International Journal of Critical Infrastructure Protection 34 (2021) 100436

Contents lists available at ScienceDirect

International Journal of Critical Infrastructure Protection

journal homepage: www.elsevier.com/locate/ijcip

Fault-tolerant AI-driven Intrusion Detection System for the Internet of Things

Faiza Medjek[a,b,*], Djamel Tandjaoui[a], Nabil Djedjig[a,b], Imed Romdhani[c]

[a] Research Centre on Scientific and Technical Information (CERIST), 03, Rue des Freres Aissou, Ben Aknoun, Algiers, Algeria
[b] Departement Informatique, Faculte des Sciences Exactes, Universite de Bejaia, Bejaia 06000, Algeria
[c] School of Computing, Edinburgh Napier University, 10 Colinton Road, EH10 5DT Edinburgh, UK

ARTICLE INFO

Article history:
Received 31 December 2019
Revised 12 October 2020
Accepted 30 March 2021
Available online 7 April 2021

Keywords:
RPL security
IoT security
IDS
Machine Learning
Deep Learning
Critical infrastructure

ABSTRACT

Internet of Things (IoT) has emerged as a key component of all advanced critical infrastructures. However, with the challenging nature of IoT, new security breaches have been introduced, especially against the Routing Protocol for Low-power and Lossy Networks (RPL). Artificial-Intelligence-based technologies can be used to provide insights to deal with IoT's security issues. In this paper, we describe the initial stages of developing, a new Intrusion Detection System using Machine Learning (ML) to detect routing attacks against RPL. We first simulate the routing attacks and capture the traffic for different topologies. We then process the traffic and generate large 2-class and multi-class datasets. We select a set of significant features for each attack, and we use this set to train different classifiers to make the IDS. The experiments with 5-fold cross-validation demonstrated that decision tree (DT), random forests (RF), and K-Nearest Neighbours (KNN) achieved good results of more than 99% value for accuracy, precision, recall, and *F*1-score metrics, and RF has achieved the lowest fitting time. On the other hand, Deep Learning (DL) model, MLP, Naïve Bayes (NB), and Logistic Regression (LR) have shown significantly lower performance.

1. Introduction

Critical infrastructures (CIs) cover various socio-economic sectors such as healthcare, agriculture, industry, gas and water distribution, transportation, energy, communications, information technology, etc. CIs are continuously changing and adapting to changes in technology. Indeed, Cyber-Physical Systems (CPS) and the Internet of Things (IoT) have emerged as core components in all advanced Cis, such as Industry 4.0 [1,2]. Since CIs are vital to daily human lives, their protection from cyber-attacks by malicious entities that cause significant impacts on the targeted CIs and their services is a serious concern. Consequently, to secure CIs, it is necessary to secure IoT networks [3].

IoT [4] consists of physical objects, usually known as things (devices) that sense, collect, and might process CIs related information. On one side, these objects are resource-constrained as they are powered by batteries and have limited computation and storage capability. On the other side, billions of these devices are interconnected and connected to the Internet under lossy and noisy communication environments such as Wi-Fi, ZigBee, Bluetooth, LoRa, GSM, WiMAX or GPRS. IoT applications have emerged in several aspects. Nevertheless, the IoT's networks rise challenges in designing efficient and secure routing protocols [5,6]. Several efforts have been made by standardisation entities to specify efficient routing protocols for the IoT. Finally, the IPv6 Routing Protocol for Low Power and Lossy Networks (RPL) [7] was designed and standardised by the IETF ROLL working group to overcome the routing challenges underpinning IoT networks. RPL specification considers limitations in both the energy power and the computational capabilities of such networks.

Besides the different characteristics of IoT components, the rapid growth of IoT applications and the increasing number of smart objects in IoT networks result in producing a massive amount of data and traffic leading to increase the IoT's vulnerabilities, and consequently, the RPL's threats [5,6]. Although the RPL specification introduces mechanisms aiming to achieve confidentiality, integrity and replay protection through control messages encryption, local and global repairs and loops detection, RPL is still susceptible to internal attacks [6]. Indeed, there are vulnerabilities from inside the RPL network that go beyond the encryption and authentication defence for the RPL communications [8,9]. In such cases, Intrusion Detection Systems (IDSs) are required as a second line of defence, where IDSs analyse activities and nodes' behaviour to detect intruders that are trying to disrupt the network.

* Corresponding author.
E-mail address: medjek-f@dtri.cerist.dz (F. Medjek).

https://doi.org/10.1016/j.ijcip.2021.100436
1874-5482/

Even though there are several methods to implement an IDS, artificial intelligence-based technologies, such as Machine Learning (ML) techniques are highly recommended as IDS since they are ideal for classification problems. Indeed, ML can predict the expected behaviour of a system by learning from previous experiences without explicit programming. Therefore, ML can be applied at RPL nodes, fog/edge nodes and (or) cloud nodes to extract and analyse from large-scale data, and hence detect malicious behaviour. Consequently, the AI-assisted security analysis approaches might transform the end-to-end IoT security into an intelligence-based security system.

In this paper, different ML algorithms and a Deep Learning (DL) model are explored to develop an efficient IDS for RPL to detect and classify unseen routing attacks. A comprehensive evaluation of several experiments of these ML and DL classifiers are shown on the developed datasets. In this paper, our contributions are as follows.

- We implement common attacks against RPL under different topologies.
- We generate new 2-class (i.e., NoAttack-Attack) and multi-class (i.e., NoAttak-6 attacks) datasets.
- We demonstrate with empirical results that specific ML algorithms perform better on the developed datasets then the sequential Deep Learning (DL) model presented in the literature [10].
- We introduce a new IDS scheme for securing the RPL-based networks in the context of Industry 4.0.

The paper is organised as follows. Section 2 presents basic understanding of Industry 4.0 and RPL. In Section 3, we overview the materials and methodology used in this work. First, we present the ML and DL models and the metrics used for the models' evaluation. Second, we detail the simulation settings and the features' selection process. In Section 4, we present and discuss the results. Section 5 introduces our ML-based IDS. Section 6 sketches the ML and DL IDSs' related work to secure RPL and IoT networks. Finally, Section 7 rises conclusions and gives future works.

2. Background

2.1. Industry 4.0

Industry 4.0 is a critical infrastructure sector. It is a new industrial revolution that has widely attracted researchers, manufacturers, and smart application developers. According to the literature, Cyber-Physical Systems (CPS), Internet of Things (IoT), big data, artificial intelligence, cloud computing, and other paradigms are requirements and parts of the visionary concept of Industry 4.0 [1,2]. Fig. 1 depicts the key technologies used to support Industry4.0's platform in the form of four linked layers.

- Objects layer: This layer is composed of machines, robots, devices, actuators, and sensors. Sensors are integrated into all physical aspects such as machines and robots to connect the physical things with virtual models. Machines in the Industry 4.0 factory are autonomous systems that can make their own decisions based on tools from the other layers, such as ML algorithms.
- Network layer: One of the underpinning features of Industry 4.0 is the complete connectivity between everything (i.e., people, process, data, and things). The data handling systems connect to the wireless sensors network and aggregate outputs, while Internet getaways use Wi-Fi and wired networks to perform further processing. IoT provides advanced connectivity of systems, services, and physical objects to one another and to the Internet via wired and wireless technologies, which permits to create customised services for the end user's needs.
- Cloud layer: One crucial aspect of Industry 4.0 is handling a considerable amount of data collected from machines, processes, and products. This data revolution demands new techniques and technologies for efficient collection, storage, retrieval, communication, and real-time analysis. Industry 4.0 uses cloud technologies (e.g., servers, repositories, and databases) to store and process data. Besides, Big data analytics and artificial intelligence techniques (ML, DL, etc.) are used in various areas of the Industry 4.0, such as quality management, preventive maintenance, fault diagnosis and prognosis, decision-making, etc.
- Application layer: The application layer relies on all the previous layers. It represents the management and enterprise planning level where reside the different applications that allow the decision-makers and managers to monitor all levels of the business from manufacturing, to sales, to purchasing, to finance and payroll, and of course intelligent security management.

2.2. The routing protocol for low-power and lossy networks

RPL [7] is a distance vector routing protocol that organises the physical network into a logical representation as a Directed Acyclic Graph (DAG) to route traffic/packets. The DAG is composed of one or multiple DODAGs (Destination Oriented DAGs) with one root per DODAG. Each root, called border router (BR), is connected to the Internet, and other potential roots (BRs) via a backbone. Each device/node in the DODAG has many attributes such as an IPv6 address (ID), a list of parents with one preferred-parent, a list of discovered neighbours, and a Rank. The Rank of a node identifies the node's position relative to the BR, respecting the rule that the parent has a lower Rank than the node itself. Specifically, the Rank values should increase from the BR towards the leaf nodes, and decrease from the leaf nodes toward the BR.

RPL supports point-to-point (P2P between nodes), multipoint-to-point (MP2P from nodes to the BR) and point-to-multipoint (P2MP from the BR to nodes) traffics. Packets should be transmitted either upward (MP2P) towards the BR or downward (P2MP) towards leaf nodes, respecting the Rank rule as defined in [7]. When a node receives a packet upward, the sender must have a Rank higher than that node and vice versa, when a node receives a packet downward, the sender must have a Rank lower than that node. Fig. 2 a shows a sample of an RPL topology with three DODAGs, where Ranks increase from the BR towards leaf nodes.

RPL uses specific control messages and a trickle mechanism to construct and maintain the DODAG. The trickle algorithm regulates the transmission rate of control messages. It enables data traffic to discover and fix routing inconsistencies quickly. The DODAG Information Object (DIO) messages are used to carry the relevant information and configuration parameters that enable a node to construct and maintain the DODAG. DIO messages convey node/link metrics and constraints (e.g., node energy, hop count, throughput, latency, link colour, and ETX -Expected Transmission Count) and the Objective Function (OF) to use to optimise the path construction and to calculate node Rank. The DODAG Information Solicitation (DIS) messages are used to discover neighbourhood and network topology. Finally, the DODAG Destination Advertisement Object (DAO) messages allow nodes to propagate their destination information upward along the DODAG to the BR so that the downward routes from the BR to its associated nodes can be constructed and updated (i.e., routing tables construction and update).

