Annals of Telecommunications (2021) 76:27–42
https://doi.org/10.1007/s12243-020-00796-x

Rogue device discrimination in ZigBee networks using wavelet transform and autoencoders

Mohammad Amin Haji Bagheri Fard[1] · **Jean-Yves Chouinard**[1] · **Bernard Lebel**[2]

Received: 12 July 2019 / Accepted: 10 August 2020 / Published online: 18 September 2020

Abstract

In modern wireless systems such as ZigBee, sensitive information which is produced by the network is transmitted through different wired or wireless nodes. Providing the requisites of communication between diverse communication system types, such as mobiles, laptops, and desktop computers, does increase the risk of being attacked by outside nodes. Malicious (or unintentional) threats, such as trying to obtain unauthorized accessibility to the network, increase the requirements of data security against the rogue devices trying to tamper with the identity of authorized devices. In such manner, focusing on Radio Frequency Distinct Native Attributes (RF-DNA) of features extracted from physical layer responses (referred to as preambles) of ZigBee devices, a dataset of distinguishable features of all devices can be produced which can be exploited for the detection and rejection of spoofing/rogue devices. Through this procedure, distinction of devices manufactured by the different/same producer(s) can be realized resulting in an improvement of classification system accuracy. The two most challenging problems in initiating RF-DNA are (1) the mechanism of features extraction in the generation of a dataset in the most effective way for model classification and (2) the design of an efficient model for device discrimination of spoofing/rogue devices. In this paper, we analyze the physical layer features of ZigBee devices and present methods based on deep learning algorithms to achieve high classification accuracy, based on wavelet decomposition and on the autoencoder representation of the original dataset.

Keywords Physical layer · Wireless networks · ZigBee devices · Data preamble · RF-DNA · Autoencoder learning · Wavelet-transform

1 Introduction

In recent decades, the development of wireless communication networks has lead to the use of portable devices anytime and anywhere. This desired wireless device portability for legitimate users, has also lead to vulnerability threats, like eavesdropping of unauthorized listeners, resulting in increasing the risks of information leakage for instance.

✉ M. A. Haji Bagheri Fard
mohammad-amin.haji-bagheri-fard.1@ulaval.ca

Jean-Yves Chouinard
jean-yves.chouinard@gel.ulaval.ca

Bernard Lebel
bernard.lebel@ca.thalesgroup.com

[1] Department of Electrical and Computer Engineering, Université Laval, Quebec City, Canada

[2] Thales Canada Inc. - TRT, Quebec City, Canada

Consequently, different security protocols such as Wi-Fi Protected Access (WPA) and WPA2 provided a higher degree of security for short or high range radio communication systems over the last years [1]. In 2019, the Wi-Fi Alliance presented a new standard, WPA3, enhancing the security level in communication systems [2]. One of the communication protocols is ZigBee, introduced in 1999 [3], which is considered an attractive wireless system for commercial and military applications, because of its low cost and low complexity [4].

Despite the advantages in security protocols and systems in the last decade, fast evolution of physical attacks by rogue (unauthorized) guests (unseen devices that attempt to access the wireless network by falsifying their bit-level credentials to match the identity of the known/authorized devices) to the ZigBee networks makes physical layer attacks prevention and countermeasures very complicated, because of the intrinsic importance of physical layer attacks in comparison with cryptanalytic attacks [5]

An approach to improve the security of data communication through a vulnerable network channel consists in defining RF Distinct Native Attributes (RF-DNA) features of hardware devices (PHY layers) [6], which are inherently unique for a given device [7]. In this paper, these RF-DNA features are analyzed and processed for the discrimination and rejection of spoofing devices.

The structure of this paper is as follows. First, in Section 2, a short review of the current research work on rogue devices discrimination is presented. Later, Section 3 introduces the methodology adopted in this paper for security purpose classification. In Section 4, outcome of the proposed method on real data is explained. Finally, Section 5 summarizes different findings of this research work.

2 Related works

2.1 Classification methods

Classification of devices for discrimination of authorized (spoofed) identities from unauthorized (rogue or spoofing) ones has been one of the most investigated areas in security in the last several years. In conjunction with evolution of physical attacks in last years, security strategies emphasize on the extraction of physical features of authorized devices to detect the incoming attacks effectively. A common feature extraction approach for ZigBee devices consists of analyzing a fixed length header, such as a preamble or a synchronization header (SHR), to obtain statistical parameters such as the mean, variance, skewness, and kurtosis features of the physical signal characteristics such as its amplitude, phase, and frequency, on equal length sub-regions (time windows) of the received signal [8–11]. It has been shown that phase is the most appropriate physical characteristic for the classification of ZigBee devices. Beside the reported methods, another common approach is to employ RF fingerprinting by measuring the transient behavior of a device. In [12], the authors present a classification research method using the RF fingerprinting concept, focusing on the extraction of features from the amplitude of the transient parts of the Wi-Fi transmissions, acquired from 8 IEEE 802.11b Wi-Fi cards.

2.2 Deep learning classification methods

Significant improvements in computational hardware capabilities during the last few years have permitted the implementation of deep learning methods for feature extraction and classification. In [13], the authors introduced a high performance classification scheme based on convolutional neural networks (CNN) operating on the time domain complex baseband signals. Moreover, a CNN used wireless interference identification for classification purposes in IEEE 802.11, 802.15.4, and 802.15.1 protocols, by identifying the channel frequency and the type of wireless technology employed [14].

Although the strategies focusing on a fixed length preamble [8–11, 15] or signal transients [12] and [16] showed satisfactory discrimination accuracy, one of the recent approaches for feature extraction proposed by [17] emphasizes the feature extraction using a deep learning model on the steady state component of the initial transmission samples (or data points).

The method presented in [17] is not restricted by which part of the signal is used and does not assume that a preamble is being transmitted. For this algorithm, a priori knowledge of the transmitted signal is not required for feature extraction such as in [8–11], [15]. This allows the network to learn the features that best distinguish the devices without requiring any a priori knowledge of the target devices features. In [17], frequency compensation of the signal was also used for the first time in RF fingerprinting experiments. Removing device-dependent carrier frequency offsets which may appear in low signal-to-noise ratio (SNR) transmissions results in the rejection of unauthorized (rogue or spoofing) devices trying to mitigate this feature by use of a precise local oscillator. On the other hand, compensation of frequency offsets lowers the probability of frequency variations at baseband. While the results from [17] are promising, the training dataset, i.e., the dataset used for training the model toward achieving its tasks, contained data points from devices that were also included in the test set, i.e., the dataset used for evaluating the results of the model. As the test set does not contain devices that were never used in training the model, we cannot conclude on the performance of the model when facing new, unseen units. The work of this paper addresses this issue. The idea behind proposed scenario of this work is *one-vs-all*.

2.3 Transform-based classification methods

Other device classification methods based on Hilbert-Huang [18] and wavelet [19] transforms have demonstrated successful RF fingerprinting performances in this field, using neural networks to develop models of nonlinear power amplifiers and to perform predistortion [20], [21].

In this paper, an approach combining the advantages of the time-scale features of wavelet transform, with feature extraction and classification using deep learning designs, is presented.

3 Proposed classification method

In this paper, a binary classification system is used to provide a mechanism of device discrimination into

two classes: legitimate devices and (unauthorized) rogue devices. The classification strategy is *one-vs-all* where the system generates a model for each specific device, and considers the detection of all other devices other than the main target device. If a model is made for a specific device, when any new device enters the network, this model tries to detect if this device is an authorized device, will be granted access to the network. If not, it will be rejected by the network.

3.1 Dataset acquisition

For training a model, the first step is the dataset acquisition from real devices. This consists in creating the data points of different devices from the signal acquisitions of the ZigBee devices. The IEEE 802.15.4 protocol (ZigBee protocol) communicates through the 11 channels from 2.4 to 2.48355 GHz, each with a 2-MHz bandwidth. The central frequency of each channel can be calculated:

$$\begin{aligned} F_c &= 2405 + 5(h-11) \quad [\text{MHz}] \\ h &= 11, 12, \ldots, 26 \end{aligned} \tag{1}$$

where h indicates the channel index. Different manufacturers set the central carrier frequency of their ZigBee units to different channels in this frequency range. For instance, RZUSBSticks work in channel 20 with a central frequency of 2.45 GHz, whereas XBEE Digi units and Texas Instruments devices use channel 11 with a central frequency of 2.405 GHz. The responses of these devices are captured as successive partial signals, or bursts.

Each burst begins with a known preamble [22], followed by 8 successive modulated I and Q components of zero symbols, labeled with indices 1 to 8 as shown in Fig. 1. Each preamble contains a repetitive pattern of a single symbol. The duration of each symbol is 16 s, and thus the length of 8×16 s = 128 s.

