Authorized and Rogue Device Discrimination Using Dimensionally Reduced RF-DNA Fingerprints for Security Purposes in Wireless Communication Systems

Thèse

Mohammad Amin Haji Bagheri Fard

Doctorat en génie électrique
Philosophiæ doctor (Ph. D.)

Québec, Canada

Résumé

La nature des réseaux de capteurs sans fil comme ZigBee, permettant la communication entre différents types de nœuds du réseau, les rend très vulnérables à divers types de menaces. Dans différentes applications des technologies sans fil modernes comme SmartHome, les informations privées et sensibles produites par le réseau peuvent être transmises au monde extérieur par des moyens filaires ou sans fil. Outre les avantages offerts, cette intégration augmentera certainement les exigences en matière de protection des communications. Les nœuds capteurs du réseau étant souvent placés à proximité d'autres appareils, le réseau peut être plus vulnérable aux attaques potentielles. Cette recherche de doctorat a pour but d'utiliser les attributs natifs distincts de radiofréquence RF-DNA sécurisés produits par le processus d'empreinte numérique dans le but de fournir un support de communication sans fil sécurisé pour les communications de réseau ZigBee. Ici, nous visons à permettre une discrimination d'appareil en utilisant des préambules physiques (PHY) extraits des signaux émis pas de différents appareils. Grâce à cette procédure, nous pouvons établir une distinction entre différents appareils produits par différents fabricants ou par le même fabricant. Dans un tel cas, nous serons en mesure de fournir aux appareils des identifications physiques de niveau binaire non clonables qui empêchent l'accès non autorisé des appareils non autorisés au réseau par la falsification des identifications autorisées.

Abstract

The nature of wireless networks like ZigBee sensors, being able to provide communication between different types of nodes in the network makes them very vulnerable to various types of threats. In different applications of modern wireless technologies like Smart Home, private and sensitive information produced by the network can be conveyed to the outside world through wired or wireless means. Besides the advantages, this integration will definitely increase the requirements in the security of communications. The sensor nodes of the network are often located in the accessible range of other devices, and in such cases, a network may face more vulnerability to potential attacks. This Ph.D. research aims to use the secure Radio Frequency Distinct Native Attributes (RF-DNA) produced by the fingerprinting process to provide a secure wireless communication media for ZigBee network device communications. Here, we aim to provide device discrimination using Physical (PHY) preambles extracted from the signal transmitted by different devices. Through this procedure, we are able to distinguish between different devices produced by different manufacturers, or by the same one. In such cases, we will be able to provide devices with unclonable physical bit-level identifications that prevent the unauthorized access of rogue devices to the network through the forgery of authorized devices' identifications.

Contents

List of Tables

List of Figures

List of Acronyms

AODVjr	Ad hoc On-demand Distance Vector junior routing
AE	Autoencoder
ALOHANET	Additive Links On-line Hawaii Area
APS	Application Support
AUC	Area Under the Curve
BCE	Binary Cross-Entropy
BPTT	Backpropagation Through Time
CNN	Convolutional Neural Network
CM	Confusion Matrix
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CRC	Cyclic Redundancy Check
DARPA	Defense Advanced Research Projects Agency
DDWT	Dyadic Discrete Wavelet Transform
DoS	Denial of Service
DSSS	Direct Sequence Spread Spectrum
FCS	Frame Check Sequence
FFD	Full-Function Device
FNR	False Negative Rate
FPGA	Field Programmable Gate Arrays
FPR (p_{FA})	False Positive Rate (False Alarm Probability)
GPS	Global Positioning System
GSM	Global System for Mobile communications
I	In-phase Component
IC	Integrated Circuit
IF	Intermediate Frequency
IR	Infrared
LANs	Local Area Networks
LR	Likelihood Ratio
LSB	Least Significant Bit
LSTM	Long Short-Term Memory
MAC	Media Access Control
MDA	Multiple Discriminant Analysis
MFR	MAC Footer
MHR	MAC Header
MLD	Maximum Likelihood Decoder

