*Journal of **AI** and **D**ata **M**ining*
Published online:

Detecting Sinkhole Attack in RPL-based Internet of Things Routing Protocol

M. Yadollahzadeh-Tabari[1*] and Z. Mataji[2]

1. Department of Computer Engineering, Babol Branch, Islamic Azad University, Babol, Iran.
2. Department of Computer Engineering, Mazandaran Institute of Technology, Babol, Iran.

Received 15 January 2020; Revised 02 April 2020; Accepted 06 July 2020
**Corresponding author: m_tabari@baboliau.ac.ir (M. Yadollah zadeh-Tabari).*

Abstract
The Internet of Things (IoT) is a novel paradigm in computer networks is capable of connecting things to the internet via a wide range of technologies. Due to the features of the sensors used in the IoT networks and the unsecured nature of the internet, IoT is vulnerable to many internal routing attacks. Using the traditional IDS in these networks has its own challenges due to the resource constraint of the nodes and the characteristics of the IoT network. A sinkhole attacker node in this network attempts to attract traffic through an incorrect information advertisement. In this research work, a distributed IDS architecture is proposed in order to detect the sinkhole routing attack in the RPL-based IoT networks, and this is aimed to improve a true detection rate and reduce the false alarms. For the latter, we used one type of post-processing mechanism in which a threshold is defined for separating suspicious alarms for further verifications. Also the implemented IDS modules are distributed via the client and router border nodes that make it energy efficient. The required data for interpretation of the network's behavior is gathered from the scenarios implemented in the *Cooja* environment with the aim of *Rapidminer* for mining the produced patterns. The produced dataset is optimized using the *genetic* algorithm by selecting appropriate features. We investigate three different classification algorithms, and in its best case, *Decision Tree* could reach a 99.35 rate of accuracy.

Keywords: *Internet of Thing, 6LoWPAN, Intrusion Detection System, Routing Security, RPL's Attacks, Sinkhole Routing Attack.*

1. Introduction

Internet of Thing (IoT) is a hybrid network of small devices that are used to monitor and control the physical environment by collecting, analyzing, and processing the data [1,2,3]. The term internet of thing generally refers to a set of standards, protocols, devices, and technologies required for connecting and transferring information between smart devices with each other or humans [4]. The main idea of IoT is to develop an independent world by smart things and a wide range of technologies such as RFID, ZigBee, Wi-Fi, and 3G/4G/5G that are accessible from anywhere, and are capable of transferring information and make decisions anytime based on the internet infrastructure [5,6]. Each IoT devices includes three main parts. One for sensing data from the environment, a processor unit or microcontroller (MCU) that processes data and runs software stacks interfaced to a wireless device for connectivity, and finally, a communication block for sending and receiving data. Due to the lack of IPv4 addresses, the only choice for addressing massive IoT devices is IPv6, which is possible by 6LoWPAN, an adaption layer to use IPv6 on the resource-constrained IoT devices. 6LoPWAN is a vital technology for supporting the IoT communications. Figure 1 shows the position of 6LowPAN in the IPv6 protocol stack. As it can be seen, 6LoWPAN has set an adaption layer on the top of the data link layer to fragment the IPv6 packets into smaller sections that are required to a lower layer to support the end to end IPv6 communications among the power-constrained devices or powerful internet entities.

Securing communications in the IoT devices using standard mechanisms such as the encryption and authentication techniques is inefficient for these networks as these preventive mechanisms cannot identify all the possible threats, especially internal routing attacks that are for the different nature of wireless communication. Thus IoT is vulnerable from both the internet and the inside network [2].

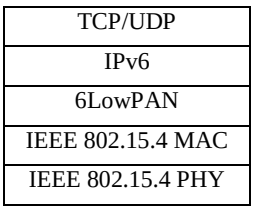


Figure 1. IPv6 protocol stack [7].

Some of the current projects have provided methods for increasing security, authentication, confidentiality, access control, privacy, and trustworthiness between the users and things [2,9]. However, in spite of these mechanisms, the IoT systems are still vulnerable to attacks aimed at a disturbance in the network. For this reason, another defensive line is required to identify the attackers. This goal can be achieved by using the intrusion detection systems to detect threats and vulnerabilities in the networks. However, using the traditional detection techniques is difficult for IoT. This is due to the specific restrictions such as the resource constraints, devices type, and specific protocol stack and standards [1,8,10].

Among the IoT routing protocols, RPL is the most famous and usable one. This is a distance vector routing protocol. It uses some sort of Directed Acyclic Graphs (DAGs) for finding the destination node. Also the sinkhole attack, which is one of the most effective attacks in the IoT networks, operates mostly on networks with the RPL protocols. In the sinkhole attacks [11], an attacker advertises a beneficial routing path and thus makes many node route traffics through it. In RPL, an attacker can launch a sinkhole attack by advertising a better rank, thus making nodes down in the DODAG to select it as parent.

The principal purpose of this research work was to improve the detection rate of the sinkhole routing attack and decrease the false alarm rates in the RPL routing protocol on the 6LoWPAN-based networks, which is a standard for network layer in the IoT networks. This will be done through classification and analysis of normal and threat behavioral patterns using data mining techniques. Before that, we used the genetic algorithm to specify the effective features in the constructed dataset. We used three learning algorithms separately in order to train the classification models and compared their results. Also for reducing the false alarms rate, a mechanism based on the generated alarms of k nearest neighbors of nodes was used. We simulated the RPL network in the *Cooja* simulator and the learning algorithms implemented in the *Rapidminer* data mining software as an offline process. Also the implemented IDS modules were distributed via the client and router border nodes that made it energy-efficient. Briefly, the paper includes the following contributions:

- Designing an anomaly-based and distributed IDS architecture in the IoT networks based on the RPL protocol in collaboration with the nodes.
- Using the genetic algorithm as a feature selection mechanism in order to improve the classifier accuracy.
- Defining a threshold-based post-processing technique in partnership with the
- k nearest neighbors of the nodes for filtering the suspicious alarms from the real attack ones.
- Evaluating the proposed method using multiple experiments in terms of the attack detection accuracy and energy consumption.

The rest of this paper is organized as what follows. Section 2 overviews the foundation of the IoT networks, and identifies the security issues, RPL routing protocol, and sinkhole attack. At the end of Section 2, the related works are presented. The model proposed for detecting the sinkhole attack in the 6LoWPAN platform based on the RPL routing protocol is represented in Section 3. The performance of the proposed model in the simulated scenarios is presented in Section 4 by reporting the simulation results. The paper is concluded in Section 5.

2. Routing Protocol for Low Power and Lossy Networks (RPL) & Sinkhole Attack

Due to the inherent characteristics of each application and used restricted devices, employing an appropriate routing solution in the 6LoWPAN environment is a challenging task. In most different LoWPAN cases, the sensors are connected to a small set of devices named root that is responsible for gathering data. For this purpose, for providing routing in 6LoWPAN, RPL was developed as a routing protocol for low-power and Lossy Networks (LLN) for the Routing Over Low-power and Lossy Networks (ROLL). This protocol operates at the network layer and is able to create a path and distribute the routing information between

nodes in an optimizing way. RPL is a distance vector routing protocol. Accordingly, the information path considered as a set of Directed Acyclic Graphs (DAGs) is further classified as a set of Destination-Oriented Directed Acyclic Graph (DODAG). In RPL, each DODAG consists of sensor nodes and a sink node that is responsible for collecting data of the sensor nodes [8]. In RPL, packets forward according to three traffic patterns; one-to-one, one-to-many, and many-to-many communications by supporting different operations such as unidirectional traffic to the DODAG root, bidirectional traffic between the 6LoWPAN nodes, and bidirectional traffic between the 6LoWPAN devices and the DODAG root [3,5,8].