A low-power and lossy network may run multiple logically independent instances of RPL concurrently. Each such instance may serve different and potentially antagonistic constraints and OFs. An RPL node may belong to multiple RPL Instances, and it may act as a router in some and as a leaf in others. An RPL Instance is a set of

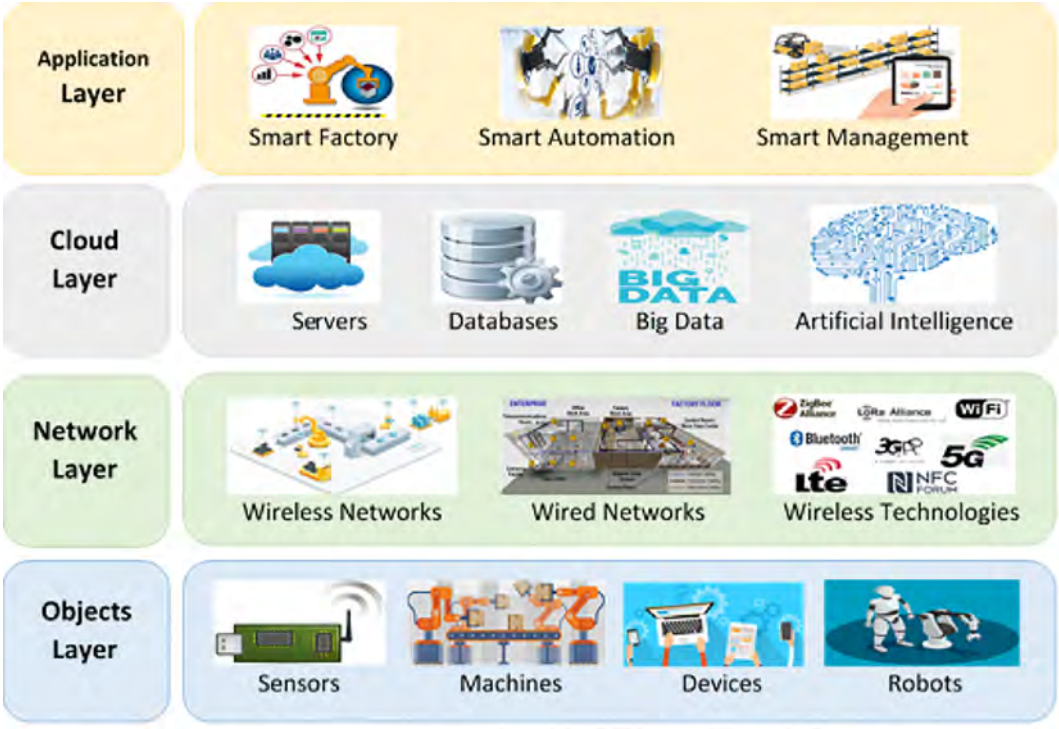


Fig. 1. Industry 4.0 components.

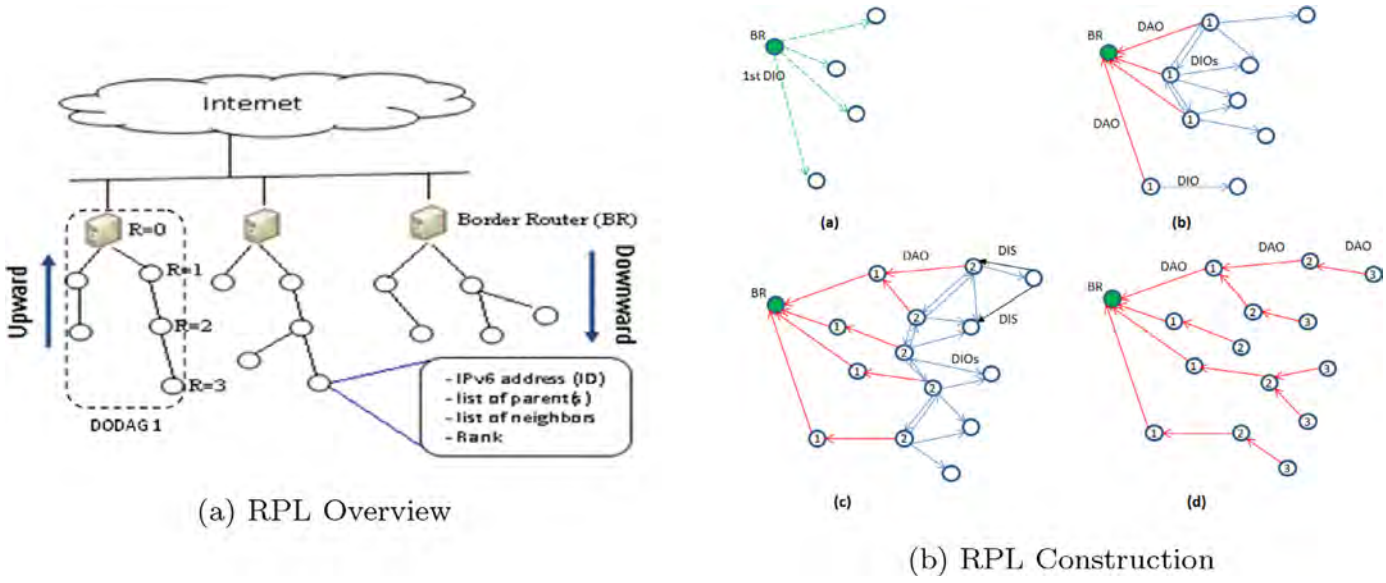


Fig. 2. The routing protocol for low power and lossy networks.

one or more DODAGs that share the same RPL Instance Identifier (RPLInstanceID). DODAGs with the same RPLInstanceID share the same OF.

Fig. 2 b portrays an example of the steps to follow to construct the RPL network. (a) the BR broadcasts an initial DIO message containing configurations such as its Rank (default equal to 0), the RPLInstanceID, the DODAG ID, the DODAG Version, the OF, Trickle timer variables, and the metrics/constraints to use. (b) When a node receives a DIO message from the BR, it selects the BR as its parent, calculates its Rank (R=1), sends a DAO to its parent, and broadcasts an updated DIO to its neighbours. (c) On receiving DIOs from nodes of Rank 1, each neighbour selects a set of parents, selects a preferred parent of Rank 1, calculates its Rank (R=2), sends a DAO to its parent, and broadcasts a DIO to its neighbours. (d) All neighbouring nodes repeat the process until each node joins the RPL network. Once the construction is completed, the maintenance begins respecting the Trickle timer. Thus, in the steady-case, the interval of the trickle timer increases, and the transmission rate will be slowed (i.e., fewer control messages). Otherwise, if there are inconsistencies (e.g., altered DIO messages, a node joining the DODAG, etc.), which involve changes in the topology, the Trickle timer will be reset to a lower value, and the transmission rate will be fastened.

The characteristics of RPL and its functioning makes it vulnerable to existing and newly designed threats. Section 3.1 presents vulnerabilities of RPL that adversaries used to trigger attacks.

3. Materials and methods

3.1. RPL security issues

Various RPL attacks have been analysed in the literature, precisely, Rank attacks, Neighbour attack, DAO attacks, DIS attack, Version number attack, Local repair attack, HelloFlooding attacks, Selective forwarding attack, Sinkhole and Blackhole attacks, Wormhole attack, and Sybil and CloneID attacks [9,11]. Most of the efforts to secure RPL focused on detecting and/or countering Rank, HelloFlooding, Selective forwarding, Sinkhole, Blackhole, Version number, and Wormhole attacks as they represent the most harmful attacks [12]. In this paper, we investigated the detection of the following six attacks.

3.1.1. Decreased Rank (DR) attack

In the DR attack, an adversary node illegitimately advertises a better Rank equal to a lower value throw DIO messages. As a consequence, the neighbour nodes may update their routing table using DAO messages and select the attacker as their new preferred parent. As a result, the malicious node may manage and manipulate more network traffic, and thus trigger other attacks such as eavesdropping, deleting and modifying data.

3.1.2. Sinkhole (SH) attack

In the SH attack, the malicious node advertises itself as the BR (best path) in order to be chosen as a preferred parent by its neighbours, and thus to route traffic through it similarly to DR.

3.1.3. Blackhole (BH) attack

In the BH attack, the intruder drops all control and data packets routed through it. This attack can be considered as a DoS attack. The BH attack is even more dangerous if combined with the DR or SH attacks since the attacker is in a position where huge traffic is routed through it. This attack increases the number of exchanged DIO messages leading to instability of the network and data packets delay.

3.1.4. Selective forwarding (SF) attack

In the SF attack, a misbehaving node aggressively drops data packets and forwards only the control messages traffic leading to a DoS attack. The SF attack is a particular case of the BH attack.

3.1.5. Helloflooding (HF) attack

According to the RPL functioning, when a new node aims to join the DODAG, it sends DIS messages. In the HF attack also known as DIS attack, the malicious node multicasts periodically DIS messages to its neighbours that have to reset their trickle timers and send DIO messages. As a result, the network is overloaded with fake control messages.

3.1.6. Version number (VN) attack

Each time a rebuilding (i.e., a global repair) of the DODAG is necessary, the DODAG version number field (in the DIO message) is incremented by the root (BR) and propagated unchanged down the DODAG graph. Because there is no mechanism in RPL to protect the version number field from modification, the malicious node triggers VN attack by illegitimately increasing the version number of the DOGAG, which triggers the global repair mechanism, forcing other nodes to update their routing tables, and thus reconstructing the RPL topology from scratch.

3.2. Methods for anomaly detection

Since the aim of an IDS is to decide whether a packet either belongs to normal or attack traffic, intrusion detection is considered as a classification problem. Thus, IDS implementation can be based on different Machine Learning (ML) classifiers. Indeed, ML-based IDS can change its execution strategy as it acquires new information. This property makes ML desirable to use for any situation. There exist four groups of ML algorithms: supervised, unsupervised, semi-supervised, and reinforcement learnings. In this paper, we focus on the supervised algorithms, which consist of a training phase and a testing phase. At the training phase, the algorithms learn the relationship between the input values (i.e., training data) and the labels (e.g., the label 0 for normal behaviour, and the label 1 four malicious behaviour). At the testing phase, the algorithms try to predict the output values (i.e., classes, labels) of the testing data. In the following, we present the ML classifiers and the DL model investigated in this study. The ML classifiers represent the frequently applied algorithm for intrusion detection.

3.2.1. Decision tree (DT)

DT classifier represents the standard for partition-based models. DT main idea is to "*break up a complex decision to into a union of several simpler decisions, hoping the final solution obtained would resemble the intended desired solution*" [13]. Hence, DT splits data into many branch-like segments such as in the tree structure, leaves represent classifications also known as labels, intermediate nodes represent features, and branches are conjunctions of features that lead to classifications.

3.2.2. Random forests (RF)

RF classifier [14] constructs a figured large number of uncorrelated DTs. Each DT predicts a classification for a sampled input data from the original dataset. Furthermore, each DT selects a subset of features from the original set for the fullest growing at each node randomly. Finally, RF collects the predictions and selects the most voted one as the final classification. RF is extensively used in data science since it has high accuracy level, speed, and stability, it is easy to parametrise, robust against overfitting, it can be applied to large-scale datasets, and is not sensitive to noise in datasets. RF is also handy for feature selection as it determines the importance of different features during the classification process.

3.2.3. K-Nearest Neighbour (KNN)

KNN classifier [15] is a non-parametric supervised ML technique that relies on similarity or distance in feature space to classify samples. In KNN, testing sample (i.e., unlabelled data) is assigned to the class that is most frequently occurred amongst the K nearest neighbours in the training set. The number K, as the square root of the total number of samples in the training dataset. KNN is widely used because it is simple, very scalable, and very fast to converge.

3.2.4. Naïve Bayes (NB)

NB or Bayesian learning [16] is a probabilistic classifier based on probabilities of hypotheses. NB applies Bayes' theorem to calculate probabilities, with the strong assumption that features are independent given class variable, which means that the probability of one feature does not affect the probability of the other one. A prior probability is assigned to each candidate hypothesis based on prior knowledge. NB uses training samples to increase or decrease the probability of a hypothesis to be correct. It classifies a testing sample by assigning the most probable target class.

