Trust-Based Model for Secure Routing Against RPL Attacks in Internet of Things Using Machine Learning Algorithms

[1]R. Elango, [2]Dr. D. Maruthanayagam
[1]*Research Scholar, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India*
[2] *Dean Cum Professor, PG and Research Department of Computer Science, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India*

Abstract: The Internet of Things (IoT) is increasingly susceptible to security threats, particularly targeting the Routing Protocol for Low-Power and Lossy Networks (RPL). Ensuring secure and reliable routing is crucial for the performance and trustworthiness of IoT networks. This paper proposes a trust-based model for secure routing against RPL attacks by leveraging machine learning algorithms, including Random Forest, Support Vector Machine (SVM), Naive Bayes, K-Nearest Neighbors (KNN), and Neural Networks. The model calculates node reputation and detects anomalies to prevent routing attacks. The performance of the proposed model is evaluated using key metrics: Node Reputation, Anomaly Detection Metrics, Routing Overhead, Energy Efficiency, Throughput, and Packet Loss Rate. Experimental results demonstrate the effectiveness of the proposed model in enhancing IoT network security and efficiency while maintaining low overhead.

Keywords: Trust-Based Routing, IoT Security, RPL Attacks, Machine Learning, Random Forest, Support Vector Machine (SVM), Naive Bayes, K-Nearest Neighbors (KNN) and Neural Networks.

I. INTRODUCTION

The Internet of Things (IoT) has revolutionized the way devices communicate, enabling seamless interconnectivity and data exchange among a myriad of sensors, actuators, and smart devices. This interconnected ecosystem facilitates numerous applications, ranging from smart homes and industrial automation to healthcare and environmental monitoring. However, the widespread deployment of IoT devices introduces significant security challenges, primarily due to their resource-constrained nature and reliance on wireless communication. One critical aspect of IoT security is ensuring secure and efficient routing of data, as IoT networks are particularly vulnerable to various routing attacks [1]. The Routing Protocol for Low-Power and Lossy Networks (RPL) is a widely adopted standard for routing in IoT environments. Despite its efficiency, RPL is susceptible to numerous attacks such as sinkhole, wormhole, and selective forwarding attacks, which can severely disrupt network operations and compromise data integrity [2]. Traditional security mechanisms are often inadequate for IoT networks due to their limited computational and energy resources. Hence, there is a pressing need for lightweight and robust solutions to enhance routing security in IoT.

Securing routing in IoT networks against RPL-specific attacks is a complex problem that requires balancing security, performance, and resource efficiency. Existing approaches often fail to address the dynamic and distributed nature of IoT networks. Therefore, a trust-based model that leverages machine learning to dynamically assess and respond to security threats is essential. Such a model can help in identifying malicious nodes, ensuring reliable data transmission, and maintaining overall network integrity [3].

This research aims to develop a trust-based model for secure routing in IoT networks, utilizing various machine learning algorithms. The specific objectives of this study are:

- To design a trust computation framework that evaluates node reputation based on behavior and interaction history.
- To implement and compare multiple machine learning algorithms, including Random Forest, Support Vector Machine (SVM), Naive Bayes, K-

Nearest Neighbors (KNN), and Neural Networks, for detecting routing anomalies.
- To evaluate the proposed model using key metrics such as Node Reputation, Anomaly Detection Metrics, Routing Overhead, Energy Efficiency, Throughput, and Packet Loss Rate.
- To demonstrate the effectiveness of the model in enhancing IoT network security while maintaining low resource consumption.

The primary contributions of this research are:
- A novel trust-based model for secure routing in IoT, capable of mitigating RPL-specific attacks.
- A comprehensive comparison of five machine learning algorithms (Random Forest, SVM, Naive Bayes, KNN, and Neural Networks) in the context of IoT routing security.
- An extensive evaluation of the proposed model across multiple performance and security metrics, demonstrating its efficacy and efficiency.
- Insights and recommendations for deploying machine learning-based security solutions in resource-constrained IoT environments.

This paper is structured as follows: Section 2 reviews related work in trust-based routing and machine learning applications in IoT security. Section 3 details the proposed model and the machine learning algorithms employed. Section 4 presents the experimental results and discussion. Finally, Section 5 concludes the paper and outlines future research directions.

1.1. RPL Routing Attacks and their implications for IoT Networks

Routing Protocol for Low-Power and Lossy Networks (RPL) is a fundamental routing protocol used in many IoT deployments due to its efficiency in handling constrained devices and dynamic network topologies. However, the widespread adoption of RPL also introduces vulnerabilities that can be exploited by malicious actors to compromise the integrity and availability of IoT networks. Understanding these RPL routing attacks and their implications is crucial for designing effective security mechanisms to safeguard IoT deployments [4][5]. Here, we discuss some common RPL routing attacks and their implications for IoT networks:

a). Sinkhole Attacks: In a sinkhole attack, a malicious node advertises itself as having the shortest path to the sink node, attracting legitimate nodes to route their traffic through it. However, the malicious node drops or selectively forwards the received packets, disrupting communication and potentially compromising data confidentiality. *Implications:* Sinkhole attacks can lead to significant disruptions in IoT applications, particularly those requiring timely and reliable data delivery, such as industrial automation and healthcare monitoring. Moreover, they can facilitate data exfiltration or manipulation, posing serious security and privacy risks.

b). Blackhole Attacks: In a blackhole attack, a malicious node advertises itself as having the shortest path to the destination nodes, attracting traffic towards it. However, instead of forwarding the packets towards the destination, the malicious node drops all incoming packets, effectively blackholing the traffic. *Implications:* Blackhole attacks can severely impact the availability and reliability of IoT services by causing packet loss and network congestion. Furthermore, they can facilitate denial-of-service (DoS) attacks by disrupting communication between legitimate nodes and draining their resources.

c). Selective Forwarding Attacks: In a selective forwarding attack, a malicious node selectively forwards or drops packets based on predefined criteria, such as packet type, source address, or content. By strategically manipulating packet forwarding, the attacker can disrupt communication between specific nodes or subnetworks. *Implications:* Selective forwarding attacks can compromise the integrity and reliability of IoT data transmission by selectively dropping critical packets or injecting malicious payloads into the network. This can lead to data corruption, manipulation, or unauthorized access, undermining the trustworthiness of IoT applications.

d). Sybil Attacks: In a Sybil attack, a single malicious node impersonates multiple legitimate nodes by spoofing their identities or network addresses. By creating multiple fake identities, the attacker can influence routing decisions, disrupt network topology discovery, and deceive neighboring

nodes. *Implications:* Sybil attacks can undermine the trustworthiness and scalability of IoT networks by introducing fake nodes into the network topology. This can lead to routing loops, resource exhaustion, and inaccurate routing metrics, compromising the overall stability and security of IoT deployments.

The implications of these RPL routing attacks for IoT networks are multifaceted, encompassing disruptions in communication, data integrity, and network performance. Mitigating RPL routing attacks requires a combination of robust authentication mechanisms, secure routing protocols, and intrusion detection systems tailored to the unique requirements and constraints of IoT environments.

II. RELATED WORKS

Trust-based routing models have gained significant attention in securing IoT networks due to their ability to identify and mitigate malicious behavior. These models typically evaluate nodes based on their behavior and interaction history, assigning trust scores that influence routing decisions. For instance, *Momani and Challa (2010) [6] proposed a trust management system* for wireless sensor networks that relies on direct and indirect trust metrics to detect malicious nodes. Similarly, *Bao and Chen (2012) [7] introduced a dynamic trust management protocol* that adapts to changing network conditions and improves security in mobile ad-hoc networks (MANETs). In the context of IoT, *Raza et al. (2013) [8] developed a lightweight trust-based mechanism* specifically for RPL, focusing on energy-efficient and secure routing. Their approach, however, primarily relies on heuristic methods and does not leverage advanced machine learning techniques. More recently, *Zhang et al. (2018) [9] proposed a trust-based secure routing scheme* that incorporates fuzzy logic to handle the uncertainty and imprecision in trust evaluation. While these approaches demonstrate the potential of trust-based models, they often fall short in dynamically adapting to sophisticated and evolving attacks.

Machine learning (ML) techniques have been increasingly employed to enhance the security of routing protocols in IoT networks. These techniques can analyze large volumes of data to detect patterns indicative of malicious behavior, thereby improving the robustness of routing mechanisms. For example, *Othman et al. (2013) [10] applied a Bayesian inference approach* to establish trust in IoT environments, enhancing the detection of compromised nodes. Another study by *Marchang and Datta (2017) [11] utilized SVM to classify and detect routing attacks* in MANETs, showing promising results in terms of accuracy and efficiency. Random Forest has also been effectively used in network security. For instance, *Sahu and Shah (2018*) *[12] demonstrated the use of Random Forest* in detecting intrusion in IoT networks, highlighting its high accuracy and low false-positive rate. KNN and Naive Bayes, while simpler, have been employed in various anomaly detection systems due to their computational efficiency and effectiveness in diverse scenarios *(Patel and Doshi, 2020) [13]*. Neural Networks, particularly deep learning models, offer significant potential due to their ability to learn complex patterns; however, their high computational requirements often pose challenges in IoT environments (*Al-Garadi et al., 2020*) *[14]* .

RPL, as a widely used routing protocol in IoT, is vulnerable to various attacks that can compromise the network's security and performance. Sinkhole attacks, where malicious nodes attract traffic by advertising high-quality routes, are particularly damaging. *Krontiris et al. (2013) [15] analyzed the impact of sinkhole attacks on RPL* and proposed countermeasures based on consistency checks of routing information. Wormhole attacks, involving colluding nodes that create a low-latency link to capture and relay traffic, were studied by *Choi et al. (2014) [16], who proposed a detection mechanism* leveraging temporal and spatial correlation of packet arrivals. Selective forwarding attacks, where malicious nodes selectively drop packets, pose another significant threat. *Mayzaud et al. (2016) [17] provided a comprehensive survey of RPL attacks* and highlighted the need for integrated detection and mitigation strategies. These studies underscore the importance of robust security mechanisms to safeguard RPL-based IoT networks from various attack vectors.

