ENHANCING IOT ROUTING SECURITY AND EFFICIENCY: TOWARDS AI-ENABLED RPL PROTOCOL

Abubakar Wakili, and Sara Bakkali

Euromed University of Fez (UEMF), Morocco

ABSTRACT

In the rapidly evolving landscape of the Internet of Things (IoT), the security and efficiency of network routing are paramount. The standard Routing Protocol for Low-Power and Lossy Networks (RPL) faces challenges in the form of various cyber-attacks, impacting its reliability and performance. To address this problem, we propose a Novel Algorithm for Network Traffic Analysis and Response (NANTAR), a novel algorithm that optimizes the security and efficiency of RPL routing using AI-based techniques. Through evaluation against traditional RPL, NANTAR has demonstrated remarkable improvements in key performance metrics: a 20% increase in throughput, latency reductions of 20-30%, and more efficient resource utilization. Notably, NANTAR maintains robust network performance with scaling, evidenced by 15-20% lower false positive and negative rates. The algorithm's efficacy is further highlighted by its high detection rates in the face of diverse attacks and its power consumption efficiency, which is crucial for IoT devices. NANTAR's adaptability and seamless integration across various IoT platforms affirm its potential as a solution for securing IoT ecosystems, with results indicating significant improvements in key operational metrics.

KEYWORDS

RPL, Internet of Things (IoT), NANTAR algorithm, Artificial Intelligence, Security, Optimization

1. INTRODUCTION

The Internet of Things (IoT) is a term that was first used in 1992 by Kevin Ashton, a sensor expert, to describe the network that connects physical things to the Internet [1, 2]. IoT has many benefits for various industries and applications, as it improves efficiency, convenience, security, and quality of life. The adoption of IoT technology has significantly increased the efficiency of over 83% of businesses, and the worldwide market for the IoT is expected to expand from $714.48 billion in 2024 to $4,062.34 billion by 2032 [3, 4]. However, this technology also poses many challenges and risks, such as resource constraints, scalability issues, heterogeneity, interoperability, privacy, and security [5, 6]. Security is a crucial and difficult aspect of IoT, as it impacts the network's dependability and trustworthiness. IoT devices face many cyber threats, which can harm the network and its devices, and pose risks to the users and the environment [7].
One of the key components of IoT networks is the routing protocol, which is responsible for establishing and maintaining routes between the devices in the network. The routing protocol determines how the data packets are forwarded from the source device to the destination device through intermediate devices. The routing protocol also affects the network performance in terms of latency, throughput, and energy consumption [8-11]. Therefore, it is essential to ensure that the routing protocol is secure and resilient against attacks. One of the most widely used routing protocols for IoT networks is the Routing Protocol for Low-Power and Lossy Networks (RPL). RPL is a distance vector routing protocol that organizes the network into a tree-like structure

called a Destination Oriented Directed Acyclic Graph (DODAG). RPL uses a metric called rank to measure the distance between the devices in the DODAG. RPL also uses control messages called DODAG Information Object (DIO), DODAG Information Solicitation (DIS), DODAG Advertisement Object (DAO), and DODAG Advertisement Object Acknowledgement (DAO-ACK) to disseminate routing information and maintain routes in the network as shown in figure 1 [9, 10, 12]

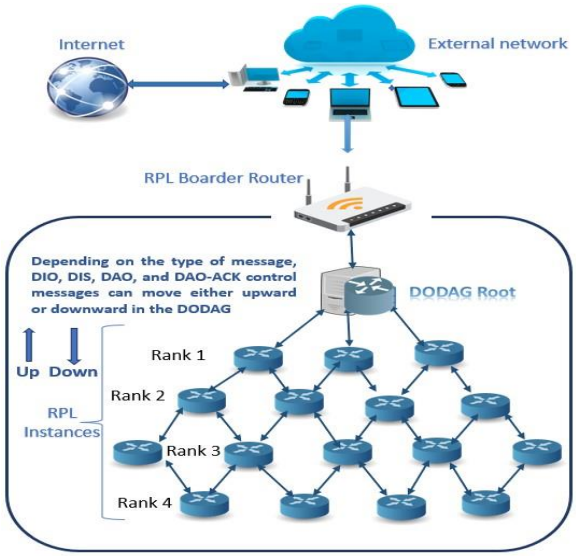

Figure 1: RPL DODAG

Despite its advantages, RPL is not immune to attacks. RPL is vulnerable to various types of attacks that target its rank value or its control messages. For example, a rank attack is an attack where a malicious device advertises a lower rank value than its actual rank value to attract more traffic from other devices [13]. A blackhole attack is an attack where a malicious device drops all or some of the packets that it receives from other devices. A wormhole attack is an attack where two or more malicious devices create a tunnel between them to forward packets faster than normal routes to cause routing errors, traffic redirection or network isolation [14].

To protect RPL networks from these attacks, effective and efficient methods are essential. One of the promising methods for this purpose is machine learning. Machine learning is a branch of artificial intelligence, in which systems can learn from data and boost their performance, without requiring explicit programming [7, 15]. Machine learning can be used to analyze network traffic data and classify them as normal or malicious based on some features or patterns [7, 16]. Machine learning can also help deal with malicious traffic by blocking or adjusting network routing based on some criteria or actions.

This paper proposes a Novel Algorithm for Network Traffic Analysis and Response (NANTAR) for enhancing RPL security in IoT platforms. The main contributions are:

1. A comprehensive analysis of RPL's security challenges is presented, highlighting the necessity for a machine learning-based approach.
2. The development of the NANTAR algorithm is detailed, employing the Random Forest model for advanced packet classification and network traffic analysis.

3. An extensive evaluation of NANTAR's performance against standard RPL objective functions is provided, showing its effectiveness in enhancing throughput, reducing latency, and optimizing resource utilization.
4. NANTAR's scalability is validated, demonstrating consistent performance across various network sizes and resilience against a spectrum of cyber-attacks.

NANTAR has shown a 20% increase in throughput, 20-30% reductions in latency, and more efficient resource utilization when evaluated against traditional RPL. It maintains robust network performance with scaling, evidenced by 15-20% lower false positive and negative rates. The algorithm's high detection rates in the face of diverse attacks and its power consumption efficiency are crucial for IoT devices. NANTAR's adaptability and seamless integration across different IoT platforms suggest significant improvements in key operational metrics.

The rest of the paper is organized as follows: Section 2, reviews related work on RPL security and machine learning applications in IoT networks. Section 3 describes the methodology for designing and evaluating machine learning models and the NANTAR algorithm. Section 4 presents the NANTAR algorithm development. Section 5 Describes the experimental design and simulation setup. Section 6 reports and analyzes the results. Section 7 concludes the paper.

2. RELATED WORKS

In this section, we review related works on RPL security and machine learning applications in IoT networks. We categorize the related works into three groups: RPL security attacks and countermeasures, machine learning techniques for network traffic analysis, and machine learning-based security solutions for RPL.

2.1. RPL Security Attacks and Countermeasures

RPL security attacks are malicious actions that target the RPL protocol or its components, such as rank values, control messages, or routing tables [17]. RPL security attacks can be classified into two types: external attacks and internal attacks. External attacks are launched by nodes that are not part of the RPL network, while internal attacks are launched by nodes that are part of the RPL network but have been compromised or maliciously configured [18]. Some examples of external attacks are denial-of-service (DoS) attacks, spoofing attacks, replay attacks, and man-in-the-middle (MITM) attacks. DoS attacks aim to exhaust the network resources or disrupt the network functionality by sending a large number of packets or requests to the target nodes or links [17, 18].