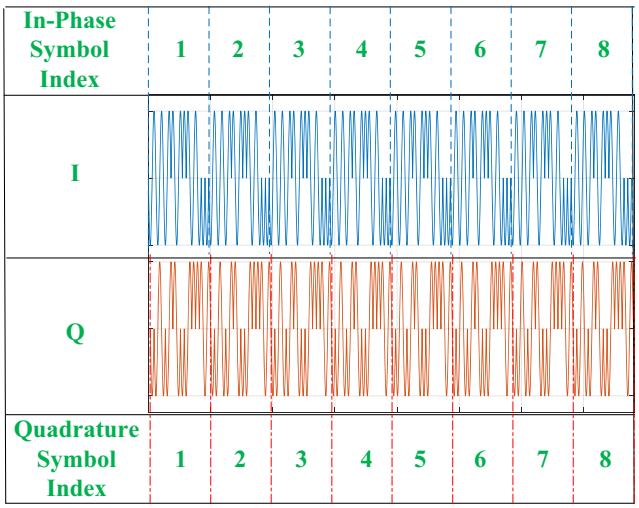


Fig. 1 Modulated I and Q components of the IEEE 802.15.4 reference preamble

3.2 Received preamble extraction

In real-life systems, the beginning of a preamble is not exactly located at the beginning of the signal burst. Therefore, the beginning of a preamble must be extracted from the received signal. For this purpose, a symbol for each of the 8 successive sub-regions is convoluted with the received signal. In such manner, there will be 8 successive peaks in the calculated convolution coefficients. The first one determines the beginning of the preamble in the received burst.

After the determination of the beginning of preamble, and knowing its length, extraction of the preamble itself can be done.

3.3 Dataset phase and frequency compensation

The received signals must be phase and frequency compensated because of the time-varying difference in the modulation and demodulation frequencies. This results in a shift in the slope of the phase of the received and reference preambles: phase compensation consists of reducing the slope difference between these two. The dataset phase and frequency compensation are done as follows:

1. Generation of a reference symbol and corresponding theoretical (reference) preamble, as shown in Fig. 1.
2. Calculation of the phase error:

$$\varphi_{err} = \varphi(ref_{pr}) - \varphi(rec_{pr}) \tag{2}$$

where $\varphi(ref_{pr})$ and $\varphi(rec_{pr})$ are the phase of the reference and the received (non-compensated) preambles, respectively.
3. As will be shown in Section 4.2.2, the phase error is almost linear. Therefore, the corrected phase of each data point can be calculated by fitting a first degree polynomial (linear regression):

$$\Delta\varphi_{linear} = \frac{\varphi_{err_2} - \varphi_{err_1}}{N} \times n + b \tag{3}$$

where b is a constant, φ_{err_1} and φ_{err_2} are the first and the last elements in φ_{err}, $n = 0, 2, 3, \ldots, N-1$ and N is the preamble length. The corrected phase of the received preamble, φ_{corr} is given by

$$\varphi_{corr} = \varphi(rec_{pr}) - \Delta\varphi_{linear} \tag{4}$$

where φ_{corr} is the corrected phase of the received preamble.

4. Using Eq. 5, the compensated in-phase (I) and also quadrature (Q) components of preamble are obtained as:

$$\begin{aligned} I_{comp} &= A(\cos(\varphi_{corr})) \\ Q_{comp} &= A(\sin(\varphi_{corr})) \end{aligned} \tag{5}$$

with A being the amplitude of received preamble:

$$A = \sqrt{\Re(rec_{pr})^2 + \Im(rec_{pr})^2}$$

where $\Re(rec_{pr})$ and $\Im(rec_{pr})$ are the real and imaginary parts of the received preamble rec_{pr}, respectively.

An example of a phase compensated signal is presented in Section 4.2.2. Once phase compensation is done, the extracted preambles can be processed for feature extraction, signal analysis, etc.

3.4 Dataset transformation

In this paper, the discrete wavelet transform is investigated as a means to improve the discrimination process between the devices. Before feeding the data points of the dataset to the classifier, a special kind of domain transformation, the dyadic discrete wavelet transform (DDWT), is applied to the dataset. The dyadic wavelet transform of the received ZigBee signal, $r(t)$, is given by [23, 24]:

$$c_{j,k} = \frac{1}{\sqrt{2^j}} \int_{-\infty}^{\infty} r(t)\psi\left(\frac{t - k2^j}{2^j}\right)dt \tag{6}$$

where $j, k = 0, 1, 2, ...$, and $\psi(t)$ is the wavelet window. The decomposition level of the wavelet coefficients is determined by the wavelet parameters (j, k). As the size of the wavelet window changes, the number of features extracted will change too.

The Haar wavelet window is used in this paper. All the classifications with the wavelet transform, referred to as the *DDWT dataset*, are done based on the extracted details at the first wavelet decomposition level. For comparison purposes, the dataset obtained before wavelet transform calculation (received, extracted, and phase and frequency compensated dataset with respect to Sections 3.1, 3.2, and 3.3) will be referred to as the *RAW dataset* in the remaining of the paper.

3.5 Model definition

As explained in Section 2.1, methods focusing on known RF-DNA features such as statistical parameters (mean, variance, skewness, and kurtosis) of amplitude, phase, and frequency [8–11] try to extract the best features measured data, resulting in the maximum possible classification rate. This is the reason why phase is known as the most effective feature of this collection of RF-DNA set for classification purposes. Based on this aim, the question that arises here is about the feature selection mechanism. Should the feature extraction be limited to this known set of statistical information of PHY parameters, or is it possible to use other elements more effective than those? The strategy that is used in this paper focuses on utilization of an autoencoder to extract the most dominant features of the input data, which give the maximum inter-class and minimum intra-class distances [25]. Later, feeding the extracted features to a fully connected classifier will result in acceptable correct classification and rejection rates. The proposed *Autoencoder (AE) combined with Fully Connected Classifier* is shown in Fig. 2.

3.5.1 Autoencoder

An autoencoder consists of two parts, an encoder and a decoder, as shown in Fig. 2. The input vector $\bar{x}_u$ represents a data point:

$$\bar{x}_u \in \{\bar{x}_0, \bar{x}_1, \cdots, \bar{x}_{U-1}\} \tag{7}$$

where U is the total number of data points in the dataset. Vector $\bar{x}_u$ contains N samples:

$$\bar{x}_u = \left[x_{u_0}, x_{u_1}, \cdots, x_{u_{N-1}}\right] \tag{8}$$

The output of encoder, $\bar{x}_{u_F}$, is a simplified representation of $\bar{x}_u$. The decoder is designed so that its output, $\hat{\bar{x}}_u$, tries to reproduce the original dataset, $\bar{x}_u$, from the encoder's representation, $\bar{x}_{u_F}$, by minimizing the difference between $\bar{x}_u$ and $\hat{\bar{x}}_u$ as illustrated in Fig. 2. The mechanism of decreasing this difference is through the mean squared error (MSE).

Feature extraction by the encoder Feature extraction is to map the high-dimensional data to a simplified low-dimensional space [26]. This transformation can be either linear or nonlinear. Specifically, considering a given data

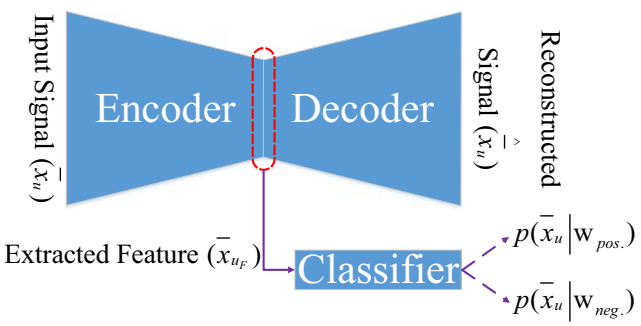


Fig. 2 Model symbolic representation

point $\bar{x}_u$, feature extraction generates new feature $\bar{x}_{u_F}$. The encoder can be described as a function f that maps an input $\bar{x}_u$ to a hidden representation $\bar{x}_{u_F}$:

$$\bar{x}_{u_F} = f\left(\bar{x}_u\right) = s_f\left(\omega \bar{x}_u + b_{\bar{x}_u}\right) \tag{9}$$

where s_f is a linear or a nonlinear activation function. The encoder is parameterized by a weight matrix ω and a bias vector $b_{\bar{x}_u} \in \mathbb{R}^n$.

Input reconstruction by the decoder The decoder function g maps the hidden representation $\bar{x}_{u_F}$ back to a reconstruction (or reproduction) vector $\hat{\bar{x}}_u$:

$$\hat{\bar{x}}_u = g\left(\bar{x}_{u_F}\right) = s_g\left(\omega' \bar{x}_{u_F} + b'_{\bar{x}_{u_F}}\right) \tag{10}$$

where s_g is the decoder's activation function, typically either the identity (yielding linear reconstruction) or a sigmoid (as nonlinear function). The decoder's parameters are a bias vector $b'_{\bar{x}_{u_F}}$ and weight matrix ω'.

Training an autoencoder involves finding parameter $\theta = (\omega, \omega', b_{\bar{x}_u}, b'_{\bar{x}_{u_F}})$ using a loss function that minimizes the difference between the original space $\bar{x}_u$ and the reconstruction space $\hat{\bar{x}}_u$.

3.5.2 Classification

After extraction of the features from the input dataset, these extracted features are fed to the classifier section depicted in Fig. 2. The typical classification structure used in the literature involves 2 connected layers. However, such a structure may overfit the training data, unless the training dataset is very large [27].

Since the strategy in this paper is *one-vs-all*, there are 2 classifier outputs, each presenting the conditional probability of the data points belonging to either *positive* or *negative* class (see Fig. 2). w_{pos} and w_{neg} represent the positive and negative (mutually exclusive) classes, respectively, that is:

$$p\left(\bar{x}_u \left| w_{pos}\right.\right) = 1 - p\left(\bar{x}_u \left| w_{neg}\right.\right) \tag{11}$$

3.6 Model training/validation/testing

The classifier model selection requires three (3) tasks: a training phase, a validation phase, and a testing phase [28]. During the training and validation phases, a new model is generated for the discrimination of one specific device from the others. Then, a testing phase with a new device is done to assess the reliability of the classifier model.