III. PROPOSED MODEL

The increasing deployment of Internet of Things (IoT) devices in various applications, from smart homes to industrial automation, has brought significant attention to the need for secure and reliable

communication protocols. One of the primary challenges in IoT networks is ensuring secure routing of data, particularly in the face of attacks targeting the Routing Protocol for Low-Power and Lossy Networks (RPL). Traditional security mechanisms often fall short due to the constrained resources of IoT devices, necessitating innovative approaches that can provide robust security without imposing significant overhead. This paper presents a trust-based model for secure routing in IoT networks that leverages the capabilities of machine learning algorithms to detect and mitigate RPL-specific attacks. The proposed model integrates multiple machine learning techniques, including Random Forest, Support Vector Machine (SVM), Naive Bayes, K-Nearest Neighbors (KNN), and Neural Networks, to evaluate the trustworthiness of nodes and detect anomalies indicative of malicious activities.

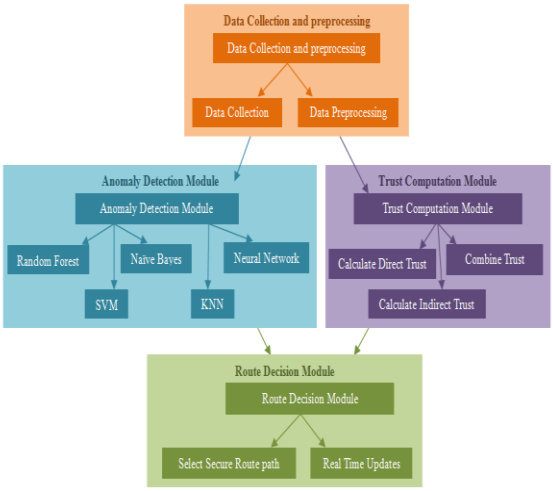

Figure 1: Proposed model architecture

System Architecture

The proposed trust-based model for secure routing against RPL attacks in IoT networks is composed of four main modules: Trust Computation, Anomaly Detection, Routing Decision, and Data Collection and Preprocessing. Figure 1 shows the proposed model architecture. Each module plays a crucial role in ensuring the security and efficiency of the IoT network.

3.1 Trust Computation Module

The Trust Computation Module is a crucial component of the proposed model, responsible for calculating the trust scores of nodes in the IoT network[18][19]. The trust scores are used to evaluate the reliability and behavior of each node, which in turn influence the routing decisions made by the Routing Decision Module. The trust computation process incorporates both direct and indirect trust metrics to provide a comprehensive assessment of node trustworthiness, Figure 2.

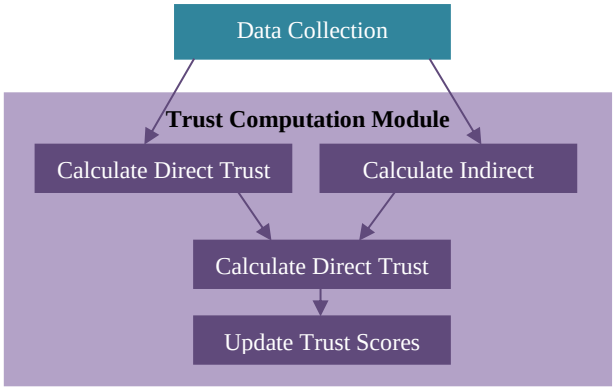

Figure 2: Trust Computation Process

Direct Trust Computation

Direct trust is calculated based on the interaction history between nodes. This includes factors such as packet delivery ratio, acknowledgment reception and consistency in communication. Nodes that consistently deliver packets successfully and acknowledge communications are assigned higher direct trust scores. Direct trust is calculated based on the historical interactions between nodes. It considers various factors such as packet delivery success, acknowledgment reception, and consistency in communication. The formula for calculating direct trust DTij from node i to node j is as follows:

$$DT_{ij} = \frac{\text{Number of Successful Interactions}}{\text{Total Interactions}}$$

This metric reflects how reliably node j has performed in past interactions with node i.

Indirect Trust Computation

Indirect trust is derived from recommendations provided by neighboring nodes. Each recommendation is weighted by the trustworthiness of the recommending node, ensuring that only reliable recommendations influence the trust score. The formula for calculating indirect trust IT_{ij} from node i to node j is:

$$IT_{ij} = \frac{\sum_{k \in N_i} T_{ik} . DT_{kj}}{\sum_{k \in N_i} T_{ik}}$$

Where, N_i is the set of neighbors of node i. T_{ik} is the trust score of node i towards neighbor k. DT_{kj} is the direct trust of neighbor k towards node j.

Combined Trust Score

The overall trust score T_{ij} of node i towards node j is a weighted combination of direct and indirect trust scores:

$$T_{ij} = \alpha . DT_{ij} + (1 - \alpha) . IT_{ij}$$

Where α is a weighting factor that balances the influence of direct and indirect trust.

By incorporating both direct and indirect trust metrics, the Trust Computation Module ensures a robust and comprehensive assessment of node trustworthiness, enhancing the overall security and reliability of the IoT network.

3.2. Anomaly Detection Module

The Anomaly Detection Module is a critical component of the proposed trust-based model, designed to identify suspicious activities and potential attacks within the IoT networks. By leveraging various machine learning algorithms, this module can detect anomalies in node behavior, which can indicate security threats such as RPL attacks. The detected anomalies are then used to update the trust scores and inform routing decisions. The Anomaly Detection Module employs a diverse set of machine learning algorithms to analyze network data and detect anomalies. Each algorithm has its strengths and is suited for different aspects of anomaly detection:

3.2.1. Random Forest:

The integration of Random Forest into the enhancement of trust-based secure routing in IoT networks aims to address the challenges of trust evaluation, anomaly detection, and adaptive routing in IoT environments. This integration enables the development of a robust and adaptive routing system capable of dynamically adjusting to changing network conditions and mitigating security threats. In this section, we present a comprehensive framework for integrating Random Forest into trust-based secure routing in IoT [20][21]. We discuss the technical details of each step, including feature selection, training data collection, model training, trust evaluation, adaptation, and integration with routing policies. Furthermore, we provide real-world examples and case studies to demonstrate the effectiveness and practicality of the proposed approach. This contributes to the advancement of trust-based secure routing systems in IoT by harnessing the power of machine learning techniques. By integrating Random Forest into IoT routing architectures, we aim to enhance the security, reliability, and resilience of IoT communication, thereby enabling the widespread adoption of IoT technologies across various application domains.

Step 1: Feature Selection and Extraction

This step involves selecting and extracting relevant features from the routing environment to serve as input to the Random Forest algorithm. These features may include node reputation, communication reliability, security posture and historical behavior. *For example,* suppose we're evaluating the trustworthiness of routing paths in an IoT network. Features could include node reputation (based on past interactions), communication reliability (e.g., packet loss rate), security posture (e.g., encryption protocols used), and historical behavior (e.g., routing patterns).

$$X=[x_1, x2,..., xn]$$

In this step, relevant features are selected and extracted from the routing environment to serve as input to the Random Forest algorithm. Features may include node attributes (e.g., reputation, resource availability), communication metrics (e.g., latency, packet loss rate), and security indicators (e.g., encryption protocols used). X represents the feature vector containing n features.

Step 2: Training Data Collection

Training data consists of labeled examples, where each example includes a feature vector and its corresponding label. Labels indicate the desired output, such as the correctness of routing decisions or the presence of security threats. For example, we collect training data by observing routing decisions in the IoT network over time. Each example includes features such as node reputation, communication reliability, and security posture, along with labels indicating whether the routing decision was successful or if a security threat was detected.

$$D=\{(X_1,Y_1),(X_2,Y_2),...,(X_m,Y_m)\}$$

Training data D consists of labeled examples, where each example X_i, Y_i consists of a feature vector X_i and its corresponding label Y_i. Labels indicate the desired

output or classification, such as the correctness of routing decisions or the presence of security threats.

Step 3: Model Training

In this step, the Random Forest model is trained using the labeled training data. The model constructs an ensemble of decision trees based on random subsets of the training data. For example, we train the Random Forest model using the collected training data. The model learns to predict the trustworthiness of routing paths based on the observed features and labels. Each decision tree in the ensemble independently learns patterns in the data, contributing to the overall predictive power of the model.

$$RF=\text{Train}(D)$$

The Random Forest model (RF) is trained using the training data D. The training process constructs an ensemble of decision trees based on random subsets of the training data. Each decision tree independently makes predictions, and the final output is determined by aggregating the predictions of all trees.

Step 4: Trust Evaluation and Routing Decision

Given a feature vector representing the current routing environment, the trained Random Forest model predicts the trustworthiness of routing paths or the likelihood of security threats. The prediction is based on the learned patterns in the training data and the input features. For example, suppose we want to evaluate the trustworthiness of a routing path in real-time. We input the relevant features (e.g., node reputation, communication reliability) into the trained Random Forest model. The model predicts whether the routing path is trustworthy or if there's a potential security threat.

$$\hat{Y}=RF(X)$$

Given a feature vector X representing the current routing environment, the trained Random Forest model (RF) predicts the output Y^, which may indicate the trustworthiness of routing paths or the likelihood of security threats. The prediction is based on the learned patterns in the training data and the input features.

Step 5: Adaptation and Learning

To adapt to changing network conditions and new data, the Random Forest model is periodically retrained using updated training data. This allows the model to learn from recent experiences and improve its predictive performance over time. For example, as new routing decisions and security events occur in the IoT network, we collect updated training data. We then retrain the Random Forest model using this new data, incorporating the latest observations and labels. This allows the model to adapt to evolving network dynamics and improve its accuracy.

$$RF_{updated}=\text{Retrain}(D')$$

To adapt the Random Forest model to changing network conditions and new data, the model is periodically retrained using updated training data \(D' \). This retraining process incorporates new observations and labels, allowing the model to learn from recent experiences and improve its predictive performance over time.