3.2.5. Multi-layer perceptron (MLP)

MLP Classifier [17] is a feed-forward Artificial Neural Network (ANN) model connecting multiple hidden layers in a directed graph, where each layer is fully connected to the next one. An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a non-linear activation function. MLP employs the backpropagation supervised learning technique for training the

Table 1
Confusion matrix.

	Predicted 0	Predicted 1
Actual 0	True Positive (TP)	False Negative (FN)
Actual 1	False Positive (FP)	True Negative (TN)

network. It maps the set of input data to a suitable output set inspired by the way biological nervous systems of the brain process information.

3.2.6. Logistic regression (LR)

LR classifier [18] is a mathematical modelling approach used to describe the relationship of a dependent variable (i.e., outcome) and one or more independent variables (i.e., predictors). LR is applicable when the outcome is a binary variable that contains data coded as 1 (yes, success) or 0 (no, failure). Thus, the LR model predicts $P(Y{=}1)$ as a function of X.

3.2.7. Deep learning classifier (DL)

DL is a particular ML technique that implements the learning process elaborating the data through Artificial Neural Networks (ANNs) [19,20]. ANNs have self-learning capabilities that enable them to produce better results as more data becomes available. An ANN has artificial neurons interconnected through at least three layers: the input layer, one or many hidden layers, and the output layer. Each neuron has inputs (e.g., features from a dataset or outputs from other nodes) and produces a single output which can be sent to multiple other neurons. The outputs of the neurons in the output layer return the final result, such as the classification of a sample as an attack or not. Each connection in the network is assigned a weight that represents its relative importance. The weights are adjusted during training to find patterns and make better predictions. Several hyperparameter need to be set before the learning process begins, such as the number of hidden layers and the number of neurons per layer, the activation function, the learning rate, batch size, and the number of epochs. ANNs are categorised into supervised (e.g., MLP) and unsupervised learning (e.g., DL) [20]. The adjective "deep" in DL comes from the fact that the classification is conducted by training data, with many layers in hierarchical networks with unsupervised learning.

There are several DL models based on the used architectures and techniques [21]. The Deep Recurrent Neural Network (RNN), the Deep Auto-Encoder, the Deep Boltzmann Machine (DBM), and the Deep Believe networks (DBN) belong to the Generative Architecture (GA) class. The Deep Convolutional Neural Network and the Deep Recurrent Neural Network (when the output is taken to be the predicted input data in the future) belong to the Discriminative Architecture (GA) class. GA models are graphical models that "*are intended to characterise the high-order correlation properties of the observed or visible data for pattern analysis or synthesis purposes, and/or characterise the joint statistical distributions of the visible data and their associated classes.*" [21]. DA models "*are intended to directly provide discriminative power for pattern classification, often by characterising the posterior distributions of classes conditioned on the visible data.*" [21].

3.3. Performance evaluation metrics

A clean and unambiguous way to present the prediction results of a classifier is to use a Confusion Matrix (CM). For a binary classification problem such as intrusion detection, the matrix has two rows and two columns as depicted in Table 1, where 0 and 1 are labels for normal and attack, respectively. Across the top are the predicted class labels and down the side are the observed class labels. Each cell contains the number of predictions made by the classifier that fall into that cell. TP (normal samples correctly classified), TN (attack samples correctly classified), FN (attack samples incorrectly classified) and FP (normal samples incorrectly classified) are used to determine the different metrics to assess the performance of a classifier [22]. In this paper, we use accuracy, precision, recall and $F1$-score metrics to compare the classifiers presented in Section 3.2.

Accuracy Accuracy is defined as the ratio of correct predictions to the total number of all predictions, as in Eq. (1).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

Precision Precision is also called the Positive Predictive Value (PPV). It is the number of positive predictions divided by the total number of positive class values predictions, as in Eq. (2).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

Recall The recall metric is also called sensitivity, True Positive Rate (TPR), or Detection Rate (DR). It is defined as the ratio of positive predictions to the number of positive class values in the test data, as in Eq. (3).

$$\text{Recall} = \text{DR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

F1-score $F1$-score is also called the F score or the F measure. It is defined as the harmonic mean of precision and recall. It, thus, conveys the balance between the precision and the recall as in Eq. (4).

$$\text{F1} - \text{score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

3.4. Dataset generation

There exist datasets available publicly that are commonly used for intrusion detection research such as KDDCUP 99, ISCX and NSL-KDD. Nevertheless, one gap in the field of IoT networks is the unavailability (privacy) of developed RPL related datasets such as the IRAD dataset [10]. Indeed, building an RPL ML-based IDS requires one or multiple RPL-related datasets, where the model can learn from. The datasets shall represent normal and malicious RPL-based traffic. In our experiments, we elaborated two types of datasets: one dataset for each attack and one multi-class dataset for all the attacks defined in Section 3.1.

To generate the datasets, we developed the necessary scripts following the methodology in [10]. Indeed, to obtain results that can be compared to those presented in [10], the same features extraction and selection procedure have been used. Nevertheless, the authors in [10] did not provide a detailed description of the simulation settings used to generate the datasets.

3.4.1. Simulations setting

We used the simulator Cooja-Contiki for our experiments. Contiki is a powerful tool for building 6LoWPAN-IoT networks and has realistic results. Because the simulation of large topologies requires high memory and computing power, we deployed the simulator on a virtual machine with 48 GB RAM and 8 VCPUs on a server. We simulated three topologies of 25, 50 and 100 TelosB nodes (Sky motes) with one root. Firstly without any attack, and then with 2, 4, and 10 malicious nodes, respectively. We implemented each attack separately. We used UDGM with distance loss as link failure model as it provides a real-world emulation of the lossy links and shared media collision among RPL's nodes. The simulations duration is one hour with one packet of 46 bytes sent each 10 s. We used RPL-collect package for packets generation.

Furthermore, we used the cooja-radio-logger-headless plug-in to capture traffic and generate PCAP files. We exploited the PCAP

Table 2
Simulation parameters.

Parameter	Value
Simulator	Cooja-Contiki 3.0
Simulation time	3600 s (1 h)
MAC	ContikiMAC
Number of nodes	25, 50, 100
Number of malicious nodes	2, 4, 10
Transmission range	50 m
Interference range	60 m
TX, RX	100%, 90%
Network area	300×300 m^2
Link failure model	UDGM with distance loss
Traffic rate	One packet sent every 10 s
Packet size	46 bytes

files to generate the datasets and extract features. Table 2 summarises the parameters of the simulations.

3.4.2. Feature engineering and selection

We implemented Python scripts for datasets' generation. We used Pandas, Numpy, and Scikit-learn libraries for features engineering, extraction and selection, Matplotlib and Seaborn libraries for data visualisation and plotting, and Scikit-learn and Keras libraries for data analysis.

Features extraction and transformation. We used Wireshark tool to transform the generated PCAP files to CVS files. The latter were pre-processed using Python scripts. Initially, each CVS dataset includes six features: the packet sequence number (N^o), simulation time (Time), source IPv6 address of the node (Source), destination IPv6 address of the node (Destination), the packet length (Length), and the packet information (Info). Firstly, we simplified data in the CVS files such that nominal attributes are converted into discrete ones. For instance, source and destination IPv6 addresses were reduced to nodes' ID, and the broadcast address to the value 9999. In addition, the packet information DIS, DAO, DIO, Ack, and UDP was encoded 1, 2, 3, 4, and 5, respectively. To calculate datasets' feature values correctly, we first sorted the CVS files by simulation time. Then, we divided all the simulation time into periods of one-second duration (i.e., 1000 ms windows); to make better use of the extracted features for intrusion detection.

- When DR, SH, and VN attacks are triggered, normal nodes add the malicious node to their routing table and send their packets through it. Consequently, the number of received packets of the malicious node increases, as well as DIO and DAO counts.
- When HF attack is performed, the malicious node sends illegitimacy DIS message pushing the neighbouring nodes to exchange control messages. As a result, the number of transmitted packets increases, as well as DIS packets count.
- When BH and SF attacks are triggered, the number of DIO and DAO increases while the number of data packets (i.e., transmitted and received packets) decreases.

As a result of the analysis above, eleven extra features have been calculated per one-second window duration, and have been added to the datasets. These features are: Transmission Rate (TR), Reception Rate (RR), TR/RR, Transmission Total Time (TTT), Reception Total Time (RTT), Transmission Average Time (ATT), Reception Average Time (ART), Source count (SrcCount), Destination count (DestCount), DAO packets count (DAO), DIS packets count (DIS), and DIO packets counts (DIO).

The normal traffic was labelled as 0 while the traffic with malicious behaviour (i.e., each attack related dataset) was labelled as 1. Indeed, the datasets generated from networks where an attack was triggered were labelled 1 as the entire networks were affected by the malicious activities.

Feature normalisation is used to make convergence quicker and limit the influence of small or large values in the training set, thus increasing the performance of the learning algorithm. We implemented a Python script to mix the normal and malicious datasets for each topology. We firstly applied quantile transformation to the datasets to adjust feature values distribution to normal distribution. We secondly used min-max scaling to scale all feature values to the range [0,1]. Afterwards, we concatenated all datasets resulting from dif-

Table 3
The generated datasets for the IDS use.

Datasets	Scenarios	Nb nodes	Attackers	Packets counts
Decreased	DR_25	25	2	503,232
Rank	DR_50	50	4	5,134,640
(DR)	DR_100	100	10	7,466,588 **Total = 1,3104,460**
				Total = 1,3104,460
Sinkhole	SH_25	25	2	513,653
(SH)	SH_50	50	4	873,932
	SH_100	100	10	2,735,976
				Total = 4,123,561
Blackhole	BH_25	25	2	499,951
(BH)	BH_50	50	4	899,333
	BH_50	50	4	6,727,132
				Total = 8,126,416
Selective	SF_25	25	2	506,444
Forwarding	SF_50	50	4	891,441
(SF)	SF_100	100	10	3,409,921
				Total = 4,807,806
HelloFlooding	HF_25	25	2	842,548
(HF)	HF_50	50	4	2,088,476
	HF_100	100	10	10,263,539
				Total = 13,194,563
Version	VN_25	25	2	2,718,314
Number	VN_50	50	4	3,585,999
(VN)	VN_100	100	10	15,205,283
				Total = 21,509,596
Multi-Class	MC	25/50/100	2/4/10	**Total = 51,326,396**

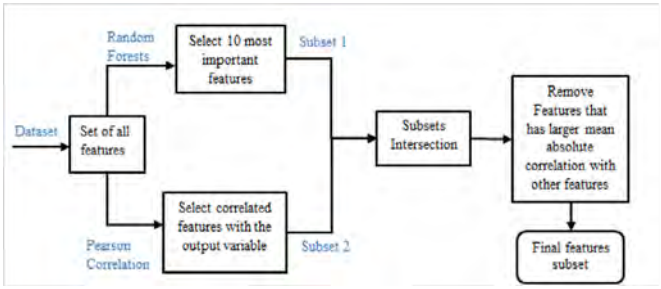


Fig. 3. Feature selection process.

ferent topologies (three topologies) of each routing attack. As a result, we got six datasets, as detailed in Table 3.

