



AIEMLA: artificial intelligence enabled machine learning approach for routing attacks on internet of things

Saurabh Sharma¹ · Vinod Kumar Verma¹

Accepted: 20 April 2021 / Published online: 3 May 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract

The Internet of things (IoT) is emerging as a prime area of research in the modern era. The significance of IoT in the daily life is increasing due to the increase in objects or things connected to the internet. In this paper, routing protocol for low power and lossy networks (RPL) is examined on the Contiki operating system. This paper used RPL attack framework to simulate three RPL attacks, namely hello-flood, decreased-rank and increased-version. These attacks are simulated in a separate and simultaneous manner. The focus remained on the detection of these attacks through artificial neural network (ANN)-based supervised machine learning approach. The accurate detection of the malicious nodes prevents the network from the severe effects of the attack. The accuracy of the proposed model is computed with hold-out approach and tenfold cross-validation technique. The hyperparameters have been optimized through parameter tuning. The model presented in this paper detected the aforesaid attacks simultaneously as well as individually with 100% accuracy. This work also investigated other performance measures like precision, recall, F1-score and Mathews correlation coefficient (MCC).

Keywords IoT · Security · RPL attacks · ANN · MCC · Validation · Accuracy

1 Introduction

The recent advances in IoT led to the development of a huge number of intelligent objects or things linked with each other via internet. These intelligent IoT devices use sensors to gather different types of data. Many devices utilize actuators to perform some action to control the environment. There have been substantial growths in IoT applications adaptable worldwide. IoT has applications in the numerous fields [1, 2] like smart cities and home automation, smart lightning or smart metering,

✉ Vinod Kumar Verma
vinod5881@gmail.com

¹ Sant Longowal Institute of Engineering and Technology, Longowal 148106, Punjab, India

healthcare domain and ambient-assisted living [3]. In addition, IoT has been realized in the areas of industrial automation, traffic monitoring and controlling, smart logistics, environmental monitoring, military target tracking, and surveillance [4]. Wireless sensor networks (WSN) are considered as the backbone of IoT. Many research works [5–10] have been reported in the literature for the improvement of WSN. It is expected that the usage of sensor-based IoT networks and their applications will flourish their horizon in the upcoming era. With this expansion, the security issues for the information breach lead to additional challenges in IoT domain [11]. The prior recognition of a malicious node can indirectly enhance the performance parameters like a lifetime, energy efficiency, accuracy, trustworthiness, availability, stability, scalability, security in the network. On the other hand, machine learning algorithms can be used in various scenarios like identification of voice and images and detection of fraud, etc. The huge amount of data generated by the IoT objects and their diversity add more issues for traditional data processing techniques. Machine learning can be treated as a significant computational paradigm to counter security concerns in IoT networks [60]. The data produced in the IoT networks may be provided as an input to a machine learning model to infer various relationships like behavior, patterns, anomalies, etc., on the consistent pattern, the behavior of the attackers can also be identified with such techniques. This lead to smarter network enabled with automatic recognition and handling of the malicious objects. This research work focus on the detection of three well-known security attacks on RPL through an ANN-based supervised machine learning approach. The main contributions of this paper includes: (i) Examining the proposed supervised machine learning model for detection of three RPL attacks in four scenarios. (ii) The proposed model recognized the fraudulent nodes on three attacks (iii) The accuracy of the model has been enhanced up to the maximum possible value, *i.e.*, 100% with the optimized value of hyperparameters (iv) The accuracy of the model has been authenticated with two validation methods, namely hold-out and tenfold cross-validation method (v) Overfitting of the model has been avoided through tenfold cross-validation (vi) The hyperparameters (namely batch size, number of epochs and the optimizer) have been optimized via parameter tuning.

Rest of the paper is organized as follows: Section 2 outlines the present growth in IoT domain, RPL and machine learning. Section 3 presents the essential concepts of RPL and artificial neural networks (ANN). Section 4 depicts the proposed model, and Sect. 5 describes the simulation scenario. Section 6 illustrates the results, and conclusion is presented in Sect. 7.

2 Related work in IoT, RPL and machine learning

IoT and machine learning are among the most stimulating domains in the present scenario. RPL is used as a key protocol for routing in IoT networks and prone to numerous attacks. Many attack detection and prevention schemes have been

illustrated in the literature. This section classifies the related work into three sub-sections as below:

2.1 Surveys

Many efforts have been reported in context of the IoT security by the researchers. The following works have been presented by the different authors for IoT security. Gaddour et al. [12] illustrated a review on RPL including design objectives and routing specifications of the protocol. The performance has been investigated on energy consumption, latency and loss associated with this protocol. Pongle et al. [13] provided a summary of attacks applicable to RPL along with numerous intrusion detection systems or mitigation schemes. Biljana et al. [14] highlighted the gap among the current smart home utilities and their incorporation to IoT networks. Also, a comprehensive model has been suggested for different smart objects within a cloud-oriented IoT scenario. Mahdavinejad et al. [15] explained the machine learning (ML) algorithms for IoT data analysis. The different ML schemes have been examined for smart cities to resolve data related issues in IoT. The strength and weaknesses of ML schemes for analyzing IoT data have been elucidated. Zarpelao et al. [16] explained a survey on intrusion detection in IoT focusing on key tendencies, open challenges and upcoming research opportunities. Khan et al. [17] demonstrated an analysis on security issues in IoT architecture. The specifications for the security requirements and challenges, safety issues and block chain support have been elaborated. Costa et al. [18] presented an appraisal on machine learning-based IDS approaches for attacks identification in IoT. This review covered a wide range of safety issues in the IoT scenario. Sengupta et al. [19] classified the attacks on the basis of objects vulnerability along with different levels of generalized IoT architecture. Verma et al. [20] presented investigations on the RPL attacks focusing on resources, topology and traffic. Specifically, the countermeasures for RPL-based network layer have been elucidated comprehensively. Avila et al. [21] presented a structured review of different mitigation schemes on RPL attacks. This analysis has focused on supervising networks, parent node selection and authentication. Almusaylim et al. [22] presented a study on RPL in context of resources, topology, traffic and different attacks. Simha et al. [23] focused on RPL and Contiki operating system for investigation over destination-oriented directed acyclic graph (DODAG) construction. The aspects related to suitability of Contiki and Cooja for RPL implementation have been explained.

2.2 Attacks and mitigation schemes

The major efforts by the researchers on attacks and mitigation schemes have been discussed as follows. Wallgren et al. [24] demonstrated various security attacks for mitigation schemes in the RPL based IoT networks. A lightweight heartbeat protocol has also been presented to prevent from different attacks. Sehgal et al. [25] examined the DODAG inconsistency attacks and elaborated its consequences. Mayzaud et al. [26] focused on the DODAG version system and examined the impact of possible attacks. Results revealed that the attacks impact the rise in overhead by 18%,

doubled the latency and reduced the delivery ratio by 30%. Mayzaud et al. [27] worked for the improvement of a mitigation scheme for topological inconsistency attacks. Mayzaud et al. [28] elucidated the nomenclature of RPL attacks and relative investigation for each of the attacks along with remedial measures. Ahmed et al. [29] reported a two-step process for avoiding the blackhole attack. Thanigaivelan et al. [30] presented a summary of a distributed internal anomaly detection scheme for IoT. Also, network fingerprinting has been used by the system for identifying any topological changes. Aris et al. [31] investigated version number attacks in RPL-based IoT networks and focused on static and mobile nodes with different cardinalities. Diro et al. [32] represented the utility of deep learning for attacks identification and compared with traditional schemes. Perazzo et al. [33] examined a wormhole attack in IEEE 802.15.4 based sensor networks to highlight the efficacy of the attack. Jyothisree et al. [34] reported the two specific attacks, namely blackhole and version number. The relative detection and prevention scheme for identifying both types of attacks have been elaborated. Ahutu et al. [35] proposed a media access control (MAC) centralized routing protocol (MCRP) for the detection of the wormhole attack in WSN. Sahay et al. [36] illustrated the increased version attacks using sky motes and observed the average energy consumption by ordinary and fraudulent nodes. Pu et al. [37] presented the analysis on the Sybil attack in RPL-based IoT networks and suggested the remedial measure to avoid the Sybil attack.

2.3 Intrusion detection systems

Several intrusion detection systems (IDS) have been proposed by researchers in the past. Raza et al. [38] presented a novel approach SVELTE to detect instruction detection for RPL attacks. The attacks recognized through these IDS are the spoofing, sinkhole and selective forwarding routing attacks. However, the attacks have not detected 100% accuracy. Enhanced IDS for sensing a denial-of-service attack in 6LoWPAN-based networks was proposed in reference [39, 40]. Kasinathan et al. [39] constructed the IDS through ebbits network model and verified it via a penetration testing system. However, Rghioui et al. [40] provided the significant statistics for problems concerning the kind of detection method, features and deployed agents. Pongle et al. [41] presented a real-time intrusion and wormhole attack detection in IoT. This model sensed the wormhole attack and attacker using the neighbor information of the node. The presented model is energy efficient and uses only a fixed number of UDP packets for recognizing the attack. Sheikhan et al. [42] explained a hybrid IDS for IoT incorporating the MapReduce policy. The anomaly oriented and misuse-based identification has been achieved through mediator agents. The internal attacks are identified by unsupervised optimum path forest (OPF) scheme. Le et al. [43] proposed a specification-based IDS for detecting attacks. This IDS has been influenced by a profiling method generating a high-level abstract of network operations to validate node activities. The simulation result depicted the recognition of the RPL topological attacks with the presented IDS. Jarrah et al. [44] discussed a semi-supervised multilayered clustering model (SMLC) for intrusion detection. The SMLC highlighted the learning from partially labeled data, and its performance