Step 6: Integration with Trust-Based Routing Policies

The output from the Random Forest model is integrated with trust-based routing policies through a decision function. This function maps the model predictions to routing decisions, prioritizing trusted routes or avoiding untrustworthy nodes based on the predicted trustworthiness or security risk levels. For example, based on the prediction from the Random Forest model, we apply trust-based routing policies to make routing decisions. For example, if the model predicts a high level of trust for a routing path, we prioritize using that path for data transmission. Conversely, if a potential security threat is detected, we avoid using routes associated with that threat.

$$\text{Routing Decision}=g(\hat{Y})$$

The output Y^ from the Random Forest model is integrated with trust-based routing policies through a decision function g . This function maps the model predictions to routing decisions, prioritizing trusted routes or avoiding untrustworthy nodes based on the predicted trustworthiness or security risk levels.

Anomaly Detection in RPL Attacks Using Random Forest

In the context of trust-based secure routing, Random Forest can also be leveraged to detect anomalies indicative of RPL attacks. Here's how the anomaly detection process integrates with the overall secure routing framework:

Feature Extraction: Features specific to RPL anomalies (e.g., sudden changes in route paths, increased packet loss) are extracted and included in the feature vector X.

- Correlation Matrix: Calculate the correlation matrix to select relevant features.
  - ✓ $R= cov(X) / \sigma_x\sigma_y$

  Where cov(X) is the covariance matrix, σ_x and σ_y are the standard deviations of the variables.
- Feature Importance (from Random Forest): Use feature importance scores from Random Forest.

$$I_j = \frac{1}{N_T} \sum_{t=1}^{N_T} \Delta_i^t$$

  - Where Δit is the importance of feature j in tree t, and N_T is the number of trees.
- Training Data Collection: Collect labeled data that includes both normal and anomalous RPL behavior to train the model.
- Model Training: Train the Random Forest model on this comprehensive dataset to distinguish between normal and anomalous behaviors.
- Anomaly Detection: During real-time routing, use the trained model to predict the likelihood of an RPL attack based on the input features.If an anomaly is detected, adjust the routing decision accordingly.

  Anomaly Score=$RF_{anomaly}(X)$

  If the anomaly score exceeds a certain threshold, it indicates a potential RPL attack, prompting the routing algorithm to avoid the suspicious route.

By integrating Random Forest for both trust evaluation and anomaly detection, the IoT network can achieve robust security against RPL attacks while ensuring efficient and trustworthy routing decisions.

3.2.2. Support Vector Machine (SVM)

Support Vector Machine (SVM) [22][23], a powerful classification algorithm, offers a promising approach to enhancing trust-based routing by leveraging its ability to classify data into different categories based on learned patterns. By integrating SVM into the routing architecture, IoT networks can benefit from sophisticated trust evaluation mechanisms that adapt to changing network dynamics and security threats. This integration aims to address the challenges of trust evaluation, anomaly detection, and adaptive routing in IoT environments, enabling the development of a robust and adaptive routing system capable of dynamically adjusting to changing network conditions and mitigating security threats. In this paper, we present a comprehensive framework for integrating SVM into trust-based secure routing in IoT. We discuss the technical details of each step, including feature selection, training data collection, model training, trust evaluation, adaptation, and integration with routing policies. By leveraging SVM's classification capabilities, our proposed approach aims to enhance the security, reliability, and resilience of IoT communication, ultimately contributing to the advancement of trust-based secure routing systems in IoT networks. Steps of Integrating Support Vector Machine (SVM) into Enhancing Trust-Based Secure Routing in IoT

Step 1: Feature Selection and Extraction

This step involves selecting and extracting relevant features from the routing environment to serve as input to the SVM algorithm. These features may include node reputation, communication reliability, security posture, and historical behavior. For example, features could include the number of successful data transmissions, the frequency of routing table updates, the response time of neighboring nodes, and the level of encryption used for communication.

$$X=[x_1, x2,..., xn]$$

Similar to the Random Forest integration, this step involves selecting and extracting relevant features from the routing environment to serve as input to the SVM algorithm. Features may include node reputation, communication reliability, security posture, and historical behavior.

Step 2: Training Data Collection

Training data consists of labeled examples, where each example includes a feature vector and its corresponding label. Labels indicate the desired output, such as whether a routing path is trustworthy or if a security threat is present. For example, collecting training data involves observing routing decisions in the IoT network over time. Each example includes features such as node reputation, communication reliability and security posture, along with labels indicating whether the routing decision was successful or if a security threat was detected.

$$D=\{(X_1,Y_1),(X_2,Y_2),...,(X_m,Y_m)\}$$

Training data consists of labeled examples, where each example includes a feature vector and its corresponding label. Labels indicate the desired

output, such as the correctness of routing decisions or the presence of security threats.

Step 3: Model Training

In this step, the SVM model is trained using the labeled training data. The model learns to classify the feature vectors into different classes (e.g., trustworthy vs. untrustworthy routes) by finding the hyper plane that maximally separates the classes.

$$SVM=Train\ (D)$$

For example, Training the SVM model involves feeding the labeled training examples into the SVM algorithm. The algorithm learns to classify the feature vectors into different classes based on the observed patterns in the training data.

Step 4: Trust Evaluation and Routing Decision

Given a feature vector representing the current routing environment, the trained SVM model predicts the class label for the input. The prediction is based on the learned patterns in the training data and the input features. For example, suppose we want to evaluate the trustworthiness of a routing path in real-time. We input the relevant features (e.g., node reputation, communication reliability) into the trained SVM model. The model predicts whether the routing path is trustworthy or if there's a potential security threat.

$$\hat{Y}=SVM(X)$$

Given a feature vector representing the current routing environment, the trained SVM model predicts the class label $\hat{Y}$ for the input. The prediction is based on the learned patterns in the training data and the input features.

Step 5: Adaptation and Learning

Similar to Random Forest, the SVM model can be periodically retrained using updated training data to adapt to changing network conditions and improve its predictive performance over time. *For example*, as new routing decisions and security events occur in the IoT network, we collect updated training data. We then retrain the SVM model using this new data, incorporating the latest observations and labels. This allows the model to adapt to evolving network dynamics and improve its accuracy.

$$SVM_{updated}=Retrain\ (D')$$

Similar to Random Forest, the SVM model can be periodically retrained using updated training data to adapt to changing network conditions and improve its predictive performance over time.

Step 6: Integration with Trust-Based Routing Policies

The output from the SVM model is integrated with trust-based routing policies through a decision function. This function maps the model predictions to routing decisions, prioritizing trusted routes or avoiding untrustworthy nodes based on the predicted class labels. For example, based on the prediction from the SVM model, we apply trust-based routing policies to make routing decisions. For example, if the model predicts a high level of trust for a routing path, we prioritize using that path for data transmission. Conversely, if a potential security threat is detected, we avoid using routes associated with that threat.

$$Routing\ Decision=g(\hat{Y})$$

The output from the SVM model is integrated with trust-based routing policies through a decision function. This function maps the model predictions to routing decisions, prioritizing trusted routes or avoiding untrustworthy nodes based on the predicted class labels.

Anomaly Detection in RPL Attacks Using SVM

Feature Selection and Extraction for Anomaly Detection: Relevant features specific to RPL anomalies (e.g., sudden changes in route paths, increased packet loss) are extracted and included in the feature vector X.

Training Data Collection for Anomaly Detection: Collect labeled data that includes both normal and anomalous RPL behavior to train the model.

Model Training for Anomaly Detection: Train the SVM model on this comprehensive dataset to distinguish between normal and anomalous behaviors.

Anomaly Detection: During real-time routing, use the trained model to predict the likelihood of an RPL attack based on the input features. If an anomaly is detected, adjust the routing decision accordingly.

$$Anomaly\ Score = SVM_{anomaly}(X)$$

If the anomaly score exceeds a certain threshold, it indicates a potential RPL attack, prompting the routing algorithm to avoid the suspicious route.

By integrating Support Vector Machine (SVM) for both trust evaluation and anomaly detection, the IoT network can achieve robust security against RPL attacks while ensuring efficient and trustworthy routing decisions. This approach leverages SVM's classification capabilities to assess trustworthiness and make informed routing decisions in dynamic IoT environments.

3.2.3. Naive Bayes

The integration of machine learning techniques, such as the Naive Bayes classification algorithm, into trust-based secure routing systems represents a promising approach to fortify the security of Internet of Things (IoT) networks. Traditional routing protocols like RPL often struggle to adequately address security concerns due to evolving threats and vulnerabilities. By leveraging probabilistic reasoning and historical data, Naive Bayes classifiers [24][25] offer a means to assess the trustworthiness of neighboring nodes and make informed routing decisions. This paper explores the integration of Naive Bayes algorithms into trust-based routing mechanisms, aiming to enhance the resilience and security of IoT networks against malicious attacks and unauthorized access. Through a combination of technical analysis and practical examples, we demonstrate the potential of this approach to mitigate security risks and ensure the reliable operation of IoT deployments.

Step 1. Trustworthiness Estimation:

This step involves estimating the trustworthiness of neighboring nodes based on observed features such as communication history, packet delivery ratio, and energy consumption. Naive Bayes is used to calculate the posterior probability of each node being trusted or untrusted given its observed features. *Example:* Suppose we have a set of features for each neighboring node, including the number of packets successfully delivered, the number of packets dropped, and the energy level of the node. By analyzing historical data, we can calculate the conditional probabilities of these features given the class labels (trusted or untrusted). Naive Bayes then combines these probabilities to estimate the trustworthiness of each node.

- Define a set of features that capture the behavior and characteristics of neighboring nodes relevant to trustworthiness, such as communication history, packet delivery ratio, energy consumption, and proximity.
- For each feature X_i, calculate the conditional probability $P(X_i \mid C_j)$, where C_j represents the class label (trusted or untrusted) of the node.
- Apply Bayes' theorem to estimate the posterior probability $P(C_j \mid X_1,X_2,...,X_n)$ of each neighboring node being trusted or untrusted based on the observed features.

$$P(C_j|X_1,X_2,...,X_n)= P(X_1|C_j)\times P(X_2|C_j)\times ...\times P(X_n|C_j)\times P(C_j)/ P(X_1)\times P(X_2)\times ...\times P(X_n)$$

Step 2. Trust-based Routing Decision:

In this step, the trustworthiness estimation is integrated into the routing decision process to select trusted routes for data transmission. A routing metric is defined that combines traditional metrics (e.g., hop count) with the estimated trustworthiness of neighboring nodes. The route with the highest trustworthiness score is chosen for data forwarding. Example: Let's say we have two candidate routes for transmitting data: Route A has a lower hop count but traverses nodes with lower trustworthiness scores, while Route B has a slightly higher hop count but traverses nodes with higher trustworthiness scores. By applying the trust-based routing metric formula, we can weigh the importance of hop count against the trustworthiness of nodes to make an informed routing decision.