According to the DODAG architecture, the nodes are organized hierarchically toward the DODAG root, which is the final destination to avoid loop in the network, meaning that in RPL, the nodes have the role of a child or a parent. A child node or a leaf collects data as a source node, and a parent node is responsible for routing the child's node data toward the root. The DODAG root can be a router that connects the IoT networks to the internet. In RPL, each node has a rank that represents the node's position to the DODAG root, i.e. the node with a lower rank has an optimized path to the DODAG root. In order to create the topology, that is DODAG, each node selects a set of nodes as parents to forward its packets to the DODAG root. The nodes selected as the parents have a better path to the root. In other words, they have lower ranks than their children. RPL uses three control messages to form the network topology and manage the routing information, DIO, DAO, and DIS.

1. DIO (DODAG Information Object): used to distribute rank and the objective function to calculate the node rank from the root to the nodes, form and update the network topology. **2. DAO** (DODAG Advertisement Object): to propagate the destination information toward the root for supporting the downward RPL traffic. **3. DIS** (DODAG Information Solicitation): to request the graph's information by the neighbor nodes for joining the network. Figure 2 represents an RPL DODAG graph including the root (sink), leaves, two control messages, DIO, and DAO along with their propagating in the graph that how RPL supports the upward and downward traffics.

2.1. Sinkhole routing attack

In the sinkhole attack, the malicious node attempts to attract traffic through incorrect information advertisement. In fact, it advertises a falsified optimal path that makes the other nodes transfer their data packets through it. Then after receiving traffic illegally, use it to do its illegitimate purposes as dropping or modifying. Consequently, the attacker affects a part of the network through a disturbance on it. For more details, in the Sinkhole attack, the attacker decreases its ETX metric, which is to calculate the rank value form the network topology [3,12].

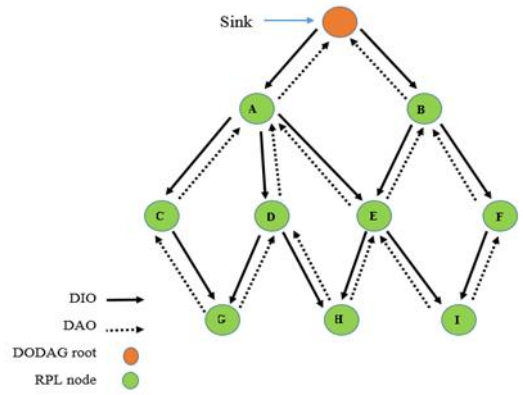


Figure 2.An RPL network control messages DIO and DAO.

Figure 3 shows a network under the sinkhole attack. As it can be seen, an attacker tries to attract the data packets of the neighbouring nodes and change the optimized topology. As a result, the paths will not be optimal and cause delay, data modifying, and data loss, and decrease the performance of the network.

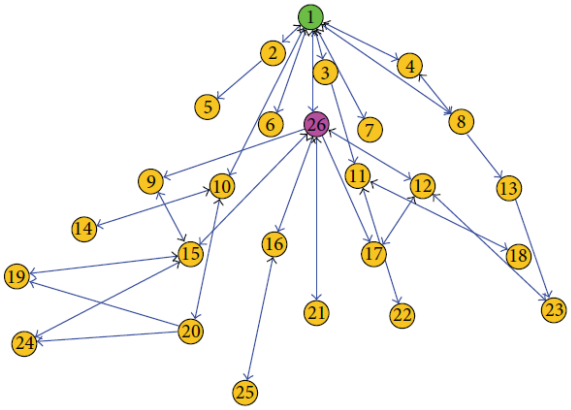


Figure 3.A network under sinkhole attack [12].

3. Related works

In this section, we present a literature review of the intrusion detection systems designed for the IoT networks. It should be noted that the provided solutions have not been investigated thoroughly, and the research works in this field are in the early stages. In the following, we will discuss several proposed solutions in terms of the placement strategy, detection method, security threats, and validation approach.

Raza *et al*. [4] have introduced SVELTE, a hybrid IDS against RPL routing attacks such as sinkhole and selective forwarding. The aim of SVELTE is to balance the storing cost of the signature-based method and the computational cost related to the anomaly-based method. In SVELTE, the border router is responsible for the main operations and processes, and the network's nodes host the lightweight agents to send data to the border router for analyzing. The simulations results have shown that SVELTE can detect all malicious nodes. However, the true positive rate is not 100%, and they have some false alarms during the detection.

Cervantes *et al*. [2] have proposed an intrusion detection system called INTI in order to detect the sinkhole attacks on 6LoWPAN. INTI employed distributed placement and combined the concepts of the anomaly-based method for monitoring packet exchange between nodes, and signature-based method to extract two kinds of node evaluation; trust and reputation. It assumed that there was a hierarchical structure and each node monitors a superior node by estimating its inbound and outbound traffic. If inbound and outbound traffic is not equal, that node is considered as an attacker. The experiments showed that INTI had a better performance for detecting sinkhole attacks than SVELTE. The impacts of the proposed model on the constrained nodes were not discussed.

Le *et al*. [11] have designed a network monitoring and an RPL specification-based IDS with a finite state machine for malicious checking in each monitor node. The proposed IDS is based on a hybrid placement and focuses on new topology attacks; rank attack and local repair attack. The authors developed and evaluated their work [13].

The authors have organized the network into clusters as demonstrated in [13]. There is an IDS agent in each cluster head that eavesdrops the cluster members' communications. Also the cluster members report the related information about itself and other neighbors to the cluster head. Based on the defined rules as specifications, the attack can be detected. The simulation results have shown that the proposed IDS has a high accuracy rate in detecting the RPL topology attacks and creates 6.3% overhead.

Wallgren *et al*. [14] have employed a centralized IDS on the border router, and using a heartbeat protocol to send the ICMPv6 echo requests to the network's internal nodes at a regular interval. If no response is received, the attacks or availability issues are expected. The presented model was investigated against the RPL routing attacks. In spite of additional traffic in the network, the authors have shown in the experiments that there is no need to allocate an additional memory to run the heartbeat.

Krimmling and Peter [15] have used the anomaly-based and signature-based techniques as a hybrid detection method to detect routing and man in the middle attacks. They did their experiments through an evaluation framework that they proposed themselves. According to the results obtained, the provided IDS could detect most of the attacks.

4. Proposed model

Our proposed detection model concentrates on detecting the sinkhole attacks in the 6LoWPAN platform based on the RPL routing protocol with the aim to improve the attack detection rate. In the suggested sinkhole detection approach, at first, the detection agents are learned using the data mining algorithms. In order to create a normal profile of the network's behavior, we use the learning algorithms such as Decision Tree, Support Vector Machines (SVM), and Bayesian Classifier. Then the learned models are used to detect the sinkhole attack. Finally, using a mechanism to validate the generated alarms, we will reduce the false detection rate, which decreases the usage of computational resources for processing the generated alarms and improves a true detection rate.

In our proposed model, we use a hybrid placement strategy. Then the intrusion detection agents place both on the border router and the hosts. The central IDS agent that demands more energy will be located on the border route that is responsible for the principal computations and actions.

Also a type of lightweight detection agent is implemented on each of the network nodes without the need for robust processing resources. These agents are responsible for collecting the required information for network analysis, and send the specific information of the nodes to the border router in a pre-defined time interval. Using this hybrid approach, the computational overhead of monitoring the total network traffic and information from the border router will be decreased. Figure 4 shows the hybrid placement of our used IDS agents in an RPL-based IoT network, which will be explained then.

