.75 and precision 0.38 in the largest and most active network. The limitations are only detection, short time window of 120 seconds, and self-reported counts are not validated, which means that the same type of attacker could simply lie about the counts, and colluding attackers is beyond its scope (Sharma et al., 2023).

The lightweight concept is taken to the extreme by Rouissat et al. (2023) who propose a “parent supervision” scheme. Each parent sees the version number it's own children are advertising, and

a child that raises it wrongly is marked as suspicious in a DAO to the root, which then broadcasts a blacklist, which means that the attacker is dropped and if he wants to be a parent again, he has to be replaced, and the scheme reuses the same header tags, so it doesn't need new messages, encryption, or monitor nodes. The attack increased the overhead of control messages from 673 to 4,736 messages in a 16-node Cooja run, while the delivery dropped from 98.6% to 8.5%, and the defense reduced overhead by 77.1%, restored delivery to 98.5%, and used only approximately 5.8 KB of additional memory (Rouissat et al., 2023). The drawbacks are that it is evaluated on a single small network executed once, it depends on the trustworthiness of the sink, it does not make any comparisons with other schemes, and its alert tags are not secured, which means that a node might falsely accuse an honest parent (Rouissat et al., 2023).

Almusayym and his colleagues (2020) present an attack-resistant RPL protocol named SRPL-RP which is able to resist two attacks simultaneously: the rank attack and the version number attack (VNA). There are three steps to the method. First, for detection, each node appends a time-stamp threshold to its DIO messages, verifies whether the sending node ID is in a monitoring table, and compares node ranks (parent, child, and sibling ranks) to detect fake ranks or an illegitimate version number raise. The second part of mitigation is that a node will only accept a higher version number if at least half of the nodes in its neighbour table agree to the same version number, and will cancel any fake raise with a majority vote. Third, in isolation, if an attacker is detected, it is placed in a blacklist table and an alert message is sent to other nodes to avoid it. They simulate 20 nodes (1 root, 14 normal, 5 malicious nodes: 4 rank attackers, 1 version number attacker) in a 100m x 100m area with the Cooja/Contiki simulator using emulated Wismote motes, with 3 topologies (grid-center, grid-random, random), and run 60 times for 60 minutes per run. The best reported results are the packet delivery ratio (PDR) of 98.48%, control traffic of 991 packets/second, average energy of 1231.778 joules and accuracy of 98.17%, outperforming the compared defenses Standard RPL with Attacks, SBIDS and RPL+Shield. The main limitations are that the study is simulation-only on a small, static 20-node network with just one version number attacker, it assumes the root/sink cannot be attacked (its ID is encrypted and trusted), and its new control signals (timestamp, alert messages, and the majority-vote version field) are themselves unauthenticated and are never tested against a node that forges or colludes to fake them; mobility is only named as future work. This paper is relevant to the version-number-attack gap because it is one of the few defenses that both detects and mitigates VNA, but, like the other defenses, relies on the sink and introduces new and unprotected control signals, so its readiness against forged signals, colluding attackers, an untrusted sink, mobility, and larger networks has not been proven in reality.

Verma and Ranga (2019) propose a network intrusion detection system (ELNIDS) to detect routing attacks in RPL-based IoT networks. They train four ensemble classifiers (Boosted Trees, Bagged Trees, Subspace Discriminant and RUSBoosted Trees) and use a majority voting approach to classify each traffic record as an attack or normal (Verma & Ranga, 2019). They use and test their own synthetic dataset, RPL-NIDDS17, which is generated using the NetSim simulator with 20 features and traces of 7 attacks: Sinkhole, Blackhole, Sybil, Clone ID, Selective Forwarding, Hello Flooding, and Local Repair (Verma & Ranga, 2019). The highest accuracy and the area under the ROC curve was obtained by Boosted Trees (94.5% and 0.98 respectively), and the lowest accuracy was obtained by Subspace Discriminant (77.8% and 0.87 respectively) (Verma & Ranga, 2019). The biggest drawback is that this is an offline study based on a pre-recorded dataset, as stated by the authors, running ELNIDS on real smart nodes is future work, and they do not report energy, memory, control overhead, or delay, and they assume the presence of a powerful always-listening sniffer (Verma & Ranga, 2019). Most importantly for this review, ELNIDS doesn't test forged control signals, colluding attackers, untrusted sink, or mobility, so the version number attack is not covered. This paper is valuable as a representative of the overall "detection-only, simulation dataset" trend that this study suggests is not yet battle-ready for VNA defense in the real world.