3.6.1 Dataset subdivision into training, validation, and testing datasets

Before training the model, the dataset is first subdivided into three different datasets: a training dataset, a validation dataset, and a testing dataset.

Selection of positive and negative devices The classification strategy adopted for this paper is the *one-vs-all* strategy. One device, device_m ($0 \leq m \leq M-1$), is selected as the *positive device* (with data points labeled + 1) while the other $M-1$ devices are identified as *negative devices*, allocated to a *negative class* (-1) labeled data points. Either device in the dataset can be labeled as the *positive device*: thus M different scenarios are possible, each considering a different device as the positive device with $+1$ labels.

Device allocation for training, validation and testing After the selection of device_m and labeling the data points, the devices are selected for training, validation, and testing. The procedure of device allocation to each of these steps is as follows:

1. In this step, device_m and at least one other device from the same or another manufacturer are selected for the training dataset.
2. Validation: In the validation procedure, beside the models used in training, there should be one or more additional devices which have never been *seen* by the model during the training procedure. Feeding the previously unseen devices improves the performance efficiency of the model.
3. Testing: Beside the devices already selected for training and validation, new devices are essential for the correct final evaluation of the generated model.

Data points allocation for training, validation, and testing To ensure that the classification model can distinguish between the data points of the desired device, device_m, from the other devices, a sufficiently large number of data points from each of the devices allocated to the training should be kept in training dataset. Then, after allocation of the devices for training, validation, and testing phases, the data points themselves are assigned for each of them. Therefore, almost 60% of the data points from training devices are used for training. Next, 10% of the validation devices data points are allocated to validation, and finally, the remaining data points from all devices should be used as testing data points. In this work, none of the devices used in training, validation, and testing has data points in common: each data point from each device is used only in one of these steps.

3.6.2 Model training and validation procedure

During the training of the model, the part of the dataset of the devices assigned to the training is fed to the classifier. The output of the decoder and also the classifier output will ideally converge to a unique solution. The reconstructed data should approximate the input dataset as much as possible. The MSE is used for assessing the output accuracy of reconstruction shown in Fig. 2 during the training or the validation processes. The MSE between the input and the reconstructed data points, $\bar{x}_u$ and $\hat{\bar{x}}_u$, at the decoder output is expressed as:

$$\begin{aligned} MSE_{\left(\bar{x}_u, \hat{\bar{x}}_u\right)} &= \frac{1}{N} \sum_{n=0}^{N-1} \left(\bar{x}_{u_n} - \hat{\bar{x}}_{u_n}\right)^2 \\ MSE_{tr/val} &= \frac{1}{U_{tr/val}} \sum_{u_{tr/val}} MSE_{\left(\bar{x}_u, \hat{\bar{x}}_u\right)} \end{aligned} \tag{12}$$

where n is the sample index of a single data point (either $\bar{x}_u$ or $\hat{\bar{x}}_u$), N is the number of samples in a data point, and $u_{tr/val}$ and $U_{tr/val}$ are the index and number of the training/validation data points, respectively.

The binary cross-entropy $H(y, p)$ between the distribution of extracted labels $y(x)$ and the distribution of the input data point $p(x)$ is used to assess the accuracy of classifier output during the training process [29]:

$$\begin{aligned} H(y, p) &= \mathbb{E}_y \left[-\log p\right] = -\sum_{x \in \mathcal{X}} y(x) \log\left(p(x)\right) \\ H(y, p) &= -\left(y \log\left(p\right) + (1 - y) \log\left(1 - p\right)\right) \end{aligned} \tag{13}$$

The mean square error (Eq. 12) and the cross-entropy (Eq. 13) should evolve simultaneously during the training, to ensure that the accuracy of the decoder output improves in such a way that it provides meaningful features for the classifier, and makes the label allocation more accurate. At the end of training process, among all models, the one which has the minimum validation cross-entropy loss function value of the classifier's output is selected as the best model for testing purposes.

3.7 Model testing

The last step in the procedure of device discrimination is the testing procedure. Although the model was evaluated during the validation process with a new group of data points (or even devices), since the design of the model is based on its optimization for the best possible classification of validation data points, there is a risk of overfitting the model to the validation dataset. Therefore, testing the model is required. This involves the verification of the classifier: the output probabilities and classified labels of the classifier are extracted and verified by machine learning evaluation methods, such as the *confusion matrix* (CM), and the *receiver operating characteristics* (ROC) curves.

4 Experimental results

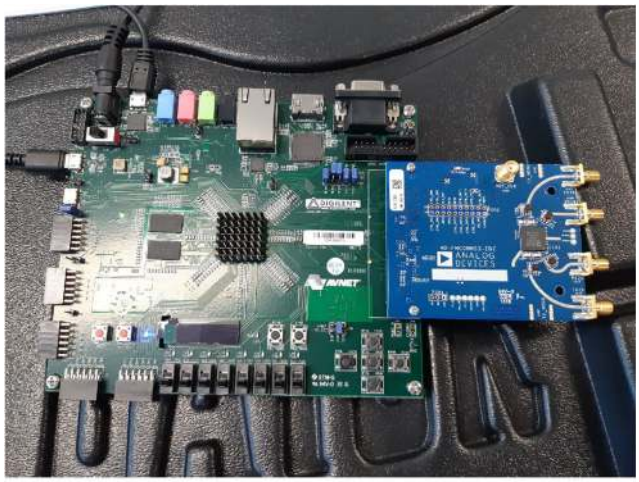
4.1 Experimental equipment setup

Figure 3 shows the laboratory transmitter and receiver for the signal measurements and dataset acquisition. As depicted, a Zynq XC7Z020 FPGA was used as the signal receiver. Eight (8) different ZigBee wireless devices were tested, including five (5) RZUSBSticks (labeled RZ_1, RZ_2, RZ_3, RZ_4, and RZ_5), one (1) XBEE Digi module (AR_1), and two (2) Texas Instruments devices (TI_1 and TI_2).

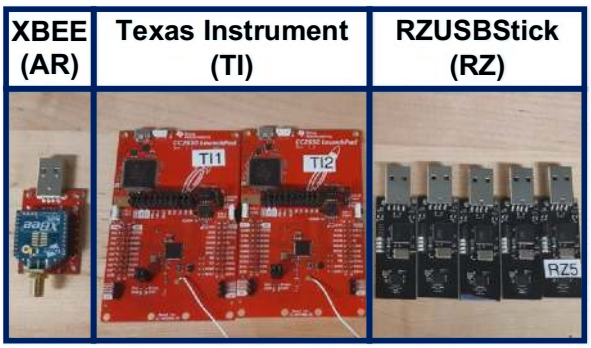
4.2 Preprocessing of acquired signals

4.2.1 Preambles extraction from the received signal bursts

As shown in Fig. 1, a preamble must include 8 successive repeated modulated zero symbols. As explained in



(a)



(b)

Fig. 3 Data acquisition. (a) Receiver. (b) Devices

Section 3.2, in practical situations the beginning of a preamble may not be located exactly at the beginning of the burst. Therefore, the first step in processing the sampled signals is to determine the beginning of the preamble by convolving the received signal with a single known symbol (each of 8 successive reference O-QPSK symbols in Fig. 1). The result of the convolution and resulting 8 successive peaks for a burst from ZigBee device RZ_1 are shown in Fig. 4. As shown, in majority of the cases, there is a large difference between the amplitude of resulting peak from the first symbol and others. After extraction of 7 successive equal peaks (numbers 2 to 8), the starting moment of preamble would be 2 symbol lengths before the second peak. Since the sampling frequency is set to 40 MHz, and based on the Section 3.1, the length of a symbol is as follows: 16 s × 40 MHz = 640 samples. Because the sample index for the second peak in Fig. 4 is 1310, the starting index is as follows: $1310 - 2 \times 640 = 30$ samples.

After determining the starting sample, the preamble can be extracted from the received signal. As mentioned in Section 3.1, the length of a preamble is 128 s, and since the sampling frequency is set to 40 MHz, the number of samples in a preamble is 5120. Knowing the length and exact location of the beginning of the preamble, extraction of the preamble can be done. As an example, the real and imaginary components of an extracted preamble of device RZ_1 are shown in Fig. 5.

4.2.2 Phase and frequency compensation

As discussed in Section 3.3, after extraction of the preamble, one can compare it with a reference preamble. The phase difference between reference and received preambles should be reduced as much as possible, with respect to Eqs. (3), (4), and (5). Figure 6 depicts the extracted preamble phases of device RZ_1 before and after phase and frequency compensation. The black line shows the phase of preamble before compensation, the green line refers to the preamble's phase after compensation, and the orange line illustrates the reference phase. As seen in the enlarged inset of this figure, the phase of compensated and reference preambles overlap with each other (green and orange lines, respectively), showing the effectiveness of compensation strategy.

The effect of phase compensation on the received signal is illustrated in Fig. 7 for the same device. As shown in the insets of this figure, comparison of the compensated real and imaginary signals with related parts of the reference preamble, shows a good concordance between the compensated and reference ones, confirming the efficiency of the presented phase and frequency compensation approach, like Fig. 6.