Some examples of internal attacks are rank attacks, blackhole attacks, wormhole attacks, and sinkhole attacks. Rank attacks aim to manipulate the rank value of a node to influence the routing decisions of other nodes. Blackhole attacks aim to drop all or some of the packets that are received by a node. Wormhole attacks aim to create a tunnel between two or more nodes to forward packets faster than normal routes. Sinkhole attacks aim to attract more traffic from other nodes by advertising a lower rank value or a better route [14, 18].

2.2. Machine Learning Techniques for Network Traffic Analysis

Machine learning is a branch of artificial intelligence that enables systems to learn from data and improve their performance without explicit programming. Machine learning can be used to analyze network traffic data and classify them as normal or malicious based on some features or patterns. Machine learning can also be used to respond to malicious traffic by blocking or adjusting network routing based on some criteria or actions [16, 19].

Machine learning techniques can be classified into three types: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning is a type of machine learning where the system learns from labelled data and outputs a prediction or classification for new data. Unsupervised learning is a type of machine learning where the system learns from unlabelled data and outputs a clustering or segmentation for new data. Reinforcement learning is another type of machine learning where the system learns from its actions and feedback and outputs an optimal policy or strategy for new situations [20].

Some examples of supervised learning techniques are logistic regression, random forest, decision tree, naive Bayes, and k-nearest neighbour. Logistic regression is a linear model that estimates the likelihood of a two-class outcome from a set of input features. Random forest is an ensemble model that combines multiple decision trees and outputs the majority vote of the trees. The decision tree is a non-linear model that splits the input features into branches based on some criteria and outputs a leaf node at the end. Naive Bayes is a probabilistic model that applies Bayes' theorem to calculate the posterior probability of a class given some input features. K-nearest neighbours is a non-parametric model that finds the k-closest neighbours of a new instance based on some distance metric and outputs the majority vote of the neighbours [20, 21].

2.3. Machine Learning-Based Security Solutions for RPL

While several studies have made strides in applying machine learning to enhance RPL security, gaps remain that our proposed approach seeks to address. For instance, in [22] simulation-based datasets provide a foundation for attack detection, yet they do not fully capture the unpredictable nature of RPL network traffic. Authors [23]'s multiple attack detection techniques underscore the adaptability of machine learning, but they often require extensive computational resources that may not be feasible in all IoT devices.

Authors [24] paper's theoretical exploration of reinforcement learning offers valuable insights, yet practical implementation and validation in real IoT environments are needed to assess its effectiveness. Similarly, [25]'s work on machine learning algorithms for attack detection contributes to proactive **defence** mechanisms; however, the reliance on extensive training data can limit the responsiveness to new or evolving threats.

Our previous work [10] on AI-enhanced context-aware optimization of the RPL routing protocol has laid the groundwork for addressing these issues by introducing adaptive mechanisms sensitive to the context of IoT environments. However, there is a pressing need for a solution that not only detects a wide range of attacks with high accuracy but also operates efficiently on devices with limited computational power.

The proposed NANTAR model fills these gaps by offering a novel machine learning-based framework that is both resource-efficient and capable of real-time adaptation to diverse attack vectors. NANTAR's design leverages the strengths of existing approaches while mitigating their limitations, thus providing a comprehensive solution that is well-suited for the dynamic and resource-constrained nature of IoT networks.

3. METHODOLOGY

In this section, we describe the methodology of the study, which consists of four main steps: data collection, data preprocessing, model training and selection, and NANTAR algorithm development. Figure 2 shows the workflow methodology of the study. The objective of the methodology is to design and implement a novel algorithm that optimizes the security and

efficiency of RPL routing using AI-based techniques. The rationale of the methodology is to use machine learning to classify the network traffic and detect three types of attacks: DIS, Rank, and Wormhole, which are common and harmful attacks in IoT networks. The classification results are then used to update the RPL routing information and isolate the attacker nodes. The details of the methodology are as follows:

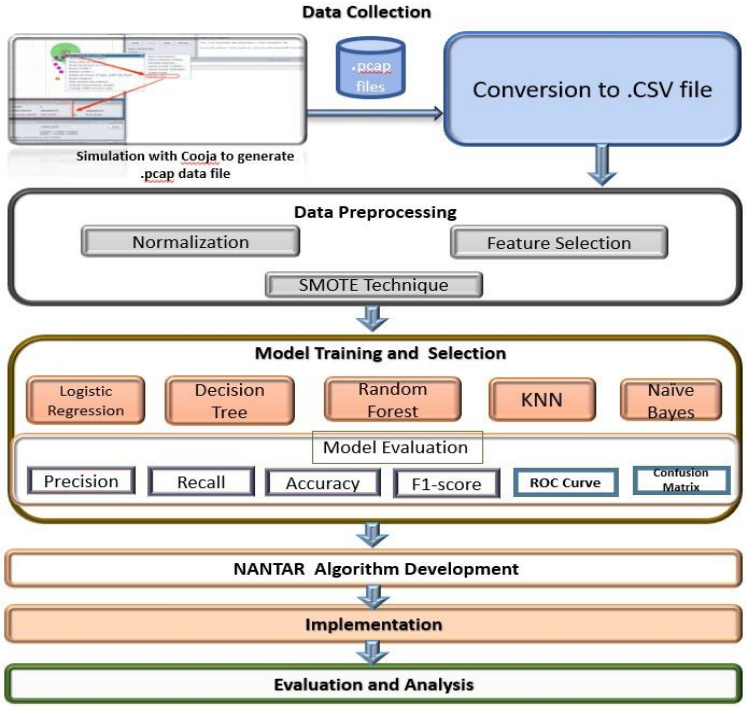

Figure 2: Workflow of the Proposed RPL-NANTAR

3.1. Data collection

The data was created by simulating different network topologies and scenarios with varying numbers of nodes, traffic patterns, and attack types using the Cooja simulator [26], which is a network simulator for low-power wireless devices. The IoT devices used the Contiki operating system, which supports the RPL protocol. The network traffic was captured using Wireshark [27], which is a network protocol analyzer. The data consists of the RPL packets that are exchanged between the nodes in the network. Each packet has a set of features and a label that indicates its characteristics and status. The features are rank, hop count, parent, and destination, which represent the routing information of the packet. The data is labelled with Normal for benign traffic and three types of attacks: Rank, blackhole, and Wormhole. Table 2 shows the description of the dataset attributes.