has been compared with tri-training, random forest, bagging and AdaboostM1 models. SMLC outperformed tri-training facilitating equivalent detection accuracy with 80% labeled training data. Lee et al. [45] examined the competence of four types of ANN models and compared these models with the multinomial logit model (MNL) for prediction performance. The prediction accuracy of four ANNs has been evaluated through hold-out and tenfold cross-validation methods along with sensitivity analysis. Anitha et al. [46] examined ANN (artificial neural network)-based intrusion detection system for IoT networks named ANNIDS through multilayer perceptron (MLP). These IDS focused on identifying the DIS attack and the version attack. Bhosale et al. [47] proposed the IDS for recognizing the wormhole attacks in RPL-based IoT using the key parameter RSSI (received signal strength indicator). This mitigation scheme has been used for avoiding the wormhole attack and considered the benefits of centralized and distributed approaches. The attack has been detected with 90% accuracy. Kfoury et al. [48] presented a self-organizing map (SOM)-based IDS for three well-known RPL attacks. SOM has been used as an unsupervised machine learning method in categorical problems. The anticipated IDS categorized the distinct RPL attacks by a graphical map depiction. Sharma et al. [49] simulated four dissimilar RPL attacks for investigating the impact on cyber-physical systems. The offered model predicted and categorized the four attacks with 99.33% accuracy via a random forest classifier. Qureshi et al. [50] presented a new architecture using two segments, namely threshold modulation and attack detection. The deployed architecture identifies four types of RPL attacks successfully. The simulations conducted revealed a higher delivery rate with diminished delay and packet loss.

3 RPL, security attacks and artificial neural networks (ANN)

This section explains the RPL, security attacks and artificial neural networks (ANN) in context of IoT.

3.1 Routing protocol for low power and lossy networks (RPL)

RPL is a key routing protocol generating destination-oriented directed acyclic graphs (DODAG) to skip loops. RPL uses three forms of message transfer labeled as multipoint to point (MP2P), point to multipoint (P2MP), and point to point (P2P), respectively [51]. MP2P supply message transfer from non-root nodes to the root node of the DODAG. P2MP facilitates the transmission of messages from the root node to all other non-root nodes. P2P supports communication among individual nodes. In addition, RPL facilitates four different modes of operation (MOP) called MOP0, MOP 1, MOP 2 and MOP 3. MOP0 uses MP2P communication and MOP1 improves MOP0 by incorporating P2P communication managed through the root node. In this case, the individual root node possesses the routing table. MOP2 improves the MOP1 by allowing the non-root nodes to store routing tables. Also, MOP2 allows P2P message transfer as managed by the non-root node. MOP3 boosts MOP2 by linking P2MP message transfer. RPL possesses four types of control

messages labeled as DODAG information solicitation (DIS), DODAG information object (DIO), DODAG advertisement object (DAO) and DAO acknowledgment (DAO-ACK). DIS message is used for finding the currently available DODAG. DIO message provides the announcement for acquainting other nodes regarding available DODAG. DAO message facilitates for connection with the presently available DODAG. DAO-ACK shows the DAO message indicating the acceptance or rejection of the call for connecting DODAG. An ordinary node decides its parent through an objective function. This objective function analyzes the rank of the competitor neighbor nodes and elects one node as the parent. Various services demand optimization based on their specific requirement, like healthcare services demand minimum delay. Similarly, smart city applications demand minimum power expenditure. Therefore, dissimilar objective functions are required for different circumstances. RPL possesses two standard objective functions labeled as objective function zero (OF0) [52] and minimum rank with hysteresis objective function (MRHOF) [53].

3.2 Security attacks on RPL

Many security attacks may occur in RPL. Three major attacks, namely hello flood, decreased rank and increased version attack have been used in this paper.

3.2.1 Hello flood attack

This attack creates a lot of traffic in the network and causes the unavailability of resources in the worst case. The fraudulent node increases the number of DIS messages to be communicated at multiple places in the network. To achieve this objective, malicious node reduces the period among two sequential DIS messages labeled as `RPL_CONF_DIS_INTERVAL` and `RPL_CONF_DIS_START_DELAY`.

During the solitary DIS message transmission, a bunch of DIS messages are transferred. The fraudulent node transmits numerous DIS messages lead to rise in traffic of the IoT network. Thus, the adjacent nodes getting DIS messages response with the DIO messages. This leads to the transmission of a bunch of DIO messages results in the further rise in traffic. Therefore, a huge share of substantial energy is misused, particularly by the adjacent nodes of the malicious node. Consequently, significant parameters like lifetime, availability, stability, energy efficiency of the network are severely affected.

3.2.2 Increased version attack

The RPL protocol generates DODAGs for skipping loops and DIO control packets. Each DIO control packet has a version number associated with it. This version number is initialized by the root and remains constant throughout the traversal. The root node can alter this version number only when the DODAG is reconstructed. This phenomenon referred as global repair. The fraudulent nodes intentionally modify the version number of the DIO control packet and transfer the modified DIO message to neighbors. On receiving the modified DIO message, the neighbors show

their exclusion in the new DODAG. This causes the needless reconstruction of currently available DODAG. Frequent reconstruction raises the traffic in the network and affects the key parameters of the network like lifetime, availability, energy efficiency, etc.

3.2.3 Decreased rank attack

During DODAGs generation, the RPL protocol assigns rank to each node in the network. The ordinary nodes elect their parent by analyzing the ranks of the competitor neighbor nodes. The fraudulent node announces its forged rank to adjacent neighbors. This led to the election of a fraudulent node as the parent of adjacent neighbor nodes. This makes other nodes to pass their messages through fraudulent node. In RPL, this objective is accomplished by firstly, assigning zero value to two constants labeled as RPL_CONF_MIN_HOPRANKINC and RPL_MAX_RANKINC. Further, the constant INFINITE_RANK is assigned to a relatively low value. For keeping the links with the higher rank nodes, the function call for re-evaluating ranks is skipped. Consequently, the fraudulent node announces its predecessor's rank as its rank to cheat other nodes. This attack raises the traffic only at the fraudulent node and can be used for spying the downward nodes in the DODAG.

3.3 Artificial neural networks (ANN)

The notion of an artificial neural network (ANN) enables us to automate the learning process [54]. ANN represents the working of the human brain and remains essential for a group of processing units. These units collaborate to behave like a human brain where each unit pretends like a biological neuron. The processing units are organized in a sequence of layers to form the ANN capable of learning through instances. This learning process empowers the ANN to predict the output for the corresponding set of inputs. Each ANN must have at least two layers, namely input layer and output layer. Depending on the nature of the problem, one or more hidden layers can also be incorporated between the input and output layer. The number of units in the input layer represents the number of features in the dataset used as independent variables in the ANN. Similarly, the number of units in the output layer denotes the

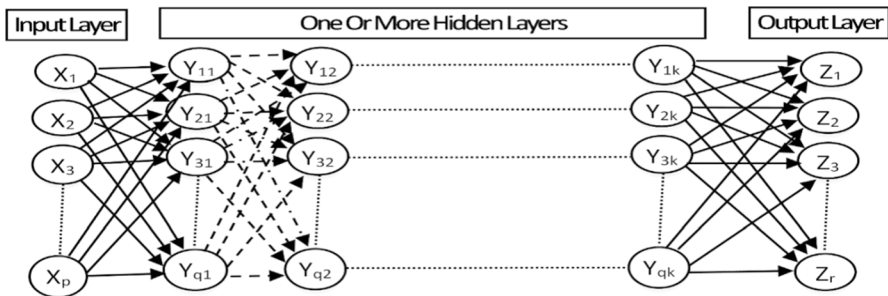


Fig. 1 Basic architecture of artificial neural networks

number of desired categories required to classify the dataset instances. The number of hidden layers and the number of units in each hidden layer can vary from problem to problem. Each link among the units is assigned a specific weight. This weight defines the impact of the source unit. Figure 1 represents the basic architecture of an artificial neural network where p number of input features from a dataset is required to be fed into the ANN to get r number of desired outputs. Let $x_1, x_2, x_3, \dots, x_p$ are the p input features and $w_1, w_2, w_3, \dots, w_p$ are the p number of weights. Here, w_1 is assigned to link with x_1, w_2 is assigned to link with x_2 and so on. The basic architecture of each processing unit is represented in Fig. 2.

Let z be the output and b be the overall bias value. Let f denotes the activation function and $k \in (1, p)$. The output of each unit is represented by Eq. (1).

$$z = f\left(\sum_{k=1}^p w_k x_k + b\right) \tag{1}$$

If the bias value is zero, the output is denoted by Eq. (2)

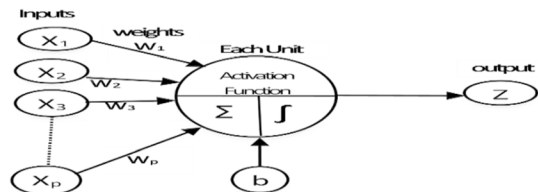
$$z = f\left(\sum_{k=1}^p w_k x_k\right) \tag{2}$$

Inputs and relative outputs are required to train the ANN. The weights are attuned during the training phase to minimize the error. After training, this trained ANN is used further to test the dataset.