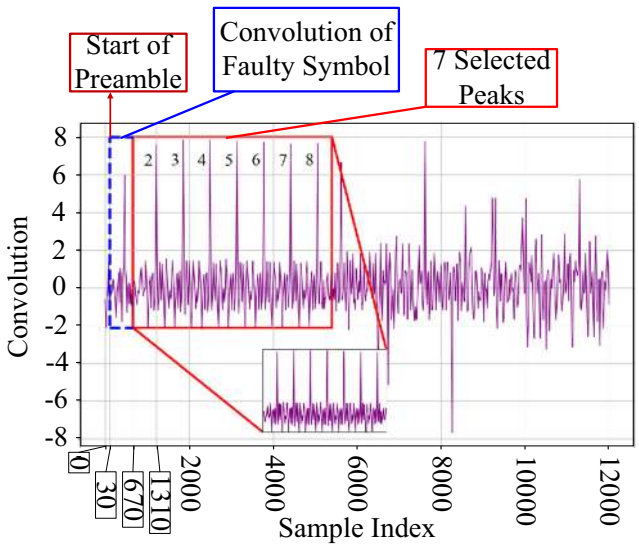


Fig. 4 Convolution of the received signal burst with the reference O-QPSK symbol

4.3 Datasets processing

4.3.1 DDWT dataset generation

After preprocessing the data, with respect to Sections 3.2 and 3.3, the DDWT is applied to the resulting data points as described in Section 3.4.

4.3.2 Model generation

The model, summarized in Table 1, is constructed with respect to the model of Fig. 2. The model is composed of 22 layers described in each row. The second column (titled as layer name), shows the name of layer, starting with En., Dec., or Cl. referring to encoder, decoder, or classifier part of Fig. 2. The third column determines the type of the layer as input layer, convolutional, maxpooling, upsampling, or dense layer. Besides, at the end of layer type for each row, 1D or 2D refers to the one- or two-dimentional size of input/output data to that layer in encoder/decoder part. The fourth column as output shape shows the size of output data from each layer. Finally, last two columns show the activation function (the function which is applied to the output data of layer), and the number of parameters in each layer which should be set during the training of the model. It is worth mentioning that the larger the number of parameters in the model, the more computational power will be needed for training. In this paper, the presented structure of the model has 708,003 trainable parameters.

In this work, the model training, validation, and testing were performed using a computing unit GPU NVIDIA Quadro K620 hardware, and Python 3.6, Tensorflow 1.0.8, and Keras 2.2.0 software. The whole set of data points was

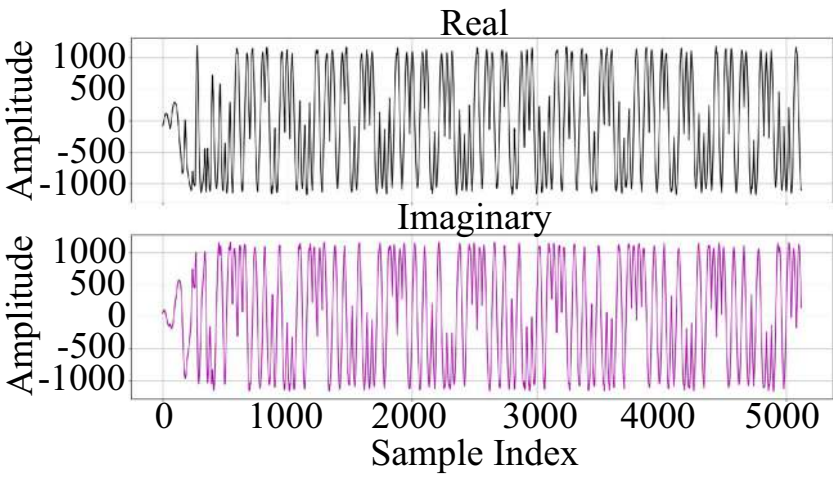


Fig. 5 Real and imaginary components of an extracted preamble from device RZ_1

fed to the classifier batch by batch. The batch size was set to 20 data points.

As mentioned in Section 3.5, the device feature extraction and classification are based on deep learning, and more specifically on autoencoders, shown in rows 0 to 18 of Table 1.

An *InputLayer* with 9 successive *MaxPooling/Conv.* layers, referred to as the encoder, extract the features from the input data, and reduce the dataset size from (5120, 2) at the input layer to (2, 1) at the output of 32 filters of the encoder layer, shown in rows 0 to 9 of Table 1.

In decoder part for the autoencoder, 9 successive *Upsampling/Deconv.* layers (rows 10 to 18 of Table 1) are used to reconstruct the dataset at the output of decoder layer with the same size as the input layer.

In both the encoder and the decoder, no dropout or batch normalization is used.

The classification layers consist of 2 successive fully connected (dense) layers. Rows 20 and 21 of Table 1

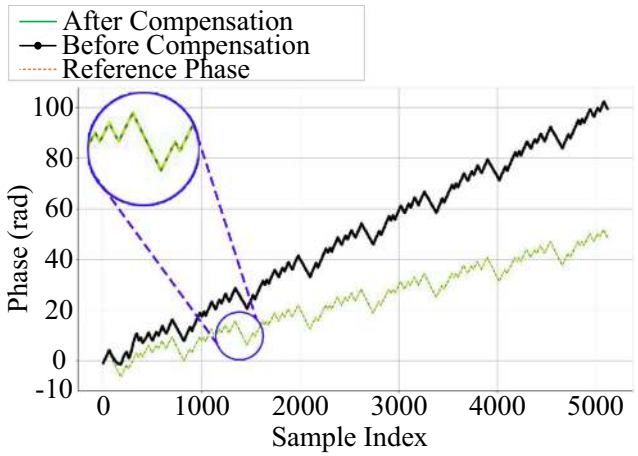


Fig. 6 Phases of received, reference and compensated preambles from device RZ_1

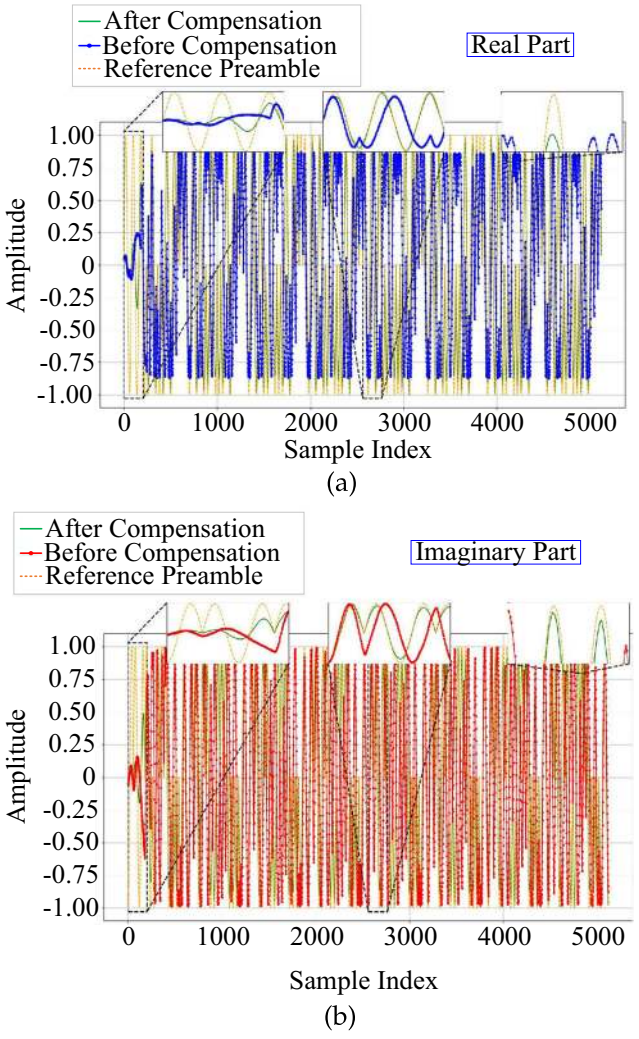


Fig. 7 Example of (a) real and (b) imaginary components of received, reference, and compensated preambles obtained from a burst of device RZ_1

Table 1 Autoencoder model summary

Index	Layer Name	Layer Type	Output Shape	Activ. Func.	Parameters No.
0	En.layer_0	InputLayer	(None, 5120, 2, 1)	-	0
1	En.layer_1	Conv2D	(None, 4800, 1, 32)	Relu	20576
2	En.layer_2	MaxPooling1D	(None, 960, 1, 32)	-	0
3	En.layer_3	Conv1D	(None, 800, 1, 32)	Relu	164896
4	En.layer_4	MaxPooling1D	(None, 160, 1, 32)	-	0
5	En.layer_5	Conv1D	(None, 80, 1, 32)	Relu	82976
6	En.layer_6	Conv1D	(None, 40, 1, 32)	Relu	42016
7	En.layer_7	Conv1D	(None, 20, 1, 32)	Relu	21536
8	En.layer_8	Conv1D	(None, 10, 1, 32)	Relu	11296
9	En.layer_9_encoder	MaxPooling1D	(None, 2, 1, 32)	-	0
10	Dec.layer_0	UpSampling1D	(None, 10, 1, 32)	-	0
11	Dec.layer_1	Conv1DTranspose	(None, 20, 1, 32)	Relu	11296
12	Dec.layer_2	Conv1DTranspose	(None, 40, 1, 32)	Relu	21536
13	Dec.layer_3	Conv1DTranspose	(None, 80, 1, 32)	Relu	42016
14	Dec.layer_4	Conv1DTranspose	(None, 160, 1, 32)	Relu	82976
15	Dec.layer_5	UpSampling1D	(None, 800, 1, 32)	-	0
16	Dec.layer_6	Conv1DTranspose	(None, 960, 1, 32)	Relu	164896
17	Dec.layer_7	UpSampling1D	(None, 4800, 1, 32)	-	0
18	Dec.layer_8_decoder	Conv2DTranspose	(None, 5120, 2, 1)	Relu	20545
19	Cl.layer_0	Flatten	(None, 64)	-	0
20	Cl.layer_1	Dense	(None, 320)	Sigmoid	20800
21	Cl.layer_2_classifier	Dense	(None, 2)	Sigmoid	642
Total parameters:		708,003			
Trainable parameters:		708,003			

identify these layers, using a *Sigmoid* as an activation function for binary (or binomial) discrimination.