- *Features selection.* After the features extraction stage, CVS files are processed to select relevant attributes, which is one of the core concepts in ML. This stage identifies and removes unneeded, irrelevant, weakly relevant, and redundant features from the dataset that do not contribute to the accuracy of the classifier or may decrease its accuracy. In other words, this step would permit to increase accuracy while reducing training time[1] and avoiding bias and model overfitting.
  There exist several feature selection methods in the literature, the filter methods, the wrapper methods, and the embedded methods. We avoid using wrapper methods because they are computationally costly and are not the most efficient in massive datasets. In this paper, we combined an embedded method using the Random Forests classifier, and a filter method applying Pearson correlation, where correlation states how the features are related to each other and to the output variable. Fig. 3 summarises the features selection process.

  Step 1. We first carried out RF-feature-importance function to highlight the ten most important features for each dataset. When RF classifier is trained, it evaluates each attribute to create splits and gives a score for each feature of the dataset; the higher the score more relevant is the feature towards the output variable (i.e., Label 0 or 1). In this approach, for the tree building process, only a subset of the data samples is chosen with replacement, which is known as bootstrap aggregating or bagging. Nevertheless, this is a biased approach, as it tends to inflate the importance of continuous features or high-cardinality categorical variables. To reduce selection bias, we used cross-validation[2] for feature selection, as reported in [23]. In Fig. 4, the red bars depict the features' importance of the Forests, along-with their inter-trees variability for the Selective-Forwarding dataset, where the *x*-axis represents the features indexes, and the *y*-axis represents the importance values.

  Step 2. Secondly, we applied the correlation matrix using the Pearson correlation method on the original features set. Fig. 5 portrays the correlation matrix for the Selective-Forwarding dataset. We checked the correlation of each feature with the output variable, and we selected a subset of features using a threshold of 0.3 for the correlation.

  Step 3. Afterwards, we selected a new subset of features that represents the intersection of both subsets from the previous two steps.

  Step 4. According to [24], redundant features should be eliminated since they affect the speed and the accuracy of learning algorithms. Consequently, in the final step, we checked the correlation of selected features subset with each other using a threshold of 0.8 for the correlation. If these attributes are correlated with each other, we kept only one of them and dropped the rest.

Initially, in each dataset, there are 17 features: Time, Source, Destination, Length, Info, TR, RR, TR/RR, SrcCount, DestCount, TTT, RTT, ATT, ART, DAO, DIS, DIO. After performing the steps mentioned above, the total number of attributes is reduced for each dataset, as presented in Table 4.

3.4.3. Multi-class dataset generation

To generate a multi-class dataset, firstly, we performed the features extraction steps from Section 3.4.2. Secondly, we labelled the normal traffic as 0, and the traffic with malicious behaviour as 1, 2, 3, 4, 5, and 6 for BH, SH, HF, DR, VN, and SF, respectively. Afterwards, we performed the features transformation steps. We mixed the normal and malicious datasets of all attacks for each topology. We then applied quantile transformation to the datasets to adjust feature values distribution to normal distribution. We used min-max scaling to scale all feature values to the range [0, 1]. Next, we concatenated all datasets resulting from different topologies and got one 7-class dataset for all attacks and topologies, as in Table 3. Finally, we performed the features selection steps and got the ones in Table 4.

4. Classifiers evaluation and discussion

To determine the best performing algorithm to classify RPL routing attacks using our datasets, we evaluated the performance of the ML and DL algorithms for 2-class (i.e., normal and attack) and 7-class (i.e., normal and six attacks) datasets. In this paper, we implemented the following classifiers:

- DecisionTreeClassifier,
- RandomForestClassifier with number of estimators (i.e., the number of trees in the forest) equal to 10 and 100,
- KNeighborsClassifier with k=1 and k=10,
- Gaussian Naive Bayes classifier (GaussianNB),
- MLPClassifier with one hidden layer, 100 neurons, and the 'relu' activation function,
- and LogisticRegression (Logit) classifier with 'sag' solver.
- sequential DL model using Keras[3] library. The model includes 1 input layer, 5 hidden layers and 1 output layer. 50 neurons are used in the first and fifth layers, whereas 100 neurons are used in the second and fourth layers, and 300 neurons in the third layer. The DL model is the same as the one in [10]. Our choice is made to be used as a benchmark to compare ML algorithms to the DL model proposed in [10].

4.1. Two-class classification results

For ML models, we applied 5-fold cross-validation on the datasets. For the DL model, we used 90% of the data for training and 10% for model evaluation. We saved all trained models for possible future use. We assessed the accuracy, precision, recall, and F1_score performance metrics for each algorithm and obtained the results on each dataset, as presented in Fig. 6.

[1] Throughout this paper, we use the terms "training time" and "fitting time" interchangeably.

[2] In the cross-validation process, the data is splitting into k equal folds (k=5). The model is trained on $k-1$ folds and evaluated on the remaining holdout fold. These two steps are performed k times, each time holding out a different fold. Finally, the performance are aggregated across all k folds.

[3] Keras is an open-source neural-network library written in Python.

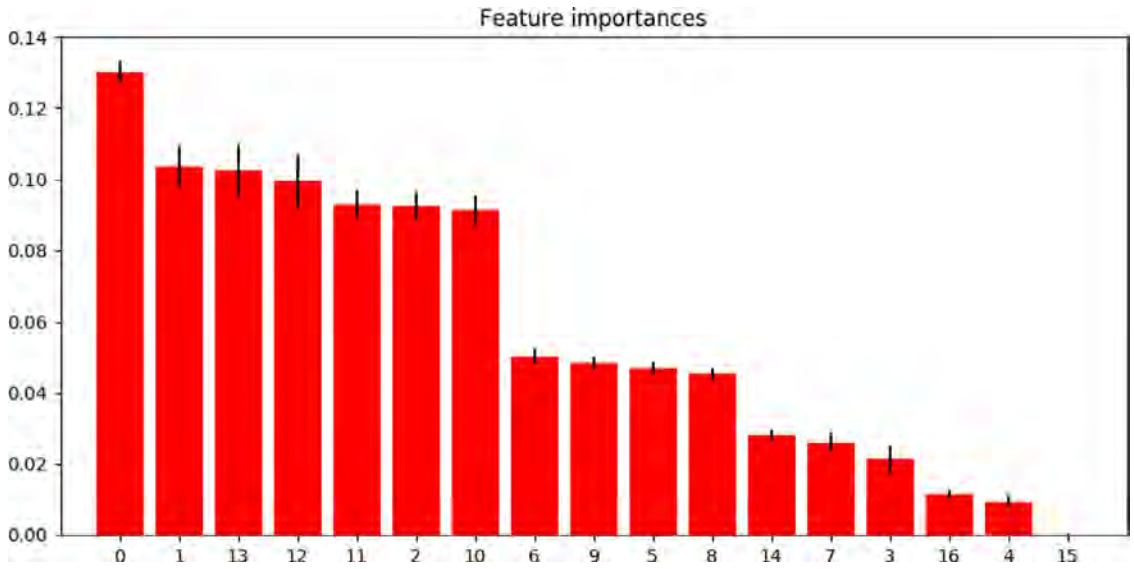


Fig. 4. Feature importance datagram for SF dataset.

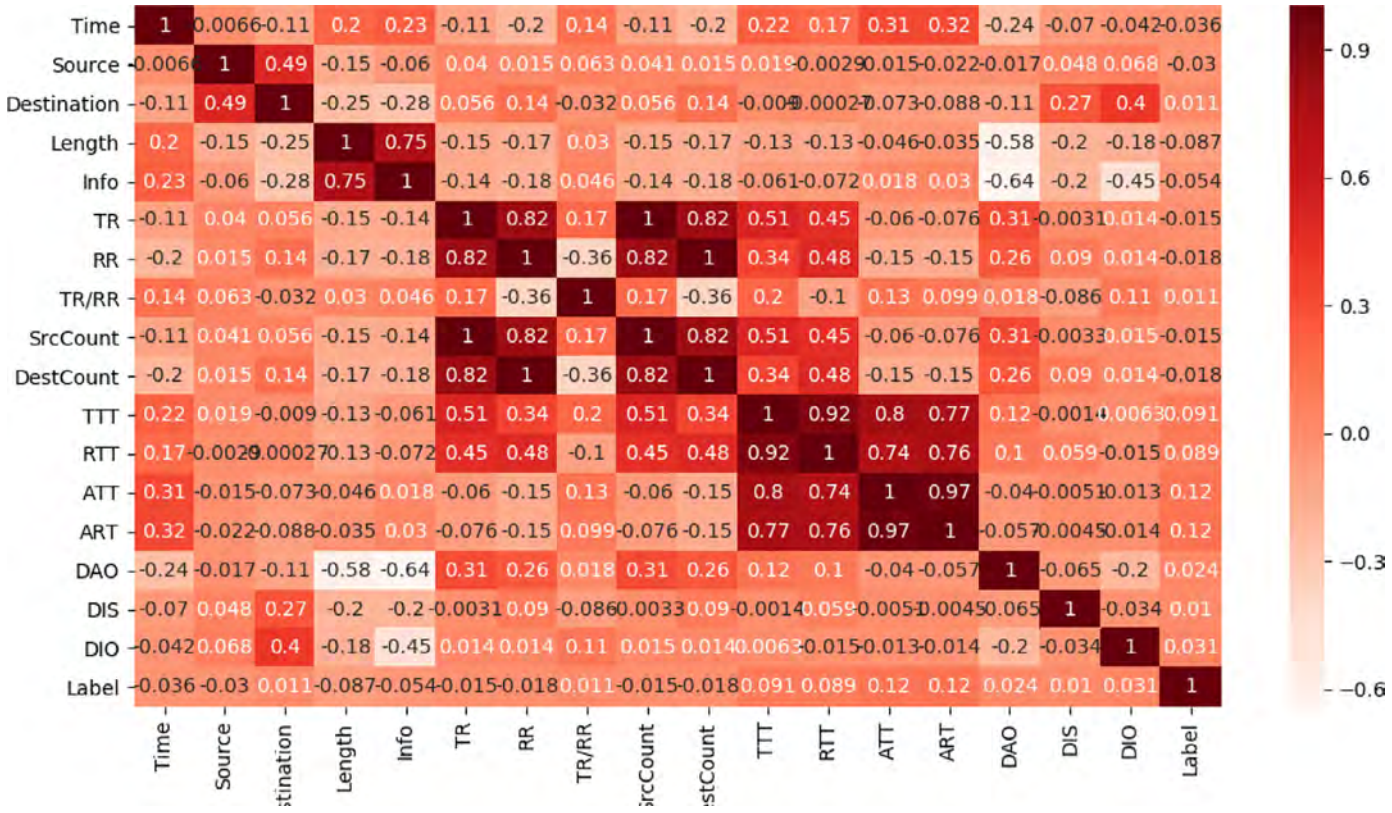


Fig. 5. Correlations between different features for the SF dataset.

Table 4
Features per dataset after data pre-possessing.