4 Proposed ANN-based IDS for RPL attacks in IoT networks

This section describes the proposed ANN-based system to detect three security attacks as shown in Fig. 3. Initially, the Contiki [55] operating system is equipped with the Cooja [56] simulator. Next, the RPL attacks framework [57] is used to implement the three security attacks labeled as hello flood, increased version and decreased rank attack. The power tracker module of the Cooja simulator is used for estimating the energy consumption of all nodes. Each node utilizes power in four ways, *i.e.*, ON (powered on and idle), TX (transmission), RX (reception) and INT (interference). All these four metrics are scaled for the alive state of the node. This is done to compute the percentage energy consumption of each node. The RPL attacks framework produces the DODAG graphs, power consumption graphs and the packet capture files during simulation. All these outputs of the

Fig. 2 Basic architecture of each processing unit in ANN



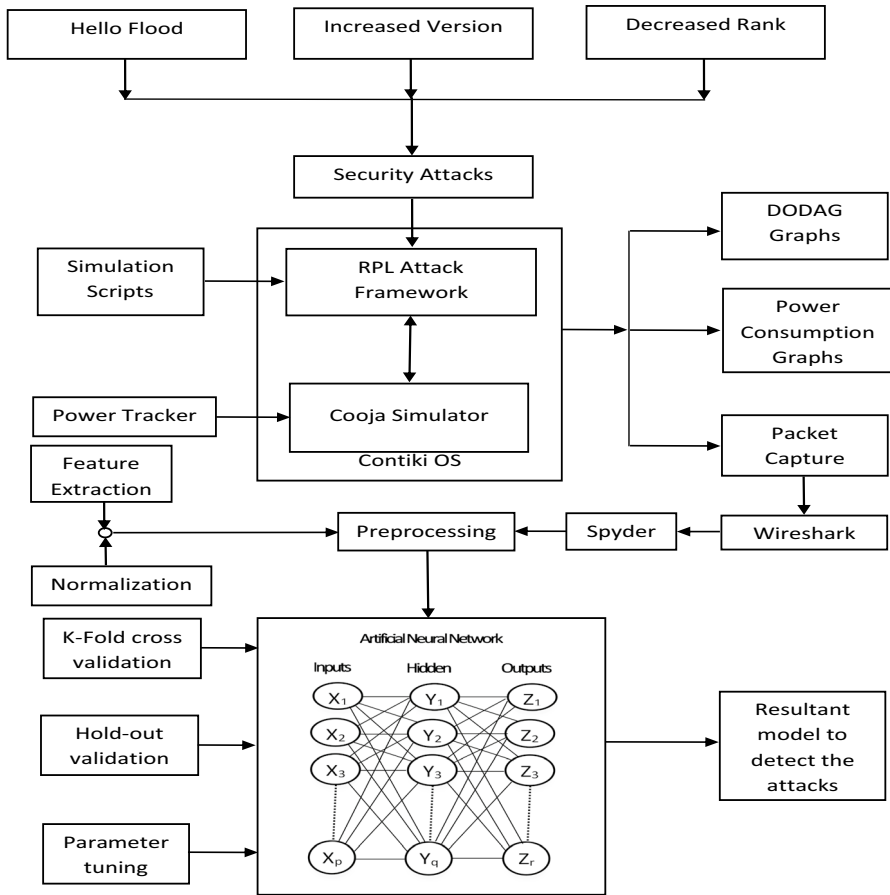


Fig. 3 Proposed system for sensing RPL attacks in IoT networks

RPL attacks framework reflects the behavior of the network in the presence or absence of the attack. Firstly, Cooja simulations are performed for each of the three attacks separately. Next, the simulation is made with all the attacks implemented in parallel. After performing the required simulations, the packet capture files (*pcap*) are converted into the comma-separated values (*csv*) using Wireshark tool. These *csv* files are then fetched as a Python dataframe through the Spyder tool. Afterward, the python data frame is preprocessed for the training and testing phase through the ANN. The preprocessing includes dealing with the missing data, extracting the required features and normalizing the data. Further, the ANN model is trained and tested through the dataset to detect the attacks. The training uses the alteration of weights through back propagation to improve the resultant model. The testing involves the validation of the resultant trained model to detect

the attacks with the appropriate accuracy. The correctness of the anticipated model is evaluated with a hold-out method and a k -fold cross-validation method.

The hyperparameters for ANN learning are optimized through parameter tuning. The ANN used in Fig. 3 assumes p , q and r number of units available in the input, hidden and output layer, respectively. This indicates the fact that the number of units in each layer may be different.

4.1 Algorithm for proposed ANN-based IDS

This algorithm uses RPL attacks framework as an input to perform the Cooja simulations and provides the resultant model as an output to detect the RPL attacks.

Step 1: Contiki operating system downloading and installation.

Step 2: Configure the Cooja simulator and RPL attacks framework within the Contiki operating system.

Step 3: Implement the three security attacks.

Step 4: Develop the simulation scenario and perform the simulations for each attack separately and simultaneously.

Step 5: Examine the power consumption graphs and DODAG graphs to analyze the behavior of the network in the presence of different attacks.

Step 6: Access the packet capture (*pcap*) files through the Wireshark tool and convert the data into comma-separated values (*csv*).

Step 7: Import the comma-separated values (*csv*) file into the Python dataframe through the Spyder tool.

Step 8: Perform preprocessing to deal with the missing values, extracting the required features and normalization.

Step 9: Split the dataset into the training set and testing set.

Step 10: Classify the fraudulent and ordinary packets through the ANN.

Step 11: Generate the resultant model to detect the attacks.

Step 12: Optimize the hyperparameters through parameter tuning.

Step 13: Validate the accuracy of the model through hold-out and k -fold cross-validation methods.

This algorithm is implemented to generate the resultant model predicting the attacks individually and simultaneously. The performance of the model is evaluated in terms of accuracy and loss. The accuracy of the model may be referred as the ratio of the total number of correct predictions to the total number of predictions. Similarly, the loss of the model is calculated through the cross-entropy function. It remained directly proportional to the deviation among the probabilities of predicted and actual values. The higher the deviation larger can be the loss. The aim remains on the better performance of the model through accuracy maximization and loss minimization.

Few other performance measures like precision, recall and f1-score have also been investigated. Let TP, TN, FP and FN represent the true positives, true negatives, false positives and false negatives from the confusion matrix, respectively. Equations (3), (4) and (5) determine the values of *precision*, *recall* and *F1-score*, respectively.

$$\text{Precision} = (TP)/(TP + FP) \quad (3)$$

$$\text{Recall} = (TP)/(TP + FN) \quad (4)$$

$$\text{F1 - score} = 2 \times (\text{Recall} \times \text{Precision}) / (\text{Recall} + \text{Precision}) \quad (5)$$

The Mathews correlation coefficient (MCC) of the model is also computed from the confusion matrix as given by Eq. (6).

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (6)$$

5 Simulation setup

The three well-known security attacks have been implemented using the proposed model. Table 1 illustrates the simulation parameters with focus on four types of simulation scenarios. Three simulation scenarios used three attacks individually and fourth simulation scenario implements the three attacks simultaneously. These scenarios serve as the building blocks incorporating malicious sensors for investigations. All four scenarios incorporate a 200 m × 200 m simulation area with execution time equal to two minutes. In addition, other parameters like interference and transmission range, minimum and maximum distance from the root node, number of normal and root nodes, and number of simulation epochs have been evaluated in these scenarios. The transmission range remained up to 50 m and the interference range lies between 50 to 100 m. The transmission range is depicted by the green circle around the node and the interference range is shown by the gray concentric ring around the green circle. The minimum and maximum distance of a node from the root node is 20 m and 200 m, respectively. Each of the nodes used in the simulation is based on the Zolertia Z1 [58] platform. All the scenarios have one root node and ten normal nodes. Out of the four simulation scenarios, root mote type and sensor mote type are selected as a dummy in scenarios 1 and 2. However, scenario 3 and 4 have used both root mote type and sensor mote type as an echo. Out of quadrants and grid algorithms, the quadrants algorithm has been used for the generation of simulated sensor-based IoT networks. The number of simulation repetitions has been kept as one in all the simulations. In each of the simulations, the root and normal nodes have been represented in green and yellow colors, respectively. The node number (or ID) for the root node in all simulation is zero. However, the normal node identifiers belong to the (1, 10) domain. The malicious node is depicted in pink color and has ID 11. In the first three simulation scenarios, the three attacks were implemented separately. In the fourth scenario, all the three attacks have been implemented simultaneously along with three malicious nodes. The malicious node for decreased rank, increased version and hello flood attack has been shown in pink, light yellow and light blue colors, respectively. The ID for the malicious node corresponding to the

Table 1 Simulation parameters

Simulation parameters	Values in		
	Hello flood attack	Increased version attack	Decreased rank attack
Area	200 m	200 m	200 m
Building block	Hello-flood	Increased-version	Decreased-rank
Time	120 s	120 s	120 s
Transmission range	50 m	50 m	50 m
Interference range	100 m	100 m	100 m
Root minimum distance	20 m	20 m	20 m
Root maximum distance	200 m	200 m	200 m
No. of normal nodes	10	10	10
Target and malicious target	Z1	Z1	Z1
Epochs	1	1	1
WSN generation algorithm	Quadrants	Quadrants	Quadrants
Root mote type	Dummy	Dummy	Echo
Sensor mote type	Dummy	Dummy	Echo
Malicious mote type	Malicious-sensor	Malicious-sensor	Malicious-sensor
No. of malicious nodes	1	1	1
Number of root nodes	1	1	1
			All 3 attacks simultaneously
			200 m
			All 3 building blocks
			120 s
			50 m
			100 m
			20 m
			200 m
			10
			Z1
			1
			Quadrants
			Echo
			Echo
			Malicious-sensor of all three types
			3 (one for each attack)
			1

decreased rank, increased version and hello flood attack remained 11, 12 and 13, respectively.