4.4 Dataset processing

4.4.1 Training and validation of model using acquired datasets

After defining the model, it is trained using the training part of the dataset. The chosen optimizer function for training the model is the adaptive learning rate optimization algorithm *Adam* [29, 30] with a learning rate value $lr = 0.0001$. Meanwhile, the number of epochs for training are selected to be 100 with early stopping.

Data points allocation for training, validation and testing There are two datasets: the RAW and DDWT datasets.

Then, different *scenarios* for training, validation, and testing, based on Section 3.6.1, are provided in Table 2. Based on this assumption, and with respect to Table 2, in each scenario one device is considered the authenticated device and is labeled as +1, while the others are rogue devices (labeled as −1). Besides, as mentioned in Section 3.6.1, at least one of these spoofing devices must be used in training. For such purpose, 3 devices are used as rogue devices for training (total of 4 devices). On the other hand, since there should be new device(s) in validation, one new device is added at validation phase for each scenario (5 devices for validation). Finally, all used devices in the training and validation phases along with 3 new devices (added up to 8 devices) are used for testing. As shown, the whole set of testing devices in each scenario is divided into 3 different groups: group *A*, group *B*, and group *C*. Group *A* consists of devices which have been seen by the model during the training phase (although a new set of data points from these devices will be used in the testing phase). Group *B* includes devices which have never been seen by the model before the testing phase, but for which at least a device from the *same family of devices* (devices from the same manufacturer, such as devices RZ_1 and RZ_2) are used in the training phase. Finally, in group *C* are devices that neither them, nor their family members have been seen by the model during the training phase.

After allocating the devices for training, validation, and testing, referring to Section 3.6.1, the assigned percentages of the data points from the dataset for each stage are 60%, 10%, and 30%, respectively. The number of allocated data points from each device for each stage is indicated in the last row of Table 2. The total number of data points for each device in the dataset is higher than 11,000 data points.

Decoding and classification convergence During the training of the model, a decoder loss function, such as the MSE, is measured to ensure that the reconstructed data points at

Table 2 Different scenarios for label allocation

	Positive Device	Training	Validation
1	RZ_1	RZ_1, RZ_2, RZ_3, TI_2	$RZ_1, RZ_2, RZ_3, TI_1, TI_2$
2	RZ_2	RZ_1, RZ_2, RZ_3, TI_2	$RZ_1, RZ_2, RZ_3, TI_1, TI_2$
3	RZ_3	RZ_1, RZ_2, RZ_3, TI_2	$RZ_1, RZ_2, RZ_3, TI_1, TI_2$
4	RZ_4	RZ_1, RZ_4, RZ_3, TI_2	$RZ_1, RZ_4, RZ_3, TI_1, TI_2$
5	RZ_5	RZ_1, RZ_5, RZ_3, TI_2	$RZ_1, RZ_5, RZ_3, TI_1, TI_2$
6	TI_1	RZ_1, RZ_2, RZ_3, TI_1	$RZ_1, RZ_2, RZ_3, TI_1, TI_2$
7	TI_2	RZ_1, RZ_2, RZ_3, TI_2	$RZ_1, RZ_2, RZ_3, TI_1, TI_2$
8	AR_1	RZ_1, RZ_2, AR_1, TI_2	$RZ_1, RZ_2, AR_1, TI_1, TI_2$
	No.	24000 (6000/device)	5000 (1000/device)

	Positive Device	Testing		
		A	B	C
1	RZ_1	RZ_1, RZ_2, RZ_3, TI_2	RZ_4, RZ_5, TI_1	AR_1
2	RZ_2	RZ_1, RZ_2, RZ_3, TI_2	RZ_4, RZ_5, TI_1	AR_1
3	RZ_3	RZ_1, RZ_2, RZ_3, TI_2	RZ_4, RZ_5, TI_1	AR_1
4	RZ_4	RZ_1, RZ_4, RZ_3, TI_2	RZ_2, RZ_5, TI_1	AR_1
5	RZ_5	RZ_1, RZ_5, RZ_3, TI_2	RZ_2, RZ_4, TI_1	AR_1
6	TI_1	RZ_1, RZ_2, RZ_3, TI_1	RZ_4, RZ_5, TI_2	AR_1
7	TI_2	RZ_1, RZ_2, RZ_3, TI_2	RZ_4, RZ_5, TI_1	AR_1
8	AR_1	RZ_1, RZ_2, AR_1, TI_2	RZ_3, RZ_4, RZ_5, TI_1	-
	No.	> 32000 (4000/device)		

the decoder output is as close as possible to the input data points. Also, to verify the efficiency of the trained autoencoder at each iteration, a validation dataset is fed to the classifier to test the accuracy of model.

4.5 Evaluating (testing) the classification method

After the training and validation, the trained classification model is tested against new devices previously unseen by the classifier. As stated in Section 4.4.1, 30% of data points in dataset are allocated to testing. The resulting percentage values of correct and false classification of data points for each device are discussed in terms of confusion matrices and receiver operating characteristic (ROC) plots.

4.5.1 Confusion matrix results

The confusion matrices for each of the 8 scenarios of Table 2 for both RAW and DDWT datasets are computed, and the ranges of classification rates for groups A, B, and C of Table 2 are reported in Table 3.

The ranges reported in this table correspond to the lowest and highest rates of classification for groups A, B, and C of RAW or DDWT datasets, in each scenario.

For example, in the first scenario of RAW dataset (for model trained based on positive device RZ_1), the range of correct classification rate for group A is $[0.99, 1.0]$. This means that, using RAW dataset, looking at the classification rates for testing devices used in training, the minimum and maximum classification rates are 0.99 and 1.00, respectively. However, for group B, the unseen devices for testing which at least have one family member in training, gives the minimum and maximum classification rates of 0.37 and 0.100. Finally, the group C, with device AR_1 as an unseen device which does not have any family member in training or validation, gives a classification rate of 1.00.

The minimum correct classification rate for group A devices in both of RAW and DDWT datasets are close, with maximum difference of 7%.

There is a degradation in results of groups B and C compared with group A. The results for group B show that for 5 scenarios of RAW dataset (RZ_2, RZ_3, RZ_4, TI_1, and AR_1) and 5 scenarios of DDWT dataset (RZ_3, RZ_4, RZ_5, TI_1, and AR_1), the range of correct classification rate starts from a value higher than 0.6 and reaches to 1.0, and the worst cases for both RAW and DDWT datasets belong to the TI_2 scenario with the minimum classification rate equal to 0.01 and 0.00, respectively. Also, the results for group C illustrate that for all models trained using RAW dataset, and all scenarios except for the RZ_2 device for DDWT dataset, the range of correct classification rate is higher than 0.7. The worst case for RAW dataset in group C is obtained from RZ_3 model, equal to 0.72, and the similar factor for the DDWT dataset in group C is 0.39 for RZ_2 scenario.

4.5.2 Receiver operating characteristics

The ROC plots are shown in Figs. 8 and 9. For each ROC plot, the area under the curve (AUC), is given in the legend. The devices selected as groups A, B, and C, are labeled with A, B, and C, respectively.

Table 3 Correct classification rate range for the RAW and DDWT datasets

Scenario	Positive Device	RAW Dataset		
		A	B	C
1	RZ_1	[0.99, 1.00]	[0.37, 1.00]	1.00
2	RZ_2	1.00	[0.73, 1.00]	0.74
3	RZ_3	[0.97, 1.00]	1.00	0.72
4	RZ_4	[0.99, 1.00]	[0.90, 1.00]	1.00
5	RZ_5	[0.91, 1.00]	[0.32, 1.00]	0.95
6	TI_1	[0.99, 1.00]	1.00	1.00
7	TI_2	1.00	[0.01, 1.00]	1.00
8	AR_1	[0.87, 1.00]	[0.99, 1.00]	-
Scenario	**Positive Device**	**DDWT Dataset**		
		A	B	C
1	RZ_1	[0.99, 1.00]	[0.42, 0.99]	0.95
2	RZ_2	[0.97, 0.99]	[0.24, 0.99]	0.39
3	RZ_3	[0.98, 1.00]	[0.85, 1.00]	0.77
4	RZ_4	1.00	[0.85, 1.00]	1.00
5	RZ_5	[0.98, 1.00]	[0.64, 1.00]	0.97
6	TI_1	1.00	1.00	0.98
7	TI_2	[0.99, 1.00]	[0.00, 1.00]	0.99
8	AR_1	1.00	1.00	-

A summary of the ROC plots for A, B, and C groups of devices is given in Table 4 for both the RAW and DDWT datasets. The correct classification rate of target (authorized or spoofed) device, p_d (detection probability), and its corresponding misclassification rate of rogue (unauthorized or spoofing) device, p_{fa} (false alarm probability), in this table are related to the worst cases using RAW and DDWT datasets.