In this article, Yavuz et al. (2018) create a deep-learning approach to identify three RPL routing attacks in IoT networks: decreased rank, hello-flood, and version number (VN) modification. They are able to simulate real RPL code using Cooja/Contiki and create networks of 10 to 1000 nodes and convert the captured packet traces (PCAP) to CSV files. They extract features from these files (e.g., transmission rate, reception rate, DIO/DAO/DIS packet counts), remove weak features and train a 7-layer neural network (Keras, AdaDelta, sigmoid output) using their large "IRAD" data set of 64.2 million values. The VN model achieves an average precision of 0.94, recall of 0.94, F1 of 0.95 (AUC 94.7) and training accuracy of 95.2% for the version number attack, while the hello-flood model performs best, with an F1 of close to 99%. The major drawback for this study is that this work is detection-only and offline: it is only a detection, it does not do anything to mitigate or recover from the attack, and it requires an external device or special node to gather all packet traces, it does not run on the constrained motes. It is also simulation-only (no real hardware), assumes a small, fixed number of attacker nodes (5–20% malicious), and never tests an attacker that is able to forge control signals or collude (no detection is done within the RPL protocol on captured signals). So, Yavuz et al. (2018) demonstrate that VN attacks can be detected at scale, but they leave open the actual gap to the real world that this study aims to address: no mitigation, no untrusted-sink or forged-signal attacker, no mobility, and no hardware validation.

Alzubi et al. (2020) develop an anomaly-based intrusion detection system (IDS) and attempt to increase its speed and accuracy by selecting the most informative traffic features. They suggest a new feature selection method, modified binary grey wolf optimisation (MBGWO) which introduces a fourth “omega” wolf into the standard binary grey wolf optimiser, so that four wolves as opposed to three select the next set of features (Alzubi et al., 2020). They test it in MATLAB on the offline NSL-KDD benchmark dataset using a support vector machine (SVM) with an RBF kernel as the classifier (Alzubi et al., 2020). The results are promising: MBGWO achieves 99.22% accuracy, 99.10% detection rate and the false-positive rate is very low (0.0064%) with only 14 features reduced from 41 (Alzubi et al., 2020). This paper is methodologically relevant to the gap, but not directly comparable: it shows the feature-reduction pathway (41 features down to 14 with the same accuracy) that a heavy ML detector, like ML-LGBM proposed by Osman et al., would need to follow in order to reduce the memory usage of an on-device inference within an RPL stack to that of a real mote. It is important to note that the 99.22% accuracy reported in the headline is for the merged-and-resplit partition, while the paper itself reports an average accuracy of 81.58% (Table 3) for the standard separated NSL-KDD partition (where unseen attacks are in the test set), showing how sensitive such offline classifiers are to attacks that are not seen during training. It is similar to the other detection-only offline classifiers, in that it does not offer any in-protocol mitigation, nor any defence against forged or colluding control signals (Alzubi et al., 2020).

2.3 Summary of Reviewed Studies (Synthesis Matrix)

Study (Year)	Method / Approach	Key Findings	Main Limitations
Mayzaud et al. (2017)	Distributed passive monitoring on a separate RPL instance + ILP monitor placement; detection and localization only	Attacker always found; false positives ~20% down to ~0 with double-coverage; scales to 1000 nodes	No mitigation; single attacker; small static grid; extra monitor hardware; no energy/delay/hardware results
Osman et al. (2021)	Supervised ML (LightGBM), ML-LGBM; ~1M-record dataset; offline/centralized	99.6% accuracy; model ~348 KB; ~140 s training	348 KB cannot fit 8 KB motes; detection only; imbalanced data; not run on device
Nandhini et al. (2021)	Claim algorithm; root-assisted majority-vote verification; crypto-free	Shows attack raises energy and hop count	No detection metrics; not reproducible; majority vote fails with colluders
Alsukayti & Singh (2022)	CDRPL; in-protocol cooperative verification; detection + mitigation	>99.3% accuracy; ~1 s; >60% energy and ≥75% overhead saved; multi-attacker	Simulation only; trusts sink; new flags not tested against forging; ~8 s convergence tax
Sharma et al. (2023)	QSec-RPL; Q-Learning + Tukey-IQR outliers; mobility-aware	~583% DIO surge; fits 8 KB; ~90% avg accuracy	Detection only; accuracy 0.75 / precision 0.38 at scale; 120 s delay; counts forgeable; no colluders
Rouissat et al.	Parent supervision; reuses DAO/DIO	Overhead -77.1%; PDR 8.5% to	One 16-node run; trusts sink; alert