In the following, each one of the IDS modules on the border router and nodes is explained separately. The following two modules are in the IDS agent existing on each network node:

Data collection: A lightweight module in the network node that sends the node information to the IDS agent on the border router. It is assumed for message security mechanisms as IPsec or DTLS used to send such data.

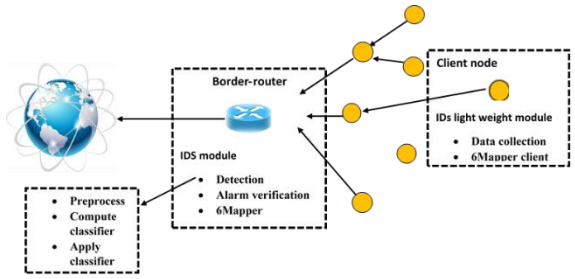


Figure 4.IDS agents' placement on the border router and nodes.

6Mapper client: It has the responsibility to send the node coordinates in the network graph, RPL identifier, and DODAG version number to the 6Mapper module on the border router to reconstruct the network graph for checking suspicious alarms in the post-processing phase.

The following modules are in the IDS agents existing on the border router:

6Mapper: Through this module, the RPL network graph is reconstructed in the border router by the information received from 6Mapper client modules in the network nodes.

Detection: responsible for the main tasks of detection including pre-processing, compute classifier, and apply the classifier sections explained below.

(1) Pre-processing: Since data collection is done by tools, they may record invalid or faulty data if the tools get in trouble. Thus the collected data must be prepared before any analysis. Figure 5 shows the pre-processing operations that are divided into two sections, the preliminary pre-processing and feature selection. In the preliminary section, the outliers and duplicates in the dataset are identified and resolved. Also the existing missing values will be initialized. The goal of the second section, meaning feature selection, is to create simpler models by selecting a subset of proper features with a minimum number of members whose probability distribution of data classes is as close as to the obtained distribution of all the primary features. We considered feature selection as an optimization problem and employed the evolutionary genetic algorithm for this purpose.

As displayed in figure 5, in the feature selection section, the first step is to initialize a random population of bit vectors of 0 and 1. It means that 1 indicates the existence of a feature and 0 denotes the absence of a feature in the subset. Then each one of these subsets should be evaluated by a fitness function in order to determine their fitness level. In our suggested scheme, we used the multiple linear regression for assessing the fitness of the generated subsets. The idea behind the linear regression [21] in a (K+1) dimensional data space is to fit a (k) dimensional hyper plane space, which minimizes the sum of squared residuals (SSR) for the points in the training data. The SSR measure calculates the difference between the target data (y_i) and the predicted values estimated by the model using the regression coefficients $(\beta_0,\ldots,\beta_k)$. This value is obtained from the following equation:

$$\min\sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{k}\beta_j x_{ij})^2 \qquad (1)$$

If we assumed n data and k dependent variable, y_i is obtained from the following equation:

$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_k x_{ik} + \grave{o} \qquad (2)$$

where $i = 1,\ldots,n, \beta$ is a $(k+1\times 1)$ dimensional matrix of the regression coefficients, $\grave{o}$ is a $(n\times 1)$ dimensional matrix of fitting error, Y is a $(n\times 1)$ dimensional matrix of the dependent variables, and X is the $(n\times k+1)$ dimensional matrix of the independent variables. Consequently, the multiple linear regression is written in the form of a matrix as follows:

$$Y = X\beta + \grave{o} \qquad (3)$$

Now, we can re-write the SSR parameter, willing to find the values of β that by replacing them in the equation, the obtained value is as low as possible:

$$\sum_{i=1}^{n}\grave{o}_i^2 = \grave{o}'\grave{o} = (y - X\beta)'(y - X\beta) \qquad (4)$$

If all $\grave{o}_i$ values are zero, the minimum value for the least squares, i.e. zero, will be obtained:

$$\hat{y} = X\hat{\beta} \qquad (5)$$

Here, $\hat{y}$ is the projection of the n-dimensional data vector y onto the hyperplane spanned by X. The $\hat{y}$ values are the predicted values in our regression model that all lie on the regression hyper-plane. Suppose that $\hat{\beta}$ satisfies the equation above.

Then the residuals $y - \hat{y}$ are orthogonal to the columns of X (by the Orthogonal Decomposition Theorem), and thus:

$$X'\left(y - X\hat{\beta}\right) = 0 \Leftrightarrow X'y - X'X\hat{\beta} = 0 \Leftrightarrow X'X\hat{\beta} = X'y \quad (6)$$

We multiplied both sides by the inverse of $X^{'}X$ in order to find the parameter estimates $\left(\hat{\beta}\right)$. Therefore, the least-squares estimator of β in vector