For instance, in the first scenario, feeding the model trained for positive device RZ_1 with the RAW dataset in Fig. 8 (b), the range of correct classification rate (p_d) for the values of p_{FA} in the $[0, 1.0]$ is almost equal to 1.0, corresponding to the group A of testing devices. As mentioned in Table 2, this group of devices is those used in training, too. In addition, looking at the same plot, group B gives the p_d values higher than 0.9 for p_{FA} in [0.2, 1.0] for the same set of dataset. Finally, the group C (as a new device which does not have any family member in the training) presents the correct classification rate (p_d) almost equal to 1.0 for all possible values of p_{FA}. A similar definition can be allocated to other rows of this table in both RAW and DDWT datasets.

As observed, although the p_d for a specific range of p_{fa} for devices from group A in the RAW dataset is better than DDWT dataset, the ROC plots for devices of groups B and C show the advantage of feeding the classifier with the DDWT dataset in specific cases. For instance, in one of these cases (device TI_2), using the DDWT dataset increases the worst p_{fa} range by about 25% for the same correct classification rate p_d in the group B of devices: that is from 0.65 to 1 for the RAW dataset, and from 0.4 to 1 for the DDWT dataset. Therefore, although the comparison of confusion matrix results shows a better classification rate using the RAW dataset, the ROC plots of Fig. 9 results in a better performance using the DDWT dataset for the cases where the classifier *tolerates* larger false alarm range values p_{fa}.

4.6 Classifier performance comparison

Before comparison, the following points of work strategy adopted in this paper are summarized:

- First, device discrimination is repeated for two different types of datasets: the RAW and the DDWT datasets.
- As indicated in Table 2, focusing on each device allows us to consider different scenarios for device allocation for training, validation, and testing leading to a specific model for each device as authorized/spoofed unit.
- The devices used in the testing phase are separated into 3 groups (with respect to Sections 3.6.1 and 4.4.1): A, B, or C.

Referring to this short summarization, 3 cases of comparison are done with respect to their approach about each of the mentioned criteria above, based on the following definitions:

$AUC_{\min}$: Minimum AUC.		
$p_{FA_{0.9}}^{AUC_{\min}}$: p_{FA} for 0.9 of p_d related to $AUC_{\min}$.		
$a(t)$:Amplitude	$\varphi(t)$: Phase	$f(t)$: Instant.−frequency
$AUC_{\max}$: Maximum AUC.		
$p_{FA_{0.9}}^{AUC_{\max}}$: p_{FA} for 0.9 of p_d related to the $AUC_{\max}$.		
δ^2: Variance	γ: Skewness	κ: Kurthosis

To have a better comparison of the results presented in Figs. 8 and 9 with those reported in the literature, the

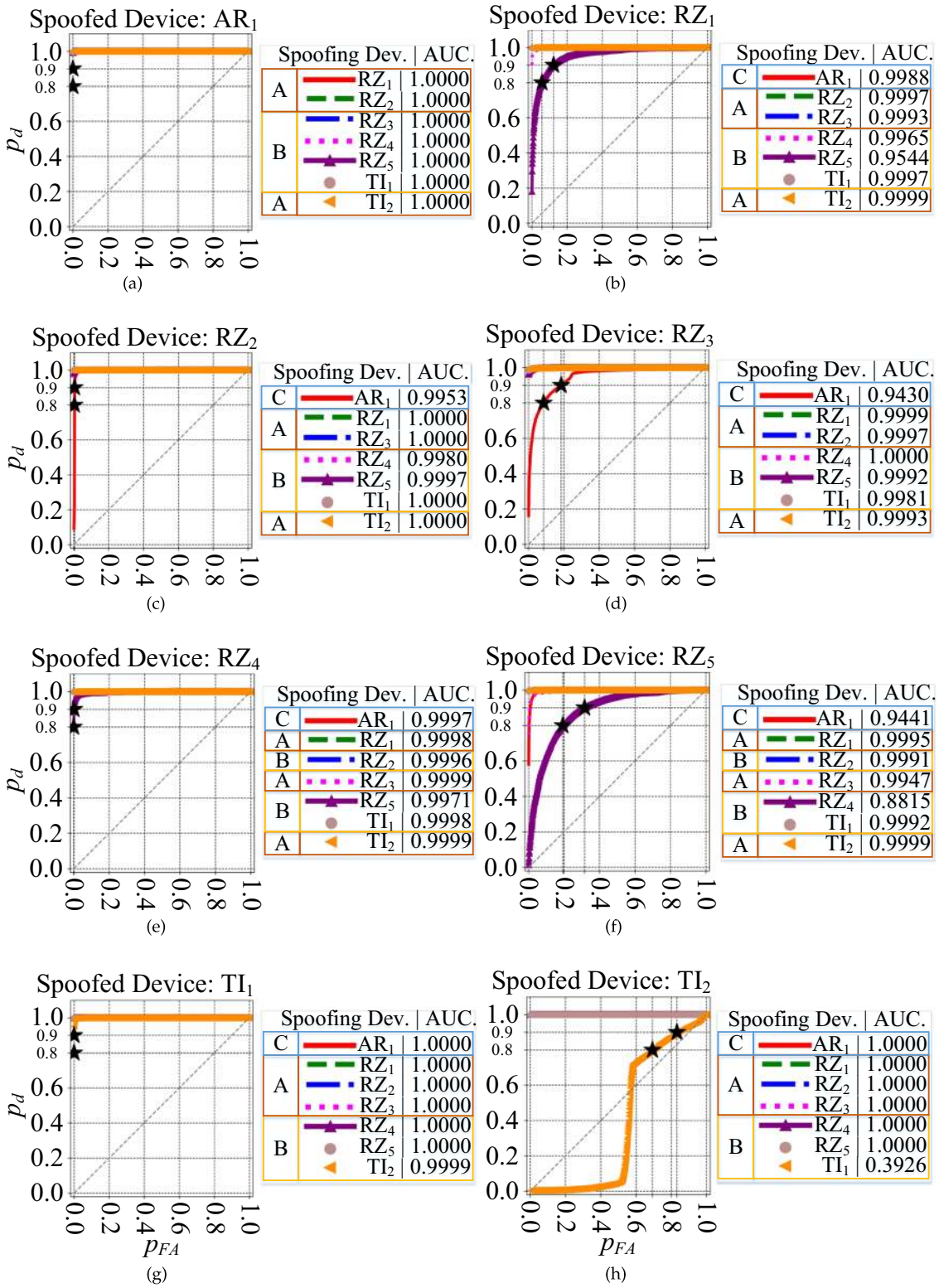


Fig. 8 ROC plot for all 8 scenarios of Table 2 for the RAW dataset and spoofed devices (a) AR_1 (b) RZ_1 (c) RZ_2 (d) RZ_3 (e) RZ_4 (f) RZ_5 (g) TI_1 (h) TI_2

close-up of groups *A*/*B* of RAW/DDWT datasets are presented in Fig. 10. The main purpose for showing these close-up plots is to have a precise comparison of $AUC_{\min}$ and its related $p_{FA,9}^{AUC_{\min}}$ (or $AUC_{\max}$ and its corresponding $p_{FA,9}^{AUC_{\max}}$) for each of these groups of devices with reported values in the literature.