6 Results and discussions

This section depicts the results and outcomes for the deployed four simulation scenarios. Let the simulation scenario for individual hello flood attack be called scenario 1. Similarly, the simulation scenarios dedicated to increased version and decreased rank attacks are called scenario 2 and scenario 3, respectively. Likewise, the simulation scenario where all the 3 attacks were implemented simultaneously was named as scenario 4. Figure 4a, b, c, d shows the network configuration used in a scenario 1, 2, 3 and 4, respectively. The Cooja simulation generates the DODAG graphs, power expenditure graphs and the pcap files to trace the message transfer. The DODAG graphs generated for each scenario are shown in Fig. 5. Figure 5a, b, c, d represents the DODAG graphs for scenario 1, 2, 3 and 4, respectively. Figure 5

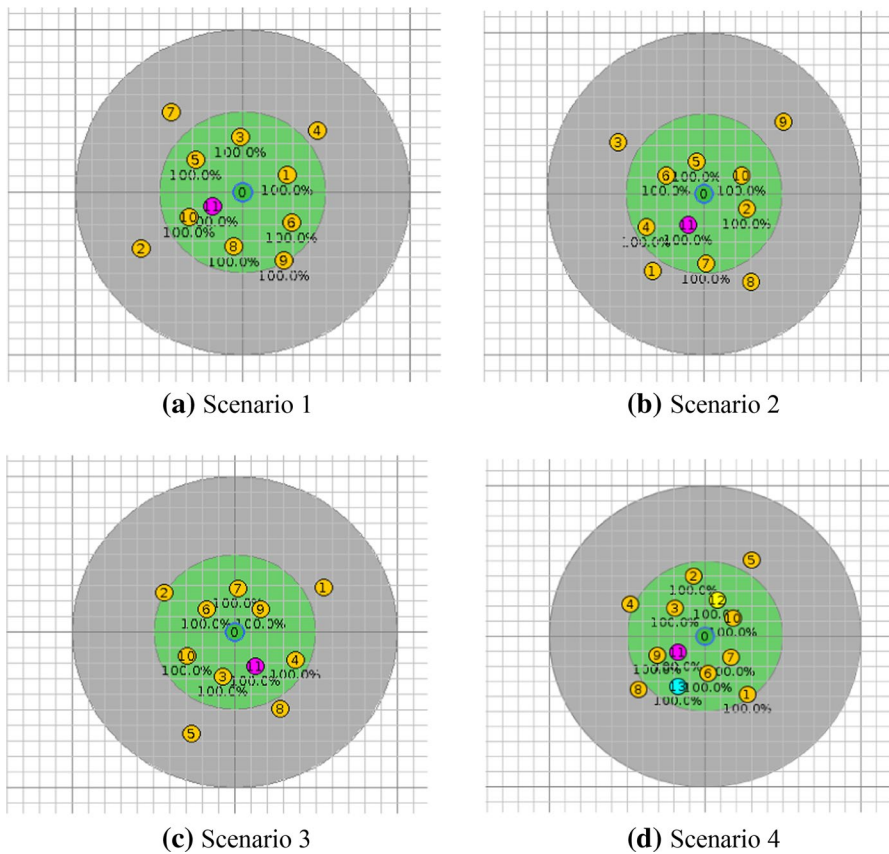


Fig. 4 Network configuration for all the four scenarios

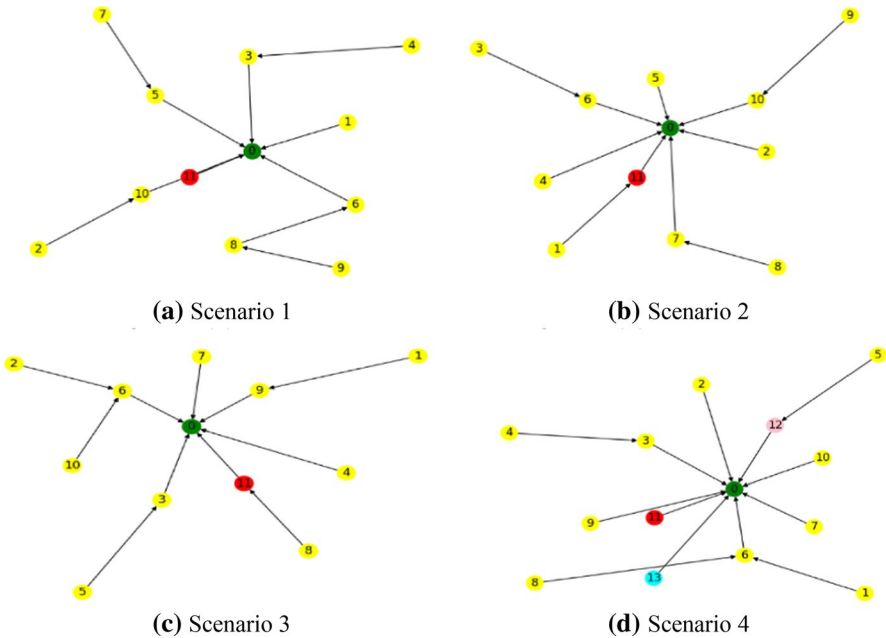


Fig. 5 DODAG graphs generated in each scenario

shows that the DODAG graphs for scenarios 2, 3 and 4 and impact of the malicious nodes in the downward direction. This is due to the downward nodes directing the traffic through the fraudulent node. The fraudulent node 11 affects the DODAG in scenarios 2 and 3. Similarly in scenario 4, the malicious node that affects the DODAG is node 12.

However, the other malicious nodes, namely node 11 and 13, do not affect the DODAG graph in scenario 4. The malicious node in scenario 1 does not affect the DODAG. The power consumption graphs created for each scenario are shown in Fig. 6. Figure 6a, b, c, d denotes the power expenditure per mote occurred in scenario 1, 2, 3 and 4. Here, the energy expenditure is evaluated as INT (interference), ON (powered on and idle), TX (transmission) and RX (reception) times.

Figure 6a shows that the malicious node for hello flood attack consumes the maximum power among all nodes. It increases the power consumption of its adjacent neighbors. Similarly, Fig. 6b shows that the fraudulent node in scenario 2 provokes other nodes in the network to consume more power. However, Fig. 6c reflects that the fraudulent node does not affect the power consumption of any node in the network in scenario 3. Figure 6d shows that the power consumption of all the nodes is affected when all the attacks are implemented simultaneously. The *pcap* file provides the information about the following parameters, namely 'Number', 'Time', 'Source', 'Destination', 'Protocol', 'Length' and 'Info'. The first parameter 'Number' represents the serial number of the packet and used as an index for pre-processing the dataframe. 'Time' and 'Length' parameters are considered as the quantitative independent variables representing the time and length of the packet.

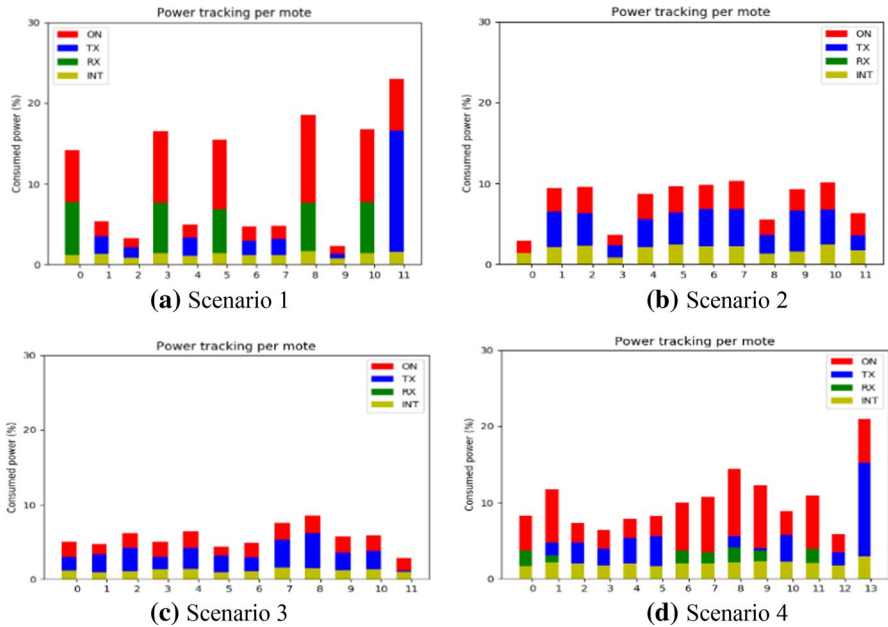
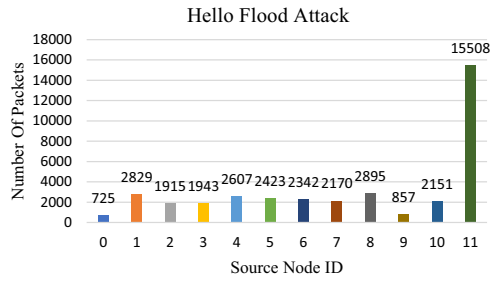


Fig. 6 Power consumption per mote in each scenario

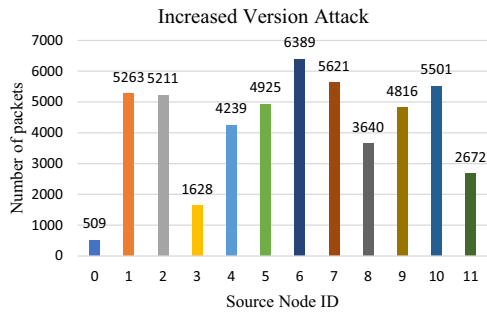
However, the remaining four features are considered as the qualitative independent variables. The ‘Source’ and ‘Destination’ features specify the source and destination IP address, respectively. The ‘Protocol’ column shows the relative protocol and the ‘Info’ parameter represents the metadata. An output-dependent variable is vertically appended to the dataset which classifies the fraudulent and ordinary packets. It is observed that during preprocessing each of the categorical variables possessing c categories is represented with ‘ $c-1$ ’ dummy variables with one exception. This is done to get rid of the dummy variable trap. However, the exceptional case is of the dependent variable in scenario 4. Here, the dependent variable is represented with all ‘ c ’ categories to provide sufficient information for the proposed model. The remaining categories have been detected efficiently, even with the ‘ $c-1$ ’ dummy variables. To allow the proposed model to detect all ‘ c ’ categories optimally in scenario 4, the output dependent variable is represented with ‘ c ’ dummy variables. In each of the scenarios, the *pcap* file is imported into a Python dataframe through Wireshark and Spyder tools. Next, these dataframes have been preprocessed to generate the bar plots. These bar plots illustrate the packet counts generated by each node in different scenarios. Figure 7 represents these bar plots, where Figs. 7a, b, c, d depicts results for scenarios 1, 2, 3 and 4, respectively.