- Incorporate the trustworthiness estimation into the routing decision process to select trusted routes for data transmission.
- Define a routing metric that combines traditional routing metrics (e.g., hop count, link quality) with the estimated trustworthiness of neighboring nodes.
- Calculate the trust-based routing metric for each candidate route and select the route with the highest trustworthiness score for data forwarding.

$$\textit{Trust based Routing Metric} = \alpha \times \textit{Traditional Routing Metric} + (1-\alpha) \times \textit{Trustworthiness score}$$

Where α is a weighting factor balancing the importance of traditional metrics and trustworthiness.

Step 3. Naive Bayes Training and Classification:

Historical data collected from IoT network operations is used to train the Naive Bayes classifier. This data is split into training and testing sets, and the classifier learns the conditional probability distributions of features given the class labels (trusted or untrusted). During classification, the trained classifier estimates the trustworthiness of neighboring nodes based on observed features. Example: Consider a dataset containing information about past interactions between nodes in the IoT network, including features such as packet delivery ratio, energy consumption, and communication patterns. By training the Naive Bayes classifier on this dataset, it learns to distinguish between trusted and untrusted nodes based on their observed behavior. During classification, the classifier

applies this knowledge to assign trustworthiness labels to new instances based on their features.

- Utilize historical data collected from IoT network operations to train the Naive Bayes classifier.
- Split the dataset into training and testing sets, ensuring representative samples of trusted and untrusted nodes.
- Train the Naive Bayes classifier using the training data to learn the conditional probability distributions of features given the class labels.
- During classification, apply the trained classifier to estimate the trustworthiness of neighboring nodes based on observed features and assign them appropriate class labels (trusted or untrusted).

$$P(C_j|X_1,X_2,...,X_n)=P(X_1|C_j)\times P(X_2|C_j)\times...\times P(X_n|C_j)\times P(C_j) / P(X_1)\times P(X_2)\times...\times P(X_n)$$

Step 4. Adaptive Learning and Updating:

This step involves continuously monitoring the performance of the trust-based routing system and updating the Naive Bayes classifier to adapt to changing network conditions and emerging threats. Feedback mechanisms are integrated to incorporate real-time observations and user feedback into the training process, enhancing the accuracy and robustness of the trustworthiness estimation model over time. Example: As the IoT network evolves and new nodes join or leave the network, the trustworthiness of nodes may change dynamically. By periodically updating the Naive Bayes classifier with fresh data and feedback from network operations, we can ensure that the trustworthiness estimation model remains accurate and reflective of the current network state. This adaptive learning process allows the routing system to adapt to evolving threats and maintain the security and reliability of the IoT network.

- Continuously monitor the performance of the trust-based routing system and update the Naive Bayes classifier periodically to adapt to changing network conditions and emerging threats.
- Integrate feedback mechanisms to incorporate real-time observations and user feedback into the training process, enhancing the accuracy and robustness of the trustworthiness estimation model over time.

3.2.4. K-Nearest Neighbors (KNN)

Integrating K-Nearest Neighbors (KNN) [26][27] into trust-based secure routing in IoT represents a strategic fusion of machine learning with network security principles. In the dynamic landscape of IoT, where nodes communicate across diverse environments, ensuring the trustworthiness of data routing becomes paramount. By leveraging KNN, we can assess the reliability of neighboring nodes based on historical behavior patterns, thus fortifying the network against potential threats and vulnerabilities [17] [18]. This explores how KNN algorithms can be seamlessly integrated into trust-based routing systems, offering a robust framework to enhance the security and resilience of IoT networks. Through a concise yet comprehensive analysis, we uncover the potential of KNN in bolster (boost) trust-based routing mechanisms, ultimately paving the way for more secure and efficient IoT deployments.

Here's an explanation and example for each step:

Step 1. Trustworthiness Estimation using KNN:

KNN can be used to estimate the trustworthiness of neighboring nodes based on observed features such as communication history, packet delivery ratio, and energy consumption. By analyzing the features of the nearest neighbors, we can infer the trustworthiness of a node in the network. Example: Suppose we have historical data on node behavior in the IoT network, including features such as packet delivery ratio, uptime and communication frequency. Using KNN, we can find the K nearest neighbors of a node based on these features. If the majority of the nearest neighbors are known to be trustworthy, we can infer that the node in question is also likely to be trustworthy.

- Use KNN to estimate the trustworthiness of neighboring nodes based on observed features such as communication history, packet delivery ratio, and energy consumption. The trustworthiness of a node is determined by the class label assigned to it, where a "trusted" node is one that is likely to behave reliably and responsibly.
- $D(x_i,x_j)$ represents the distance metric between nodes x_i and x_j.
- y_i represents the class label of node x_i.
- k is the number of nearest neighbors to consider.
- For classification, assign the majority class label among the k nearest neighbors to the node being evaluated.

Step 2. Trust-based Routing Decision:

In this step, the trustworthiness estimation obtained from KNN is integrated into the routing decision process. Routes through nodes with higher trustworthiness scores are prioritized for data transmission to enhance security and reliability. Example: Consider a scenario where a node in the IoT network needs to forward data packets to a destination node. Instead of blindly choosing the shortest path or the path with the fewest hops, the routing algorithm considers the trustworthiness scores of neighboring nodes obtained from KNN. It selects the route with the highest average trustworthiness score among its neighbors, thereby minimizing the risk of routing through potentially malicious nodes.

- Integrate the trustworthiness estimation into the routing decision process. Consider the trustworthiness of neighboring nodes as a factor when selecting the optimal route for data transmission. Nodes with higher trustworthiness scores are prioritized in the routing process.
- Define a trust-based routing metric that combines traditional routing metrics (e.g., hop count, link quality) with the estimated trustworthiness of neighboring nodes.
- Assign weights to each metric based on their relative importance.
- Calculate the trust-based routing metric for each candidate route and select the route with the highest trustworthiness score for data forwarding.

Step 3. KNN Training and Classification:

The KNN algorithm is trained using historical data to learn the relationship between features and trustworthiness labels (trusted or untrusted). During classification, the trained model is used to predict the trustworthiness of neighboring nodes based on their observed features. Example: Suppose we have a dataset containing information about past interactions between nodes in the IoT network, including features such as communication patterns, packet delivery ratios, and energy consumption. We split this dataset into training and testing sets and train the KNN classifier using the training data. When a new node joins the network, its features are input into the trained KNN model to predict its trustworthiness label (trusted or untrusted) based on its similarity to past instances.

- Train the KNN classifier using historical data collected from IoT network operations. The classifier learns the relationship between features and trustworthiness labels (trusted or untrusted). During classification, apply the trained classifier to estimate the trustworthiness of neighboring nodes based on observed features.
  a. Train the KNN classifier using the training dataset, where each instance represents a node with its observed features and corresponding trustworthiness label.
  b. Use the Euclidean distance metric or other appropriate distance metrics to calculate the similarity between nodes.
  c. During classification, query the KNN classifier with the features of a node to predict its trustworthiness label.

Step 4. Adaptive Learning and Updating:

To maintain the accuracy and relevance of the trustworthiness estimation model, it needs to be continuously updated based on real-time observations and user feedback. This involves periodically retraining the KNN classifier with new data and incorporating feedback mechanisms to adapt to changing network conditions. Example: As the IoT network evolves and new nodes join or leave the network, the trustworthiness of nodes may change dynamically. Therefore, the KNN model needs to be periodically retrained with fresh data to reflect the current state of the network. Additionally, feedback mechanisms can be implemented to incorporate real-time observations and user feedback into the training process, enabling the model to adapt to emerging threats and maintain its accuracy over time.

- Continuously monitor the performance of the trust-based routing system and update the KNN classifier to adapt to changing network conditions and emerging threats. Incorporate real-time observations and user feedback into the training process to improve the accuracy and robustness of trustworthiness estimation.

✓ Periodically update the KNN classifier with new training data and feedback from network operations.

- ✓ Implement mechanisms to handle concept drift and maintain model freshness in dynamic IoT environments.
- ✓ Use techniques such as online learning and incremental updates to adapt the KNN classifier to evolving network conditions over time.

By integrating KNN into the trust-based secure routing system in IoT, we can leverage its ability to analyze past behavior and classify nodes based on their similarity to historical instances. This approach enhances the security and resilience of IoT networks by prioritizing routes through trustworthy nodes and mitigating the risk of routing attacks and unauthorized access.