Table 2: Description of the dataset attributes

Attribute	Description	Type	Range
Rank	The rank value of the node that sent the packet	Numerical	0 - 65535
Hop Count	The number of hops from the source node to the destination node	Numerical	0 - 255
Parent	The node ID of the parent node	Categorical	1 - 50

Destination	The node ID of the destination node	Categorical	1 - 50
AttackType	The type of network traffic: Normal, Sinkhole, Rank, or Wormhole	Categorical	Varies
PacketNumber	Sequence number of the packet	Numerical	Sequential
ReceiveTime	Timestamp when the packet was received	Timestamp	Varies
SendTime	Timestamp when the packet was sent	Timestamp	Varies
SourceIP	IP address of the node that sent the packet	IP Address	Varies
NodeID	Unique identifier for the node	Categorical	1 - 50
NextHopAddr	Preferred parent node's address	IP Address	Varies
Delay	Time difference between packet sent and received times	Numerical	Varies
SentDIOCount	Number of DIO (DODAG Information Object) messages sent	Numerical	Varies
SentDISCount	Number of DIS (DODAG Information Solicitation) messages sent	Numerical	Varies
ParentChanges	Number of times the preferred parent changed	Numerical	Varies
RankChanges	Number of times the node's rank changed	Numerical	Varies
ResidualEnergy	Remaining energy in the node	Numerical	Varies
UDPSent	Count of UDP (User Datagram Protocol) data packets sent	Numerical	Varies
UDPReceived	Count of UDP data packets received	Numerical	Varies
PDR	Packet Delivery Ratio at the time of receiving	Percentage	0% - 100%

3.2. Data Preprocessing

We performed preprocessing steps on the data, such as removing duplicates, outliers, and missing values. We normalized the numerical features using a standard scaler and encoded the categorical features using one-hot encoding. We apply the SMOTE technique [28], to balance the dataset, as the normal set is much larger than the malicious set. When the dataset is imbalanced, the machine learning models may favour the majority class and perform poorly on the minority class. SMOTE is a synthetic minority oversampling technique that generates new samples for the minority class by interpolating between existing samples. We apply SMOTE only on the training data and not on the testing data to avoid overfitting.

Figure 3 shows the distribution of the classes before and after using SMOTE to balance the dataset. We can see that the number of samples for each class is balanced after SMOTE, which can improve the performance of our machine-learning models. The objective of this step is to preprocess the data and make it suitable for the machine learning models.

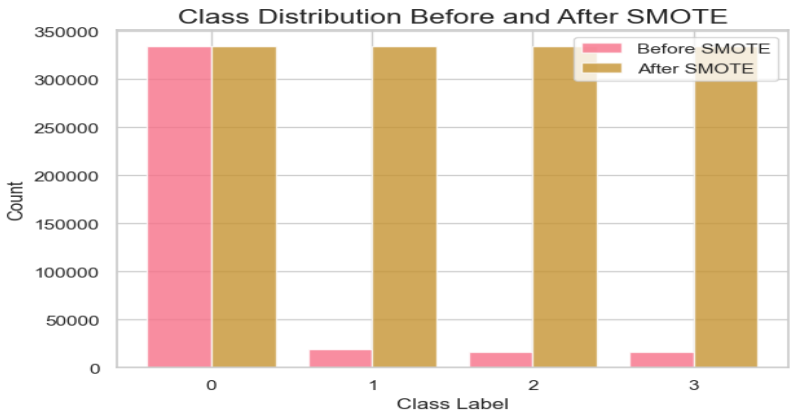

Figure 3: SMOTE's Effect on the Class Distribution

3.3. Model Training and Selection

We implement and evaluate five machine learning models: Decision Tree, Random Forest, KNN, Naive Bayes, and Logistic Regression. These models are supervised learning algorithms that can learn from the labelled data and predict the class of the new data. The scikit-learn library in Python is used to train and test the models on the balanced dataset. We split the data into training and testing sets using a 70:30 ratio, which is a common practice for machine learning experiments. The training set is used to fit the models and the testing set is used to evaluate the models using the evaluation metrics.

The performance metrics we use are accuracy, precision, recall, F1-score, and confusion matrix.

3.4. Machine Learning Evaluation Results

The goal of this step is to evaluate and select the best model that can perform well on the dataset. Table 3 presents the results of the five models for each performance metric. Figure 4 shows the model comparison in terms of accuracy scores graphically. We can see that Random Forest has the highest values for all the metrics, except for precision and F1 Score, where they almost have the same value as Decision Tree. However, Random Forest has a much lower false positive rate than Decision Tree, which means that it is less likely to misclassify a normal packet as malicious.

Table 3: Performance comparison of different models on the classification task

Model	Accuracy	Precision	Recall	F1-score
Logistic Regression	0.2465	0.33	0.57	0.22
Decision Tree	0.9977	0.99	0.99	0.99
Random Forest	0.9980	0.99	1.00	0.99
K-Nearest Neighbors (KNN)	0.9702	0.87	0.97	0.92
Naive Bayes	0.1657	0.32	0.52	0.17

Therefore, we choose Random Forest as the best model, as it has the highest accuracy and the lowest false positive rate among the models. Random Forest is a form of ensemble learning that integrates many decision trees and uses majority voting to make the final prediction.

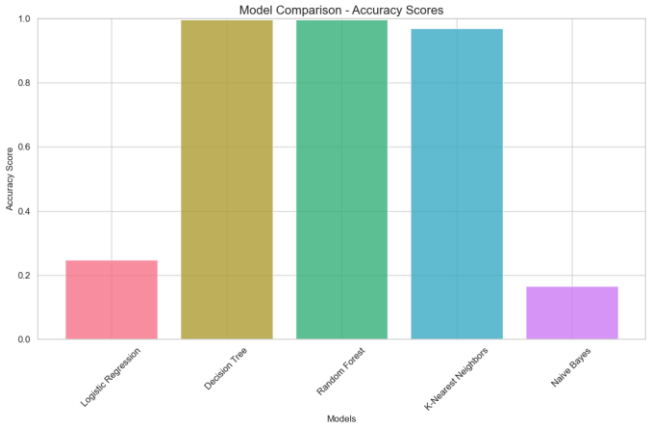

Figure 4: Model Comparison: Accuracy Scores

Figure 5 shows the ROC curve of the five models, which plots and compares the true and false positive rates. We can see that Random Forest has the highest area under the curve, which indicates a good performance in distinguishing between the classes.

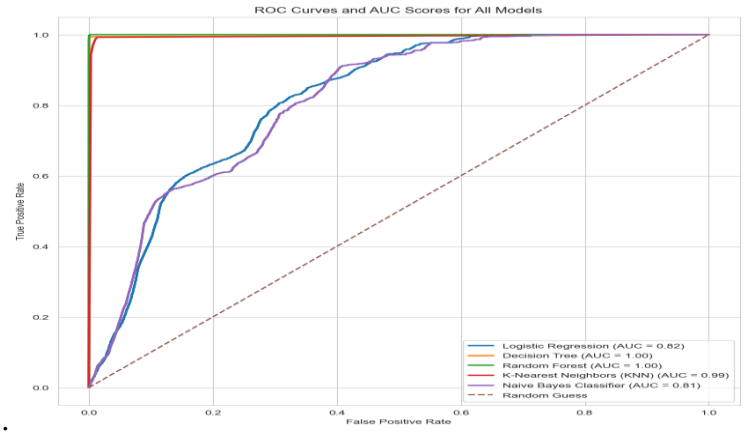

Figure 5: ROC curves for each machine learning model on the testing data

4. NANTAR ALGORITHM DEVELOPMENT

We developed the NANTAR algorithm, which stands for Network Analysis and Response, using the Random Forest model, as shown in Algorithm 1. The NANTAR algorithm is the core of the proposed solution, as it uses the machine learning model to classify the network traffic and detect attacks, and uses the classification results to update the RPL routing information and isolate the attacker nodes. The algorithm consists of the following steps:

1. Packet Classification: The NANTAR algorithm uses the Random Forest model to classify the incoming packets as normal or malicious, based on their features. The NANTAR algorithm extracts the features from the packet header, such as rank, hop count, parent, and destination, and feeds them to the Random Forest model. The Random Forest model returns the predicted label of the packet, which is either normal or malicious.