Datasets	Features	Count	Discard
DR	DAO, Length, TTT, RR, Dst, TR, Src	7	10
SH	Dst, Time, TAR, ATT, Src, RTT, TTT, TR/RR, RR, TR	10	7
BH	Dst, Length, RR, DIS, DIO, TR/RR, TTT	7	10
SF	Time, ART, ATT, Src, RTT, TTT, Dst, RR, TR, DAO	10	7
HF	RR, DIS, DAO, Length, Dst, TR	6	11
VN	ART, RR, TR/RR, Dst, RTT, TTT, DAO	7	10
Multi-class	Dst, Time, ART, ATT, Src, RTT, TTT, TR/RR, RR, TR, DAO, DIO, DIS	13	4

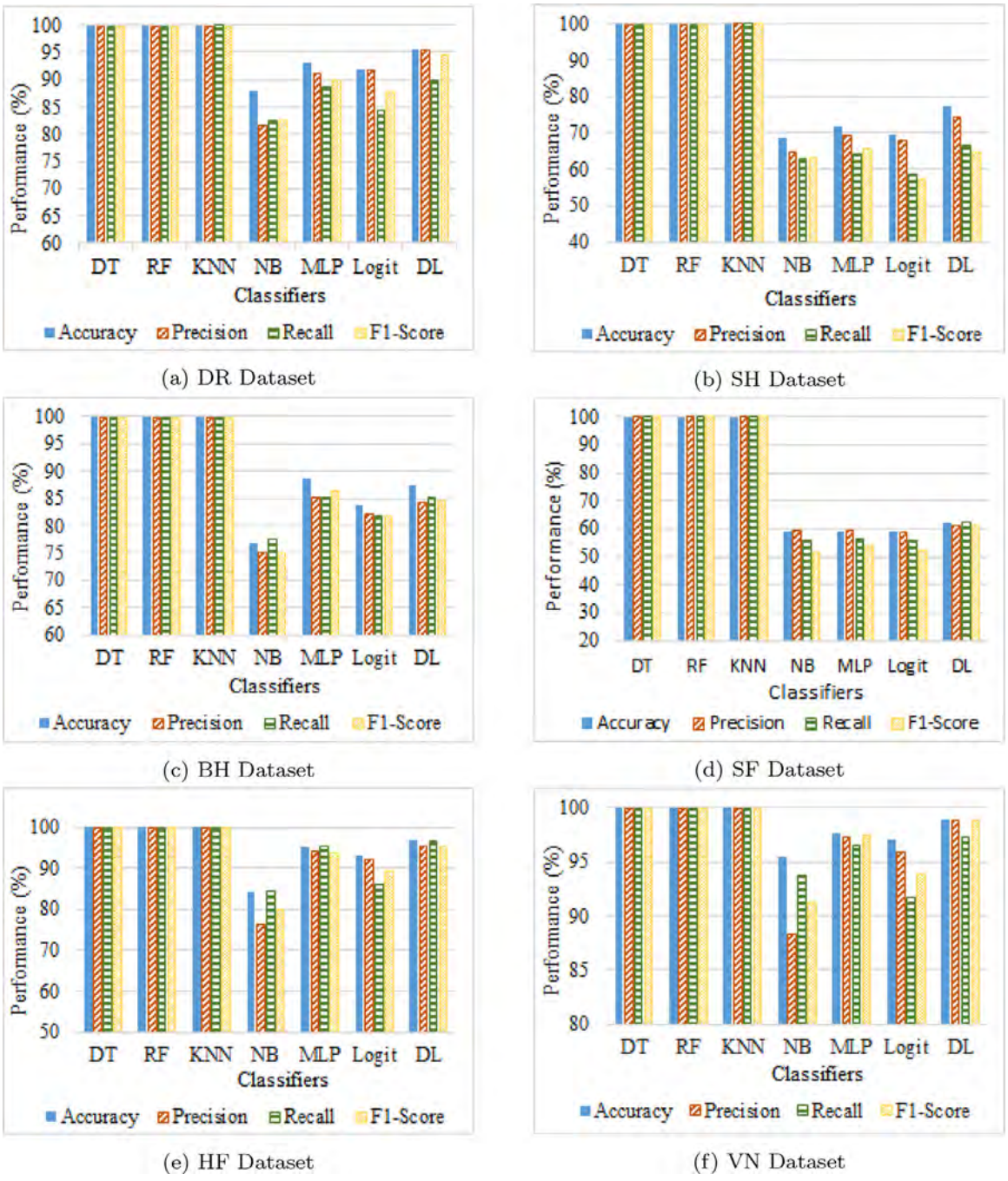


Fig. 6. Classifiers performance per dataset for 60 min simulation time.

We observe that DT, RF and KNN showed better performance for all metrics as compared to NB, MLP, Logit, and DL. We also notice that the same three algorithms achieved a classification accuracy rate of more than 99% for both 2-class and 7-class classifications, as shown in Figs. 6 and 10.

Nevertheless, compared with DT and RF, KNN is very slow to converge and gets significantly slower as the number of independent variables increases (i.e., the dataset size increases), as shown in Fig. 7. Besides, the MLP classifier also required a longer fitting time but gave better performance when the size of the dataset decreases, as can be seen in Fig. 8. On the other hand, the DL classifier takes the longest training time, and the training time increases with the dataset size.

When we compare the two ANN-based classifiers (i.e., the MLP classifier with one hidden layer and 100 neurons, and the DL model in Fig. 6), we find that for almost all datasets, the latter gives slightly better performance except for BH attack where the former gives relatively better results (88.56% Vs 87.6%). These results are due to the fact that the DL model has more capacity than MLP (i.e., the number of layers and neurons in the DL model is higher than in MLP classifier).

From another side, a DL model with increased capacity tends to yield better accuracy up to a point at which the model stops improving [25]. As an example, Fig. 9 a and b draw the DL model's loss and accuracy, respectively, of the VN dataset for 10 min' simulation time. We notice that the accuracy of the DL

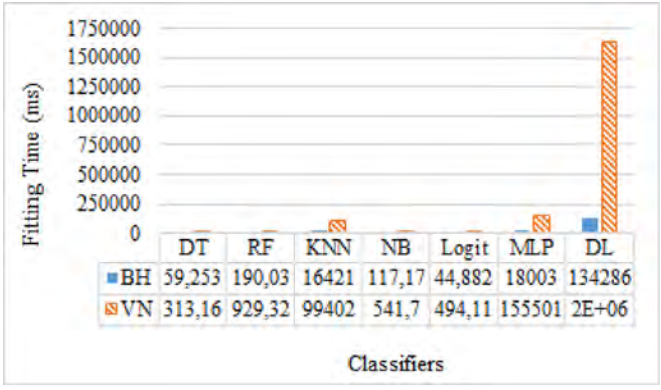


Fig. 7. Fitting time for BH dataset Vs VN dataset for 10 min simulation time.

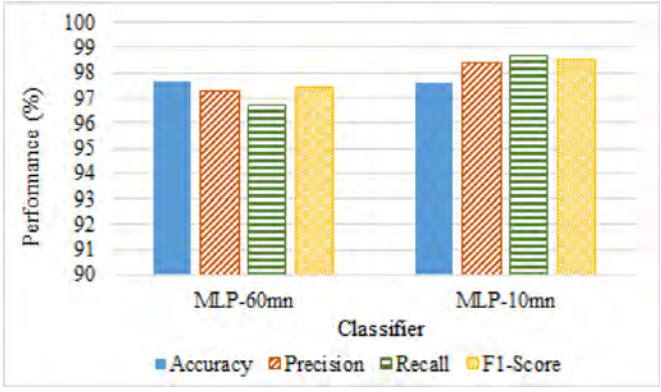


Fig. 8. MLP performance for VN datasets: 10 min Vs 60 min simulation time.

model for a larger dataset (60 min' simulation time in Fig. 6 f) is higher compared to a smaller dataset (10 min' simulation time in Fig. 9 b). Nevertheless, although DL methods are getting lots of attention lately because of their promising results in several areas, such as signal processing, natural language processing, and image recognition, the biggest the DL model, the more computational resources it requires and the longer it takes to train which is not suitable for intrusion detection in IoT networks. Furthermore, the present evaluation results confirm that the DL methods are not desirable for intrusion detection for IoT networks.

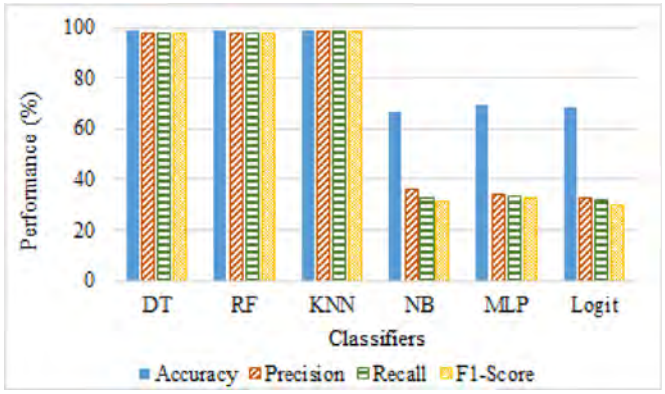


Fig. 10. 7-class classification performance.

4.2. Multi-class classification results

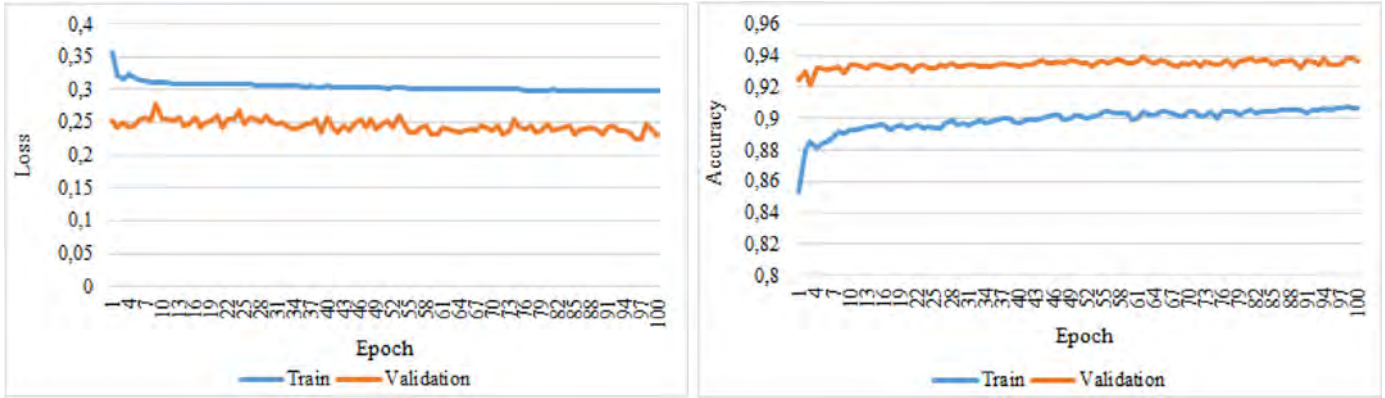
We used the multi-class dataset in Table 3. Because the 7-class dataset is too large (51,326,396 packets), we shuffled it and used half of the generated dataset to test the ML and DL classifiers. We got the performance plotted in Fig. 10. The results show that KNN has an accuracy of 99% with a detection rate of 98%. RF and DT take the second position with an accuracy of 98% and a detection rate of 98%. On one other hand, compared with 2-class classification, MPL, Logit, and NB gave a mediocre performance with precision, recall and *F*1-score around 35%. Regarding DL, the model did not converge after three weeks of execution, which makes it impractical as IDS for IoT networks, especially for real-time needs.