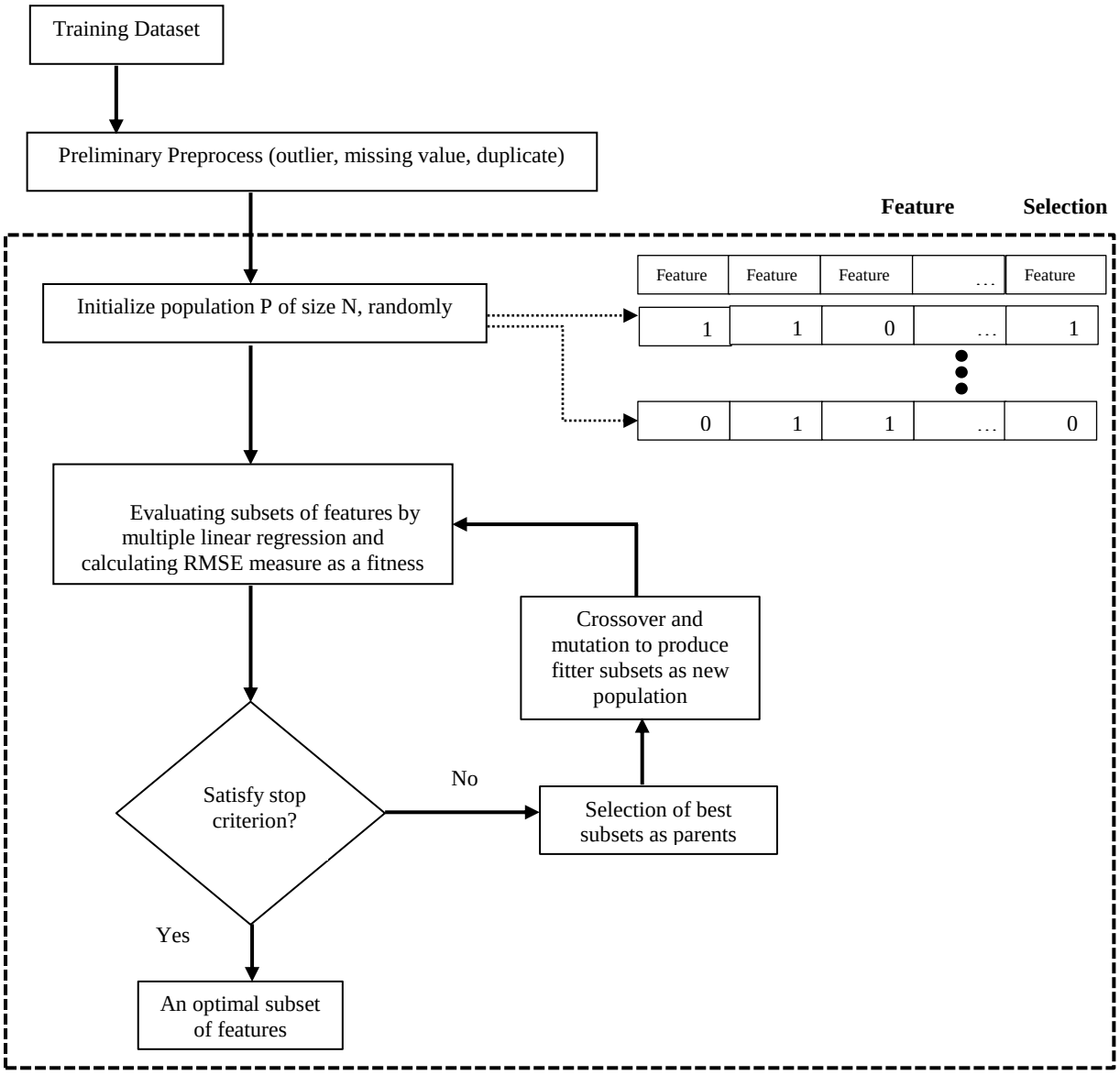


Figure 5. A flowchart of feature selection by the genetic algorithm.

form is:

$$\hat{\beta} = (X^{'}X)^{-1}X'y \quad (7)$$

$$\hat{y} = X\hat{\beta} = (X^{'}X)^{-1}X'y = Hy \quad (8)$$

where X' is the transpose of the matrix X. To calculate the inverse of matrix $(X^{'}X)$, the determinant of the equation above is required to be zero. The vector of the fitted values $\hat{y}$ in a linear regression model can be expressed as the HAT matrix. It maps the vector of the observed values y onto the vector of the fitted values $\hat{y}$ that lie on the regression hyper-plane. The regression residuals can be written in different ways as:

$$\grave{o} = y - \hat{y} = y - X\hat{\beta} = y - Hy = \left(I - H\right)y \quad (9)$$

Eventually, in order to compare the fitness of the regression line associated with each subset, we used RMSE (Root Mean Squared Error), which was based on SSR. RMSE is the square root of the variance of the residuals, and measures the error

rate of a regression model. It is expressed as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\hat{o}_i^2} = \sqrt{\frac{\sum_{i=1}^{n}\left(y_{obs,i} - \hat{y}_{pred,i}\right)^2}{n}} \qquad (10)$$

where y_o is related to the n observed values, and $\hat{y}_{pred,i}$ denotes the n predicted values by regression.

In each generation, after evaluating the generated subsets, the stop criterion is checked. If the stop criterion is satisfied, a subset with the lowest fitness is selected as an optimal subset of features; otherwise, according to the fitness values, the best subsets are elected as the parents to produce new subsets of features (children) through cross-over and mutation operations.

(2) Compute classifier: In this section, after pre-processing the submitted data from the collection modules available in the nodes, the classification models will be trained for classifying the network data as normal or attack. We used SVM, Tree, and Bayesian as the learning algorithms for creating the classification models and evaluating their performance separately. The details of the learning algorithms' deployment are described as follow.

For training the classification model by ***Decision Tree*** [13], we considered the gain ratio as an attribute selection measure for selecting the splitting criterion that separates the given dataset into individual classes. It employs a kind of normalization to information gain using a "split information" value defined analogously with *Info*(D) as:

$$\mathrm{SplitInfo_A}(\mathrm{D}) = -\sum_{j=1}^{v}\left(\left|\mathrm{D_j}\right| / \left|\mathrm{D}\right|\right) \times \log_2\left(\left|\mathrm{D_j}\right| f \ \left|\mathrm{D}\right|\right) \qquad (11)$$

D is a training set of class-labeled tuples, and $|D|$ denotes the number of tuples in D. $|D_j|$ contains the number of tuples in D that have outcome a_i of attribute A. This value represents the potential information generated by splitting the training data set, D, into v partitions, corresponding to the v outcomes of a test on attribute A. The attribute with the maximum gain ratio is selected as the splitting attribute.

For training the classification model by ***SVM*** [17], the Radial Basis Function (RBF) kernel is used to map the samples into a higher dimensional space with the aim of searching for the best separator hyperplane. For classifying a test tuple, the following decision function is used.

$$\mathrm{d}\left(\mathrm{x}^{\mathrm{T}}\right) = \sum_{i=1}^{l} \propto_i \mathrm{y_i}\phi\left(\mathrm{x_i}\right)^{\mathrm{T}}\phi\left(\mathrm{x}\right) + \mathrm{b_0} \qquad (12)$$

where y_i is the class label of the support vector x_i, x^T is a test tuple, $\propto_i$ and b_0 are the numeric parameters determined automatically by the optimization or SVM algorithm, and l is the number of support vectors.

The last trained model is based on the Bayes' theorem, which is ***naïve Bayes*** [18]. In this learning algorith for a given test tuple, X, the classifier predicts that the tuple belongs to the class having the highest posterior probability, conditioned on X.For reducing the computation in evaluating $P(X|C_i)$, the naïve assumption of class-conditional independence is made. This presumes that the attributes' values are conditionally independent from one another, given the class label of the tuple (i.e. there are no dependency relationships among the attributes). Thus:

$$\mathrm{P}\left(\mathrm{X}|\mathrm{C_i}\right) = \prod_{k=1}^{n}\mathrm{P}\left(\mathrm{x_k}|\mathrm{C_i}\right) = \mathrm{P}\left(\mathrm{x_1}|\mathrm{C_i}\right) \times \mathrm{P}\left(\mathrm{x_2}|\mathrm{C_i}\right) \ldots \times \mathrm{P}(\mathrm{x_n}\,|\,\mathrm{C_i}) \qquad (13)$$

By this equation, we can estimate the probability that X belongs to class C_i. Each tuple is represented by an *n*-dimensional attribute vector, $X = (x_1, x_2, \ldots, x_n)$. Parameter x_k refers to the value of attribute A_k for tuple X. Since in our dataset the selected features are continuous, the continuous-valued attributes are typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ, defined by:

$$\mathrm{g}\left(\mathrm{x},\mu,\sigma\right) = \frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-\frac{(\mathrm{x}-\mu)^2}{2\sigma^2}} \qquad (14)$$

So that

$$P\left(x_k|C_i\right) = \mathrm{g}\left(x_k, \mu_{C_i}, \sigma_{C_i}\right) \qquad (15)$$

At first, it is required to compute μ_{C_i} and σ_{C_i}, which are the mean (i.e. average) and standard deviations, respectively, of the values of attribute A_k for training tuples of class C_i. Then plug these two quantities into Eq. (14), together with x_k, to estimate $P(x_k|C_i)$.

(3) Apply classifier: After creating and training the classification models, using the training data and learning algorithms, normal or attack status of network's nodes is detected. According to the results of the data classification, alarms are produced to indicate the normal or attack states for each node. In figure 6, the process of computing and applying the classifiers is shown.