Figure 7 shows that the fraudulent nodes for the hello flood attack generate the maximum number of packets in the network. This is due to the fact that the malicious node generates a bunch of DIS messages instead of one. However, the malicious nodes that belong to the increased version attack incite other nodes to generate more number of packets. This occurs because the deceitful node provokes other

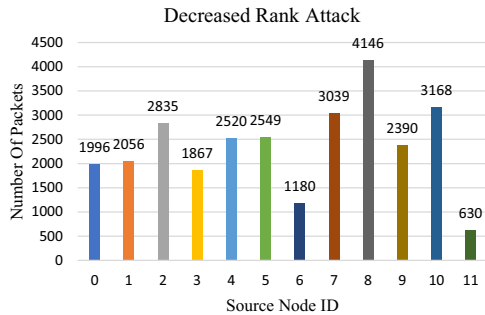
Fig. 7 Bar plots for packet count in each scenario



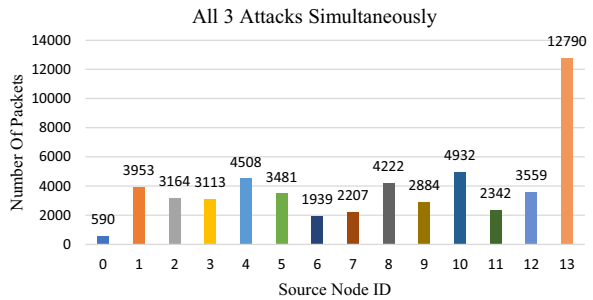
(a) Packet count in scenario 1



(b) Packet count in scenario 2



(c) Packet count in scenario 3



(d) Packet count in scenario 4

nodes to repeatedly invoke the global DODAG repairs. The duplicitous node of a decreased rank attack produces the least number of packets among all the malicious nodes. It is due to the reason that the decreased rank attack does not harm the network. It can be used for spying the communication of downward nodes in the DODAG. Figure 7 shows that 38,365, 50,414, 28,376 and 53,684 packets were generated by all the nodes in scenarios 1, 2, 3 and 4, respectively. It is noticed that the ACK packets present in the packet capture are ignored due to lag of any source or destination information. It is observed from Fig. 7 that in scenarios 1, 2 and 3, the malicious nodes generate 15,508, 2672 and 630 packets respectively. However, in scenario 4, the fraudulent nodes for hello flood, increased version and decreased rank attack generate 12,790, 3559 and 2342 packets individually. Table 2 depicts the packets generated in different scenarios.

The preprocessed data have been fed to the ANN. The proposed ANN used three layers including a hidden layer, an input layer and an output layer. Here, only one hidden layer has been used as the adding of more hidden layers does not upgrade the performance of the model. The output layer has only one unit in the first three scenarios due to the reason that two possible outcomes indicating the fraudulent and ordinary packets. However, in scenario 4, there are four possible resultants. Three of these have been dedicated to each attack and the fourth one corresponds to the absence of the attack. Therefore, the output layer has 4 units in scenario 4. Similarly, the number of units in the input layer for each scenario has been optimized. Like in scenario 1, the 'Protocol' feature has a single possible value. Consequently, it has been dropped from the dataset and only three categorical variables have been left. These features namely: 'Source', 'Destination' and 'Info' have observed 12, 9 and 3 distinct possible values, respectively. Therefore, $12 + 9 + 3 = 24$ dummy variables are required to represent the three aforesaid categorical variables. But to avoid the dummy variable trap, it is required to drop one dummy variable per category. Thus, $11 + 8 + 2 = 21$ dummy variables are required to represent the three aforementioned categorical variables. In addition, there are two numerical variables, namely Time and Length. Consequently, in total, $21 + 2 = 23$ units are required for the input layer of the proposed ANN model. Hence, the input layer consists of 23 units in scenario 1. Likewise, the input layer for scenarios 2, 3 and 4 is optimized to possess 22, 32, and 28 units, respectively. There is no thumb rule to select the optimized number of nodes in the hidden layer. This can also be achieved through parameter tuning. But, this paper has been focused on taking the average sum of the number of units in input and output layers to select the number of nodes in the hidden layer. Thus,

Table 2 Number of packets generated

Scenario	Total packets (except ACK packets)	Malicious packets
1	38,365	15,508
2	50,414	2672
3	28,376	630
4	53,684	$12,790 + 3559 + 2342 = 18,691$

the number of units selected for the hidden layer of the proposed model for each scenario is represented by Eq. (7).

$$\text{No of hidden units} = (\text{Sum of no. of units in the input layer and output layer})/2 \quad (7)$$

Therefore, the hidden layer has $(23 + 1)/2 = 12$ units in scenario 1. Similarly, the hidden layer for scenarios 2, 3 and 4 possess 11, 16 and 15 units, respectively. Table 3 depicts the number of units used in each layer of the proposed model. The activation function used in the output layer of the first three scenarios is 'sigmoid' and the 'softmax' activation function is used in the output layer of scenario 4. The 'relu' activation function is used for the hidden layer in all the scenarios. Similarly, the loss function has been used in the first three scenarios to update the weights is 'binary_crossentropy.' The 'categorical_crossentropy' has been used as a loss function in scenario 4. The performance of the model has been investigated with two types of validation methods, namely hold-out and tenfold cross-validation. The stochastic gradient descent algorithms have been used in both the validation methods 'adam' and 'rmsprop.'

6.1 Hold-out validation

In this method, the total dataset is fragmented into an 80% and 20% ratio. This 80% of the dataset is used to train the ANN. However, the remaining 20% is used for testing the ANN. The accuracy and loss of the model corresponding to each epoch obtained in the hold-out method through training and testing fragments are shown in Fig. 8. This hold-out method used the batch size = 5000 and the number of epochs = 120 in all the scenarios.

Figure 8a, b, c, d are denoted in scenarios 1, 2, 3 and 4, respectively. Each of these figures has two subparts representing the accuracy and loss of the model. Figure 8a shows two subparts, namely Figure 8a1, a2. Similarly, Fig. 8b, c, d also shows their own two subparts. There are four legends used in each of these figures denoting the scenario and optimizer used during training and testing. Like Fig. 8a has four legends namely 'S1_a_trn,' 'S1_a_tst,' 'S1_r_trn' and 'S1_r_tst.' Likewise, Fig. 8b, c, d also have their own four legends. Here, 'S' means scenario, 'a' denotes 'adam' optimizer, 'r' depicts 'rmsprop' optimizer, 'trn' represents training and 'tst' denotes testing. The legend 'S1_a_trn' represents the accuracy or loss for scenario 1 during training using 'adam' optimizer. Similarly, 'S2_r_tst' depicts the accuracy or loss for scenario 2 during testing using 'rmsprop' optimizer, and so on. It is evident from

Table 3 Number of units used in each layer of the model

Scenario	Input layer	Hidden layer	Output layer
1	23	12	1
2	22	11	1
3	32	16	1
4	28	15	4