Study (Year)	Method / Approach	Key Findings	Main Limitations
(2023)	tags; detection + mitigation + recovery	98.5%; +5.8 KB memory	tags unprotected; no comparison
Almusaylim et al. (2020)	SRPL-RP (in-protocol): timestamp threshold + node-ID monitoring table + rank comparison (detect); majority-vote version check (mitigate VNA); blacklist + alert (isolate). Cooja, Wismote, 20 nodes, 3 topologies, 60 runs	Best PDR 98.48%; control traffic 991 pkt/s; energy 1231.778 J; accuracy 98.17%; beats SBIDS and RPL+Shield (lower FP)	Simulation-only; small static 20-node net, single VN attacker; sink assumed unbreakable; new signals (timestamp, alerts, majority-vote) unauthenticated, untested against forging/collusion; mobility = future work
Verma & Ranga (2019)	Offline ensemble ML NIDS (Boosted/Bagged/Subspace/RUSBoosted Trees + majority vote); RPL-NIDDS17 NetSim dataset, 20 features	Boosted Trees best: 94.5% accuracy, 0.98 AUC; Subspace Discriminant worst: 77.8%; detects 7 routing attacks	Offline dataset only, no on-mote run; no energy/memory/overhead/delay; assumes powered sniffer; does NOT cover VNA, collusion, untrusted sink, or mobility
Yavuz et al. (2018)	7-layer DNN (Keras/AdaDelta) on Cooja-simulated RPL features (IRAD, 10–1000 nodes); detection-only, offline packet-trace classification	VN: precision 0.94, recall 0.94, F1 0.95, AUC 94.7; HF best (F1 ~99%); IRAD ~64.2M values, up to 1000 nodes	Detection-only; simulation-only, static, no mobility, no hardware; needs external capture, not on-mote; VN field stays unauthenticated; no forged/colluding attacker
Alzubi et al. (2020)	Feature-selection IDS: MBGWO (adds omega wolf) + SVM (RBF), offline on NSL-KDD in MATLAB; detection-only, general (non-RPL) traffic	99.22% accuracy, 99.10% DR, 0.0064% FPR; features 41 → 14 (but 81.58% under standard separated split)	Not about RPL/VNA; general wired IDS on static offline NSL-KDD; no IoT/mote test, no energy/PDR/delay/overhead; no mitigation; extra wolf raises runtime

2.4 Connections, Conflicts, and Research Gap

The 10 studies are clearly linked together. They all have to do with the security of RPL-based IoT networks, and most of them are only simulation-based, with the majority being tested in Cooja/Contiki on small networks (Mayzaud et al., 2017; Nandhini et al., 2021; Rouissat et al., 2023). Most of them also rely on the sink and employ only one attacker and several have introduced new control signals (alert tags, self-reported counts, majority votes) which are not authenticated (Alsukayti & Singh, 2022; Sharma et al., 2023; Rouissat et al., 2023; Almusaylim et al., 2020).

The studies are also in conflict with each other. Some only detect the attack (Mayzaud et al., 2017; Osman et al., 2021; Nandhini et al., 2021; Sharma et al., 2023; Verma & Ranga, 2019; Yavuz et al., 2018; Alzubi et al., 2020), while others also stop and recover from it (Alsukayti & Singh, 2022; Rouissat et al., 2023; Almusaylim et al., 2020). The machine-learning studies achieve very high accuracy (99.6% in Osman et al., 2021; 99.22% in Alzubi et al., 2020) but the models are heavy or offline and have not been proven to run on the small motes. The lightweight

in-protocol studies fit the motes but can have degraded accuracy when mobility and scale (0.75 in Sharma et al., 2023).