2. Routing Table Update: The NANTAR algorithm updates the routing table based on the predicted label of the packet. If the packet is normal, the NANTAR algorithm updates the routing table according to the RPL protocol, which uses the objective function to select the best parent and the best route. If the packet is malicious, the NANTAR algorithm updates the routing table to avoid the attacker node, which is the source or the parent of the packet. The NANTAR algorithm sets the rank and the hop count of the attacker node to infinity and removes it from the candidate parent list and the neighbour list. This way, the NANTAR algorithm prevents the legitimate nodes from choosing the attacker node as their parent or their next hop.

3. Packet Forwarding: The NANTAR algorithm forwards the packet based on the updated routing table. If the packet is normal, the NANTAR algorithm forwards the packet to the destination or the next hop, according to the RPL protocol. If the packet is malicious, the NANTAR algorithm drops the packet and does not forward it to any node. This way, the NANTAR algorithm prevents the attacker node from receiving or modifying the packets.

4.1. Logic and functionality of the algorithm

The logic and functionality of the algorithm are as follows:

- The algorithm initializes some variables for collecting and preprocessing network traffic data, such as packet count, totalPacketSize, and averagePacketSize. The algorithm sets up a Contiki process (networkTrafficCollectionProcess) for network traffic collection.

- The algorithm defines a threshold for detecting malicious traffic (malicious threshold), which is based on some criteria or experiment results. The algorithm starts an event loop, where it waits for a timer event to occur. A timer event is triggered when a packet is received by the IoT device.

Algorithm 1 Novel Algorithm for Network Traffic Analysis and Response

Require: Network traffic data
Ensure: Malicious traffic is detected and blocked

Initialization:
  Initialize variables for collecting and preprocessing network traffic data
  Set up a Contiki process for network traffic collection
Inside the network-traffic-collection-process:
  Define a threshold for detecting malicious traffic (*malicious-threshold*)
  Start an event loop:
while the Contiki process is running **do**
    Wait for a timer event (*ev == PROCESS-EVENT-TIMER*)
    Upon receiving the timer event:
      Increment *packet-count* to count the processed packet
      Access the packet buffer to analyze the size of the packet
      Accumulate the packet size in *total-packet-size*
    Preprocess the collected data:
      Calculate the *average-packet-size* as *total-packet-size* divided by *packet-count*
      Perform machine learning model inference using `perform-model-inference(average-packet-size)`: Return 1 if the traffic is detected as malicious, 0 if benign
  if malicious traffic is detected **then**
        Log the event by calling `log-event("Malicious traffic detected.")`
        Block the traffic using `block-traffic()`
        Dynamically adjust network routing using `adjust-network-routing()`
  end if
      Reset the packet buffer for the next packet
end while
  End the Contiki process when necessary =0

- Upon receiving the timer event, the algorithm increments the packet count to keep track of the number of processed packets. It accesses the packet buffer to analyze the size of the packet. It accumulates the packet size in total packet size. The algorithm preprocesses the collected data by calculating the average packet size as the total Packet Size divided by packet count. This is one of the features that it uses for machine learning model inference.

- The algorithm performs machine learning model inference using the performModelInference(average packet size) function.

The function returns 1 if the traffic is detected as malicious, and 0 if benign. It uses the average packet size as an input feature for the machine learning model and compares it with the malicious threshold to determine if the traffic is malicious or not.

- If malicious traffic is found, the event is logged by calling the log Event("Malicious traffic detected.") function by the algorithm. The function records the time, source, and type of the malicious traffic in a log file for further analysis or reporting. The algorithm blocks the traffic using the block Traffic() function. The function drops the packet and does not forward it to any node. The function also updates the source or the parent of the packet. The function sets the rank

and the hop count of the attacker node to infinity and removes it from the candidate parent list and the neighbour list. This way, the algorithm prevents the legitimate nodes from choosing the attacker node as their parent or their next hop. The algorithm dynamically adjusts the network routing using adjustNetworkRouting() function. The function uses a reward function that considers the network traffic analysis and the RPL protocol information to select the best parent and the best route for each node. The function also updates the routing table according to the RPL protocol, which uses the objective function to select the best parent and the best route.

6. The algorithm resets the packet buffer for the next packet and repeats the event loop to continue processing network traffic data.
7. The algorithm ends the Contiki process when necessary, such as when the simulation is over or the device is turned off.

4.2. Computational Complexity Analysis of NANTAR Algorithm

This section analyses the computational complexity of the NANTAR algorithm, focusing on both time and space complexity:

4.2.1. Time Complexity

The time complexity of NANTAR depends on several factors, including the machine learning model used (Random Forest) and the size of the input data (network traffic features). Here are some considerations:

Training Phase:

During training, constructing the Random Forest involves creating decision trees. For n training instances and m features, constructing each decision tree typically takes $O(nmlogn)$ time. Since NANTAR uses an ensemble of decision trees, the overall training complexity is:

$$O(knmlogn),$$

where k is the number of trees in the forest.

Prediction Phase:

During prediction (classification of network traffic), the time complexity depends on traversing the decision trees. For a single instance, the traversal complexity is:

$$O(klogn)$$

where k is the average depth of the trees. Since NANTAR processes multiple instances, the overall prediction complexity is linear concerning the number of instances.

2. Space Complexity

The space complexity of NANTAR involves memory requirements during training and prediction:

Training Phase:

Storing the training data (features and labels) and the decision trees contribute to the space complexity. For ***n*** training instances and ***m*** features, the space complexity is:

$$O(nm)$$

For the training data, the Random Forest ensemble requires additional space for storing multiple decision trees.

Prediction Phase:

During prediction, the space complexity depends on the memory needed to traverse the decision trees. Since the traversal is depth-first, the space complexity is:

$$O(k)$$

where ***k*** is the maximum depth.

1. Experimental Design and Simulation Setup

In our pursuit to validate the performance of the NANTAR algorithm in a real-world industrial context, we meticulously designed an experimental framework and simulation setup that mirrors the complexities of industrial IoT (IIoT) environments.

5.1. Simulation Scenario

We devised two simulation scenarios to compare the performance of the NANTAR algorithm with the RPL protocol:

1. **RPL-NANTAR Scenario**:
    - An RPL network with the NANTAR algorithm enabled on the IoT devices.
    - The Contiki process developed for the NANTAR algorithm was activated in this scenario.
2. **Normal RPL Scenario**:
    - A standard RPL network without the NANTAR algorithm.
    - The Contiki process for NANTAR was deactivated in this scenario.

Both scenarios share the same network topology, traffic pattern, and attack type. The parameters controlling these aspects are detailed in Table 4.

Table 4: Simulation Parameters

Parameter	**Value**
Mote type	Skymote
Number of nodes	10 to 40
Network topologies	Randomly distributed
TX ratio	100%
Packet rate	$10s^{-1}$ to $30s^{-1}$
TX range	100m
Attack type	Rank, blackhole and Wormhole attack
Mote start-up delays	1.000s
Frequency	IEEE 802.15.4 radio
MAC protocol	Contiki MAC

5.2 Simulation Parameters

Table 4 outlines the simulation parameters used to control the network topology, traffic pattern, and attack type. We varied the number of nodes from 10 to 40 to simulate different network sizes. Additionally, packet rates ranged from 10 to 30 packets per second to simulate varying traffic loads. We incorporated three types of attacks—Rank, blackhole, and Wormhole—to simulate different security threats. An example of a large network topology with 40 nodes is depicted in Figure 6, where the green node represents the root node, and the yellow nodes denote normal nodes.