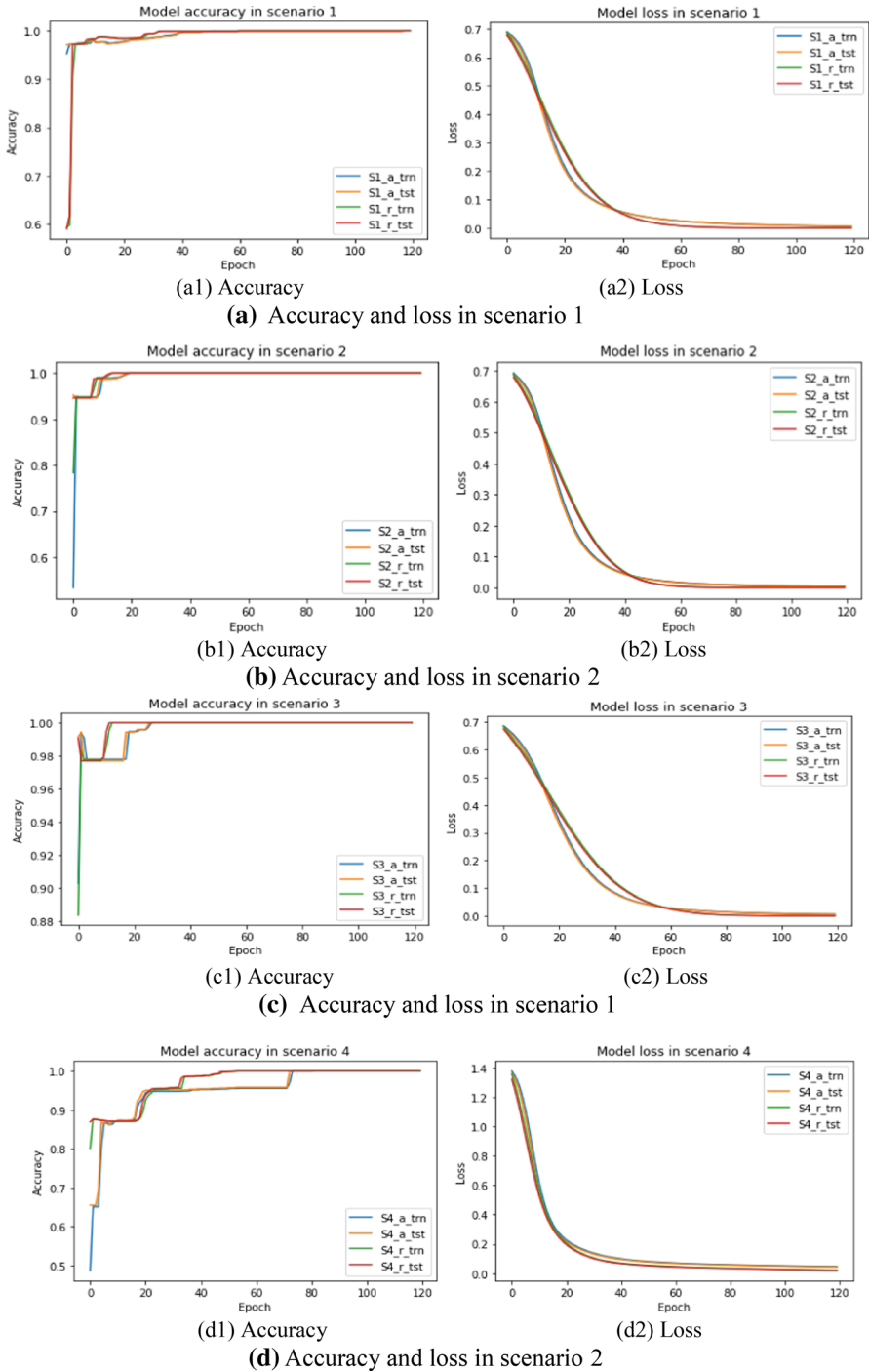


Fig. 8 Accuracy and loss in all the scenarios through hold-out validation

Fig. 8 that all four scenarios show maximum accuracy and minimum loss with the rise in the number of epochs. Generally, the accuracy increases and loss decreases with the rise in the number of epochs. Figure 8 shows that the 'rmsprop' optimizer performed better than the 'adam' optimizer as it requires less number of epochs to raise the accuracy to maximum. It is noticed that in Fig. 8, the accuracy and loss trends of testing data follow training data behavior in all the scenarios. This infers the successful training of the model. The classification report for each scenario is given in Table 4. This classification report illustrates the accuracy of the model through different performance metrics. Table 4 shows that the *precision*, *recall* and *F1-score* performance metrics possess the maximum values in each scenario. Besides, the support in a classification report denotes the total number of correct predictions for each class in a classification problem.

The MCC has been considered as a more reliable performance measure than *accuracy* and *F1-score* in a binary classification problem [59]. As a result, the MCC value has been computed for all the scenarios. The MCC evaluated by Eq. (6) applies to only binary classification problems. Therefore, the first three scenarios representing the binary classification problem possess maximum value for MCC. Scenario 4 representing the multi-classification problem possesses 0 values in the non-diagonal elements of the confusion matrix. All the FP and FN values are considered as zero and all the diagonal elements of the confusion matrix represent the correct predictions of different classes. Thus, the MCC value for scenario 4 is also evaluated as 1 according to Eq. (6). Moreover, the confusion matrix and Mathews correlation coefficient (MCC) for each scenario are illustrated in Table 5. Table 5 shows that both of the aforesaid performance measures reflect 100% accuracy. This is because the confusion matrix in each scenario has the total number of correct predictions equal to the total number of predictions. Also, the MCC computed for all the scenarios also possesses the maximum value.

Table 4 Classification report

Scenario	Classes	Precision	Recall	F1-score	Support
1	0	1	1	1	4525
	1	1	1	1	3148
2	0	1	1	1	9533
	1	1	1	1	550
3	0	1	1	1	5546
	1	1	1	1	130
4	0	1	1	1	7032
	1	1	1	1	449
	2	1	1	1	747
	3	1	1	1	2509

Table 5 Parameter tuning and confusion matrix

Scenario	To get 100% accuracy through tenfold cross-validation						Hold-out validation		
	Optimizer	Batch Size	Number of Epochs	Accuracy	Mean	Variance	Confusion Matrix	Mathews correlation coefficient (MCC)	
1	adam	500	16	100%	1	0	$\begin{bmatrix} 4525 & 0 \\ 0 & 3148 \end{bmatrix}$	1	
		1000	34						
		1500	49						
		2000	63						
		2500	73						
		3000	91						
		3500	110						
		4000	128						
		4500	129						
		5000	147						
		rmsprop	500						9
			1000						17
			1500						24
			2000						32
	2500		39						
	3000		45						
	3500		56						
	4000		62						
	4500		63						
	5000		75						

Table 5 (continued)

Scenario	To get 100% accuracy through tenfold cross-validation						Hold-out validation	
	Optimizer	Batch Size	Number of Epochs	Accuracy	Mean	Variance	Confusion Matrix	Mathews correlation coefficient (MCC)
2	adam	500	6	100%	1	0	$\begin{bmatrix} 9533 & 0 \\ 0 & 550 \end{bmatrix}$	1
		1000	10					
		1500	13					
		2000	18					
		2500	24					
		3000	31					
		3500	41					
		4000	43					
		4500	44					
		5000	45					
		rmsprop	500	6				
			1000	12				
			1500	17				
			2000	25				
	2500		29					
	3000		37					
	3500		41					
	4000		42					
	4500		43					
	5000		46					

Table 5 (continued)

Scenario	To get 100% accuracy through tenfold cross-validation						Hold-out validation		
	Optimizer	Batch Size	Number of Epochs	Accuracy	Mean	Variance	Confusion Matrix	Mathews correlation coefficient (MCC)	
3	adam	500	6	100%	1	0	$\begin{bmatrix} 5546 & 0 \\ 0 & 130 \end{bmatrix}$	1	
		1000	14						
		1500	18						
		2000	25						
		2500	29						
		3000	32						
		3500	38						
		4000	43						
		4500	48						
		5000	51						
		rmsprop	500	8					
			1000	14					
			1500	19					
			2000	26					
	2500		28						
	3000		30						
	3500		37						
	4000		41						
	4500		45						
	5000		47						

Table 5 (continued)

Scenario	To get 100% accuracy through tenfold cross-validation						Hold-out validation	
	Optimizer	Batch Size	Number of Epochs	Accuracy	Mean	Variance	Confusion Matrix	Mathews correlation coefficient (MCC)
4	adam	500	14	100%	1	0	$\begin{bmatrix} 7032 & 0 & 0 & 0 \\ 0 & 449 & 0 & 0 \\ 0 & 0 & 747 & 0 \\ 0 & 0 & 0 & 2509 \end{bmatrix}$	1
		1000	23					
		1500	31					
		2000	41					
		2500	48					
		3000	53					
		3500	72					
		4000	85					
		4500	93					
		5000	123					
	rmsprop	500	8					
		1000	14					
		1500	20					
		2000	23					
		2500	37					
		3000	42					
		3500	45					
		4000	55					
		4500	56					
		5000	64					

6.2 K-fold cross-validation

The accuracy of the model in each scenario has also been validated through the k-fold cross-validation method to avoid overfitting. The value of k is taken as 10 for this work. The tenfold cross-validation method divides the training dataset into ten subsections. Out of which, nine have been collectively used for training and the remaining one has used for testing. The ANN is trained with any nine subsections and tested with the remaining subsection. Each time a different subsection has been used for testing. The model is trained and tested repeatedly until all the subsections

are used once for testing. Thus, the proposed model has been trained and tested ten times with the mentioned number of epochs. The average accuracy of all the ten trials have been evaluated for the validation of the model. The relevant accuracy of any supervised learning model can be obtained by getting low bias and low variance value in the bias-variance tradeoff to prevent that model from overfitting. Therefore, the accuracy of the proposed model is enhanced until the accuracy reaches its maximum value in all the ten trials. The optimized value for the hyperparameters in each scenario is obtained through parameter tuning. The hyperparameters are considered to have optimized value at the instant when all the ten trails reach maximum accuracy. At this instant, the mean accuracy value for all the ten subsections is 100%, and the variance value is zero. This value corresponds to the lowest variance and the maximum accuracy (or minimum bias) in the bias-variance tradeoff. It prevents the model from overfitting. In this work, the parameter tuning is accomplished in Table 5. Here, the value of hyperparameters have been varied to find the optimum result, *i.e.*, 100% accuracy. Like, the hyperparameter 'Batch Size' is varied as [500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000]. Similarly, the hyperparameter 'Optimizer' has been varied as ['adam', 'rmsprop']. The best number of 'epochs' that are found for each combination of 'batch_size', and 'optimizer' are mentioned in Table 5. These epochs achieve maximum accuracy in all trails of ten-fold cross-validation method. The mean and variance value for each case is 1 and 0, respectively. Figure 9 depicts the relationship among various aforementioned hyperparameters for attaining 100% accuracy through tenfold cross-validation. In Fig. 9, each legend is named according to the convention where "S" depicts the scenario. The 'adam' and 'rmsprop' are the stochastic gradient descent algorithms or optimizers used in the ANN. The legend 'S1_adam' represents the number of epochs

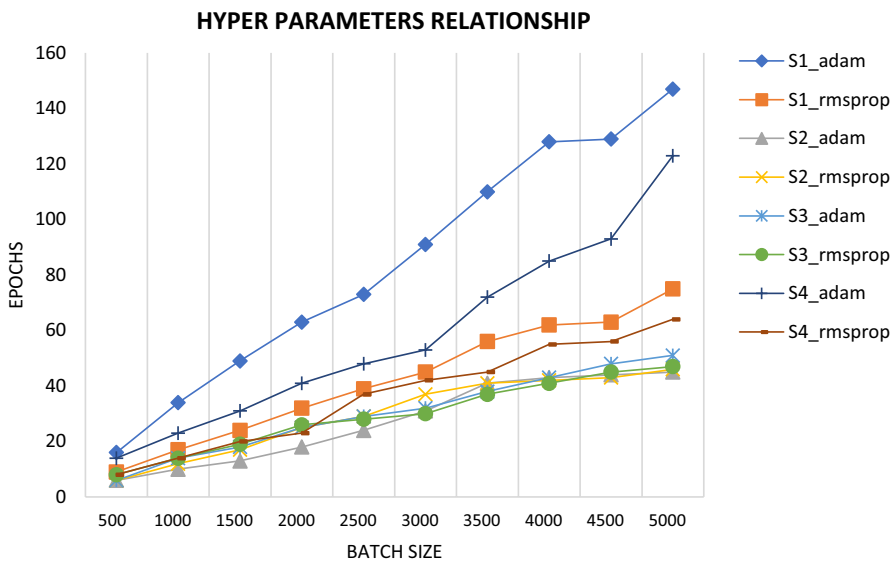


Fig. 9 Hyperparameters relationship

required in scenario 1 using ‘adam’ algorithm to achieve 100% accuracy. Similarly, ‘S2_rmsprop’ denotes the epochs required in scenario 2 using ‘rmsprop’ algorithm, and so on.