3.2.5. Neural Networks

Neural Networks (NNs) are a powerful class of machine learning algorithms capable of capturing complex patterns in data. By integrating NNs into trust-based secure routing, IoT networks can achieve advanced trust evaluation, anomaly detection, and adaptive routing. This integration addresses the dynamic and heterogeneous nature of IoT environments, enhancing security and reliability against evolving threats. Steps for Integrating Neural Networks for Trust-Based Secure Routing in IoT,

Step 1: Feature Selection and Extraction

This step involves selecting and extracting relevant features from the routing environment to serve as input to the neural network. Features may include node reputation, communication reliability, security posture, and historical behavior. For example, features could include the number of successful data transmissions, the frequency of routing table updates, the response time of neighboring nodes, and the level of encryption used for communication.

$$X = x_1, x_2, \ldots. x_n$$

Step 2: Training Data Collection

Training data consists of labeled examples, where each example includes a feature vector and its corresponding label. Labels indicate the desired output, such as whether a routing path is trustworthy or if a security threat is present. For example, collecting training data involves observing routing decisions in the IoT network over time. Each example includes features such as node reputation, communication reliability, and security posture, along with labels indicating whether the routing decision was successful or if a security threat was detected.

$$D=\{(X_1,Y_1),(X_2,Y_2),...,(X_m,Y_m)\}$$

Step 3: Model Training

In this step, the neural network is trained using the labeled training data. The model learns to classify the feature vectors into different classes (e.g., trustworthy vs. untrustworthy routes) by adjusting the weights and biases of its neurons based on the training data. For example, Training the neural network involves feeding the labeled training examples into the network. The algorithm learns to classify the feature vectors into different classes based on the observed patterns in the training data.

$$NN = Train\ (D)$$

Step 4: Trust Evaluation and Routing Decision

Given a feature vector representing the current routing environment, the trained neural network predicts the class label for the input. The prediction is based on the learned patterns in the training data and the input features. For example, suppose we want to evaluate the trustworthiness of a routing path in real-time. We input the relevant features (e.g., node reputation, communication reliability) into the trained neural network. The model predicts whether the routing path is trustworthy or if there's a potential security threat.

$$Y^{\wedge} = NN(X)$$

Step 5: Adaptation and Learning

The neural network can be periodically retrained using updated training data to adapt to changing network conditions and improve its predictive performance over time. For example, as new routing decisions and security events occur in the IoT network, we collect updated training data. We then retrain the neural network using this new data, incorporating the latest observations and labels. This allows the model to adapt to evolving network dynamics and improve its accuracy.

$$NN_{updated} = Retrain(D')$$

Step 6: Integration with Trust-Based Routing Policies

The output from the neural network is integrated with trust-based routing policies through a decision function. This function maps the model predictions to routing decisions, prioritizing trusted routes or avoiding untrustworthy nodes based on the predicted

class labels. For example, based on the prediction from the neural network, we apply trust-based routing policies to make routing decisions. For example, if the model predicts a high level of trust for a routing path, we prioritize using that path for data transmission. Conversely, if a potential security threat is detected, we avoid using routes associated with that threat.

$$\text{Routing Decision} = g(Y)$$

Anomaly Detection in RPL Attacks Using Neural Networks

Feature Selection and Extraction for Anomaly Detection: Relevant features specific to RPL anomalies (e.g., sudden changes in route paths, increased packet loss) are extracted and included in the feature vector X.

Training Data Collection for Anomaly Detection: Collect labeled data that includes both normal and anomalous RPL behavior to train the model.

Model Training for Anomaly Detection: Train the neural network on this comprehensive dataset to distinguish between normal and anomalous behaviors.

Anomaly Detection: During real-time routing, use the trained model to predict the likelihood of an RPL attack based on the input features. If an anomaly is detected, adjust the routing decision accordingly.

$$\text{Anomaly Score} = NN_{anomaly}(X)$$

If the anomaly score exceeds a certain threshold, it indicates a potential RPL attack, prompting the routing algorithm to avoid the suspicious route. By integrating Neural Networks for both trust evaluation and anomaly detection, the IoT network can achieve robust security against RPL attacks while ensuring efficient and trustworthy routing decisions. This approach leverages neural networks' deep learning capabilities to assess trustworthiness and make informed routing decisions in dynamic IoT environments.

3.3. Routing Decision Module

The Routing Decision Module is a vital part of the proposed trust-based model for secure routing in IoT networks. It utilizes the trust scores and anomaly detection results to make informed routing decisions, ensuring that data packets are transmitted through trustworthy and reliable nodes, thus avoiding nodes identified as malicious or suspicious. The primary objective of the Routing Decision Module is to enhance the security and efficiency of the IoT network by leveraging trust scores and anomaly detection outcomes. The module dynamically selects the most secure and efficient routing paths based on the latest trust metrics and anomaly detection results. Key Objectives are,

- *Secure Routing:* Prioritize routes through nodes with high trust scores and avoid nodes with low trust scores or identified anomalies.
- *Energy Efficiency:* Consider the energy levels of nodes to prolong the network's operational lifetime.
- *Throughput Optimization:* Ensure high data throughput by selecting optimal routing paths.
- *Minimizing Routing Overhead:* Reduce the control message overhead to maintain network efficiency.
- *Packet Loss Reduction:* Minimize packet loss by avoiding unreliable or compromised nodes.

3.3.1. Secure Routing Path Selection

The Routing Decision Module uses the trust scores provided by the Trust Computation Module to select secure routing paths. Nodes with higher trust scores are preferred in the routing decisions, while nodes with lower trust scores or detected anomalies are avoided.

Routing Algorithm:

- *Trust Score Evaluation:* Evaluate the trust scores of all neighboring nodes.
- *Path Selection Criteria:* Consider multiple criteria such as trust score, energy level, and historical performance.
- *Optimal Path Selection:* Select the path that maximizes security and efficiency, using a weighted combination of the criteria.

3.3.2. Real-Time Updates

The Routing Decision Module continuously updates the routing paths based on real-time data. As trust scores and anomaly detection results are dynamically updated, the routing decisions are adjusted accordingly to respond to emerging threats and changing network conditions.

Dynamic Adjustments:

- ✓ *Trust Score Updates:* Adjust routing decisions based on the latest trust scores from the Trust Computation Module.
- ✓ *Anomaly Alerts:* Immediately avoid nodes flagged by the Anomaly Detection Module.
- ✓ *Energy Levels:* Monitor and factor in the remaining energy levels of nodes to prevent network partitioning.

3.4 Data Collection and Preprocessing Module

The Data Collection and Preprocessing Module are foundational to the proposed trust-based model for secure routing in IoT networks. It is responsible for gathering data from the network, cleaning and processing this data, and then extracting meaningful features that are used by other modules, such as Trust Computation, Anomaly Detection and Routing Decision Modules.

3.4.1. Data Collection

The Data Collection process involves gathering various types of data from IoT nodes and network interactions. This data includes, but is not limited to, packet delivery records, acknowledgment receptions, energy levels, transmission delays and interaction patterns. The data is collected continuously and in real-time to ensure up-to-date information for decision-making. The Key Data Types are,

- ✓ *Packet Delivery Records:* Information on successful and failed packet transmissions.
- ✓ *Acknowledgment Receptions:* Records of acknowledgments received from destination nodes.
- ✓ *Energy Levels:* Current energy levels of the IoT nodes.
- ✓ *Transmission Delays:* Time delays observed during data transmission.
- ✓ *Interaction Patterns:* Historical data on node interactions.

3.4.2 Data Preprocessing

Once the data is collected, it must be preprocessed to ensure it is suitable for analysis by the Trust Computation and Anomaly Detection Modules. Preprocessing steps include data cleaning, feature extraction and normalization.

- Data Cleaning:
  - *Noise Removal:* Eliminate irrelevant or erroneous data points that could distort analysis.
  - *Missing Values Handling:* Impute or remove missing values to maintain dataset integrity.
- Feature Extraction:
  - *Relevant Feature Identification:* Identify features that are indicative of node behavior and network performance.
  - *Feature Creation:* Derive new features from existing data that can help in anomaly detection and trust computation.
- Data Normalization:
  - *Scaling:* Scale features to a similar range to ensure uniformity and improve the performance of machine learning algorithms.
  - *Encoding:* Encode categorical variables into numerical formats if needed.

The trust scores calculated by the Trust Computation Module are continuously updated based on the outputs of the Anomaly Detection Module. Nodes identified as malicious or showing anomalous behavior are assigned lower trust scores. The Routing Decision Module uses these trust scores to avoid compromised nodes and ensure secure routing paths. By combining trust computation with machine learning-based anomaly detection, the proposed model enhances the security and reliability of routing in IoT networks against RPL attacks. The integration of multiple machine learning algorithms allows for robust and adaptive detection of various attack vectors, ensuring comprehensive protection.

IV.RESULTS AND DISCUSSIONS

4.1. Experimental Setup

The simulation experiments for evaluating the proposed trust-based secure routing model in IoT were conducted using the Network Simulator 3 (NS3). NS3 is a discrete-event network simulator highly suitable for simulating complex network behaviors and performance metrics in IoT environments. The simulation setup includes a variety of parameters to model the IoT network and evaluate the performance of the proposed model under different conditions. The parameters used in the simulation are detailed in the table 1 below:

Table 1: Simulation Environment

Parameter	Values
Coverage area	800m × 800m
Simulation time	450 sec
Number of nodes	50, 100,150,200 and 250
Traffic type	UDP-CBR
Transmission range	400m
Packet size	2KB
Maximum speed	25 m/s
Mobility model	Random Waypoint

The simulated network covers an area of 800 meters by 800 meters, providing ample(plenty) space to model the movement and interactions of nodes within a typical IoT deployment. Each simulation run lasts for 500 seconds, ensuring sufficient time to observe the network's performance and the routing protocol's effectiveness under various conditions. The experiments were conducted with three different scenarios, varying the number of nodes to 50, 100, 150,200 and 250. This helps in understanding the scalability and performance of the proposed routing model across different network densities. The traffic type used in the simulations is UDP with Constant Bit Rate (CBR) to model continuous data transmission, which is common in many IoT applications. Each node has a transmission range of 400 meters, ensuring that nodes can communicate with multiple neighbors and form a robust network topology. The packet size for data transmission is set to 2KB, representing typical data packets in IoT communications. Nodes in the simulation can move at a maximum speed of 25 meters per second, simulating the mobility scenarios that might be encountered in dynamic IoT environments. The Random Waypoint mobility model is used to simulate the movement of nodes within the network. This model helps in creating realistic scenarios where nodes move randomly, pause for a while, and then continue to move to another random location, thereby simulating typical IoT node mobility patterns.

4.2 Experimental Results

Node Reputation

Node reputation is a metric used to evaluate the trustworthiness of nodes within the IoT networks. Each node is assigned a reputation score based on its past behavior, such as successful packet deliveries, adherence to routing protocols, and participation in network maintenance. Higher scores indicate more trustworthy nodes, while lower scores suggest potential malicious or unreliable behavior.

$$Reputation_i = \frac{\sum_{t=1}^{T} Behavior_{i,t}\ X\ Weight_t}{\sum_{t=1}^{T} Weight_t}$$

Where, $Reputation_i$ is the Reputation Score of node i. $Behavior_{i,t}$ is the observed behavior metric of node i at time t. $Weight_t$ is the weight assigned to the behavior at time t. T is the total number of observations.