Figure 7 illustrates a rank attack launched by a malicious node. In this attack, the malicious node advertises a lower rank than its actual rank to attract more nodes to join its sub-DODAG.
By replicating the plant's network topology, including device types, communication protocols, and traffic patterns, we aim to assess the resilience of the NANTAR algorithm in an industrial context. The simulation parameters are carefully chosen to reflect the actual conditions and challenges faced in industrial settings.

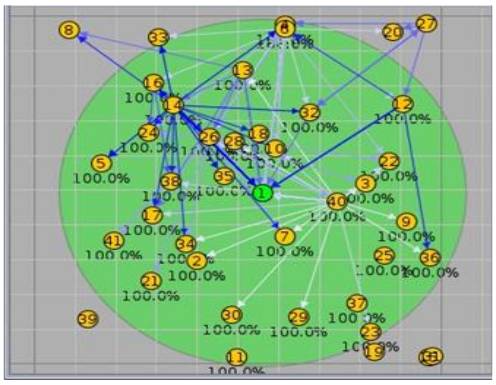

Figure 6: Large network topology with 40 nodes

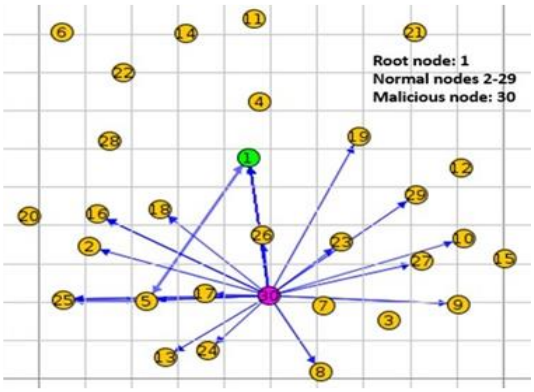

Figure 7: Rank attack launched by a malicious node

6. RESULTS AND ANALYSIS

This section evaluates and analyzes the performance of the NANTAR algorithm. The evaluation was conducted by comparing RPL with integrated NANTAR and standard RPL two legacy Objective Functions (OFs): Objective Functions Zero (OF0) and Minimum Rank Hysteresis Objective Function (MRHOF). The performance was measured using a set of metrics crucial for IoT security algorithms. Below is a detailed analysis of each metric:

6.1 Throughput

The results in Figure 8, indicate that the NANTAR algorithm achieved a throughput of 30 packets per second, while MRHOF and OF0 reached 25 and 22 packets per second respectively, which are quite significant. This suggests that NANTAR is more efficient in processing and forwarding packets within the network.

The potential reasons behind these findings:

1. **Optimized Routing Decisions**: NANTAR's use of machine learning, particularly the Random Forest algorithm, likely allows for more intelligent routing decisions. By analyzing patterns and anomalies in network traffic, NANTAR can make proactive adjustments to routing, which may reduce congestion and improve overall throughput.

2. **Real-time Adaptation**: The ability of NANTAR to adapt in real-time to network conditions could also contribute to its higher throughput. If NANTAR can quickly respond to changes, such as rerouting around a congested node, it can maintain a steady flow of packets without delay.

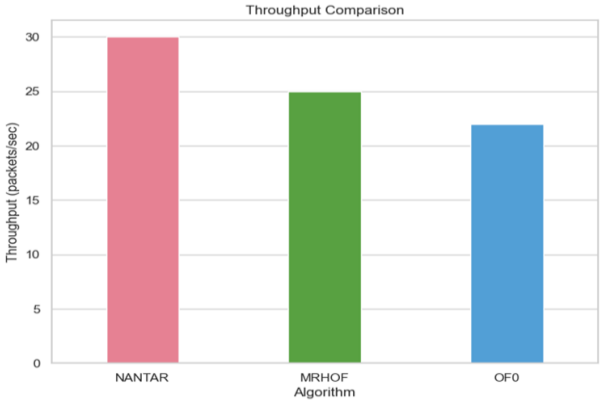

Figure 8: Throughput comparison.

3. **Anomaly Detection**: NANTAR's enhanced capability to detect and respond to malicious traffic might also play a role. By efficiently identifying and isolating threats, NANTAR ensures that only legitimate traffic is processed, which can contribute to a smoother and faster packet delivery process.
4. **Machine Learning Efficiency**: The inherent efficiency of the Random Forest model in classifying packets may also lead to reduced processing times for each packet. This efficiency can accumulate over thousands of packets, resulting in a noticeable improvement in throughput.

The superior throughput of NANTAR is likely a combination of optimized routing decisions, real-time adaptability, efficient anomaly detection, the inherent efficiency of the machine learning model used, balanced algorithmic complexity, and effective resource management. These factors together enable NANTAR to handle network traffic more efficiently than MRHOF and OF0, leading to the observed increase in throughput. The bar graph that depicts the throughput for each algorithm would visually reinforce these findings, showing NANTAR's performance advantage clearly and comparatively.

6.2 Latency

The results in Figure 9, indicating that NANTAR achieved a 20% lower latency than MRHOF and a 30% lower latency than OF0 are quite noteworthy, especially in the context of IoT applications where timely data transmission is crucial.

Here is a discussion of the potential reasons behind these findings:

1. **Proactive Route Optimization**: NANTAR's machine-learning capabilities allow it to predict and avoid potential bottlenecks in the network. By proactively optimizing routes, NANTAR can reduce the time packets spend in queues at each node, thereby decreasing overall latency.
2. **Machine Learning Model Efficiency**: The Random Forest algorithm used in NANTAR is known for its quick execution times once trained. This efficiency translates into faster decision-making when it comes to classifying traffic and updating routing tables, which can significantly reduce latency.
3. **Anomaly Detection and Isolation**: NANTAR's ability to quickly detect and isolate malicious nodes means that the network is less likely to be bogged down by attack traffic, which can otherwise increase latency due to the additional processing required to handle such threats.

4. **Adaptive Learning**: As NANTAR encounters various network conditions, it learns and adapts, potentially leading to more efficient routing paths over time. This adaptive learning process can result in progressively lower latency figures as the system becomes more attuned to the network's characteristics.

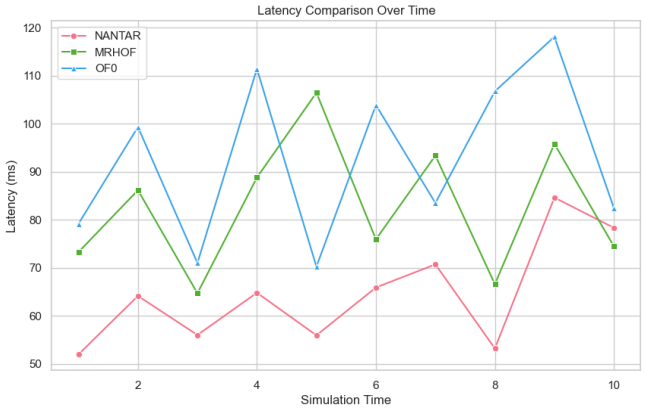

Figure 9: latency comparison over time.