It is evident from Table 5 and Fig. 9 that with the increase in batch size, more number of epochs are required to achieve 100% accuracy through tenfold cross-validation. Scenario 1 used the maximum number of epochs among all the scenarios to reach maximum accuracy. Scenario 4 deployed the second largest number of epochs for 100% accurate detection of dissimilar attacks. In scenario 1 and 4, ‘rmsprop’ outperforms ‘adam’ in each batch size as relatively less number of epochs are required to get the best accuracy. Scenario 2 and 3 required a relatively less number of epochs to get trained optimally. Also, the performances of both the optimizers in scenarios 2 and 3 have been observed.

6.3 Comparison between hold-out validation and K-fold cross-validation

The minimum number of epochs obtained in both the validation schemes reached 100% accuracy as shown in Table 6. Here, the batch size = 5000, Epochs (H) and Epochs (K) represent the minimum number of epochs obtained in hold-out and k -fold cross-validation respectively. Table 6 shows that the hold-out method relatively takes less number of epochs to reach 100% accuracy. However, comparatively k times more complex process of k -fold cross validation makes it compute-intensive but more reliable. Analyzing both the validation schemes, it is recommended that the hold-out method is suitable for relatively quick and less computing analysis. However, k -fold cross-validation is suggested for thorough, computation-intensive and more reliable investigation. The hold-out validation method may lead to overfitting. The k -fold cross-validation method helps in avoiding the overfitting of the model.

6.4 Comparison of the proposed work with other research works

Diro et al. [32] used a deep learning approach to detect the attacks. This research work has not achieved 100% attack detection accuracy even by using comparatively more number of layers, input and hidden units. Le et al. [43] detected the RPL

Table 6 Minimum number of epochs required to reach 100% accuracy

Scenario	Optimizer	Epochs(H)	Epochs(K)
1	adam	119	147
	rmsprop	62	75
2	adam	21	45
	rmsprop	15	46
3	adam	28	51
	rmsprop	13	47
4	adam	82	123
	rmsprop	55	64

attacks through a specification based IDS which took more time to recognize the attacks perfectly. However, this work uses an ANN model that identified the RPL attacks with 100% accuracy in less than 2 min. Lee et al. [45] also validated the ANN through hold-out and tenfold cross-validation methods. Authors in [45] used 60%—40% ratio for training and testing respectively in the hold-out method. However, this work used 80%—20% ratio for the same. This work used 21 neurons in ANN model and focused on Eq. (7) for using the hidden neurons. Authors in reference [45] claimed 80% prediction accuracy for the ANN model. Similarly, Bhosale et al. [47] detected attacks using RSSI in RPL-based IoT networks with 90% accuracy. However, the proposed ANN model achieved 100% prediction accuracy for the optimized value of hyperparameters. Anitha et al. [46] detected the version and DIS attack collectively through the ANN model. Authors in [46] used comparatively more number of layers and input units. The ANN model is not capable of distinguishing between both the aforesaid attacks. However, the proposed model is able to detect and differentiate between the rank, hello flood and version attacks. Kfoury et al. [48] recognized the RPL attacks through an unsupervised SOM approach. Similarly, Qureshi et al. [50] detected the RPL attacks through threshold modulation. However, this work identified the RPL attacks using a supervised ANN-based approach. Sharma et al. [49] detected four types of RPL attacks, but separately. They have not implemented the simulation scenario with all four types of malicious nodes to occur simultaneously. However, this work implemented and detected the three RPL attacks separately as well as simultaneously. Sharma et al. [49] used standard classifiers like J48, Naïve Bayes, and Random Forest. Out of which they have got the maximum accuracy = 99.33% with the random forest classifier. Even the precision and recall values are comparatively less. Moreover, this work has not used any validation method. This work built its own classifier and got the maximum values for accuracy, precision, recall and F1-score. The presented work also got the maximum value for MCC which is considered as a more reliable performance measure than *accuracy* and *F1-score* [59]. The accuracy of the proposed work has been validated through two validation methods. The presented model has been prevented from overfitting through tenfold cross-validation method. Therefore, the proposed work is more robust and reliable.

7 Conclusion

This research work has investigated and applied a supervised machine learning technique for sensing the three well-known security attacks, namely hello flood, increased version and decreased rank. The proposed machine learning approach has been used in this work along with artificial neural networks. This paper implemented the three security attacks in four different scenarios. The first three simulation scenarios implemented the three attacks separately. The fourth simulation scenario has been applied to all the attacks simultaneously. An observable effect of the security attacks on the DODAG graph, energy consumption and traffic generation has been exposed by the performed simulations. The results disclosed that the malicious node for the hello flood attack produces the maximum number of packets

among all the nodes in the network. Therefore, it raises the energy expenditure of neighbors and does not affect the DODAG construction. Similarly, the fraudulent node for the increased version attack affects the DODAG and power consumption of other nodes in the network. This occurred due to the reason that other node in the network produces more number of packets. The fraudulent node for the decreased rank attack produces the least number of packets among all the malicious nodes. It is very effective for spying purposes and affects DODAG in the downward direction. The proposed ANN-based model recognized the attacks in each scenario with 100% accuracy. The other performance measures like *precision*, *recall*, *F1-score* and Mathew's correlation coefficient (MCC) have attained maximum values with the proposed model. The hyperparameters such as batch size, number of epochs and the optimizer have been optimized through parameter tuning. The proposed model has been validated through the hold-out and tenfold cross-validation method. It is notice through parameter tuning that for achieving 100% accuracy the proposed model requires more training for larger batch size. Scenario 1 required maximum training among all scenarios. However, scenario 2 and 3 required the least training to optimally detect the attacks. The optimizer 'rmsprop' performed better than 'adam' in scenarios 1 and 4 via tenfold cross-validation. However, 'rmsprop' did relatively well in all scenarios through hold-out validation. Moreover, the hold-out method is found relatively quicker for validation with less overhead. However, k-fold cross-validation is comparatively complex, time-consuming, computation-intensive but more reliable. Overfitting of the model has also been avoided through the tenfold cross-validation method. Furthermore, the proposed model is capable of identifying the attacks simultaneously as well as distinctly. The correct prior recognition of the attack averts the network from its adverse impact. This results in the increased security of the network. The increased security automatically enhances other performance metrics of the network such as energy efficiency, network lifetime, scalability and trustworthiness. The research work may be further enhanced via integrating classes for other security attacks occurred in IoT networks. Moreover, various machine learning schemes may be explored for suggesting novel intrusion detection systems for identifying different security attacks in IoT.

References

1. Lee I, Lee K (2015) The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Bus Horiz* 58(2015):431–440
2. Chen S, Xu H, Liu D, Hu B, Wang H (2014) A Vision of IoT: Applications, Challenges, and Opportunities With China Perspective. *IEEE Int Things J* 1(4):349–359
3. Purri S, Choudhury T, Kashyap N, Kumar P (2017) "Specialization of IoT applications in health care industries", International Conference on Big Data Analytics and Computational Intelligence (ICBDACI), pp. 1–5
4. Botta A, Donato WD, Persico V, Pescapé A (2016) Integration of cloud computing and internet of things: a survey. *FuturGenerComputSyst* 56:684–700
5. Singh S, Verma VK, Pathak NP (2015) Sensors augmentation influence over trust and reputation models realization for dense wireless sensor networks. *IEEE Sens J* 15(11):6248–6254