Alarm verification: This step is known as the post-processing phase. In order to improve the accuracy and decrease the computational costs of alarm processing, the generated alarms will be revised and evaluated by this module. At first, a threshold will be defined called the degree of confidence, to separate the suspicious alarms that are required to be revised. After separating the neighbors are specified in the reconstructed DODAG graph; we considered $k=3$. For investigating and taking the final decision about a node, at first, we determined the influence level of the neighbors in decision-making about the normal or attack state. For this purpose, we used the product of the distance (from the node to its neighbor) in ETX / Hop (for the neighbor node). If the produced alarm for a neighbor node indicates a

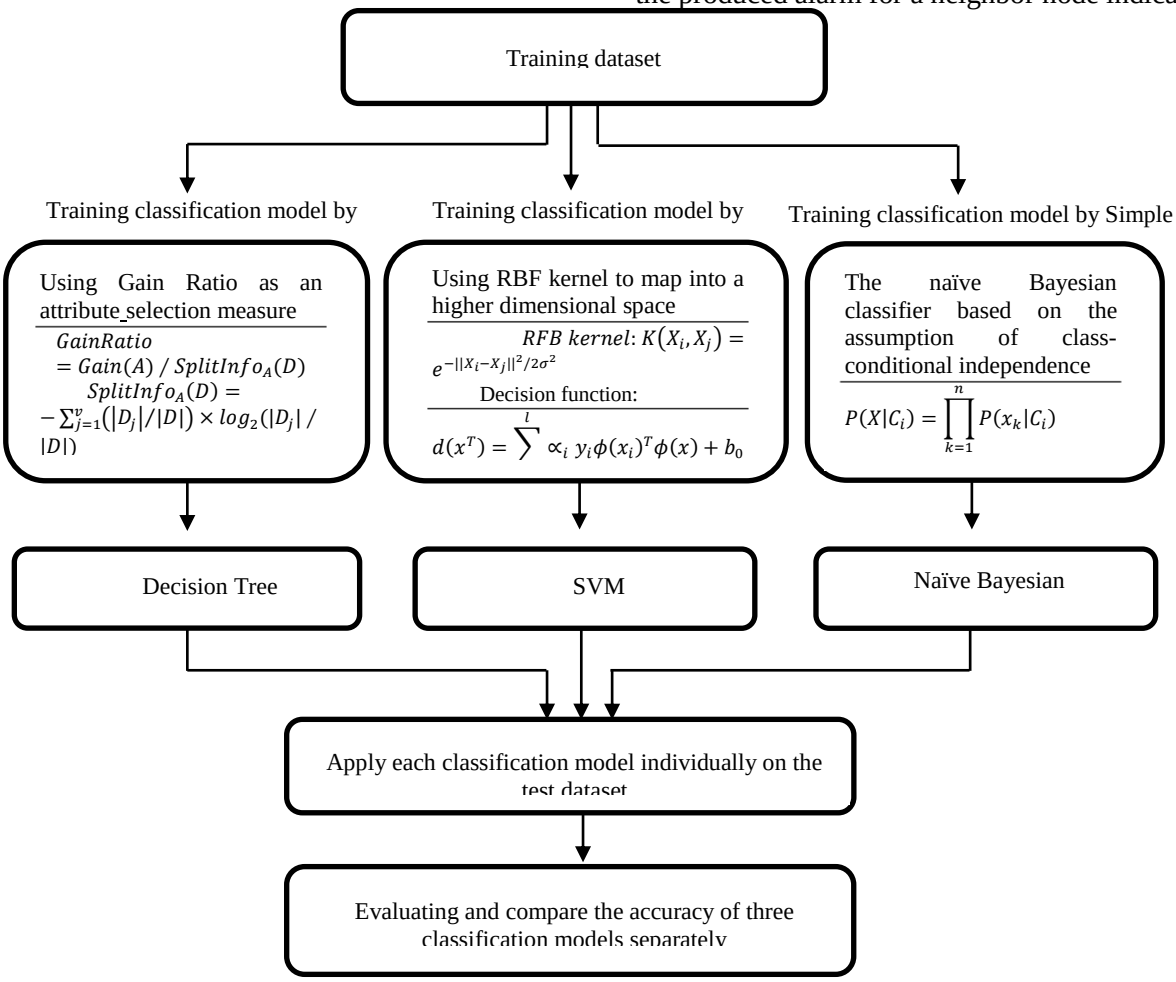


Figure 6.Computing and applying the classifier models.

This equals the ratio of ETX to Hop for $node_i$. ETX is the expected transmission count, and Hop refers to the hop count from the node to the sink. We considered $ETX / Hop \geq 13$ for the normal situation and $ETX / Hop < 13$ for the attack situation empirically. During checking the alarms, if the ETX / Hop ratio and the specified class for an alarm violate the ETX / Hop rule, it is separated as a suspicious alarm. In the attack situations, an attacker tries to provide a lower ETX, despite having high hop counts to sink. Therefore, the ETX / Hop ratio will get a lower value than a normal situation. After separating the suspicious alarms, the associated nodes that are based on their sent data, the alarms produced, and their k nearest normal state, this equation products in $(+1)$; otherwise, it will product in (-1). This leads to a considerable decrement in the false positive rate. The Parameter S will be defined for determining the final status of the suspicious alarms as equation 16. The parameter C_i is our defined threshold that we called the degree of confidence for the generated alarm related to node i.

Index k is the number of neighbors participating in the validation of the suspicious alarms. The parameter D_i is the distance from node n to its neighbor node i and index n is for the nodes whose produced alarms are separated as suspicious.

$$S_n = \sum_{i=1}^{k}(D_i \times C_i)\times(+1) + \sum_{i=1}^{k}(D_i \times C_i)\times(-1)$$

$$\Rightarrow \begin{cases} S > 0 & state = normal \\ S < 0 & state = attack \\ S = 0 \;\; k = +1 & ,calculate\, S\, again \end{cases} \quad (16)$$

$$C_i = ETX_i \; f \;\; Hops_i \quad (17)$$

The calculated values for each neighbor node are summed together and displayed with S_n. If the obtained value is positive for a node, the related alarm is classified as normal. Otherwise, for the negative obtained values, the related alarms are categorized as the attack. If S_n is zero, one will be added to the number of neighbors, and S_n will be calculated again.

5. Simulation and results

In order to measure the performance metrics and show the improvement of sinkhole detection rate, we used the *Cooja* [19] network simulator to simulate an IoT network based on the RPL routing protocol, and produced the dataset. *Cooja* is a network simulator based on *Contiki* OS, an open-source operating system for the Internet of Things that focuses on low-power devices. Also *Rapidminer*, as an analytics tool, is used for mining the produced dataset. The simulated network has two types of nodes, the server node and the sender nodes. The sender nodes send their sensed data to the server node that acts as a sink. It is assumed that the server node does not have any computational and energy constraints. The nodes are stationary and do not have mobility. The communication platform and the employed routing protocol are 6LoWPAN and RPL, respectively. We used the T_{mote} sky node type, which was compatible with other IEEE 802.15.4 [20] devices and had a low-power consumption. For simulating the network in Cooja, the network's parameters are set as follow: the Unit Disk Graph Medium (UDGM): Distance Loss is used as a radio medium, mote startup delay is set to 1000 (ms), transceiver's signal strength is set to 1-100 (dBm), and the transfer and interference ranges are set to 50 (m) and 100 (m), respectively.

The sending and receiving ratios are set to 100%, and the packet transmission rate is every 60 seconds. The transfer protocol, the network layer, and the routing protocols are UDP, IPv6, and RPL, respectively. Figure 7 shows a sample snapshot of the network in the normal and sinkhole attack states. Figure 7(a) shows the network in a normal situation, where all nodes send their data to the sink via an optimal path. In figure 7(b), the node 24 acts as an attacker and attracts the total traffic sent by neighbors to itself. Then it tries to corrupt or abuse the attracted data and makes disturbance in the network through implementation of other attacks.