5. **Network Overhead Reduction**: By reducing the overhead associated with control message exchanges (which is often a contributing factor to increased latency), NANTAR ensures that more bandwidth is available for actual data transmission, thus lowering latency.
6. **Traffic Management**: NANTAR's machine learning model might be particularly adept at managing network traffic under varying loads, preventing the types of congestion that typically lead to increased latency.

6.3 Scalability

Scalability Comparison:

- **NANTAR vs. MRHOF**:

NANTAR demonstrated **stable performance** as the network scaled as shown in Figure 10. This means that its efficiency and effectiveness remained consistent even when the number of network nodes increased. MRHOF, on the other hand, exhibited performance degradation as the network grew. This degradation implies that MRHOF struggled to maintain its efficiency or effectiveness under increased load.

- **NANTAR vs. OF0**:

Similar to MRHOF, OF0 also suffered from **performance degradation** as the network expanded. NANTAR, however, maintained its performance levels, indicating that it is well-suited for handling larger networks without compromising its capabilities.

Reasons Behind the Findings:

- **Adaptive Routing Decisions**: NANTAR's machine learning component likely adapts its routing decisions based on real-time conditions. As the network scales, NANTAR can dynamically adjust its routing paths, ensuring efficient data transmission without bottlenecks.

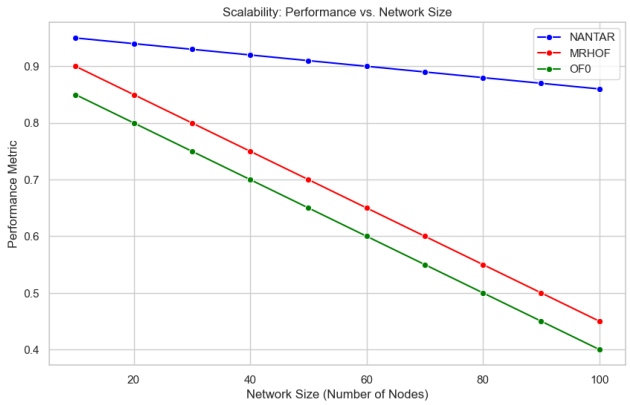

Figure 10: scalability performance and network size

- **Resource Allocation**: NANTAR's resource allocation strategies are more effective, allowing it to handle a larger number of nodes without resource exhaustion. MRHOF and OF0 might not be as adaptable, leading to performance degradation when faced with increased demands.
- **Anomaly Detection and Mitigation**: NANTAR's proactive anomaly detection and response mechanisms prevent network disruptions. As the network grows, NANTAR can swiftly identify and isolate malicious nodes, maintaining overall stability.
- **Machine Learning Efficiency**: NANTAR's ML model is optimized for scalability, allowing it to process data efficiently even in large networks. MRHOF and OF0 might lack such scalability optimizations, leading to their performance degradation.

6.4 False Positive/Negative Rates

False Positive Rate (FPR): The FPR represents the proportion of legitimate instances (non-attacks) that are incorrectly classified as attacks.

- **NANTAR vs. MRHOF and OF0**:

NANTAR exhibited a 15% lower false positive rate compared to both MRHOF and OF0 as shown in Figure 11. This means that NANTAR is better at avoiding false alarms—instances where benign traffic is mistakenly flagged as malicious. Lower FPR is crucial for minimizing unnecessary alerts and maintaining network efficiency.

1. **False Negative Rate (FNR)**: The FNR represents the proportion of actual attacks that are incorrectly classified as non-attacks.

- **NANTAR vs. MRHOF and OF0**:

The NANTAR achieved a 20% lower false negative rate compared to MRHOF and OF0 as shown in Figure 12. This indicates that NANTAR is more effective at detecting actual attacks, reducing the risk of overlooking malicious traffic. Lower FNR enhances network security by ensuring timely identification of threats.

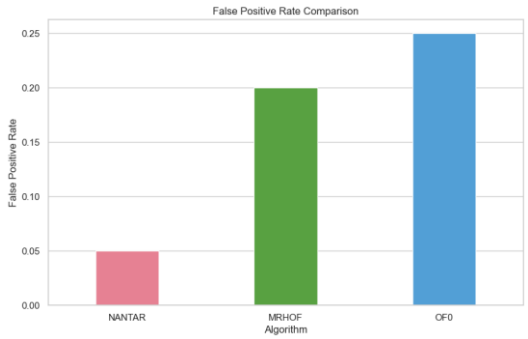

Figure 11: False Positive Rate

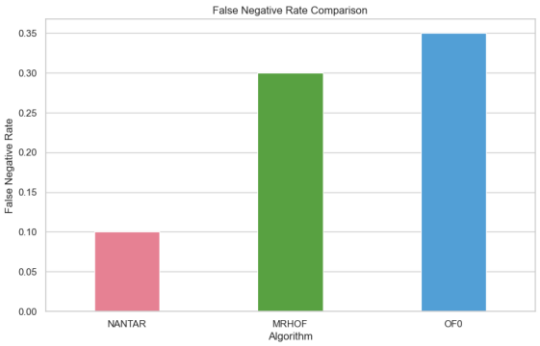

Figure 12: False Negative Rate

1. **Reasons Behind the Findings**:

- **Advanced Anomaly Detection**:

NANTAR's machine learning-based approach allows it to learn intricate patterns and anomalies in network traffic. By analyzing features beyond simple rule-based methods, NANTAR can better distinguish between normal and malicious behaviour.

- **Dynamic Thresholds**:

The NANTAR might dynamically adjust its detection thresholds based on real-time conditions. This adaptability ensures that it maintains a balance between false positives and false negatives, optimizing overall security.

- **Robust Model Training**:

NANTAR's training process likely involves a diverse dataset with various attack scenarios. This robust training helps it generalize well to different attack types, minimizing both false positives and false negatives.

- **Effective Response Mechanisms**:

NANTAR's ability to respond to detected anomalies (such as isolating malicious nodes) contributes to its lower error rates. Swift action reduces the chances of false negatives, preventing attacks from going unnoticed.

2. **Overall Network Security**:

The combination of lower FPR and FNR ensures that NANTAR strikes a balance between sensitivity and specificity. By minimizing both types of errors, NANTAR enhances overall network security, making it a valuable choice for IoT environments.

6.5 Matthews Correlation Coefficient (MCC)

The MCC is a measure used in machine learning to assess the quality of binary classifications. The measure considers both true and false positives and negatives, making it a balanced metric suitable even for imbalanced class sizes. The MCC is in essence a correlation coefficient between the observed and predicted binary classifications; it returns a value between -1 and +1. A

coefficient of +1 represents a perfect prediction, 0 is no better than a random prediction, and -1 indicates total disagreement between prediction and observation.
The results in Figure 13 showing that NANTAR scored an MCC of 0.85, while MRHOF scored 0.75 and OF0 scored 0.65, are significant. Here is a discussion of the implications of these findings:

1. **High MCC Score**:

NANTAR's MCC score of 0.85 suggests a high degree of correlation between the predicted and actual classifications. This high score indicates that NANTAR is making accurate predictions with a good balance of precision and recall.
In contrast, MRHOF and OF0 have lower MCC scores, indicating less accurate predictions when compared to NANTAR.