From the obtained results, we conclude that in terms of performance and fitting time, RF is more suitable for intrusion detection for RPL-based networks.

5. RF-Based Intrusion Detection System for RPL (RF-IDSR)

5.1. System model and assumption

As presented in Section 2.1, IoT networks play an important role in the establishment of Industry 4.0 and thus need to be secure. Because RPL is the de facto routing protocol for IoT networks, we introduce the RF-Based Intrusion Detection System for RPL, namely, RF-IDSR. RF-IDSR is an anomaly-based IDS that uses RF model to detect attacks. Fig. 11 depicts a graphical representation of the IDS architecture. RF-IDSR is a hybrid-IDS (depending on the location of the IDS modules) relying on the collaboration of three actors:



(a) Model Loss (b) Model Accuracy

Fig. 9. DL-model for VN dataset.

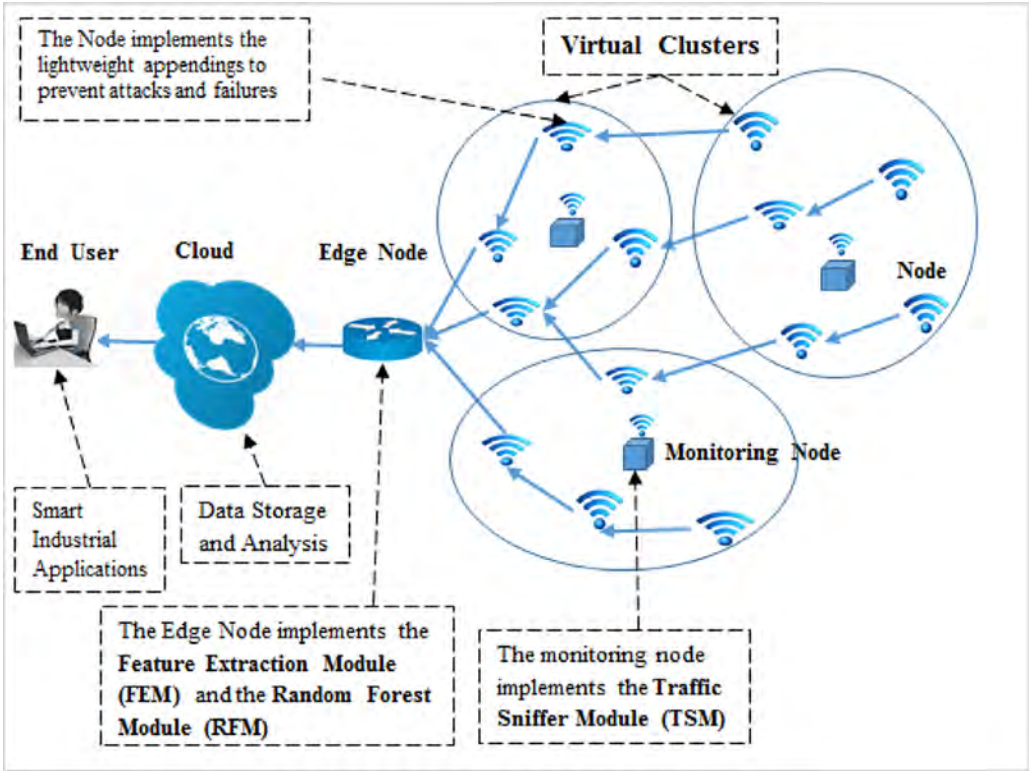


Fig. 11. RF-IDSR architecture.

the DODAG border router (BR) or edge node, the Monitoring Nodes (MNs), and the sensor nodes. In this paper, we assume the following:

- The edge node is a powerful node, which has not processing and energy consumption constraints.
- The edge node is always trusted and cannot be compromised by an adversary.
- The edge node shares a secret key with each node to encrypt and secure data packets.
- Two RPL instances coexist:
  - The first instance, called sensor network (SN), is an RPL-based network composed of both resource-constrained sensor devices and more powerful devices. The SN is used to route the sensed data to the edge node. Independently of the RF-IDSR, to prevent attacks and failures, each sensor node implements lightweight appending for the RPL protocol, as in Section 5.3.
  - The second instance, called the monitoring network, is composed of a few more powerful nodes (i.e., monitoring nodes). The MNs are powerful machines and devices, which have not resource-constraints. Notably, the MNs do not have battery depletion issues.
- We assume the MNs are synchronised with each other and with the edge node. The MNs are selected based on the geographical location of the nodes in the first RPL instance. Indeed, the architecture can be seen as a set of virtual clusters of nodes from both RPL instances, where MNs are cluster heads with enough resources (e.g., energy power) for intrusion detection purpose.

5.2. Attacks detection using RF model

One objective of the paper is to create a predictive model to classify the RPL-based network packets into two classes: Normal or Attack and identify the attack using the multi-class dataset. In this paper, we select RF as the classifier to be used to detect RPL routing attacks because of its high accuracy of prediction, computational and time efficiency, and its ability to select features according to their importance [14]. Furthermore, referring to our study, we found that compared to KNN that gives better accuracy than RF, the latter returns results in a shorter time, especially for large datasets such as DR, HF, and VN datasets. As presented in Fig. 11, the RF-IDSR is composed of three modules defined as follows.

- Distributed module: is a module placed at each MN of the second RPL instance.
  1. Traffic Sniffer Module (TSM): TSM implements Algorithm 1. The MNs use TSM to sniff the traffic from the first RPL instance. MNs must timestamp packets as they are received by their radio, generate PCAP files each 1-min window (one PCAP file per MN), and send the PCAP files to the edge node through the second RPL instance paths.

Algorithm 1 Monitoring Algorithm

1. Sniff packets
2. Timestamp packets
3. Generate a PCAP file for the last 1-min window of the sniffed packets
4. Send the PCAP file to the edge node

return PCAP

- Cenralised modules: are modules placed at the edge node.
  1. Feature Extraction Module (FEM): FEM implements Algorithm 2. FEM allows the edge node to gather all received PCAP files from the MNs, concatenate them, and process them to extract features and generate new data (equivalent to the testing data in the dataset) as presented previously in Section 3.4.2. The edge node may use the cloud to store and process the data.

Algorithm 2 Datasets Generation Algorithm

Require: PCAP files received from MNs
1. New-PCAP = Concatenate PCAP files
2. New-PCAP = Delete duplicated packets from New-PCAP
3. dataset.csv =Ttransform New-PCAP to CSV format
4. New-Data = Execute steps in Section 3.4.2
   a. Sorting dataset.csv by Time attribute
   b. Feature Transformation (Source, Destination, Info, etc.)
   c. Feature Extraction within 1-second window size
   d. Feature Selection as summarised in Fig. 3
5. Store New-Data in the cloud (optional)

return New-Data

2. Random Forests Module (RFM): RFM implements the trained RF model (from Section 4) and Algorithm 3. The edge node uses RFM to evaluate the new data generated from FEM using the trained RF model and gives predictions. Furthermore, RFM implements an update process to update, periodically, the RF model using the new data from FEM. The last point permits to enrich the learning algorithm, and thus to detect new threats. RFM may use the cloud to update the RF model and execute Algorithm 3 (see Fig. 11). The alarm may be sent to the end-user (e.g., the network administrator) and notifications to the sensor nodes to discard the malicious nodes from participating in the network operations.

Algorithm 3 Anomaly Detection Algorithm

Require: New-Data from FEM
1. Load trained RF model
2. Scores = Predict outcomes on New-Data
5. If Intrusion, raise an alarm and send notifications

return Scores

5.3. Attacks and failure prevention

In addition to the presented RF-IDSR, we propose considering three lightweight appending to RPL to prevent the HelloFlooding, version number, and global repair attacks, as well as network failure.

5.3.1. Helloflooding and DIS attacks prevention

We propose to redefine the DIO message to prevent attacks based on multicast messages such as the DIS (see Section 3.1.5) and the SybM [26] attacks. We introduce a new mechanism in RPL based on the RFC 3810 [27]. More specifically, we use the Maximum Response Code (MRC) field to reduce responses to multicast messages. In the DIO Base Object, there are two unused bytes: Flags and Reserved fields. In our approach, we use these two bytes as one MRC field, as depicted in Fig. 12. The edge node sets the MRC field. The pseudocode in Algorithm 4 summarises the proposed mechanism. The presented solution allows for reducing significantly the overhead generated by such attacks.

Algorithm 4 HelloFlooding and DIS Attacks Prevention

Require: MRC
Calculate Maximum Response Delay (MRD) using MRC as defined in the RFC 3810 [27]
if a node receives a multicast DIS message **then**
  It delays the response (sending a DIO) by a random amount of time in the range [0, MRD]
  if the number of response from its neighbours reaches the threshold defined by the edge node [7] **then**
    It cancels the pre-programmed response
  else
    once the delay has expired, it reinitialises the trickle timer and sends back a DIO
  end if
end if

5.3.2. Global repair and version number attacks prevention and detection

We authenticate the DODAG version field of the DIO message to detect and prevent the version number and global repair attacks. In our approach, the edge node uses a one-way hash chain to generate sequence numbers for the DODAG version field. A one-way hash chain is a sequence of numbers, V_i $(0 \leq i \leq n)$, generated by a one-way hash function F as in Eq. (5), where V_i is a random number generated by the edge node, and F function is the same for the edge node and all nodes in the network.

$$\forall i, 0 \leq i < n : V_i = F(V_{i+1}) \tag{5}$$

We assume that the one-way hash chain is stocked in the edge node. In addition, the first value of the one-way hash chain used to generate DODAG versions is uploaded securely into the nodes before deployment. Besides, when a new node is deployed in the network, it is pre-loaded with the first unused value of the chain.

Each global repair is identified with a DODAG version (V_i) that is the last delivered value of the one-way hash chain. Consequently, the nodes of the network can verify the new DODAG version V_{i+1} through checking whether $V_i = F(V_{i+1})$, where V_i is the previous DODAG version. On the other hand, the nodes cannot calculate the following DODAG version since the security of the one-way hash chain concept is based on the fact that knowing V_i, it is computationally infeasible to determine V_{i+1}. The pseudocode in Algorithm 5 summarises the proposed procedure.