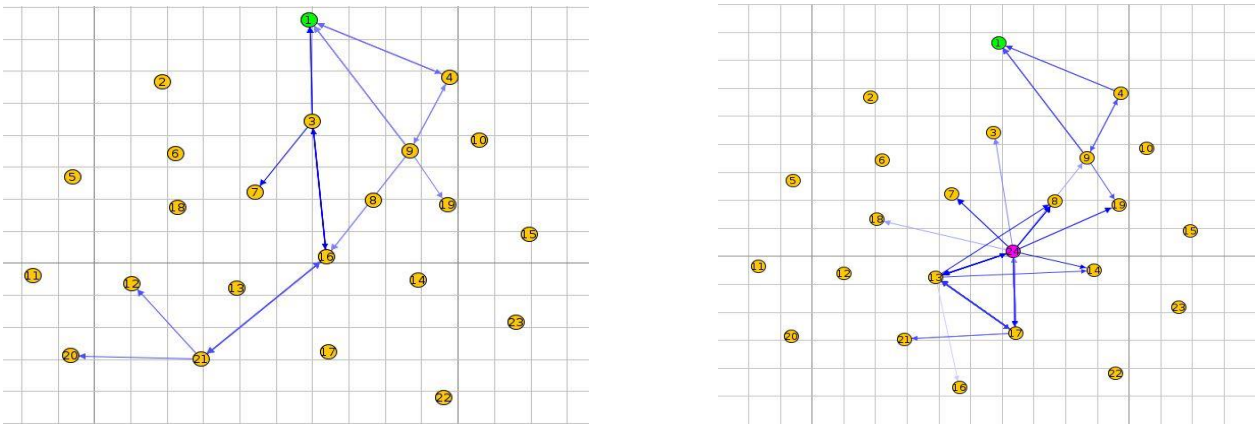
5.1. Pre-process

First, the preprocessing operations will be done in order to remove the unnecessary features to expedite the learning time and save on the computing resources and also normalizing the data. The normalization is conducted using the Z-transformation [21] method. Then for some features that there is no data for them, we used the Replace Missing Values operator to replace the missing values. Next, the outliers in the data are identified and removed by the Detect Outlier (distance) operator based on the KNN (K-Nearest Neighbors concept. We considered K = 10. As mentioned in Section 4, for feature selection, we used the evolutionary genetic algorithm to find an optimal subset of attributes for classification.

In order to evaluate subsets searched by the genetic algorithm, we used the k-fold Cross-validation method, in which k = 10. The sampling type is shuffled-sampling. In the Cross-Validation block, the linear regression is used to evaluate the subsets searched by the genetic algorithm that employs the Akaike information criterion (AIC) [21] for selecting the model. AIC is a measurement of the relative fitness of a statistical model. For implementation of the linear regression algorithm, the parameter eliminate collinear features sets with the minimum tolerance value of 0.05 for bias is used, and the ridge parameter is set to 1.0E-8. The Root Mean Squared Error (RMSE) was considered as a performance measure, and a subset with the minimum RMSE was selected as an optimal subset. After applying the pre-processing operations to the dataset, a subset of five features with an RMSE value of 0.357 was selected.

A model with the minimum AIC is the best among all the others. The parameter set for the genetic algorithm is shown in table 1. In order to find the mentioned parameter values, we conducted several experiments with different values. For the parameter selection schema, three different schemas were investigated as the Roulette Wheel, Tournament, and Ranking methods. Among them, the Tournament method has a surprisingly better AIC value. The same strategy was also conducted

to find the optimal value for *P_crossover*, type of cross-over, and *P_mutation*. To find the best value for them, we also conducted numerous experiments, and finally, we extracted the parameters making a lower AIC value. The final selected features are illustrated in table 2.



(a) network in a normal situation. **(b) network under sinkhole attack (node 24).**

Figure 7.An RPL network in the normal and sinkhole attack situations.

Table1. The genetic algorithm parameter values used for feature selection.

No	Parameter	Value
1	Population size	60
2	Maximum number of generations	100
3	Selection scheme	Tournament
4	Tournament size	0.5
6	P mutation	0.05
7	P cross-over	0.66
8	Cross-over type	Uniform
9	maximal fitness	Infinity

Table2. Features selected by the feature selection method.

No	Feature	Description
1	Hops	Number of hops from the node toward the sink node
2	Routing metric	Amount of routing metric in RPL routing
3	ETX	Number of expected transmissions of a packet received at destination successfully
4	CPU power	Amount of node power utilization
5	Transmit power	Amount of node transmission power utilization

5.2 Training phase

For the same distribution of classes, the stratified sampling is used. To increase the accuracy of the Decision Tree classification model, the *tree pruning method* [22] is used based on the determination of a certain amount of confidence level of the branches. In addition to post-pruning, pre-pruning is used as well by defining the parameters as a threshold, preventing the division of samples, and turning the node to a leaf. All the parameter values related to Decision Tree are shown in table 3. For the naïve Bayesian classifier, the *Laplace correction* [22] enables to prevent the high influence of zero probabilities. The tuned parameters for the Decision Tree and SVM algorithms are presented in tables 3 and 4, respectively.

Table 1. Parameter sets for Decision Tree.

No	Parameter	Value
1	Criterion	Gain ratio
2	maximal depth	20
3	Confidence (used for pessimistic error calculation of pruning)	0.25
4	minimal gain	0.1
5	Minimal leaf size	4
6	Minimal size for split	4
7	Number of pre-pruning alternatives	3

All the mentioned parameter values were investigated using several conducted experiments, which resulted in the highest the initial evaluation results of three classification models for the data collected at 2,4,6,8 and 10 minutes' intervals of the network's runtime are displayed in table 5.

DR (Detection Rate) is the percentage of the attack samples identified correctly divided by the total number of attack samples existing in the dataset

calculated by $DR = TP / (FN + TP) \times 100$. FPR (False Positive Rate) is the percentage of normal samples identified as attack incorrectly by IDS divided by the total number of the existing normal samples, obtained by $FPR = FP / (FP + TN) \times 100$.

Table 2. Parameter sets for SVM classifier.

No	Parameter	Value
1	kernel type	Radial
2	kernel gamma	1
3	kernel cache	200 mg
4	C (complexity constant)	0
5	convergence epsilon	0.001
6	Max iterations	100000
7	L pos (SVM constant for pos samples)	1.05
8	L neg (SVM constant for neg samples)	1.05
9	Epsilon (the insensitivity constant).	0
10	epsilon plus	0
11	epsilon minus	0.2

Table 5. The classification results for 5 different data collection time intervals before applying the alarm verification method.

Data collection Intervals	Decision Tree		SVM		Bayesian	
	DR	FPR	DR	FPR	DR	FPR
2 min	71%	8.13%	74.19%	13%	85.48%	22.76%
4 min	77%	7%	80.65%	10%	82.66%	21%
6 min	80%	7%	77%	10%	80.32%	19%
8 min	81.62%	8%	77.37%	5%	83%	19%
10 min	83.39%	7%	78.39%	6%	84.35%	20%

According to the mechanism described in Section 4, after separating and checking the suspicious alarms, the number of false alarms will be decreased for three classification models, as shown in figure 8. Also the final evaluation of the classification models is presented in table 6. As it can be seen, by training the models during the time, the detection rate has been increased and reached above 90% for the three classification models. At each time interval, the Bayesian model has the highest sinkhole attack detection rate. The lowest false detection rate associated with the Decision Tree classifier, which at 2, 4, and 6 intervals has no false detection rate, and at 6 to 10 intervals has the lowest false detection rate.

At 2 to 8 intervals, the Bayesian classifier has a higher false detection rate than the SVM and Decision Tree models. The highest FPR belongs to the SVM classifier by the 10 minutes' data collection time interval. As another metric, the *Accuracy* has been used to determine the performance of the classification models.