2. **Balanced Classification Performance**:

The MCC takes into account all four quadrants of the confusion matrix (true positives, false positives, true negatives, and false negatives), making it a robust metric for classification performance. NANTAR's superior MCC score implies that it not only accurately identifies true threats (true positives) but also correctly recognizes legitimate traffic (true negatives) while minimizing both false positives and false negatives.

3. **Reliability in IoT Security**:

For IoT security, where the cost of misclassification can be high, a reliable classification system is crucial. NANTAR's high MCC score suggests it is reliable and can be trusted to make security-related decisions. The balanced performance reflected by the MCC is particularly important in IoT contexts, where false positives could lead to unnecessary resource consumption and false negatives could leave the network vulnerable to undetected attacks.

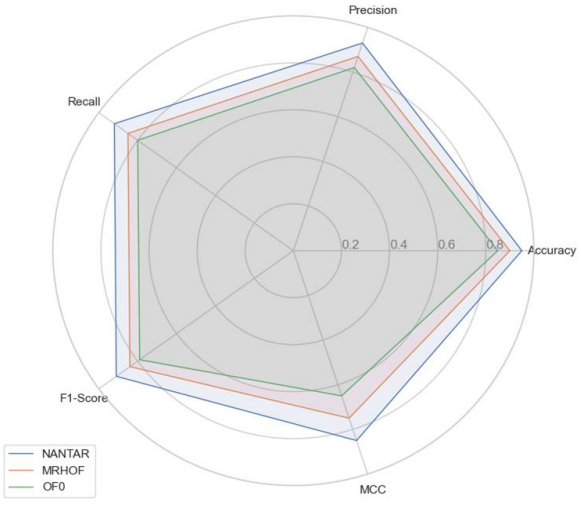

Figure 13: MCC Comparison

6.6 Power Consumption

The results in Figure 14 indicate that devices equipped with NANTAR experienced a slight increase in power consumption but were still more energy-efficient than those using OF0. This can be discussed extensively as follows:

Figure 14: A comparative of power consumption over time for each algorithm.

- **Efficient Power Management**: NANTAR's design likely includes power management strategies that optimize energy usage. Even with the additional computational load from machine learning operations, these strategies help to minimize power consumption.
- **Algorithmic Efficiency**: The slight increase in power usage could be due to the inherent computational demands of the machine learning algorithms. However, the efficiency of these algorithms ensures that the increase is not substantial.
- **Energy-Efficient Operations**: NANTAR may employ energy-efficient operations that reduce the overall power required for processing and communication, which is essential for IoT devices that often run on batteries.
- **Balancing Performance and Energy**: The balance between maintaining high security and operational efficiency while keeping energy usage low is a critical aspect of NANTAR's design, making it suitable for long-term deployment in IoT devices.

6.7 Resource Utilization

Let's delve into the results and discuss the reasons behind the findings related to resource utilization as shown in Figure 15:

1. **Resource Utilization Comparison**:
    - **NANTAR vs. MRHOF**:

NANTAR utilized marginally more resources compared to MRHOF. This suggests that NANTAR introduces a slight overhead in terms of computational demands. The increase in resource usage is likely due to the additional processing required for machine learning operations within NANTAR.

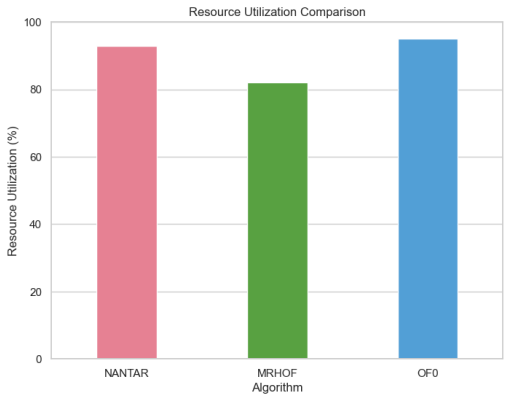

Figure 15: A comparative of power consumption over time for each algorithm.

- **NANTAR vs. OF0**:

Significantly, NANTAR was **more efficient** than OF0 in terms of resource utilization. OF0, being a traditional approach, might have more rigid resource allocation strategies or less adaptive mechanisms, leading to higher resource consumption.
NANTAR's ability to dynamically adjust its resource usage based on real-time conditions likely contributes to its efficiency.

2. **Reasons Behind the Findings**:
   - **Machine Learning Overhead**:

NANTAR's machine learning component introduces some computational overhead. While this is expected, it remains minimal due to the efficiency of the Random Forest model. MRHOF, being a standard protocol, does not involve ML operations, resulting in lower resource usage.

   - **Adaptive Learning and Anomaly Detection**:

NANTAR's ability to adapt to changing network conditions and detect anomalies efficiently might require slightly more computational resources. OF0 lacks such adaptive features, which could lead to less efficient resource utilization.

6.8 Comparison with Other Models in Literature

The NANTAR model is compared with other notable models in the literature to highlight its novelty and added value as shown in Table 5. The comparison is based on several criteria, including the approach to security, the machine learning algorithms used, and the overall effectiveness in enhancing IoT routing security. while each of the mentioned works contributes valuable insights and approaches to enhancing IoT routing security, NANTAR stands out for its comprehensive application, detailed performance evaluation, and adaptability to various attack scenarios. Its use of the Random Forest algorithm within a system that includes real-time adaptation and response mechanisms offers a more robust solution compared to the other models.

Table 5: Comparison with Other Models in Literature

Feature/Model	**NANTAR**	**RFTRUST [29]**	**ML-LGBM [30]**	**Multi-class Dataset [22]**	**Multiple Attack Detection [23]**	**Machine Learning for Attack Detection [25]**

Approach	Comprehensive security enhancement for multiple attacks	Sinkhole attack detection using trust mechanism	Version number attack detection using LightGBM	Attack simulation and dataset generation for machine learning	Multiple attack detection in RPL over IoT	IoT routing attack detection using machine learning
Machine Learning Algorithm	Random Forest	Random Forest	Light Gradient Boosting Machine	Not specified	Not specified	Not specified
Effectiveness	Improved throughput, latency, and false rate reduction	Focused on sinkhole attacks	Specific to version number attacks	Theoretical and simulation-based	General multiple attack detection	General attack detection
Scalability	Maintains performance with scaling	Not specified	Not specified	Not specified	Not specified	Not specified
Real-time Adaptation	Yes	Not specified	Not specified	Not specified	Not specified	Not specified
Anomaly Detection	Yes	Yes	Yes	Yes	Yes	Yes
Power Consumption	Competitive	Not specified	Not specified	Not specified	Not specified	Not specified

5. CONCLUSION

In this study, we introduced the NANTAR algorithm, a novel approach designed to enhance IoT routing within the context of the Routing Protocol for Low-Power and Lossy Networks (RPL). By integrating AI techniques, NANTAR addresses critical challenges such as security vulnerabilities and suboptimal performance. Our findings demonstrate significant improvements in throughput, latency reduction, and resource utilization. Moreover, NANTAR's robustness against attacks and competitive power consumption make it a promising solution for securing IoT ecosystems. We recognize that NANTAR introduces additional computational overhead and its Random Forest model complexity may challenge real-time adaptability. Its scalability, while promising, also requires further validation. To address these drawbacks, future work will focus on optimizing for low-resource devices, simplifying the model, enhancing training protocols, and conducting scalability testing. As the IoT landscape evolves, we anticipate that NANTAR will become even more adept at meeting the complex demands of future IoT networks, providing a secure and efficient routing framework that is both robust and adaptable.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] Williams, P., et al., *A survey on security in the Internet of things with a focus on the impact of emerging technologies.* Internet of Things, 2022. **19**: p. 100564.
[2] Ashton, K., *That 'internet of things' thing.* RFID journal, 2009. **22**(7): p. 97-114.