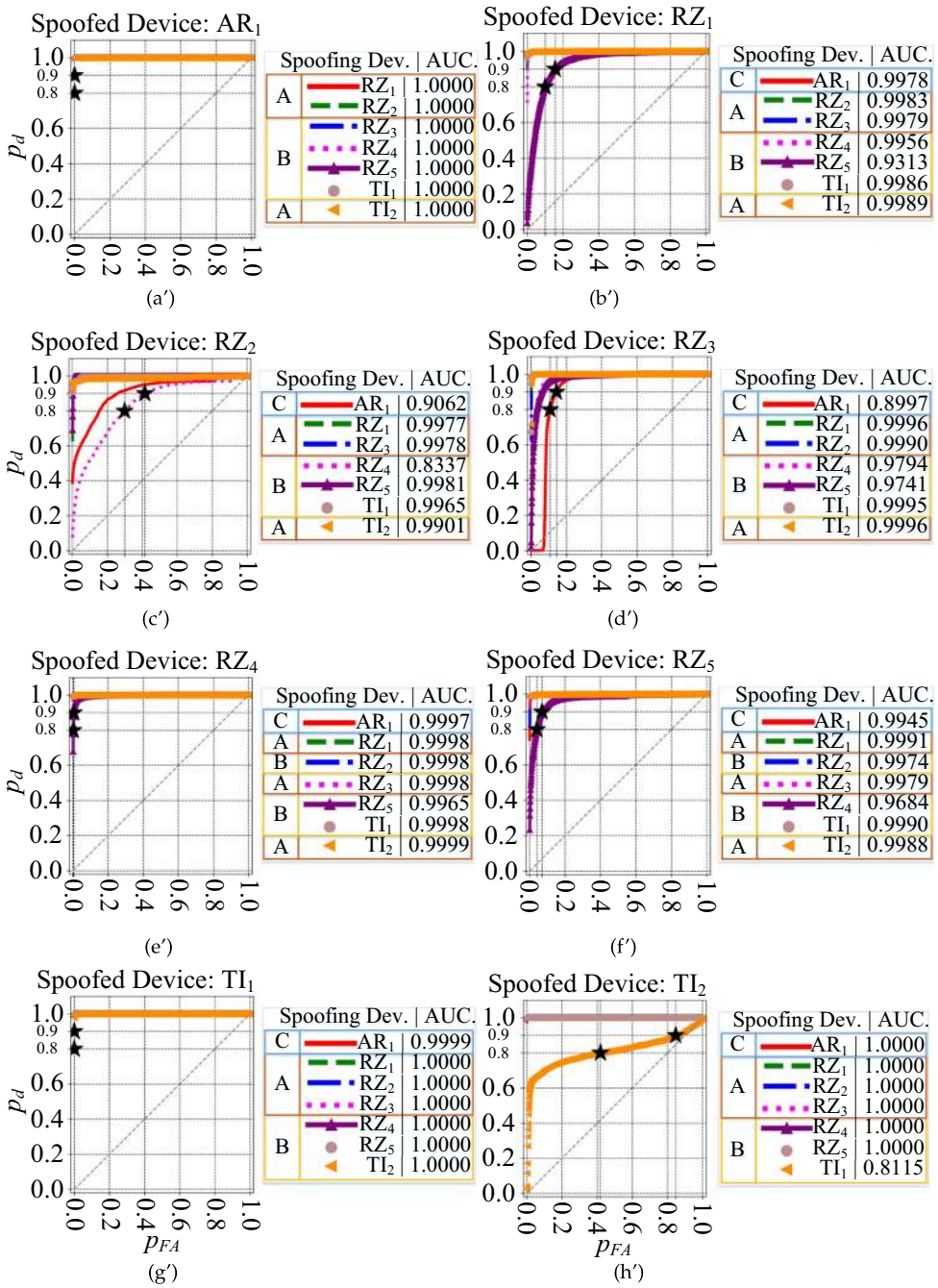


Fig. 9 ROC plot for all 8 scenarios of Table 2 for the DDWT dataset and spoofed devices (a′) AR_1 (b′) RZ_1 (c′) RZ_2 (d′) RZ_3 (e′) RZ_4 (f′) RZ_5 (g′) TI_1 (h′) TI_2

For instance, if the presented testing values in a work from the literature focus on the time domain dataset of the same devices as those used for training, then in the comparison, $AUC_{\min}/AUC_{\max}$ and its $p_{FA_{.9}}^{AUC_{\min}}/p_{FA_{.9}}^{AUC_{\max}}$ of group A of all plots of RAW dataset in Fig. 8 are presented. On the other hand, if the literature work focuses on the new devices in testing phase, from the same time domain dataset, Group B or C of the RAW dataset will be used for comparison.

The same rule applies to the case where the emphasized dataset in the literature is the transformed signal, and in such manner, the comparison will be done with DDWT dataset. To have a better understanding, some of these cases will be explained in more details, by examples, in the rest of this section.

In [17], the authors present a framework for training a convolutional neural network for the identification ZigBee devices in the time domain. Comparison with [17] is done based on group A for the 8 strategies of Table 2 with the RAW dataset. Referring to Fig. 8 for the same group of devices, the $AUC_{\min}$ is 0.9947 with spoofing/rogue device RZ_3 as shown in Fig. 8 (f) (close-up view in Fig. 10 (a)).

Table 4 ROC plot summary for the RAW and DDWT datasets

Scenario	Positive Device	RAW Dataset						DDWT Dataset					
		A		*B*		*C*		*A*		*B*		*C*	
		p_d	p_{FA}	p_d	p_{FA}	p_d	p_{FA}	p_d	p_{FA}	p_d	p_{FA}	p_d	p_{FA}
1	RZ_1	≈ 1.0	$[0,1]$	≥ 0.9	$[0.2,1]$	≈ 1.0	$[0,1]$	≥ 0.9	$[0.01,1]$	≥ 0.9	$[0.2,1]$	≈ 1.0	$[0,1]$
2	RZ_2	≈ 1.0	$[0,1]$	≥ 0.95	$[0,1]$	≥ 0.9	$[0.01,1]$	≥ 0.9	$[0.01,1]$	≥ 0.8	$[0.3,1]$	≥ 0.85	$[0.2,1]$
3	RZ_3	≈ 1.0	$[0,1]$	≥ 0.95	$[0,1]$	≥ 0.9	$[0.2,1]$	≥ 0.9	$[0.01,1]$	≥ 0.8	$[0.05,1]$	≥ 0.8	$[0.1,1]$
4	RZ_4	≈ 1.0	$[0,1]$	≥ 0.9	$[0.01,1]$	≈ 1.0	$[0,1]$	≈ 1.0	$[0,1]$	≥ 0.9	$[0.01,1]$	≥ 0.95	$[0.01,1]$
5	RZ_5	≥ 0.9	$[0.01,1]$	≥ 0.8	$[0.2,1]$	≥ 0.9	$[0.01,1]$	≥ 0.9	$[0.01,1]$	≥ 0.9	$[0.1,1]$	≥ 0.9	$[0.01,1]$
6	TI_1	≈ 1.0	$[0,1]$	≥ 0.9	$[0.01,1]$	≈ 1.0	$[0,1]$	≈ 1.0	$[0,1]$	≥ 0.95	$[0.01,1]$	≈ 1.0	$[0,1]$
7	TI_2	≈ 1.0	$[0,1]$	≥ 0.8	$[0.65,1]$	≈ 1.0	$[0,1]$	≈ 1.0	$[0,1]$	≥ 0.8	$[0.4,1]$	≥ 0.95	$[0.01,1]$
8	AR_1	≈ 1.0	$[0,1]$	≥ 0.95	$[0,1]$	-	-	≈ 1.0	$[0,1]$	≈ 1.0	$[0,1]$	-	-

Moreover, the $p_{FA,9}^{AUC_{\min}}$ is equal to 0.013. The $AUC_{\max}$ for 12 cases shown in Fig. 8 (a, c, g, h) reach the maximum value of 1.0000 and the related $p_{FA,9}^{AUC_{\max}} = 0.000$. These results are comparable to those reported by [17], that is, $AUC_{\min} = 0.9653$ and $AUC_{\max} = 0.9971$.

In [31], the dataset consists of time domain information (RAW dataset) of the signal characteristics of ZigBee devices, $a(t)$,$\varphi(t)$, and $f(t)$ and their statistical features: σ^2, γ, and κ. The devices used for testing are different from those used for training, but belong to the same family. Therefore, the results reported in [31] are compared with those obtained with group B for all 8 strategies with the RAW dataset (Table 2). Using these devices, the $AUC_{\min}$, corresponding to device TI_1 in Fig. 8 (h) is 0.3926 (close-up view in Fig. 10 (b)). As shown, $p_{FA,9} = 0.832$.

The $AUC_{\max}$ and its related $p_{FA,9}^{AUC_{\max}}$ among the different devices in Fig. 8 (a, c, d, g, h) are equal to 1.0000, and 0.000, respectively. Again, comparable results are reported in [31]: $p_{FA,9}^{AUC_{\min}}$ and $p_{FA,9}^{AUC_{\max}}$ are 0.540 and 0.000, respectively.

In [4], *Multiple Discriminant Analysis* (MDA) is employed to train and classify ZigBee devices from their RF-DNA (Radio Frequency Distinct Native Attributes). The used dataset consists of the statistical features (σ^2, γ, and κ of physical signal characteristics $a(t)$, $\varphi(t)$, and $f(t)$ and 2-D features from wavelet and Gabor transforms of the recorded signals. The devices used for the testing phase are either the same or family members of those involved in training. We compare the performance of our proposed discrimination method with those obtained in [4] for two cases.

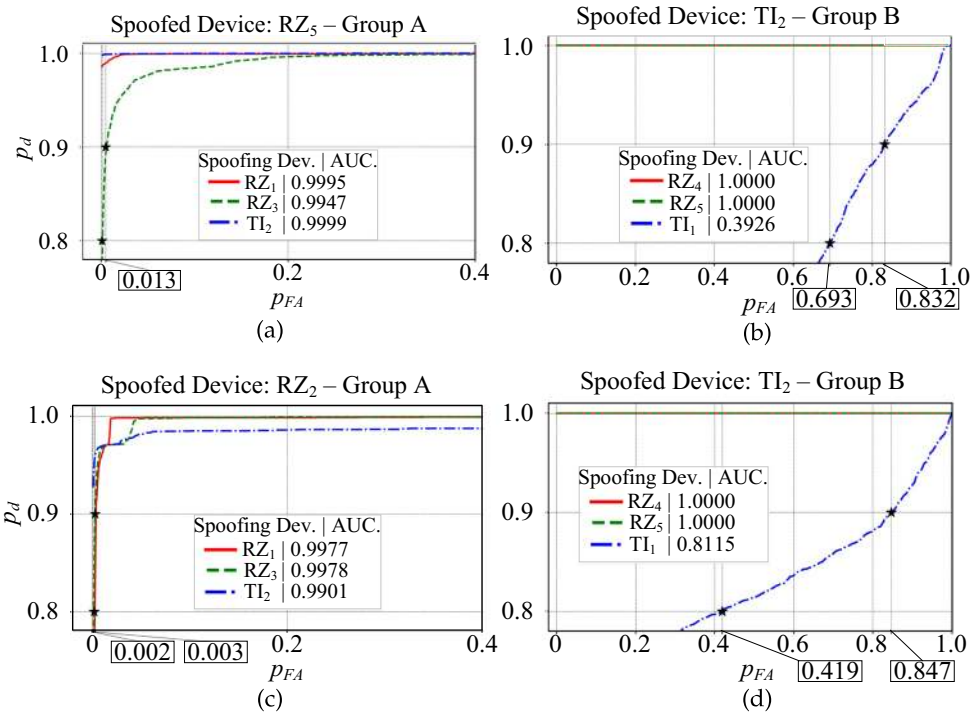


Fig. 10 Close-up view of the ROC plots around the 90% of p_d for (a) group A of RAW dataset related to RZ_5 model (b) group B of RAW dataset related to TI_2 model (c) group A of DDWT dataset corresponding to RZ_2 model (d) group B of DDWT dataset corresponding to TI_2 model