6. Verma VK, Singh S, Pathak NP (2015) Optimized battery models observations for static, distance vector and on-demand based routing protocols over 802.11 enabled wireless sensor networks. *Wireless PersCommun* 81(2015):503–517
7. Verma VK, Singh S, Pathak NP (2016) Analytical event-based investigations over delphi random generator distributions for data dissemination routing protocols in highly dense wireless sensor network. *Wireless PersCommun* 87(2016):1209–1222
8. Verma VK (2017) Pheromone and path length factor-based trustworthiness estimations in heterogeneous wireless sensor networks. *IEEE Sens J* 17:215–220
9. Verma VK, Singh S, Pathak NP (2017) Towards comparative evaluation of trust and reputation models over static, dynamic and oscillating wireless sensor networks. *Wireless Netw* 2017:1–9
10. Verma VK, Ntalianis K, Singh S, Pathak NP (2018) Data proliferation based estimations over distribution factor in heterogeneous wireless sensor networks. *ComputCommun* 124:111–118
11. Kawamoto D (2017) “IoT security incidents rampant and costly”, <https://www.darkreading.com/vulnerabilities---threats/iot-security-incidents-rampant-and-costly/d/d-id/1329367>, pp. 1–9
12. Gaddour O, Koubaa A (2012) RPL in a nutshell: a survey. *ComputNetw* 56(14):3163–3178
13. Pongle P, Chavan G (2015) A survey: attacks on RPL and 6LoWPAN in IoT. *IntConf Pervasive Comput (ICPC)* 2015:1–6
14. Biljana L, Risteska S, Kire TV (2017) A review of internet of things for smart home: challenges and solutions. *J Clean Prod* 140(2017):1454–1464
15. Mahdaveinejad MS, Rezvan M, Berekatain M, Adibi P, Barnaghi P, Sheth AP (2017) “Machine learning for internet of things data analysis: a survey”, *Dig Commun Netw-2017*, pp. 1–56
16. Zarpelão BS, Miani RS, Kawakani CT, Alvarenga SCD (2017) “A survey of intrusion detection in internet of things”, *J Netw Comput Appl-2017*, pp. 1–46
17. Khan MA, Salah K (2018) IoT security: review, blockchain solutions, and open challenges. *Future Gen ComputSyst* 82(2018):395–411
18. Costa KAPD, Papa JP, Lisboa CO, Munoz R, Albuquerque VHCD (2019) Internet of things: a survey on machine learning-based intrusion detection approaches. *ComputNetw* 151(2019):147–157
19. Sengupta J, Ruj S, Bit SD (2019) A comprehensive survey on attacks, security issues and blockchain solutions for IoT and IIoT. *J NetwComputAppl* 2019:1–51
20. Verma A, Ranga V (2020) “Security of RPL based 6LoWPAN networks in the internet of things: a review”, *IEEE Sens J-2020*, pp. 1–24
21. Avila K, Jabba D, Gomez J (2020) “Security aspects for Rpl-based protocols: a systematic review in IoT”, *Appl Sci-2020*, pp. 1–20
22. Almusaylim ZA, Alhumam A, Jhanjhi NZ (2020) Proposing a secure RPL based internet of things routing protocol: a review. *Ad Hoc Netw* 101(2020):1–17
23. Simha SNV, Mathew R, Sahoo S, Biradar RC (2020) “A review of RPL protocol using contiki operating system”, *Proceedings of the Fourth International Conference on Trends in Electronics and Informatics (ICOEI 2020)*, pp 259–264
24. Wallgren L, Raza S, Voigt T (2013) “Routing attacks and countermeasures in the RPL-based internet of things”, *Int J Dist Sens Netw-2013*, pp 1–11
25. Sehgal A, Mayzaud A, Badonnel R, Chrisment I, Schönwälder J (2014) “Addressing DODAG inconsistency attacks in RPL networks”, *Proc. of GIIS Conference*, pp 1–8
26. Mayzaud A, Sehgal A, Badonnel R, Chrisment I, Schonwalder J (2014) “A study of RPL DODAG version attacks,” in *AIMS’14* Springer, pp 92–104
27. Mayzaud A, Sehgal A, Badonnel R, Chrisment I, Schönwälder J (2015) Mitigation of topological inconsistency attacks in RPL-based low-power lossy networks. *Int J Net Manag*. <https://doi.org/10.1002/nem.1898>
28. Mayzaud A, Badonnel R, Chrisment I (2015) A taxonomy of attacks in RPL-based internet of things. *Int J Netw Sec* 18(3):459–473
29. Ahmed F, Ko YB (2016) Mitigation of black hole attacks in routing protocol for low power and lossy networks. *Sec CommNetw* 9(18):1–27
30. Thanigaivelan NK, Nigussie E, Kanth RK, Virtanen S, Isoaho J (2016) “Distributed internal anomaly detection system for Internet-of-Things”, *13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pp.319–320
31. Aris A, Oktug SF, Yalcin SBO (2016) “RPL version number attacks: In-depth study”, *2016 IEEE/IFIP Network Operations and Management Symposium (NOMS 2016)*, pp. 776–779
32. Diro AA, Chilamkurti N (2018) Distributed attack detection scheme using deep learning approach for Internet of Things. *FuturGenerComputSyst* 82(2018):761–768

33. Perazzo P, Vallati C, Varano D, Anastasi G, Dinni G (2018) "Implementation of a wormhole attack against a RPL network: challenges and effects", 2018 14th Annual Conference on Wireless On-demand Network Systems and Services (WONS), pp 95–102
34. Jyothisree MVR, Sreekanth S (2019) "Attacks in RPL and detection technique used for internet of things", *Int J Recent Technol Eng (IJRTE)* ISSN: 2277–3878, 8(1) 1876–1879
35. Ahutu OR, El-Ocla H (2020), "Centralized routing protocol for detecting wormhole attacks in wireless sensor networks", *IEEE Access* 8 pp. 63270–63282
36. Sahay R, Geethakumari G, Mitra B, Sahoo I (2020) "Efficient framework for detection of version number attack in internet of things", Abraham A., Cherukuri A., Melin P., Gandhi N. (Eds) *Intelligent systems design and applications (ISDA 2018)*. *Advances in Intelligent Systems and Computing (AISC 941)*, pp. 480–492
37. Pu C (2020) Sybil attack in RPL-based internet of things: analysis and defenses. *IEEE Int Things J* 7(6):4937–4949. <https://doi.org/10.1109/JIOT.2020.2971463>
38. Raza S, Wallgren L, Voigt T (2013) SVELTE: Real-time intrusion detection in Internet of Things. *Ad Hoc Netw* 11(2013):2661–2674
39. Kasinathan P, Pastrone C, Spirito MA, Vinkovits M (2013) "Denial-of-Service detection in 6LoWPAN based internet of things.", *Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2013 IEEE 9th International Conference on. IEEE, pp. 600–607
40. Rghioui A, Khannous A, Bouhorma M (2014) Denial-of-Service attacks on 6LoWPAN-RPL networks: Threats and an intrusion detection system proposition. *J AdvComputSciTechnol* 3(2):143
41. Pongle P, Chavan G (2015) Real Time Intrusion and Wormhole Attack Detection in Internet of Things. *Int J ComputAppl* 121(9):1–13
42. Sheikhan M, Bostani H (2016) "A hybrid intrusion detection architecture for internet of things", 8th International Symposium on Telecommunications, IEEE, pp 601–606
43. Le A, Loo J, Chai KK, Aiash M (2016) A specification-based IDS for detecting attacks on RPL-based network topology. *Information* 7(2):1–19
44. Jarrah OYA, Hammdi YA, Yoo PD, Muhaidat S, Qutayri MA (2017) "Semi-supervised Multi-Layered Clustering Model for Intrusion Detection", *Digital Communications and Networks*, pp. 1–12
45. Lee D, Derrible S, Pereira FC (2018) "Comparison of four types of artificial neural network and a multinomial logit model for travel mode choice modeling", *transportation research board-2018*, pp.1–12
46. Anitha AA, Arockiam L (2019) "ANNIDS: artificial neural network based intrusion detection system for internet of things", *Int J Innov Technol Expl Eng (IJITEE)* ISSN: 2278–3075, 8(11), 2583–2588
47. Bhosale SD, Sonavane SS (2019) "A real-time intrusion detection system for wormhole attack in the rpl based internet of things", *The 12th International Conference Interdisciplinarity in Engineering, Procedia manufacturing* 32, 840–847
48. Kfoury E, Saab J, Younes P, Achkar R (2019) A self organizing map intrusion detection system for rpl protocol attacks. *Int J InterdisTelecommunNetw* 11(1):30–43
49. Sharma M, Elmiligi H, Gebali F, Verma A (2019) "Simulating attacks for RPL and generating multi-class dataset for supervised machine learning", 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 20–26
50. Qureshi KN, Rana SS, Ahmed A, Jeon G (2020) A novel and secure attacks detection framework for Smart Cities Industrial Internet of Things. *Sustain Urban Areas* 2020:1–33
51. Sharma S, Verma VK (2020) Security explorations for routing attacks in low power networks on internet of things. *J Supercomput* 2020:1–35
52. Thubert P (2012) "Objective function zero for the routing protocol for low-power and lossy networks (RPL)", RFC 6552; IETF Secretariat: Fremont, CA, USA, pp. 1–14
53. Gnawali O, Levis P (2012) "The minimum rank with hysteresis objective function", RFC 6719; IETF Secretariat: Fremont, CA, USA, pp. 1–13
54. Jain AK, Mao J, Mohiuddin KM (1996) "Artificial neural networks: a tutorial", *Computer*, 29(3), pp. 31–44
55. Dunkels A, Gronvall A, Voigt T (2004) "Contiki—a lightweight and flexible operating system for tiny networked sensors". 29th Annual IEEE International Conference on Local Computer Networks (LCN'04), IEEE Computer Society, pp. 455–462
56. Sterlind FO, Dunkels A, Eriksson A, Finne J, Voigt T (2006) "Cross-level sensor network simulation with Cooja", *IEEE 31st Conference on Local Computer Networks, LCN '06*, IEEE Computer Society, pp. 641–648.

57. Hondt AD, Bahmad H, Vanhee J (2016) "RPL attacks framework", mobile and embedded computing LINGI2146 Report, pp.1–14
58. Hendrawan INR, Arsa IGNW (2017) "Zolertia Z1 energy usage simulation with Cooja simulator", Informatics and Computational Sciences (ICICoS) 2017 1st International Conference on, pp. 147–152
59. Chicco D, Jurmaan G (2020) The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics* 21:6. <https://doi.org/10.1186/s12864-019-6413-7>
60. Hussain F, Hussian R, Hassan SA, Hosssain E (2020) Machine learning in IoT security: current solutions and future challenges. *IEEE CommunSurv Tutor* 22(3):1686–1721

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.