This is not a new weakness; it's the historic trend of RPL version-number defence. Early schemes like VeRA (Dvir et al., 2011) were extended by introducing hash-chain and MAC mechanisms that allow version increments to be authenticated, but TRAIL (Perrey et al., 2013) demonstrated that these could be spoofed, since the integrity signals that were added were not end-to-end verifiable against an adversarial intermediate node. Each of the defences reviewed in 2022-2023 introduces an additional control signal (alert tags, self-reported counts, majority votes) to monitor the version number but does not authenticate the new signal. It is this repetition over the course of over ten years that makes it imperative to test the defence against an attacker who colludes or forges on these signals, instead of assuming the signals are correct.

These studies leave one thing apparent. They have very few simulations to test their defenses, they rely on the sink, and they assume of one non-colluding attacker. The new defense introduces its own unauthenticated control signals that create the same integrity problem exploited by the attack, but no study evaluates an attacker who is able to forge these control signals or who is colluding. No study verifies the defenses on real machines, with node mobility, or on a larger scale. As a result, the readiness of existing VNA defenses in the real world remains untested and this study aims to address this gap.

Chapter 3: Research Methodology

This chapter describes the methodology for conducting this study. It is a simulation-based study, meaning that most of the work is performed in a network simulator, not on the actual network. The primary tool is Cooja simulator running on Contiki-NG OS. Cooja enables us to create an RPL network, add attacker nodes, and observe the events that unfold, one by one, with emulated Zolertia Z1 motes (small, low-power devices with approximately 8 KB of RAM, similar to real IoT nodes). We also use real IoT hardware for one validation at the end, so it is not just a simulation study.

The work is divided into 6 phases. Each phase fulfills one of the six objectives, identifies the tools used in the phase, and specifies the metrics generated in the phase. The phases are executed in sequence since each phase provides inputs to the subsequent phase.

Phase 1: Review and Classify the Defenses

During this phase, we revisit the six core VNA defenses, more in-depth. Each paper is read and the following is noted: How it works, what it can do, and what it cannot do. We look at Mayzaud et al. (2017), Osman et al. (2021), Nandhini et al. (2021), Alsukayti and Singh (2022), Sharma et al. (2023), and Rouissat et al. (2023). These are sorted by type (detection only, or detection and mitigation) and recorded as to where they have been tested and how large. The output is a clear comparison table that is the foundation for the fair comparison in Phase 6. Tools: reading and note taking, no simulation yet. Metrics: Detection accuracy, mitigation capability, and test scale of each defense, as reported in their paper.

Phase 2: Identify the Added Control Signals and Define the Attacker Model

Each defense introduces its own set of novel control signals to counter an attack, but RFC 6550 does not validate the version-number field and many of the new signals are not protected. In this phase, we list all the added control signals in each defense (e.g. alert tags in Rouissat et al., 2023, self-reported counts in Sharma et al., 2023, and majority votes in Nandhini et al., 2021) and then introduce a more powerful attacker model. This attacker can forge these unauthenticated signals and can collude with other attacker nodes.

The attacker model also includes version-number decrease, not just increase: a malicious node can advertise a lower version number, which is not detected by the monitor based on “higher-than-my-own” (Mayzaud et al., 2017).

We also define an untrusted sink. Importantly, this is not a fully compromised root that generates valid version increments, if that were the case, then any examined defence would be a trivially

weak defence, as they all rely on the sink as their trust anchor, and there would be no comparative information. We instead model sink impersonation: an attacker that spoofs the root's identity, or that broadcasts an increment that is supposed to come from the sink (the S-DIO/S-DAO-style flags and blacklist broadcasts that several defences introduce). The result is a written attacker model and a list of weak signals for each of the defenses. Tools: analysis of the protocol and the defense designs. Metrics: none yet; this phase generates the attack definitions for Phases 3 and 4.

Phase 3: Build the Test Scenarios in Cooja/Contiki

During this stage, we develop test scenarios within Cooja, using emulated Z1 motes. We configure an RPL network with a DODAG rooted at a border router and introduce the more challenging and realistic scenarios from the research gap: node mobility (nodes move during the test), larger networks (more nodes than the small grids used before), and an untrusted sink. In Phase 2, we insert the forging and colluding attackers, causing them to increment (or decrement) the version number, and to forge the defense signals as well. The tests are fair as we are using the same base settings for each scenario set.