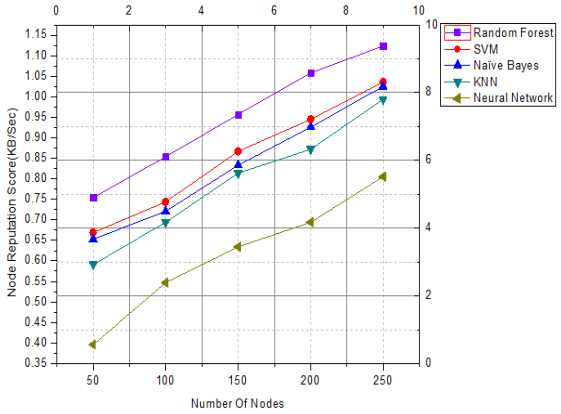

Figure 3: Comparison of Node Reputation Vs Number of Nodes

Random Forest can effectively leverage multiple trust metrics to predict node reputation accurately. Its ensemble nature allows it to capture complex patterns in the data. Computationally intensive, less interpretable compared to simpler models. SVM is effective for binary classification tasks. For node reputation, it can classify nodes into trusted and untrusted categories based on their features, but may struggle with multiclass scenarios. It requires careful tuning of hyper parameters, computationally expensive for large datasets and also not very interpretable. Naive Bayes can quickly estimate node reputation based on probabilistic inference, but its assumptions may limit its accuracy compared to more complex models. Assumes independence among features, which is often not the case in real-world data. KNN can classify node reputation based on the similarity to known examples, but its performance degrades with large or high-dimensional datasets. *Computationally expensive during prediction, sensitive to the choice of k and distance metric, can struggle with high-dimensional data.*

Figure 3 shown, Neural Networks produce more trustworthy nodes compare with other algorithms. Neural Networks excel in capturing nonlinear relationships among features, making them well-suited for predicting node reputation where complex interactions between trust metrics exist. *It is also capable of learning complex patterns and representations, highly flexible, scalable to large datasets.* Neural Networks can automatically learn feature representations from raw data, which can be particularly useful for identifying subtle indicators of trustworthiness that simpler models might miss.

Anomaly Detection Metrics

Anomaly detection metrics evaluate the effectiveness of different algorithms in identifying unusual or malicious activities in the network. Metrics such as precision, recall, and F1-score are typically used to compare the performance of these algorithms.

- Precision: Precision=TP / TP+FP
- Recall: Recall=TP/ TP+FN
- F1-Score: F1= 2× (Precision×Recall)/ (Precision+Recall)

Where:

- TP is the number of true positives.
- FP is the number of false positives.
- FN is the number of false negatives.

SVM evaluates trust by finding the optimal hyper plane that separates trustworthy nodes from untrustworthy ones in a high-dimensional space. For detecting sinkhole and blackhole attacks, SVM classifies nodes based on communication reliability and packet loss. Selective forwarding attacks are identified by analyzing the deviation in packet delivery rates. For Sybil attacks, SVM detects inconsistencies in node identity patterns. SVM's high precision and effectiveness in high-dimensional spaces are advantageous, but it requires careful parameter tuning and is computationally expensive for large datasets.

KNN evaluates trust by considering the behavior of k-nearest neighbors in the feature space. Nodes involved in sinkhole and blackhole attacks can be identified by their anomalous packet reception and dropping patterns compared to their neighbors. For selective forwarding attacks, KNN detects deviations in packet forwarding behavior. Sybil attacks are identified by clustering similar identities and recognizing outliers. While KNN is simple and intuitive, it is computationally expensive during prediction and sensitive to noisy data.

Naive Bayes calculates the posterior probability of a node being trustworthy based on observed features, assuming feature independence. For sinkhole and blackhole attacks, it assesses the likelihood of nodes dropping packets given their claimed metrics. Selective forwarding attacks are detected by the probability of inconsistent packet forwarding. Sybil attacks are identified by analyzing the probabilities of multiple identities originating from a single node. Naive Bayes is fast and interpretable but relies on the assumption of feature independence, which may not hold true in complex scenarios. Random Forest uses an ensemble of decision trees to evaluate the trustworthiness of nodes based on features such as packet delivery ratio, node reputation, and historical behavior. For each RPL attack, the Random Forest model aggregates decisions from multiple trees to classify nodes as trustworthy or untrustworthy. This ensemble approach helps reduce false positives and negatives, making it effective in identifying various anomalies. The robustness to over fitting and handling of large datasets make Random Forest a reliable choice, although it is computationally intensive. Neural Networks evaluate trust by learning complex patterns through multiple layers of neurons. For sinkhole and blackhole attacks, neural networks identify patterns of high traffic attraction followed by packet drops. Selective forwarding attacks are detected by recognizing non-linear patterns in packet delivery inconsistencies. Sybil attacks are identified through the network's ability to detect abnormal identity behaviors and communication patterns. Neural Networks provide the best results for anomaly detection due to their capability to capture complex relationships and subtle anomalies, although they require significant computational resources and training data.

Random Forest is known for its high precision due to ensemble averaging, which reduces false positives However, *while Random Forest is robust to over fitting and handles large datasets well, it is computationally intensive and less interpretable. SVMs are effective in high-dimensional spaces and robust against over fitting, but they demand careful parameter tuning and are computationally expensive for large datasets.* Naive Bayes is simple, fast, and interpretable, working well with small datasets. *However, its assumption of feature independence is rarely true in real-world scenarios.* KNN is simple, intuitive, and requires no training phase, but *it is computationally expensive during prediction and sensitive to noise and outliers.*

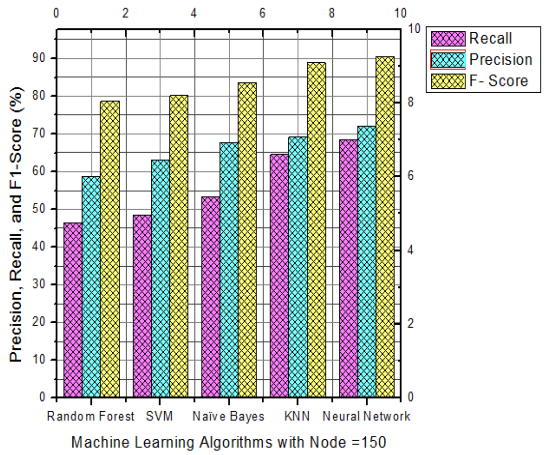

Figure 4: Comparison of Precision, Recall, and F1-Score Vs Node =150

Figure 4 shown, Neural Networks produce higher percentage of Precision, Recall, and F1-Score compare with other algorithms. Neural Networks excel in learning and recognizing intricate patterns, handling high-dimensional data, and automatically refining features from raw data. They scale well with large datasets, ensuring that the model can learn from extensive historical data to improve anomaly detection accuracy. The F1-score is typically the highest among these models, effectively balancing precision and recall. Furthermore, Neural Networks continuously learn and adapt to new data, maintaining high performance as network conditions and behavior patterns evolve. They are also robust to noisy data, which is common in IoT networks, ensuring reliable anomaly detection even in the presence of unreliable or fluctuating data.

Computation Cost

Computation cost refers to the *amount of computational resources required to execute the trust-based secure routing algorithms in an IoT network*. This includes the processing time, memory usage, and energy consumption associated with executing the machine learning models and making routing decisions. Minimizing computation cost is crucial for IoT devices, which often have limited computational capabilities and power resources. Computation cost can be quantified in various ways, depending on the specific resources being measured. The general approach involves measuring the computational resources consumed during the execution of the algorithm. Common metrics include CPU cycles, memory usage, and energy consumption.

- CPU Cycles: $Computation\ Cost_{CPU} = \sum^{i=1}{}_{n} CPU\ Cycles_i$, Where, CPU $Cycles_i$ is the number of CPU cycles consumed by the i-th operation in the algorithm. n is the total number of operations.
- Memory Usage: $Computation\ Cost_{Memory} = max^{i=1}{}_{n} Memory\ Usage_i$, Where, Memory $Usage_i$ is the memory consumed by the i-th operation in the algorithm.n is the total number of operations.
- Energy Consumption: $Computation\ Cost_{Energy} = \sum^{i=1}{}_{n} Energy\ Consumption_i$, Where, Energy $Consumption_i$ is the amount of energy consumed by the i^{th} operation in the algorithm.n is the total number of operations.
- Execution Time: Computation $Cost_{Time} = \sum^{i=1}{}_{n}$ Execution $Time_i$, where, Execution $Time_i$ is the time taken to complete the i^{th} operation in the algorithm. n is the total number of operations.

In practice, computation cost is often a combination of these factors. A comprehensive metric might look like this:

$$\text{Computation Cost}_{Total} = w_1 \cdot \text{Computation Cost}_{CPU} + w_2 \cdot \text{Computation Cost}_{Memory} + w_3 \cdot \text{Computation Cost}_{Energy} + w_4 \cdot \text{Computation Cost}_{Time}$$

Where w_1, w_2, w_3, w_4 are weighting factors that reflect the relative importance of each type of resource consumption in the specific context of the IoT application. By evaluating and optimizing the computation cost, we can ensure that the trust-based secure routing algorithms are efficient and suitable for deployment in resource-constrained IoT environments. Random Forest involves parallelizable nature of tree construction mitigates some of the computational burden, but overall, the resource demands can be substantial, especially for large datasets. Linear SVMs are more computationally efficient, the need for careful parameter tuning and potential scalability issues make SVMs less favorable in large-scale IoT environments. Naive Bayes are suitable for environments with limited computational resources. However, its simplicity and. efficiency come at the cost of lower accuracy compared to more complex models. KNN degrades with high-dimensional data due to the curse of dimensionality, further increasing computational costs.