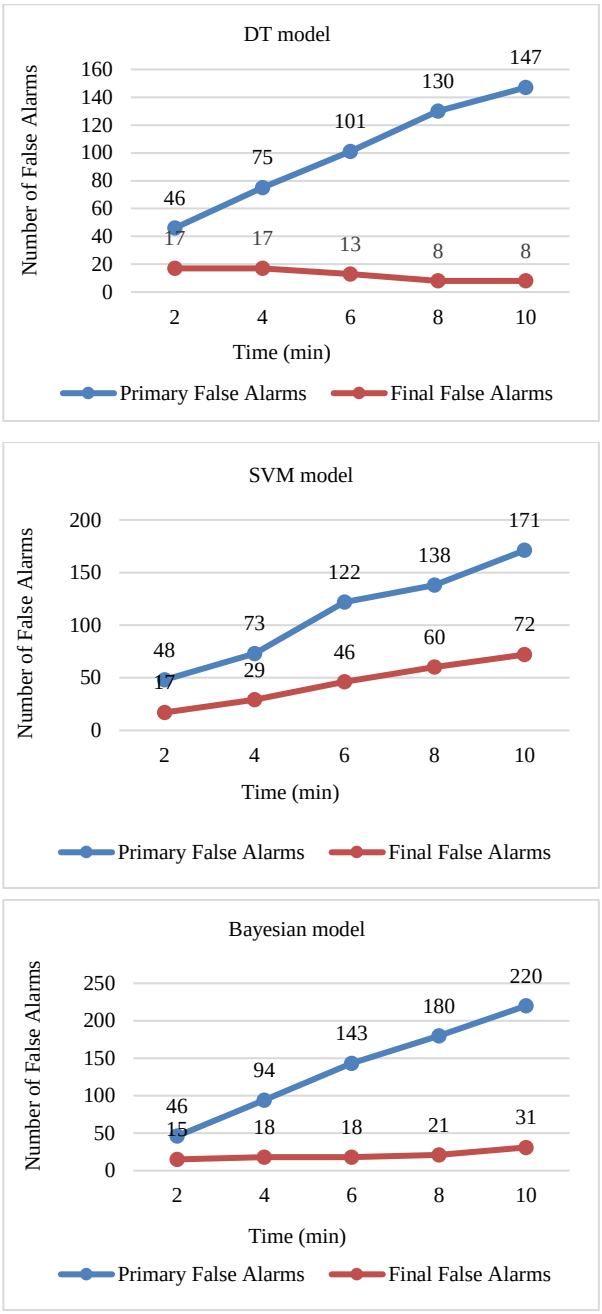


Figure 8. Number of generated false alarms by IDS for three classification models before and after checking the suspicious alarms.

Accuracy is the ratio of samples of a class that are classified correctly toward all the predicted data for that class, and is calculated by $\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{FP} + \text{TN} + \text{FN}) \times 100$

The accuracy value of the three employed classification models at specified intervals is shown in figure 9 that is generated after the alarm verification method.

Table 6. The final evaluation results related to the classification models after applying the alarm verification method.

Data collection Intervals	Decision Tree		SVM		Bayesian	
	DR	FPR	DR	FPR	DR	FPR
2 min	86.29%	0	87.01%	0.8%	93.6%	5.6%
4 min	93.14%	0	91%	2.5%	97.6%	5%
6 min	96.5%	0	90%	2%	98%	3%
8 min	98.58%	0.19%	89.5%	1.56%	98.6%	2.73%
10 min	99.02%	0.16%	98.23%	9.24%	99.03%	4.04%

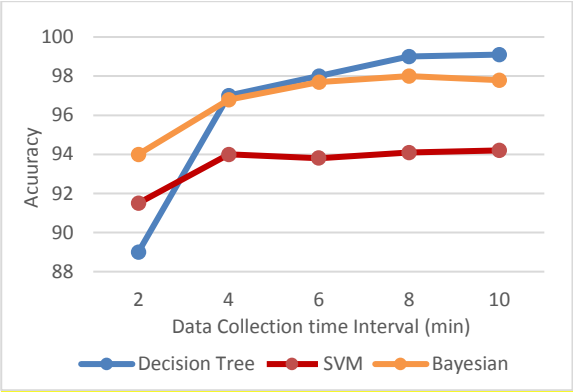


Figure 9. Comparison of precision of Decision Tree, SVM, and Bayesian classifiers at the specified time intervals.

For the first two data collection time interval minutes, the trained model by Bayesian has a higher precision than the two others. However, at other time intervals, the model trained by Decision Tree has the highest precision in classification of the normal and attack data.

5.3. Energy overhead

Given the importance of the role of energy in the IoT networks in this section, we analyzed the amount of energy overhead consumed in the proposed method. As described in Section 4, our proposed method is supposed to be offline after running the 6Mapper module in either of the client of border nodes. Then we only investigated the impact of running this module for the proposed IDS. Also since we assumed that the boundary nodes were the devices connected to unlimited electricity power, we did not calculate the impact of running the 6Mapper module on it. Then in this section, we only investigated the impact of running light-weight IDS module including (Data Collection module and 6Mapper-Clinet module) for the client nodes. We used Contiki Powertrace [18] to measure the power consumption. The output from the Powertrace application was the total time the different parts of the system were on.

We calculated the energy usage and power consumption using the nominal values, the typical operating conditions of the Tmote sky, shown in table 7.

Table 3. Energy Cnsumption for different Tmote Sky modes (mA).

Tmote sky mode	Symbol	Energy Consumption
MCU on, Radio RX	Listen	*21.8 mA*
MCU on, Radio TX	Transmit	*19.5 mA*
MCU on, Radio off	CPU	*1.8 mA*
MCU idle, Radio off	LPM	*0.0545 mA*

As explained in the introduction section from three main parts of an IoT device, the MCU and Sender/Receiver parts consume energy. It is clear that for any sending or receiving in the radio part, it is essential that the MCU part should be on.
We symbolized the time where the MCU was idle and the radio was off, as low power mode or ***LPM***. The time the MCU is on and the radio is off is referred to as the ***CPU*** time. The time the radio is receiving and transmitting with the MCU on is referred to as ***Listen*** and ***Transmit***, respectively [23]. Supposing 3 V (as a default Voltage) and 8 wakeups per second, the amount of energy consumption for the nodes is calculated using 18.

$$\text{Energy}(\text{mJ}) = \begin{pmatrix} \text{Transmit*19.5 mA} \\ \text{+Listen*21.5 mA +} \\ \text{CPU*1.8 mA +} \\ \text{LPM*0.0545 mA} \end{pmatrix} \text{*3 V} / (32768) \qquad (18)$$

We calculated this equation for all the network nodes and by dividing by the total number of nodes gives us the per node average energy consumption. Like our previous experiments, we calculated the per node energy consumption in terms of data collection interval. Another reason for this is that the energy consumption of the *Data Collection* module is directly related to the time intervals of data retrieval. The experiments were conducted in three states, where the simple RPL, RPL with Data Collection module, and RPL with *Data Collection* and *6Mapper* modules were running. Figure 10 shows the results obtained.
As it is illustrated, in a shorter Data Collection time interval, Data Collection and 6Mapper modules consume much more energy compared with the larger ones. For the latter, it does not make much difference by the pure RPL.