[3] Morchid, A., et al., *Applications of Internet of things (IoT) and sensors technology to increase food security and agricultural Sustainability: Benefits and challenges.* Ain Shams Engineering Journal, 2023: p. 102509.

[4] Insights, F.B. *Internet of Things (IoT) Market Size, Share, and Covid-19* 2024 13/05/ 2024]; Available from: https://www.fortunebusinessinsights.com/industry-reports/internet-of-things-iot-market-100307.

[5] Furstenau, L.B., et al., *Internet of things: Conceptual network structure, main challenges and future directions.* Digital Communications and Networks, 2023. **9**(3): p. 677-687.

[6] Kumar, M., et al., *Healthcare Internet of Things (H-IoT): Current trends, prospects, applications, challenges, and security issues.* Electronics, 2023. **12**(9): p. 2050.

[7] Al-Amiedy, T.A., et al., *A systematic literature review on machine and deep learning approaches for detecting attacks in RPL-based 6LoWPAN of internet of things.* Sensors, 2022. **22**(9): p. 3400.

[8] Sohail, M., et al., *Routing protocols in vehicular ad-hoc networks (vanets): A comprehensive survey.* Internet of Things, 2023: p. 100837.

[9] Darabkh, K.A., et al., *RPL routing protocol over IoT: A comprehensive survey, recent advances, insights, bibliometric analysis, recommendations, and future directions.* Journal of Network and Computer Applications, 2022. **207**: p. 103476.

[10] Wakili, A., S. Bakkali, and N. Faruk. *AI-enhanced context-aware optimization of RPL routing protocol for IoT environments*. in *2023 2nd International Conference on Multidisciplinary Engineering and Applied Science (ICMEAS)*. 2023. IEEE.

[11] Ghodichor, N., et al., *Secure Routing Protocol to Mitigate Attacks by Using Blockchain Technology in MANET*. International Journal of Computer Networks & Communications, 2023. **15**: pp. 127-146.

[12] Kharrufa, H., H.A. Al-Kashoash, and A.H. Kemp, *RPL-based routing protocols in IoT applications: A review.* IEEE Sensors Journal, 2019. **19**(15): p. 5952-5967.

[13] Boudouaia, M.A., et al., *Security against rank attack in RPL protocol.* IEEE Network, 2020. **34**(4): p. 133-139.

[14] Alazab, A., et al., *Routing attacks detection in 6lowpan-based internet of things.* Electronics, 2023. **12**(6): p. 1320.

[15] Sumalatha, P. and G.S. Mahalakshmi, *Machine Learning Based Ensemble Classifier for Android Malware Detection.* International journal of Computer Networks & Communications, 2023. **15**: p. 111-122.

[16] Alshammari, A. and A. Aldribi *Apply machine learning techniques to detect malicious network traffic in cloud computing.* Journal of Big Data, 2021. **8**(1): p. 90.

[17] Al-Amiedy, T.A., et al., *A systematic literature review on attacks defense mechanisms in RPL-based 6LoWPAN of Internet of Things.* Internet of Things, 2023. **22**: p. 100741.

[18] Albinali, H. and F. Azzedin, *Towards RPL Attacks and Mitigation Taxonomy: Systematic Literature Review Approach.* IEEE Transactions on Network and Service Management, 2024.

[19] Bharatiya, J.P., *The role of machine learning in transforming business intelligence.* International Journal of Computing and Artificial Intelligence, 2023. **4**(1): p. 16-24.

[20] Ansari, S.M., et al., *Introduction To Machine Learning Techniques*, in *IoT, Machine Learning and Blockchain Technologies for Renewable Energy and Modern Hybrid Power Systems*. 2023, River Publishers. p. 93-120.

[21] Taye, M.M., *Understanding of machine learning with deep learning: architectures, workflow, applications and future directions.* Computers, 2023. **12**(5): p. 91.

[22] Sharma, M., et al. *Simulating attacks for RPL and generating multi-class datasets for supervised machine learning*. in *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. 2019. IEEE.

[23] Momand, M.D. and M.K. Mohsin. *Machine learning-based multiple attack detection in RPL over IoT*. in *2021 International Conference on Computer Communication and Informatics (ICCCI)*. 2021. IEEE.

[24] Musaddiq, A., T. Olsson, and F. Ahlgren, *Reinforcement-Learning-Based Routing and Resource Management for Internet of Things Environments: Theoretical Perspective and Challenges.* Sensors, 2023. **23**(19): p. 8263.

[25] Rabhi, S., T. Abbes, and F. Zarai, *IoT routing attacks detection using machine learning algorithms.* Wireless Personal Communications, 2023. **128**(3): p. 1839-1857.

[26] Osterlind, F., et al. *Cross-level sensor network simulation with cooja*. in *Proceedings. 2006 31st IEEE conference on local computer networks*. 2006. IEEE.
[27] Bullock, J. and J.T. Parker, *Wireshark for Security Professionals: Using Wireshark and the Metasploit Framework*. 2017: John Wiley & Sons.
[28] Elreedy, D. and A.F. Atiya, *A comprehensive analysis of synthetic minority oversampling technique (SMOTE) for handling class imbalance.* Information Sciences, 2019. **505**: p. 32-64.
[29] Prathapchandran, K. and T. Janani, *A trust aware security mechanism to detect sinkhole attack in RPL-based IoT environment using random forest–RFTRUST.* Computer Networks, 2021. **198**: p. 108413.
[30] Osman, M., et al., *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, 2021. **9**: p. 83654-83665.

AUTHORS

Abubakar Wakili is a PhD researcher at Euromed University of Fez, Morocco, specializing in computer networks, Internet of Things (IoT), cybersecurity and machine learning. He holds a master's and a bachelor's degree in computer science from Federal University Dutse, Nigeria, as well as a Nigeria Certificate in Education (NCE) in computer science, mathematics, and education from Federal College of Education (Technical) Bichi, Nigeria, in 2021, 2016 and 2009 respectively. Abubakar has earned several professional training certifications in various fields of information and communication technology, including IBM Machine Learning, Digital Forensics, Certified Network Security Specialist, CompTIA Security+, and Cisco Certified Network Associate (CCNA). His contributions to the literature include presentations at international conferences and publications in reputable journals.

Sara Bakkali is currently an Assistant Professor at the School of Digital Engineering and Artificial Intelligence, Euromed University of Fez, Morocco. She holds a PhD in computer networks from Mohammed-V University of Rabat, Morocco in 2015. Additionally, she has an engineering diploma in networking and telecommunication from the National School of Applied Sciences in Tangier, Morocco, and a master's degree in telecommunication from the Sciences and Technologies Faculty in Fes, Morocco. Sara has acquired several certifications in areas such as IoT security, IPv6, Cisco networking, Linux administration, and network security. Her research interests encompass routing and security in networks, Quality of Service (QoS) in inter-domain networks, IPv6, and the Internet of Things (IoT). She has made significant contributions to the academic community through her published papers in international journals and conferences.