In all scenarios sets, two baselines are always given: RPL with no attack (upper bound of performance) and RPL under VNA with no defense (lower bound of performance) to give a reference point for every measured metric. Tools: Cooja simulator, Contiki-NG, Zolertia Z1 motes (emulated) and mobility model for node movement. Metrics: No results yet, this phase results in the ready-to-run test scenarios (see Experimental Parameters table below).

Phase 4: Run the Experiments and Measure Robustness

In this phase we run the scenarios from Phase 3 and collect results. Runs are repeated for each defense with the forged-signal attacker, the colluding attacker, mobility, bigger networks, and the untrusted sink to ensure stability of the results. The detection quality is measured by accuracy (together with precision, recall, F1 and false positive rate), where a single accuracy value hides the bad performance on the imbalanced attack traffic; the detection latency (time taken by the defence to detect the attacker) is measured from 1 second for CDRPL to 120 seconds for QSec-RPL; packet delivery ratio (PDR); control overhead; energy consumption and delay.

The machine-learning detector (Osman et al., 2021) is not able to run inside the protocol on a mote, thus the result is evaluated off-line on traffic traces exported from the same simulation runs that are used to assess the in-protocol defenses, but not from a separate set of traffic traces. Tools: Cooja simulator, Contiki-NG, emulated Z1 motes and Cooja logging and energy tools.

Metrics: detection accuracy (precision, recall, F1, FPR), detection latency, PDR, control overhead, energy and delay.

Phase 5: Validate one or more defense on real hardware

The simulation phase is not enough to prove readiness for the real world, and in this phase, we test at least one defense on real IoT hardware. We select a defense which is small enough to fit into a real mote and a good representative example (e.g. in-protocol defense with mitigation, e.g. CDRPL or parent supervision). The hardware used are either the FIT IoT-LAB public testbed (with remote access to hundreds of real low-power nodes) or, alternatively, Zolertia RE-Mote / Firefly boards. The defense is ported to real devices, tested using VNA and verified that it acts the same way as it did in simulation. Tools: real IoT hardware motes (FIT IoT-LAB or Zolertia RE-Mote) plus the same attack setup as in the simulator. Metrics: detection accuracy, detection latency, PDR, control overhead, energy, and delay, to enable hardware results to be compared with Cooja results.

Phases 6: Fair Head-to-Head Comparison and Real-World-Readiness Verdict

In this final stage we compared all the defenses under the same conditions. The comparison is fair since Phases 3 and 4 have the same base network, attacker model, and the same metrics for each defense. We develop a final table that indicates, for each defense, how well it can resist forged signals, colluding attackers, an untrusted sink, mobility and larger networks, based on the results of Phase 4 and the results of the hardware tests conducted in Phase 5. From this table, we provide a clear real world readiness verdict for each defense, which defenses work and which do not, and why. Tools: the outcome of Phase 4 and 5, plus comparison tables and charts.

Experimental Parameters

All of the major simulation parameters are set in the table below, so that all defenses are tested under the same, repeatable conditions. Values are selected to help the defenses go beyond the small static grids found in the original papers.

Parameter	Setting
Simulator / OS	Cooja on Contiki-NG, with emulated Zolertia Z1 motes
Network sizes	20, 50, and 100 nodes
Topology	Grid and random placement in a 200 m × 200 m area
Attacker ratio	10%, 20%, and 30% of nodes
Attacker types	Single, colluding (multi-node), and signal-forging; version increase and decrease
Sink model	Trusted (baseline) and impersonated / untrusted

Parameter	Setting
Mobility	Static and Random Waypoint (0–2 m/s)
Radio model	UDGM (distance loss) for all runs; MRM for selected runs
Application traffic	One UDP packet per node every 60 s
Simulation time	3600 s per run
Repetitions	≥ 20 runs per configuration, reported as mean $\pm$ 95% confidence interval
Baselines	RPL with no attack; RPL under VNA with no defense
Reported metrics	Detection accuracy, precision, recall, F1, FPR, detection latency, PDR, control overhead, energy, delay