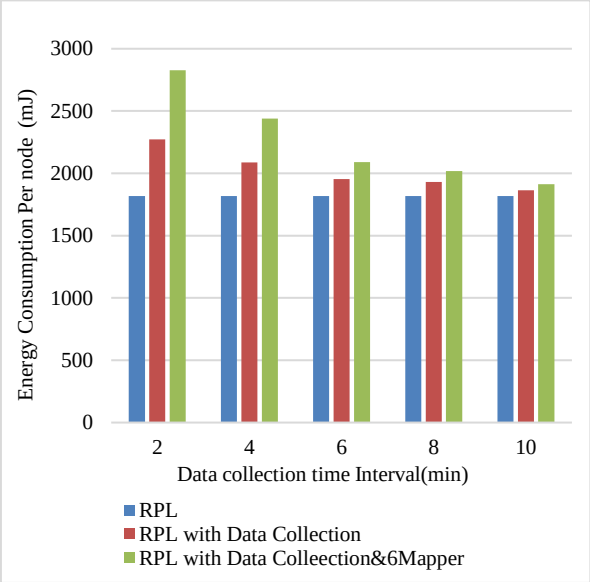


Figure 10. Comparison of energy Consumption in client nodes for different time intervals in in RPL routing protocol and RPL with Data Collection and 6Mapper client module.

6. Conclusion

In this research work, a distributed IDS architecture was proposed to detect the sinkhole routing attack on the RPL-based IoT networks, s aimed to improve a true detection rate and reduce the false alarms. The proposed approach was able to classify the network data and detect the sinkhole attack by creating the network profile and using the data mining techniques. According to our experiments, among the three employed classification models, the Decision Tree had the highest level of precision than SVM and Bayesian, regardless of the network size. Out experiments were conducted in terms of data collection time intervals in which the lower interval produced lower false alarms and a higher detection rate. Also comprehensive experiments were conducted for investigating the client node energy consumptions, which showed that a lower data collection time interval consumed more energy in comparison to the higher intervals. In this work, by checking the suspicious alarms, we were able to reduce the produced false alarms significantly. It can be considered as an advantage of our suggested model that can improve the detection precision and save the computational resources. Also due to employing the anomaly-based detection technique, this approach can be generalized and used for more routing attacks.

References

[1] Zarpelão, B. B., Miani, R. S., Kawakani, C. T., & de Alvarenga, S. C. (2017). A survey of intrusion detection in Internet of Things. Journal of Network and Computer Applications, vol. 84, pp. 25–37.

[2] Cervantes, C., Poplade, D., Nogueira, M., & Santos, A. (2015, May). Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things. 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM).

[3] Airehrour, D., Gutierrez, J., & Ray, S. K. (2016). Secure routing for internet of things: A survey. Journal of Network and Computer Applications, vol. 66, pp. 198–213. https://doi.org/10.1016/j.jnca.2016.03.006.

[4] Raza, S., Wallgren, L., & Voigt, T. (2013). SVELTE: Real-time intrusion detection in the Internet of Things. Ad Hoc Networks, vol. 11, no. 8, pp. 2661–2674. https://doi.org/10.1016/j.adhoc.2013.04.014.

[5] Bostani, H., & Sheikhan, M. (2017). Hybrid of anomaly-based and specification-based IDS for Internet of Things using unsupervised OPF based on MapReduce approach. Computer Communications, vol. 98, pp. 52–71.

[6] Zhou H. (2013). The internet of things in the cloud: A middleware perspective. Boca Raton: CRC Press, Taylor & Francis Group.

[7] Sheng, Z., Yang, S., Yu, Y., Vasilakos, A., Mccann, J., & Leung, K. (2013). A survey on the ietf protocol suite for the internet of things: standards, challenges, and opportunities. IEEE Wireless Communications, vol. 20, no. 6, pp. 91–98.

[8] Ahmadi Livani, M., Abadi, M., Alikhany, M., & Yadollahzadeh Tabari, M. (2013). Outlier detection in wireless sensor networks using distributed principal component analysis. Journal of AI and Data Mining, vol. 1, no. 1, pp. 1-11.

[9] Sathish Kumar, J., & R. Patel, D. (2014). A Survey on Internet of Things: Security and Privacy Issues. International Journal of Computer Applications, vol. 90, no. 11, pp. 20--26.

[10] Pongle, P., & Chavan, G. (2015). Real Time Intrusion and Wormhole Attack Detection in Internet of Things. International Journal of Computer Applications, vol. 121, no. 9, pp.1–9.

[11] Le, A., Loo, J., Luo, Y., & Lasebae, A. (2011, October). Specification-based IDS for securing RPL from topology attacks. 2011 IFIP Wireless Days (WD). 2011 IFIP Wireless Days (WD).

[12] Anthea Mayzaud, Remi Badonnel, & Isabelle Chrisment. (2016). A Taxonomy of Attacks in RPL-based Internet of Things. International Journal of Network Security, vol. 18, no. 3.

[13] Le, A., Loo, J., Chai, K., & Aiash, M. (2016). A Specification-Based IDS for Detecting Attacks on RPL-Based Network Topology. Information, vol. 7, no. 2, 25. https://doi.org/10.3390/info7020025.

[14] Wallgren, L., Raza, S., & Voigt, T. (2013). Routing Attacks and Countermeasures in the RPL-Based Internet

of Things. International Journal of Distributed Sensor Networks, vol. 9, no. 8, 794326.

[15] Krimmling, J., & Peter, S. (2014, October). Integration and evaluation of intrusion detection for CoAP in smart city applications. 2014 IEEE Conference on Communications and Network Security. 2014 IEEE Conference on Communications and Network Security (CNS).

[16] Van Poucke, S., Zhang, Z., Roest, M., Vukicevic, M., Beran, M., Lauwereins, B., Zheng, M.-H., Henskens, Y., Lancé, M., & Marcus, A. (2016). Normalization methods in time series of platelet function assays. Medicine, vol. 95, no. 28, e4188. doi: 10.1097/MD.0000000000004188.

[17] Pham, B. T., Jaafari, A., Prakash, I., & Bui, D. T. (2018). A novel hybrid intelligent model of support vector machines and the MultiBoost ensemble for landslide susceptibility modeling. Bulletin of Engineering Geology and the Environment, vol. 78, no. 4, pp. 2865–2886.

[18] Mukherjee, A., Mondal, S., Chaki, N., & Khatua, S. (2018). Naive Bayes and Decision Tree Classifier for Streaming Data Using HBase. In Advances in Intelligent Systems and Computing (pp. 105–116). Springer Singapore.

[19] Kugler, P., Nordhus, P., & Eskofier, B. (2013, May). Shimmer, Cooja and Contiki: A new toolset for the simulation of on-node signal processing algorithms. 2013 IEEE International Conference on Body Sensor Networks. 2013 IEEE International Conference on Body Sensor Networks (BSN).

[20] Molisch, A. F., Balakrishnan, K., Chong, C. C., Emami, S., Fort, A., Karedal, J & Siwiak, K. (2004). IEEE 802.15. 4a channel model-final report. IEEE P802, vol. 15, no. 04, 0662.

[21] Cavanaugh, J. E., & Neath, A. A. (2019). The Akaike information criterion: Background, derivation, properties, application, interpretation, and refinements. Wiley Interdisciplinary Reviews: Computational Statistics, vol. 11, no. 3, e1460.

[22] Pham, B. T., Prakash, I., Singh, S. K., Shirzadi, A., Shahabi, H., & Bui, D. T. (2019). Landslide susceptibility modeling using Reduced Error Pruning Trees and different ensemble techniques: Hybrid machine learning approaches. Catena, vol. 175, pp. 203-218.

[23] El-Azouzi, R., Menasche, D. S., Sabir, E., De Pellegrini, F., & Benjillali, M. (Eds.). (2016). Advances in Ubiquitous Networking 2: Proceedings of the UNet'16 (vol. 397). Springer.