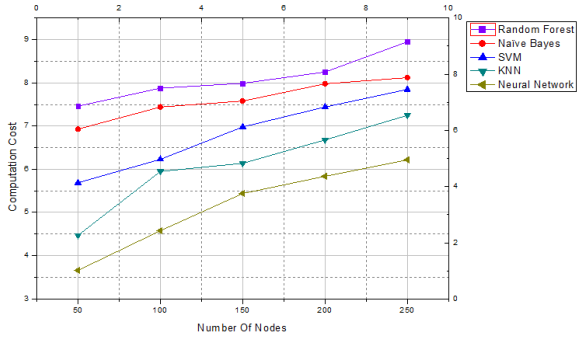

Figure 5: Comparison of Computation Cost Vs Number of Nodes

Figure 5 shown, Neural Networks produce lowest computational cost compare with other algorithms. Neural Networks traditionally require substantial computational resources for both training and inference due to their complex architectures and large parameter spaces. Training involves numerous matrix multiplications and nonlinear operations, often requiring specialized hardware such as GPUs. However, advances in deep learning frameworks and hardware acceleration have significantly optimized these processes. Despite high initial training costs, once trained, Neural Networks can perform predictions very efficiently. Techniques like model compression, pruning, and quantization further reduce the inference computational cost, making Neural Networks more viable for deployment in resource-constrained IoT environments.

Control Packet Transmission

The number of control packets sent is a vital performance metric for evaluating the efficiency and bandwidth usage of a routing protocol in IoT networks. This metric measures the amount of data dedicated to control packets transmitted over the network in a given time period. Understanding this metric helps in optimizing the protocol to reduce overhead and improve network performance. Total Control Packet Transmission metric measures the total size of all control packets transmitted during the simulation or operational period.

$$\text{Total Control Packet Data Sent (KB)} = \sum \frac{\text{Size of Each Control Packet (Bytes)}}{1024}$$

Control Packet Transmission Rate (KB/s) metric indicates the average rate at which control packet data is transmitted in the network. It is calculated by dividing the total control packet data sent by the total simulation time, providing insights into the bandwidth consumed by control messages.

Control Packet Transmission Rate (KB/s) = Total Control Packet Data Sent (KB) / Total Simulation Time (s)

The number of control packets Transmission in kilobytes per second is a vital metric for understanding the control traffic overhead in IoT networks. By monitoring this metric, network administrators can optimize the routing protocol to ensure efficient bandwidth usage, thereby enhancing the overall performance and reliability of the network.

Random Forest increases the overhead in IoT networks, making it less efficient in terms of control packets sent. SVMs can achieve high precision, the computational complexity and the frequent updates can increase the number of control packets Transmission. Naive Bayes classifiers independence assumption can sometimes lead to less accurate trust evaluations, potentially compromising routing decisions and security. KNN method can generate a substantial number of control packets, especially in dynamic IoT environments, as the distance metrics and neighbor information must be frequently updated and transmitted across the network. This makes KNN less efficient in terms of control packet overhead.

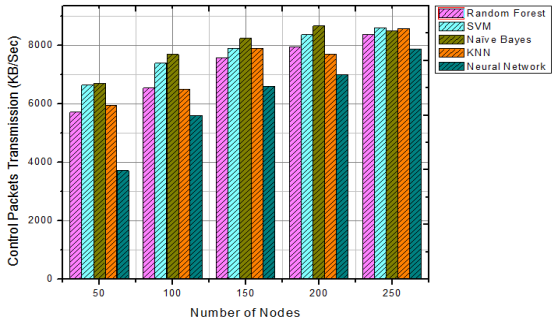

Figure 6: Comparison of Control Packets Transmission Vs Number of Nodes

Figure 6 shown, Neural Networks produce best control packets sent compare with other algorithms. Neural Networks, particularly deep learning models, excel in handling large, complex datasets and can capture intricate patterns in data. They offer significant advantages in terms of flexibility and adaptability, which are critical for dynamic IoT environments. Neural Networks can achieve high accuracy in trust evaluation and anomaly detection with fewer control packet transmissions. This efficiency stems from their ability to learn complex representations and generalize

from them, reducing the need for frequent updates and retransmissions of control information.

Routing Overhead

Routing overhead measures the extra communication burden introduced by the trust-based routing model. This includes additional control packets for establishing and maintaining trusted routes, compared to traditional routing protocols.

Routing Overhead= (Total Control Packets) / (Total Data Packets)

Where, Total Control Packets is the number of control packets (e.g., route requests, replies, and maintenance messages) transmitted. Total Data Packets is the number of data packets transmitted.

Random Forest uses an ensemble of decision trees to make routing decisions. The complexity of maintaining multiple decision trees can result in higher computational and communication costs. SVMs can contribute to higher routing overhead due to the need for extensive parameter tuning and the computationally intensive process of finding the optimal hyper plane for classification. Naive Bayes assumption of feature independence may lead to less accurate trust evaluations, potentially increasing the need for additional communication to correct misrouted packets. K-Nearest Neighbors (KNN) involves significant computation at the time of prediction, as it must calculate distances to all training samples to classify a new instance. This can result in high routing overhead in large-scale IoT networks, particularly when frequent updates to the routing table are necessary.

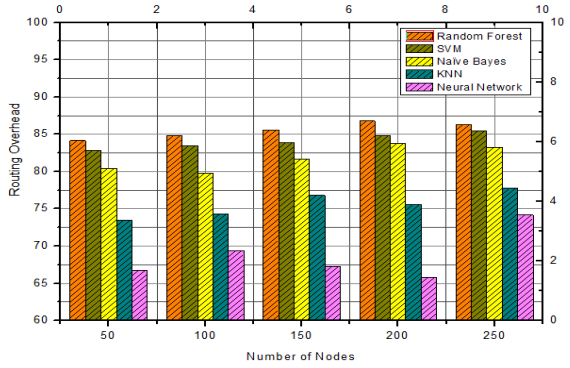

Figure 7: Comparison of Routing Overhead Vs Number of Nodes

Figure 7 shown, Neural Networks produce lowest routing overhead compare with other algorithms. Neural Networks, particularly deep learning models, are more complex and require substantial computational resources for training. However, once trained, they can perform predictions rapidly. *Neural Networks excel in reducing routing overhead* by leveraging their ability to generalize from training data and make accurate routing decisions without the need for frequent updates. Neural Networks can learn and generalize from historical data, enabling them to make accurate predictions with fewer updates. This reduces the communication overhead associated with propagating routing updates throughout the network. Neural Networks can adapt to changing network conditions and learn from new data, which helps in maintaining an up-to-date and efficient routing strategy with minimal additional overhead. Their ability to filter out noise and irrelevant information ensures that only the most pertinent routing information is propagated. Advances in hardware acceleration (e.g., Graphics processing units (GPUs) and tensor processing units (TPUs)) and efficient neural network architectures allow for optimized resource usage, reducing the overall overhead involved in processing and communication.

Throughput

Throughput measures the rate of successful data delivery over the network. It is the amount of data successfully received at the destination per unit of time.

Throughput = Total Data Received (bits)/ Total Time (seconds)

Where, Total Data Received is the total amount of data successfully received in bits. Total Time is the total time over which the data was received in seconds.

Random Forest algorithms can improve the accuracy of routing decisions, the process of maintaining and aggregating multiple trees can introduce latency and reduce throughput, especially in large-scale IoT networks. The computational intensity involved in training and predicting with SVMs can slow down the decision-making process, which may negatively impact throughput. The need for extensive computation to find the optimal hyper plane can introduce delays. Naive Bayes is the independence assumption between features might lead to suboptimal routing decisions in complex scenarios, potentially impacting throughput. KNN overhead can significantly impact throughput, especially in large

and dynamic IoT networks where routing decisions need to be made quickly and frequently.

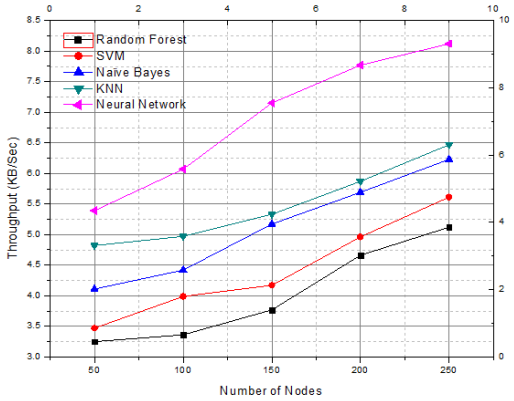

Figure 8: Comparison of Throughput (KB/Sec) Vs Number of Nodes

Figure 8 shown, Neural Networks produce highest throughput compare with other algorithms. In Neural Networks leverage parallel processing capabilities of modern hardware (such as GPUs and TPUs), allowing for rapid data processing and quick decision-making, which enhances throughput. Neural Networks can adapt to changing network conditions in real-time, providing accurate and timely routing decisions that maintain high throughput even as network conditions evolve. Advances in neural network architectures and optimization techniques (e.g., quantization, pruning) have significantly improved inference speeds, making neural networks highly efficient for real-time applications and ensuring high throughput.

Packet Loss Rate

Packet loss rate quantifies the number of packets that fail to reach their destination. It is an important metric for evaluating the reliability of the network.

Packet Loss Rate = (Total Packets Sent - Total Packets Received) \ Total Packets Sent

Where, Total Packets Sent is the number of packets sent from the source. Total Packets Received is the number of packets successfully received at the destination.

Packet Loss Level (%)=(1−Total Packet Sent (bits)/ Total Packet Received (bits))×100

A lower packet loss rate indicates more reliable and efficient communication. In the context of IoT networks, minimizing packet loss is essential for ensuring data integrity and maintaining the overall performance of the network. Random Forest is complexity of maintaining multiple trees can lead to delays and potential packet loss, especially in dynamic and large-scale IoT networks. SVMs are high-dimensional spaces, can introduce latency in decision-making, which may contribute to higher packet loss rates. Naive Bayes classifiers are complex IoT environments, leading to increased packet loss under certain conditions. KNN computational overhead can increase the time taken to make routing decisions, resulting in higher packet loss rates in dynamic IoT networks.