In the first case, where the same devices are used for training and testing, the group A of all 8 scenarios (strategies) of the DDWT dataset described in Table 2 is used. As shown in the ROC plots for this group of devices in Fig. 9, the $AUC_{\min}$ occurs with device TI_2 (Fig. 9 (c′)). $AUC_{\min}$ and its related $p_{FA,9}^{AUC_{\min}}$ (see Fig. 10 (c)), equal to 0.9901 and 0.003, respectively. The $AUC_{\max}$ and its related $p_{FA,9}^{AUC_{\max}}$ in Fig. 9 (a′, g′, h′) are equal to 1.000 and 0.000, respectively. In [4], the probabilities $p_{FA,9}^{AUC_{\min}} \approx 0.038$ and $p_{FA,9}^{AUC_{\max}} \approx 0.005$ for a similar group of devices.

For the second case, the comparison is for group B with the 8 scenarios with the DDWT dataset in Table 2. The ROC plots for the same group of devices in Fig. 9 (h′) indicate a $AUC_{\min} = 0.8115$ (see Fig. 10 (d)) and the corresponding $p_{FA,9}^{AUC_{\min}} = 0.847$. $AUC_{\max} = 1.0000$ and $p_{FA,9}^{AUC_{\max}} = 0.000$ in Fig. 9 (a′, g′, h′). In [4], $p_{FA,9}^{AUC_{\min}} \approx 0.850$ and $p_{FA,9}^{AUC_{\max}} \approx 0.030$ for a similar group of devices.

5 Conclusion

In this paper, a rogue device discrimination method in a vulnerable network channel at the physical layer is presented. The main strategy relies on discrimination of target/authorized/spoofed devices from rogue (unauthorized or spoofing) ones, using RF-DNA features. The separation of devices for training, validation, and testing is done by allocating specific devices for verification and/or testing which have never been seen by the model during training. The classifier structure consists of an autoencoder for the feature extraction process. Feature extraction is investigated for (time domain) RAW and (time-scale domain) DDWT datasets of the received RF signals. The classification rate for testing devices has shown an acceptable accuracy for both seen and new (unseen) devices. The suggested rogue device discrimination method compares favorably with recent results reported in the literature. The results are promising, since 7 out of 8 deep models for the devices in the dataset demonstrated area under the curve for all spoofing devices higher than 0.8815 for RAW dataset, and 0.8337 for DDWT dataset. Also, the related false alarm probabilities (p_{fa}) for a 90% detection probability (p_d) corresponding to the spoofing devices in all of these 7 cases are lower than 40% for both cases of datasets.

Acknowledgments The authors wish to thank Louis-N. Bélanger, Mathieu Lévesque, Guillaume Godbout and Samuel Brin-Marquis for their help with data acquisition setup development as well as Sébastien Carrier for his helpful suggestions on the data analysis. They also wish to thank Prof. Paul Fortier for his suggestions in improving the manuscript.

Funding This research work is scientifically and financially supported by NSERC (Natural Sciences and Engineering Research Council of Canada), Thales Research and Technology, Canada, and MITACS.

References

1. Arana P (2006) Benefits and vulnerabilities of Wi-Fi protected access 2 (WPA2). INFS 612: Principles and Practices of Communication Networks. George Mason University
2. Wi-Fi Alliance. WPA3 specification version 1.0, 2019
3. Benkic K, Planinsic P, Cucej Z (2007) Custom wireless sensor network based on ZigBee. In: ELMAR, 2007. IEEE, pp 259–262
4. Dubendorfer CK, Ramsey BW, Temple MA (2012) An RF-DNA verification process for ZigBee networks. In: 2012 Military Communications Conference (MILCOM 2012). IEEE, pp 1–6
5. Cobb WE, Laspe ED, Baldwin RO, Temple MA, Kim YC (2011) Intrinsic physical-layer authentication of integrated circuits. IEEE Trans Inform Forensics Secur 7(1):14–24
6. Ramsey BW (2014) Improved Wireless Security through Physical Layer Protocol Manipulation and Radio Frequency Fingerprinting. Ph.D. dissertation, Air Force Institute of Technology, Wright-Patterson Graduate School of Engineering and Management
7. Reising DR, Temple MA (2012) WiMAX mobile subscriber verification using Gabor-based RF-DNA fingerprints. In: 2012 IEEE International Conference on Communications (ICC). IEEE, pp 1005–1010
8. Reising DR, Temple MA, Jackson JA (2015) Authorized and rogue device discrimination using dimensionally reduced RF-DNA fingerprints. IEEE Trans Inform Forensics Secur 10(6):1180–1192
9. Patel H (2015) Non-parametric feature generation for RF-fingerprinting on ZigBee devices. In: 2015 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA). IEEE, pp 1–5
10. Bihl TJ, Bauer KW, Temple MA (2016) Feature selection for RF fingerprinting with multiple discriminant analysis and using ZigBee device emissions. IEEE Trans Inform Forensics Secur 11(8):1862–1874
11. Dubendorfer C, Ramsey B, Temple M (2013) Zigbee device verification for securing industrial control and building automation systems. In: International Conference on Critical Infrastructure Protection. Springer, Berlin, pp 47–62
12. Ureten O, Serinken N (2007) Wireless security through RF fingerprinting. Can J Electr Comput Eng 32(1):27–33
13. O'Shea TJ, Corgan J, Clancy TC (2016) Convolutional radio modulation recognition networks. In: International Conference on Engineering Applications of Neural Networks. Springer, Berlin, pp 213–226

14. Schmidt M, Block D, Meier U (2017) Wireless interference identification with convolutional neural networks. In: 2017 IEEE 15th International Conference on Industrial Informatics (INDIN). IEEE, pp 180–185
15. Yuan Y, Huang Z, Wang F, Wang X (2015) Radio specific emitter identification based on nonlinear characteristics of signal. In: 2015 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom). IEEE, pp 77–81
16. Zhao C, Wu X, Huang L, Yao Y, Chang Yao-Chung (2014) Compressed sensing based fingerprint identification for wireless transmitters. The Scientific World Journal, 2014
17. Merchant K, Revay S, Stantchev G, Nousain B (2018) Deep learning for RF device fingerprinting in cognitive communication networks. IEEE Journal of Selected Topics in Signal Processing 12(1):160–167
18. Yuan Y, Huang Z, Wu H, Wang X (2014) Specific emitter identification based on Hilbert–Huang transform-based time–frequency–energy distribution features. IET Commun 8(13):2404–2412
19. Bertoncini C, Rudd K, Nousain B, Hinders M (2012) Wavelet fingerprinting of radio-frequency identification (RFID) tags. IEEE Trans Ind Electron 59(12):4843–4850
20. Benvenuto N, Piazza F, Uncini A (1993) A neural network approach to data predistortion with memory in digital radio systems. In: Proceedings of ICC'93-IEEE International Conference on Communications, vol 1. IEEE, pp 232–236
21. Mkadem F, Boumaiza S (2011) Physically inspired neural network model for RF power amplifier behavioral modeling and digital predistortion. IEEE Transactions on Microwave Theory and Techniques 59(4):913–923
22. Shi G, Li K (2017) Fundamentals of ZigBee and WiFi. In: Signal interference in WiFi and ZigBee networks, chapter 2. Springer, Cham, pp 9–27
23. Mallat SG, et al. (1989) Multifrequency channel decompositions of images and wavelet models. IEEE Trans Acoustics, Speech, and Signal Processing 37(12):2091–2110
24. Nievergelt Y (1999) Wavelets made easy. Birkhäuser
25. Boehmke B, Greenwell BM (2019) Hands-on machine learning with R. CRC Press
26. Meng Q, Catchpoole D, Skillicom D, Kennedy PJ (2017) Relational autoencoder for feature extraction. In: 2017 International Joint Conference on Neural Networks (IJCNN). IEEE, pp 364–371
27. Cao X (2015) A practical theory for designing very deep convolutional neural networks. Technical Report
28. Alpaydin E (2014) Introduction to machine learning, 2nd edn. MIT Press, Cambridge
29. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge
30. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv:1412.6980
31. Ramsey BW, Stubbs TD, Mullins BE, Temple MA, Buckner MA (2015) Wireless infrastructure protection using low-cost radio frequency fingerprinting receivers. Int J Critical Infrastruct Protect 8:27–39