Algorithm 5 Global Repair and Version Number Attacks Prevention

Require: V_0 (i.e., the first value of the hash chain uploaded in all nodes before deployment)V_i (i.e., the DODAG version of the ith global repair and the last delivered value of the one-way hash chain);F (i.e., the one-way hash function implemented in both the edge node and the network's nodes);V_i $(0 \leq i \leq n)$ (i.e., the one-way hash chain stocked in the edge node);
if a node receives a DIO with a new DODAG version value V_{i+1} knowing that V_i is the current DODAG version **then**
  it calculates $F(V_{i+1})$
  if $V_i = F(V_{i+1})$ **then**
    (the node is sure that the edge node updated the DODAG version)
    it reinitialises the trickle timer
    it updates the DODAG version field to V_{i+1} and broadcast a DIO message with the new value
  else
    it discards the received DIO and considers the node from which it receives the DIO as malicious
  end if
end if

5.3.3. Fault and intrusion tolerance

In RPL, a preferred parent (PP) is used by children nodes to forward traffic until detection of a better route, a path failure or an intrusion (i.e., the PP is detected as a malicious node). Thus, other potential parents are rarely used. To enhance the resiliency and security of RPL, our approach merges intrusion-tolerance with

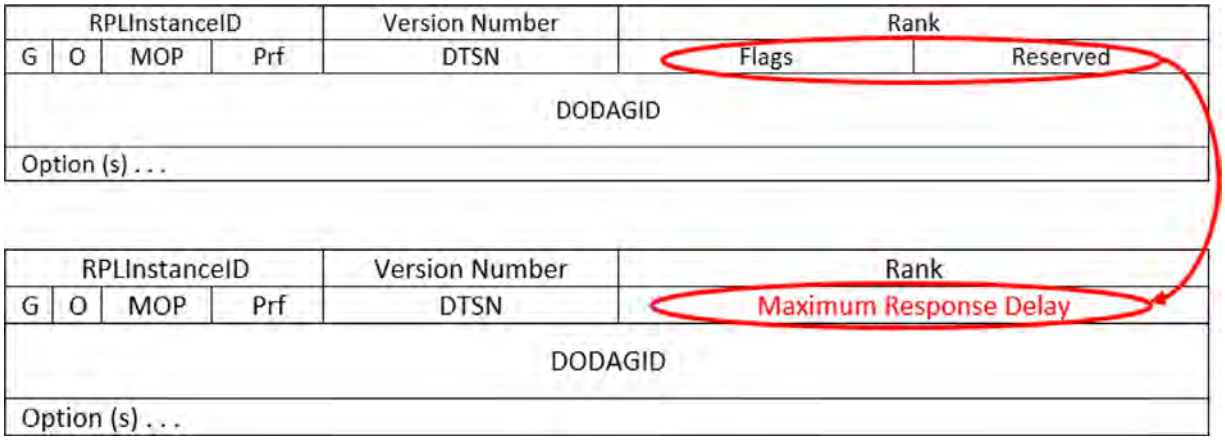


Fig. 12. New DIO message format.

fault-tolerance solutions by considering a multi-path strategy for RPL. We propose to redefine the RPL's objective function in a way to select two random parents through which the traffic is routed. Choosing only two paths reduces the network overhead while increasing the packet delivery ratio and preventing attacks, such as Wormhole, blackhole, selective forwarding and sinkhole. Indeed, a packet at each hop is routed through two potential parents. On the one hand, the parents are selected randomly, and thus, cannot be specifically targeted by an attacker; because, the attacker has no way to know the parents that will forward the traffic. On the other hand, our solution prevents path and node failure because, at each hop, the traffic is forwarded through two random parents.

In RPL, nodes ignore any DIO message from nodes of higher or equal ranks aiming to avoid loops [7]. However, in some cases, a node could have only one or two potential parents. To expand the parents' list and the selection choice, if the number of potential parents is less than or equal to two, our solution adds nodes with the same rank in the parents' list as supplementary parents.

6. Related works

There exist various anomaly-based intrusion detection techniques [28]. In this paper, we give particular attention to ML and DL anomaly-based IDSs for IoT networks. Table 5 summarises related works on ML and DL IDSs for RPL/IoT security.

6.1. ML-based IDSs

Sheikhan and Bostani [29] proposed a hybrid distributed IDS for real-time detection of the sinkhole, selective forwarding and Wormhole attacks. The model is based on the Map-Reduce approach that uses the Optimum-Path Forest (OPF), the Modification of Supervised Optimum-Path Forest (MOPF), and the Optimum Path Forest Clustering (OPFC), to classify nodes. The authors used NSL-KDD dataset. However, their approach rises high false positive and false negative rates.

McDermott and Petrovski [30] presented an experimental comparison of a Multi-Layer Perceptron Backpropagation Neural Network (BPN) and a Support Vector Machine (SVM) classifier to detect Denial of Service (DoS) attacks in Wireless Sensor Networks using the NSL-KDD dataset. The authors concluded that both techniques offer a high true-positive rate and a low false-positive rate, making both of them useful for intrusion detection.

In [31], the authors proposed a compression header analyser based IDS (CHA-IDS) to detect HelloFlood, sinkhole, and Wormhole attacks in an RPL network. The authors used Cooja-Contiki simulator [32] to generate a dataset of 77 features. They used the Best First Search (BFS) and Greedy Stepwise (GS) to perform the features searching, then the Correlation-based Features Selection (CFS) algorithm to evaluate the most significant features. The authors compared MLP, SVM, J48 (i.e., DT), NB, Logistic, and RF classifiers, and the results showed that J48 performs better than other classifiers for that specific configuration. Even though this approach presents a good background for IoT ML-based IDS, the authors considered one topology and a small network of eight nodes.

Anthi et al. [33] proposed an IDS to detect network scanning probing and simple forms of DoS attacks in IoT networks. To generate the dataset, the authors used the software Wireshark to sniff network traffic. They tested several ML classifiers and used the Naive Bayes classifier as it gave the best performance. Although this study is based on a testbed to generate datasets and detect intrusions, the authors used a small network composed of nine devices.

Hasan et al. [34] presented a study comparing several ML methods to detect threats and attacks in IoT infrastructures. The authors concluded that RF classifier performs better than the other one. An open-source dataset of 13 features from kaggle[4] was used. The dataset was gathered from a day of capture from the application layer using four simulated IoT sites. Although this work demonstrated the effectiveness of RF classifier in the context of kaggle's dataset, further studies need to take place to assess its performance for traffic from the network layer.

Authors in [35] proposed an Artificial-Bee-Colony-based classifier to detect flooding attacks in a cloud environment. They used the occurrence of Route Request packets to label nodes as normal or malicious. The proposed solution performed better than NB and DT classifiers, with accuracy values of 92.53%, 95.52%, and 97%, respectively.

6.1.1. Random forests-based IDSs for IoT

Primartha and Tama [36] used three datasets, namely, NSL-KDD, UNSW-NB15, and GPRS, to evaluate the performance of RF for IDS use. The authors assessed 10 RF classifiers with different number of trees (10, 40, 50, 80,..., 800) and RF-800 gave better results. Comparing RF-800 to Random tree+NB tree, DMND, MLP, and NBTree, RF-800 outperforms the other classifiers with 99.57% for accuracy and 0.34% for false alarm rate.

In [37], authors proposed a cloud-based IDS using RF and Neural Network. The IDS receives IoT traffic from the network device, performs features extraction, and classification on the extracted features. RF is used to detect if the data point is classified as an in-

[4] M.-O. Pahl, F.-X. Aubet, DS2OS traffic traces, 2018, (https://www.kaggle.com/francoisxa/ds2ostraffictraces). [Online; accessed 29 December 2019].

Table 5
IDS solutions for RPL/IoT networks 2017–2019... (Part 1).

Works	Algorithm	Dataset	Metrics	Attacks
[29]	Optimum-Path Forest, Supervised Optimum-Path Forest, Optimum Path Forest Clustering, MapReduce	NSL-KDD	DR, FPR, FNR	SH, SF, Wormhole
[30]	MLP-Backpropagation-NN, Support Vector Machine (SVM)	NSL-KDD	TPR, FPR	DoS
[36]	RF, Random tree+NB tree, DMND, MLP, NBTree	NSL-KDD, UNSW-NB15, GPRS	Accuracy, FAR, Significance Tests	DOS
[31]	Best First Search (BFS), Greedy Stepwise (GS), Correlation-based Features Selection (CFS), DT, MLP, SVM, NB, Logistic, RF	Authors' dataset	Accuracy, TPR, FPR, Precision	HF, SH, Wormhole
[33]	Rule-based algorithm, NB	Authors' dataset	Precision, TPR, *F*1-score	Quick/Quick Plus scan, Regular/Intense scan, SYN/UDP Flood
[37]	RF, DNN	UNSW-NB15	Precision, TPR, *F*1-score	–
[38]	Word Embedding, Text-Convolutional NN, RF	ISCX2012	Accuracy, DR, FAR	Infiltration, BFSSH, HttpDoS, DDOS
[39]	Particle Swarm Optimisation (PSO), RF, rotation forest (RoF), DNN	NSL-KDD	Accuracy, Precision, TPR, Significance tests	-
[41]	DNN with three hidden layers	NSL-KDD	Accuracy, Precision, TPR, FAR, *F*1-score	Prob, DoS, U2R, R2L
[10]	DNN with five hidden layers	Authors' RPL-based dataset (IRAD)	Accuracy, precision, recall, *F*1-score, FAR	Rank, HF, VN
[34]	RF, LR, SVM, DT, ANN	kaggle	Accuracy, Precision, TPR, FPR, *F*1-score	DoS, Data Type Probing, Malicious Control, Malicious Operation, Scan, Spying, Wrong Setup
[40]	ExtraTreesClassifier, RF	UNSW-NB15	Accuracy, precision, TPR, *F*1-score, FAR	Botnets
[35]	Defined algorithms, Artificial Bee Colony, NB, DT	Authors' dataset	Accuracy	Flooding
[42]	Random-NN-Gradient Descent Algorithm, SVM, NB, DT, MLP, RF, RF Tree, Recurent-NN, ANN	NSL-KDD	Accuracy, Precision, TRP, FPR	DoS, U2R, R2L, Probe
RF-IDSR	RF, NB, DT, KNN, MLP, RL, DNN	Authors' RPL-based dataset	Accuracy, Precision, TRP, *F*1-score	Rank, SH, BH, SF, HF, VN

trusion or not. Whereas, the Neural Network (i.e., one input layer, several hidden layers and one output layer) is used to categorise the detected intrusion. The authors used UNSW-NB15 dataset. RF gives good results for Precision, Recall, and *F*1-score (99%, 98%, and 98%, respectively).

Authors in [38] proposed TR-IDS, an IDS that uses word embedding and text-convolutional neural network (Text-CNN) techniques to extract features from the payloads in network traffic automatically, and RF for the classification. The authors used ISCX2012 dataset, from which they extracted 27 features to classify infiltration, BFSSH, HttpDoS, and DDOS attacks. They obtained the following performance: 99.13%, 99.26%, and 1.18% for accuracy, DR, and false alarm rate, respectively.

Tama and Rhee [39] proposed an IDS that uses particle swarm optimisation (PSO) for feature selection and RF classifier for attack detection. They used NSL-KDD dataset. 37 features were selected to obtain an accuracy of 99.67%. The model outperformed rotation forest (RoF) and deep neural network (DNN) classifiers.

In [40], authors proposed AD-IoT, an RF-based IDS that monitors IoT traffic in a distributed fog layer to detect IoT Botnets at fog node, and alert the administrator. The authors used UNSW-NB15 dataset. They used ExtraTreesClassifier to reduce the number of features to 12. RF classifier achieved good performance with values of 99.34%, 98%, 98%, 98%, 0.2% for accuracy, precision, recall, *F*1-score, and false alarm rate, respectively.

6.2. DL-based IDSs

One application of DL for intrusion detection in the IoT network is the work in [41]. The authors discussed the detection of Prob, DoS, U2R, and R2L attacks using fog-to-things architecture. They used NSL-KDD dataset with 128 features for detecting four classes of attacks. They also gave a comparison study of a deep neural network model with three hidden layers and a shallow neural network with as results 98.27% and 96.75% in term of accuracy, respectively.