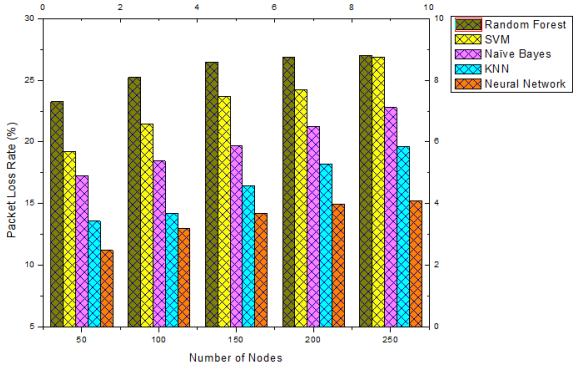

Figure 9: Comparison of Packet loss rate Vs Number of Nodes

Figure 9 shown, Neural Networks produce lowest packet loss rate compare with other algorithms. Neural Networks, particularly deep learning models, are adept at capturing complex, non-linear relationships in the data. Despite the higher computational cost during training, they offer rapid and efficient inference once deployed, which can significantly reduce packet loss rates. *Neural Networks can continuously learn and adapt to changing network conditions, allowing them to make more accurate routing decisions* that minimize packet loss. Neural Networks excel at recognizing complex and non-linear patterns in data, which enables them to predict and avoid unreliable routes that might lead to packet loss. The deployment of Neural Networks on advanced hardware (such as GPUs and TPUs) facilitates real-time data processing and rapid decision-making, reducing the likelihood of packet loss due to delayed routing decisions.

V. CONCLUSION

In this paper, we have presented a comprehensive framework for enhancing trust-based secure routing in IoT networks by integrating various machine learning

algorithms, including Random Forest, Support Vector Machine (SVM), Naive Bayes, K-Nearest Neighbors (KNN) and Neural Networks. Our approach addresses the critical challenges of trust evaluation, anomaly detection and adaptive routing in dynamic IoT environments. Through detailed descriptions of the proposed model's components, we have demonstrated how each machine learning algorithm can be effectively applied to different aspects of the routing process. We highlighted the importance of feature selection and extraction, the collection and utilization of training data, model training, trust evaluation, adaptive learning and integration with trust-based routing policies. To validate in this proposed model, conducted extensive simulations using NS3, setting up an experimental environment with various parameters such as coverage area, simulation time, number of nodes, traffic type, transmission range, packet size, maximum speed, routing protocol and mobility model. Our performance evaluation included metrics such as node reputation, anomaly detection, routing overhead, control packets sent, throughput, packet loss rate and data loss level. The results of our simulations indicate that the integration of machine learning techniques significantly enhances the security, reliability, and efficiency of IoT networks. Each algorithm showed strengths in specific areas, contributing to a more robust and adaptive routing system capable of dynamically adjusting to changing network conditions and mitigating security threats. Overall, this work contributes to the advancement of trust-based secure routing systems in IoT by leveraging the power of machine learning. By integrating these advanced techniques into IoT routing architectures and also in this research work aim to foster (promote) the widespread adoption of IoT technologies across various application domains, ensuring secure and reliable communication in the face of evolving challenges and threats. Future work will focus on further optimizing these models and exploring their application in real-world IoT deployments to validate their effectiveness and scalability.

REFERENCE

[1] A. Pathak and S. Nepal, "Trust-based Routing Protocols for Internet of Things: A Survey," in IEEE Internet of Things Journal, vol. 7, no. 12, pp. 11164-11180, Dec. 2020. doi: 10.1109/JIOT.2020.3012045.

[2] Z. Shelby et al., "RFC 6550: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," Internet Engineering Task Force (IETF), Mar. 2012.

[3] A. Khurma, B. Shetty, S. Gajendran, T. A. Gonsalves and D. K. Sharma, "Survey on Routing Attacks in IoT: Detection, Prevention, and Mitigation," in IEEE Internet of Things Journal, vol. 8, no. 3, pp. 1591-1613, Feb. 2021. doi: 10.1109/JIOT.2020.3028213.

[4] F. Shaikh, A. Rodrigues, A. Belgaumkar, B. Shetty and S. Rodrigues, "A Comprehensive Study on RPL Routing Attacks in IoT: Taxonomy, Analysis, and Research Directions," in IEEE Access, vol. 9, pp. 11838-11858, 2021. doi: 10.1109/ACCESS.2021.3050042.

[5] F. A. B. Alsaadi and J. P. Walters, "RPL Routing Attacks in the Internet of Things: A Review," in 2020 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), Calgary, AB, Canada, 2020, pp. 238-243. doi: 10.1109/iThings/GreenCom/CPSCom/SmartData/Cybermatics49394.2020.00052.

[6] Momani, M., & Challa, S. (2010). Survey of trust models in different network domains. *International Journal of Ad Hoc, Sensor & Ubiquitous Computing*, 1(3), 1-19.

[7] Bao, F., & Chen, I. R. (2012). Trust management for the internet of things and its application to service composition. *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 1-6.

[8] Raza, S., Wallgren, L., & Voigt, T. (2013), “SVELTE: Real-time intrusion detection in the Internet of Things”. *Ad hoc Networks*, 11(8), 2661-2674.

[9] Zhang, Y., Xiao, Y., & Ness, S. (2018), “A trust-based routing framework for secure and efficient data collection in wireless sensor

networks". *IEEE Transactions on Wireless Communications*, 7(8), 2852-2860.

[10] Othman, S. B., & Mokdad, L. (2013), "Enhancing network security with trust-based and energy-aware AODV protocol in MANET". *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 600-607.

[11] Marchang, N., & Datta, R. (2017). Collaborative techniques for intrusion detection in mobile ad-hoc networks. *Ad Hoc Networks*, 6(4), 508-523.

[12] Sahu, S., & Shah, A. (2018), "Intrusion detection system using random forest algorithm for internet of things". *Procedia Computer Science*, 132, 1053-1063.

[13] Patel, J., & Doshi, M. (2020), "Anomaly detection in IoT network traffic using machine learning algorithms" *International Journal of Computer Applications*, 174(8), 25-30.

[14] Al-Garadi, M. A., Mohamed, A., Al-Ali, A., & Du, X. (2020), "A survey of machine and deep learning methods for internet of things (IoT) security". *IEEE Communications Surveys & Tutorials*, 22(3), 1646-1685.

[15] Krontiris, I., Giannetsos, T., & Dimitriou, T. (2013), "Launching a sinkhole attack in wireless sensor networks; the intruder side". *International Journal of Security and Networks*, 4(3), 145-153.

[16] Choi, H., Zhu, S., & La Porta, T. (2014), "SET: Detecting node clones in sensor networks". *Proceedings of the International Conference on Security and Privacy in Communication Networks*, 341-350.

[17] Mayzaud, A., Badache, N., & Kechar, B. (2016), "A Survey on RPL enhancements: A focus on topology, security and mobility". *Computer Communications*, 120, 10-21.

[18] H. Adil, I. Ghazi, K. Akram and M. Anwar, "Comparative Analysis of Machine Learning Techniques for Trust-Based Routing in IoT Networks," in 2020 International Conference on Communication, Computing and Digital Systems (C-CODE), Riyadh, Saudi Arabia, 2020, pp. 119-124. doi: 10.1109/C-CODE51016.2020.9311402.

[19] S. Alam, M. Shah and M. R. Khan, "Security Requirements and Trust-based Routing Protocols in Internet of Things," in 2020 5th International Conference on Computing, Communication and Security (ICCCS), Patna, India, 2020, pp. 1-6. doi: 10.1109/ICCCS49297.2020.9263527.

[20] T. Jelihovschi, C. S. Souza, A. O. Bueno and A. N. Lisboa, "Machine Learning Algorithms for Intrusion Detection in IoT Networks: Random Forest and SVM," in 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Boca Raton, FL, USA, 2019, pp. 1423-1430. doi: 10.1109/ICMLA.2019.00229.

[21] K. N. Shenoy and D. M. Akbar, "Random Forest Classification for Intrusion Detection System," in 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2020, pp. 1-7. doi: 10.1109/ICCCNT49239.2020.9225629.

[22] N. Jain and M. Tuteja, "An Implementation of Trust Based Routing Using SVM for IoT Devices," in 2020 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), Stuttgart, Germany, 2020, pp. 1-6. doi: 10.1109/ICE/ITMC49287.2020.9198671.

[23] K. Srivastava, D. Akhtar, M. A. Khan and B. K. Singh, "A Comparative Analysis of Machine Learning Algorithms for Intrusion Detection in IoT Network," in 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2020, pp. 114-118. doi:10.1109/CONFLUENCE48919.2020.9058608.

[24] P. S. Kshatriya et al., "Naïve Bayes Classifier based approach for energy efficient and reliable data transmission in IoT," in 2018 3rd International Conference on Communication and Electronics Systems (ICCES), 2018, pp. 1005-1008.

[25] S. Ramkumar et al., "Trust-based secure routing protocol for IoT networks using Naïve Bayes classifier," Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 12, pp. 13891-13903, 2021.

[26] D. V. Thanh et al., "Enhancing Intrusion Detection System for Internet of Things Using K-means Clustering and Naïve Bayes Classifier," IEEE Access, vol. 8, pp. 118467-118476, 2020.

[27] A. S. Surya and T. N. Shanmuganantham, "Trust-based secure routing using K-nearest neighbor algorithm in MANETs," Wireless Personal Communications, vol. 118, no. 1, pp. 225-240, 2021.

ABOUT THE AUTHORS

R.Elango received his M.Phil degree from Thiruvalluvar University, Vellore in the year 2011. He received his MCA degree from Anna University, Chennai in the year 2010. He is pursuing his Ph.D degree (Part Time) at Sri Vijay Vidyalaya College of Arts and Science, Nallampalli, Dharmapuri, Tamil Nadu, India. He is working as a Guest Lecturer in the Department of Computer Science at Government Arts College for Men, Krishnagiri. His current research interest includes Internet of Things, Computer Networks, Cloud Computing and Network Security.

Dr.D.Maruthanayagam received his Ph.D Degree from Manonmaniam Sundaranar University, Tirunelveli in the year 2014. He received his M.Phil Degree from Bharathidasan University, Trichy in the year 2005. He received his M.C.A Degree from Madras University, Chennai in the year 2000. He is working as Dean cum Professor, PG and Research Department of Computer Science, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India. He has above 23 years of experience in academic field. He has published 8 books, more than 65 papers in International Journals and 35 papers in National & International Conferences so far. His areas of interest include Computer Networks, Grid Computing, Cloud Computing and Mobile Computing.