Expected Outcomes

At the conclusion of this study, we hope to achieve the following:

1. Clear categorization of the six core defenses, by action (detection only, detection and mitigation) and by location (inside RPL, outside RPL) and by test against (detection, detection and mitigation).
2. A list of all the controllable signals that can be forged and a clear attacker model, where each defense introduces its own unauthenticated signals (alert tags, self-reported counts, majority votes) and the attacker is defined that is able to forge these signals, collude with other defenders, or impersonate the sink.
3. Robustness under more challenging settings: detection accuracy (precision/recall/F1/FPR), detection latency, PDR, control overhead, energy, and delay for each defense in the presence of forged signals, colluding attackers, an untrusted sink, mobility, and larger networks. Some defenses will lose accuracy and/or be too cumbersome.
4. Hardware-validation results reporting if the defense simulation results still occur on actual motes, or if memory, energy and timing limitations alter the results.
5. A fair table of head-to-head comparisons of defenses with a real-world-readiness ranking under the same circumstances.
6. Practical guidance for future VNA defenses (e.g., new control signals protection, don't trust the sink, test mobile and at scale), to ensure that new defenses do not create the same integrity gap that the attack exploits.

Ethical Considerations

This research does not involve human participants, personal data or animal subjects; all experiments are run on network simulators and on non-personal IoT testbed hardware and informed-consent or privacy procedures are not required. The study does, however, adhere to the

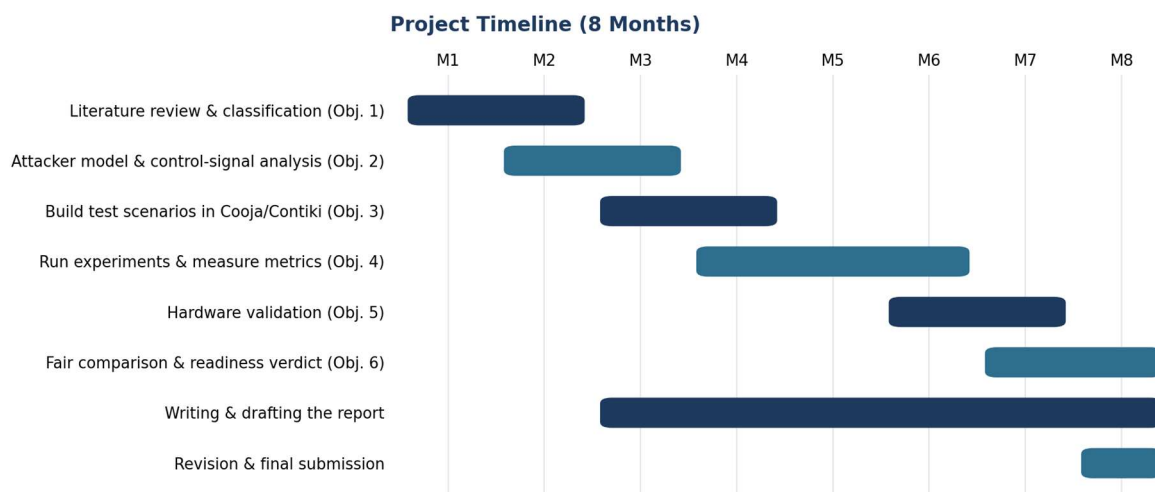
standard practice in the research integrity arena: all reviewed techniques are referenced to their original authors; reported values are referenced to their source papers; and any re-implementation of a published defense is explicitly labeled as such and not written as the original authors' code. The work creates and analyzes an attack module (a forging and colluding VNA), and is performed and documented for defensive evaluation purposes only, on test networks that are not connected to production or third-party networks and do not have the capability to impact production or third-party networks.

Limitations

There are a number of limitations recognized. First, the evaluation is mostly simulation-based, but while Cooja with emulated Z1 motes is a very faithful and popular environment, it is not able to capture all the physical-layer effects of a real deployment, and for this reason at least one defense is also validated on hardware. Second, the published descriptions of the defenses are re-implemented in the code because the original source code is typically not available (one paper, Nandhini et al., 2021, does not explicitly include important parameters); any differences between the original intent of the authors and our re-implementation are possible and are reported transparently. Third, hardware validation is constrained by the number of testbed nodes and thus hardware results reflect feasibility, not behavior. Fourth, the study assesses and compares current defenses but does not suggest an entire new routing protocol; a new routing protocol is suggested for future work based on the findings of the study.