Authors in [10] have also applied a DL approach with five hidden layers to detect RPL routing attacks. The authors generated datasets for decreased rank, HelloFlooding, and version number attacks relaying on several topologies. The obtained performance results in terms of *F*1-score for each dataset are 94.7%, 99%, and 95%, respectively.

Qureshi et al. [42] proposed a deep random neural network-based heuristic Intrusion Detection System (RNN-IDS) for IoTs. They trained RNN-IDS using the Gradient Descent Algorithm (GD). The authors used KDDTrain20 from NSL-KDD dataset to train the classifier with variant learning rates, and with both reduced features (29) and all features (41). The RNN-IDS accuracy reached up to 95.2% and gave better performance than SVM, NB, DT, MLP, and others.

Besides the works in [10] and [31], one major drawback of the works mentioned earlier is using datasets from open-sources that are not designed explicitly for IoT networks. In addition, these studies did not use features that are relevant for RPL-based IoT routing attacks detection. In this paper, we presented a comparative study of both classical ML techniques and the DL model in [10] to propose an IDS for detecting attacks against RPL. Compared to other works, our paper provides much detailed description of the simulation settings used to generate large datasets specially produced for IoT routing attacks. Furthermore, our work gives attention to the classification of multiple classes, which is more challenging than binary classification.

7. Conclusion

In this paper, as a first step, we studied the applicability of ML and DL techniques for intrusion detection in RPL-based IoT networks. We demonstrated that with the selection of the appropriate features, high performance had been achieved. In the 2-class classification, the decision tree (DT), random forests (RF), and K-Nearest Neighbours (KNN) classifiers recorded more than 99% for each of the following metrics: accuracy, precision, recall, and *F*1-score. The recorded detection rate (Recall), precision, and *F*1-score for multi-class classification were more than 98% for the three classifiers, while the KNN accuracy was 99%. Besides, RF recorded the lowest fitting time.

On the other hand, the DL model, MLP, Naïve Bayes (NB), and Logistic Regression (LR) classifiers recorded lower performance. We changed the DL model to get better performance, but the overall classification performance stayed almost the same. Different hidden layers and parameters were used, however no positive impact on the accuracy, recall, precision, and *F*1-score.

The present evaluation results showed that RF is a good classifier for RPL networks threats detection. As a second step, we introduced an RF-based IDS, named RF-IDSR, to provide both fault tolerance and intrusion tolerance and detection for RPL-based Industry 4.0 networks. RF-IDSR uses RF classifier to categorise RPL-based attacks using a multi-class dataset. Furthermore, we presented lightweight appending to RPL to prevent the HelloFlooding, version number, global repair attacks, and network failure.

We plan to implement RF-IDSR and evaluate its performance in both a simulation environment and an experimental environment. We plan to generate new datasets after implementing the appending above and see how they influence the results in the present work. Besides, one future work is to extend our datasets with other RPL's attacks.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is financially supported by the Research Center on Scientific and Technical Information (CERIST).

References

[1] K. Zhou, T. Liu, L. Zhou, Industry 4.0: towards future industrial opportunities and challenges, in: Proceedings of the 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2015, pp. 2147–2152, doi:10.1109/FSKD.2015.7382284.
[2] S.I. Tay, T.C. Lee, N. Hamid, A. Ahmad, An overview of industry 4.0: definition, components, and government initiatives, J. Adv. Res. Dyn. Control Syst. 10 (2018) 1379–1387.
[3] K. Kimani, V. Oduol, K. Langat, Cyber security challenges for IoT-based smart grid networks, Int. J. Crit. Infrastruct. Protect. 25 (2019) 36–49, doi:10.1016/j.ijcip.2019.01.001.
[4] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of Things (IoT): a vision, architectural elements, and future directions, Fut. Gen. Comput. Syst. 29 (7) (2013) 1645–1660, doi:10.1016/j.future.2013.01.010.
[5] J.H. Kim, A survey of IoT security: risks, requirements, trends, and key technologies, J. Ind. Integr. Manag. 2 (02) (2017) 1750008, doi:10.1142/S2424862217500087.
[6] M. Frustaci, P. Pace, G. Aloi, G. Fortino, Evaluating critical security issues of the IoT world: present and future challenges, IEEE Internet Things J. 5 (4) (2017) 2483–2495, doi:10.1109/JIOT.2017.2767291.
[7] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, R. Alexander, RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks, Internet Engineering Task Force, 2012.
[8] T. Tsao, R. Alexander, M. Dohler, V. Daza, A. Lozano, M. Richardson, A Security Threat Analysis for the Routing Protocol for Low-Power and Lossy Networks (RPLs), RFC7416 (2015) 131.
[9] F. Medjek, D. Tandjaoui, I. Romdhani, N. Djedjig, Security threats in the internet of things: RPL's attacks and countermeasures, in: Security and Privacy in Smart Sensor Networks, IGI Global, 2018, pp. 147–178, doi:10.4018/978-1-5225-5736-4.ch008.
[10] F.Y. Yavuz, D. Ünal, E. Gül, Deep learning for detection of routing attacks in the Internet of Things, Int. J. Comput. Intell. Syst. 12 (1) (2018) 39–58, doi:10.2991/ijcis.2018.25905181.
[11] A. Jain, S. Jain, A survey on miscellaneous attacks and countermeasures for RPL routing protocol in IoT, in: Emerging Technologies in Data Mining and Information Security, Springer, 2019, pp. 611–620.
[12] S. Choudhary, N. Kesswani, A survey: intrusion detection techniques for internet of things, Int. J. Inf. Secur. Priv. (IJISP) 13 (1) (2019) 86–105.
[13] S.R. Safavian, D. Landgrebe, A survey of decision tree classifier methodology, IEEE Trans. Syst. Man Cybern. 21 (3) (1991) 660–674, doi:10.1109/21.97458.
[14] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[15] J.M. Keller, M.R. Gray, J.A. Givens, A fuzzy k-nearest neighbor algorithm, IEEE Trans. Syst. Man Cybern. (4) (1985) 580–585, doi:10.1109/TSMC.1985.6313426.
[16] T.M. Mitchell, et al., Machine Learning, 45, McGraw Hill, Burr Ridge, IL, 1997, pp. 870–877.
[17] M. Riedmiller, Advanced supervised learning in multi-layer perceptrons-from backpropagation to adaptive learning algorithms, Comput. Stand. Interfaces 16 (3) (1994) 265–278.
[18] D.W. Hosmer Jr, S. Lemeshow, R.X. Sturdivant, Applied Logistic Regression, 398, John Wiley & Sons, Ltd, 2013, doi:10.1002/9781118548387.
[19] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, Nature 521 (2015) 436–444.
[20] A. Zappone, M.D. Renzo, M. Debbah, Wireless networks design in the era of deep learning: model-based, AI-based, or both? IEEE Trans. Commun. 67 (2019) 7331–7376.
[21] L. Deng, A tutorial survey of architectures, algorithms, and applications for deep learning, APSIPA Trans. Signal Inf. Process. 3 (2014).
[22] M. Hossin, M.N. Sulaiman, A review on evaluation metrics for data classification evaluations, Int. J. Data Min. Knowl. Manag. Process 5 (2) (2015) 1, doi:10.5121/ijdkp.2015.5201.
[23] J. Krawczuk, T. Åukaszuk, The feature selection bias problem in relation to high-dimensional gene data, Artif. Intell. Med. 66 (2016) 63–71, doi:10.1016/j.artmed.2015.11.001.
[24] L. Yu, H. Liu, Efficient feature selection via analysis of relevance and redundancy, J. Mach. Learn. Res. 5 (Oct) (2004) 1205–1224.
[25] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[26] F. Medjek, D. Tandjaoui, I. Romdhani, N. Djedjig, Performance evaluation of RPL protocol under mobile Sybil attacks, in: Proceedings of the 2017 IEEE Trustcom/BigDataSE/ICESS, IEEE, 2017, pp. 1049–1055.
[27] R. Vida, L. Costa, Rfc 3810, Multicast Listener Discovery Version 2 (2004).
[28] M.F. Elrawy, A.I. Awad, H.F.A. Hamed, Intrusion detection systems for IoT-based smart environments: a survey, J. Cloud Comput. 7 (1) (2018) 21, doi:10.1186/s13677-018-0123-6.
[29] M. Sheikhan, H. Bostani, A security mechanism for detecting intrusions in internet of things using selected features based on MI-BGSA, Int. J. Inf. Comput. Technol. Res. 9 (2) (2017) 53–62.

[30] C.D. McDermott, A. Petrovski, Investigation of computational intelligence techniques for intrusion detection in wireless sensor networks, Int. J. Comput. Netw. Commun. 9 (4) (2017).
[31] M.N. Napiah, M.Y.I.B. Idris, R. Ramli, I. Ahmedy, Compression header analyzer intrusion detection system (CHA-IDS) for 6loWPAN communication protocol, IEEE Access 6 (2018) 16623–16638, doi:10.1109/ACCESS.2018.2798626.
[32] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, T. Voigt, Cross-level sensor network simulation with COOJA, in: Proceedings of the 31st IEEE Conference on Local Computer Networks, 2006, pp. 641–648.
[33] E. Anthi, L. Williams, P. Burnap, Pulse: an adaptive intrusion detection for the internet of things, in: Living in the Internet of Things: Cybersecurity of the IoT – 2018, IET, 2018, pp. 1–4, doi:10.1049/cp.2018.0035.
[34] M. Hasan, M.M. Islam, M.I.I. Zarif, M.M.A. Hashem, Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches, Internet Things 7 (2019) 100059, doi:10.1016/j.iot.2019.100059.
[35] S.R. Kalaivani, A. Vikram, G. Gopinath, An effective swarm optimization based intrusion detection classifier system for cloud computing, in: Proceedings of the 5th International Conference on Advanced Computing and Communication Systems (ICACCS), 2019, pp. 185–188.
[36] R. Primartha, B.A. Tama, Anomaly detection using random forest: a performance revisited, in: Proceedings of the 2017 International Conference on Data and Software Engineering (ICoDSE), 2017, pp. 1–6.
[37] T. Mohamed, T. Otsuka, T. Ito, Towards machine learning based IoT intrusion detection service, in: Proceedings of the IEA/AIE, 2018.
[38] E. Min, J. Long, Q. Liu, J. Cui, W. Chen, TR-IDS: anomaly-based intrusion detection through text-convolutional neural network and random forest, Secur. Commun. Netw. 2018 (2018) 4943509:1–4943509:9.
[39] B.A. Tama, K.-H. Rhee, An Integration of PSO-Based Feature Selection and Random Forest for Anomaly Detection in IoT Network, 2018.
[40] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, H. Ming, Ad-IoT: anomaly detection of IoT cyberattacks in smart city using machine learning, in: Proceedings of the 9th IEEE Annual Computing and Communication Workshop and Conference (CCWC), 2019, pp. 0305–0310.
[41] A.A. Diro, N. Chilamkurti, Distributed attack detection scheme using deep learning approach for internet of things, Fut. Gen. Comput. Syst. 82 (2018) 761–768, doi:10.1016/j.future.2017.08.043.
[42] A.-U.-H. Qureshi, H. Larijani, J. Ahmad, N. Mtetwa, A Heuristic Intrusion Detection System for Internet-of-Things (IoT), 2019.