Gantt Chart

The plan considers an eight-month timeframe (M1–M8). The following chart indicates the time periods in which each task is active, and then there is a month by month grid.



Task	M1	M2	M3	M4	M5	M6	M7	M8
Literature review and classification (Obj. 1)								
Attacker model and control-signal analysis (Obj. 2)								
Build test scenarios in Cooja/Contiki (Obj. 3)								
Run experiments and measure metrics (Obj. 4)								
Hardware validation (Obj. 5)								
Fair comparison and readiness verdict (Obj. 6)								
Writing and drafting the report								
Revision and final submission								

References

1. Almusaylim, Z. A., Alhumam, A., & Jhanjhi, N. Z. (2020). Detection and mitigation of RPL rank and version number attacks in the Internet of Things: SRPL-RP. *Sensors*, 20(21), 5997. <https://doi.org/10.3390/s20215997>
2. Alsukayti, I. S., & Singh, A. (2022). A lightweight scheme for mitigating RPL version number attacks in IoT networks. *IEEE Access*, 10, 111115–111133. <https://doi.org/10.1109/ACCESS.2022.3215460>
3. Alzubi, Q. M., Anbar, M., Alqattan, Z. N. M., Al-Betar, M. A., & Abdullah, R. (2020). Intrusion detection system based on a modified binary grey wolf optimisation. *Neural Computing and Applications*, 32(10), 6125–6137. <https://doi.org/10.1007/s00521-019-04103-1>
4. Dvir, A., Holczer, T., & Buttyán, L. (2011). VeRA – Version number and rank authentication in RPL. In 2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems (pp. 709–714). IEEE. <https://doi.org/10.1109/MASS.2011.76>
5. Mayzaud, A., Badonnel, R., & Chrisment, I. (2017). A distributed monitoring strategy for detecting version number attacks in RPL-based networks. *IEEE Transactions on Network and Service Management*, 14(2), 472–486. <https://doi.org/10.1109/TNSM.2017.2705290>
6. Nandhini, P. S., Srinath, P., Veeramanikandan, P., & Malliga, S. (2021). Version attack detection using Claim algorithm in RPL based IoT networks: Effects and performance parameters evaluation. In 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC) (pp. 209–215). IEEE. <https://doi.org/10.1109/ICOSEC51865.2021.9591735>
7. Osman, M., He, J., Mokbal, F. M. M., Zhu, N., & Qureshi, S. (2021). ML-LGBM: A machine learning model based on Light Gradient Boosting Machine for the detection of version number attacks in RPL-based networks. *IEEE Access*, 9, 83654–83665. <https://doi.org/10.1109/ACCESS.2021.3087175>
8. Perrey, H., Landsmann, M., Ugus, O., Schmidt, T. C., & Wählisch, M. (2013). TRAIL: Topology authentication in RPL. *arXiv preprint arXiv:1312.0984*.
9. Rouissat, M., Belkheir, M., & Mokaddem, A. (2023). Parent supervision lightweight solution against version number attacks for IoT networks. *Research Square* (preprint). <https://doi.org/10.21203/rs.3.rs-2542886/v1>
10. Sharma, G., Grover, J., & Verma, A. (2023). QSec-RPL: Detection of version number attacks in RPL based mobile IoT using Q-Learning. *Ad Hoc Networks*, 142, 103118. <https://doi.org/10.1016/j.adhoc.2023.103118>
11. Verma, A., & Ranga, V. (2019). ELNIDS: Ensemble learning based network intrusion detection system for RPL based Internet of Things. In 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU) (pp. 1–6). IEEE. <https://doi.org/10.1109/IoT-SIU.2019.8777504>

12. Yavuz, F. Y., Unal, D., & Gul, E. (2018). Deep learning for detection of routing attacks in the Internet of Things. *International Journal of Computational Intelligence Systems*, 12(1), 39–58. <https://doi.org/10.2991/ijcis.2018.25905181>