MSB	Most Significant Bit
MSE	Mean Squared Error
MSDU	MAC Service Data Unit
NLP	Natural Language Processing
O-QPSK	Orthogonal Quadrature Phase Shift Keying
PHY	Physical
PHR	PHY Header
PN	Pseudo-Noise
PPDU	PHY Protocol Data Unit
PSDU	PHY Service Data Unit
PUFs	Physical Unclonable Functions
Q	Quadrature Component
RF	Radio Frequency
RF-COA	RF Certificates of Authenticity
RF-DNA	Radio Frequency Distinct Native Attributes
RFD	Reduce-Function Device
RFID	Radio-Frequency Identification
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristics
ROI	Region of Interest
RSS	Received Signal Strength
RZ	RZUSBSTICK
SDK	Software Development Kit
SFD	Start of Frame Delimiter
SHR	Synchronizer Header
SNR	Signal to Noise Ratio
TI	Texas Instruments
TNR	True Negative Rate
TPR (p$_d$)	True Positive Rate (Detection Probability)
USB	Universal Serial Buss
Wi-Fi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WPA	Wi-Fi Protected Access
WPA2	Wi-Fi Protected Access 2
WPA3	Wi-Fi Protected Access 3
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Networks
XOR	Exclusive OR function
ZC	ZigBee Coordinator
ZDO	ZigBee Device Object
ZR	ZigBee Router
ZTC	ZigBee Trust Centre

List of symbols

A or $(a(t))$	Amplitude of received preamble
$a_{j,k}$	DDWT approximation coefficient at scale j and translation k
AUC_{max}	Maximum AUC
AUC_{min}	Minimum AUC
$\mathbf{b}_f$	Forget gate bias vector
$\mathbf{b}_g$	Input branch bias vector
$\mathbf{b}_h$	Output bias vector
$\mathbf{b}_i$	Input gate bias vector
$\mathbf{b}_o$	Output gate bias vector
$\mathbf{b}_{\mathbf{x}_n}$	Bias vector for extraction of feature vector $(\mathbf{x}_{n_F})$
$\mathbf{b'}_{\mathbf{x}_{n_F}}$	Weight matrix for reconstruction of output vector $(\hat{\mathbf{x}}_n)$
$c_{j,k}$	CWT coefficient at scale j and translation k
D_m	m^{th} Device
$d_{j,k}$	DDWT details coefficient at scale j and translation k
$D_{nT,te,l}$	l^{th} nontarget testing device
$D_{nT,tr,o}$	o^{th} nontarget training device
$D_{nT,val,q}$	q^{th} nontarget validation device
$D_{T,te,z}$	z^{th} target testing devices
$DDWT$	Dyadic discrete wavelet transform
f or $(f(t))$	Instantaneous-frequency
$f_g(.)$	Input branch function
$f_h(.)$	System output function
$g\left[k\right]$	DDWT high-pass filter
h	Channel index
$h\left[k\right]$	DDWT low-pass filter
$\mathbf{h}_k$	System state vector at time-step k
$H(y,p)$	Binary cross-entropy
$\Im$	Imaginary value
I_{Comp}	Compensated in-phase part of preamble
k	Sample index from a data point
K	Length of a data point input
K_{RNN}	Length of a data point input to an RNN cell unit
L	Overall output error
L_n	Output error for data point n
$L_{n,k}$	Output error at time-step k for data point n

LR_{nT}^{max}	Maximum likelihood ratio for nontarget devices class
LR_{nT}^{min}	Minimum likelihood ratio for nontarget devices class
LR_{T}^{max}	Maximum likelihood ratio for target devices class
LR_{T}^{min}	Minimum likelihood ratio for target devices class
m	Index of m^{th} device in the dataset with size M
M	Number of devices in the dataset
M_{te}	Number of testing devices
$M_{T,te}$	Number of target testing devices
$M_{nT,te}$	Number of nontarget testing devices
MSE_{tr}	Mean squared error for training dataset reconstruction
MSE_{val}	Mean squared error for validation dataset reconstruction
n	Index of the data point in the dataset with size N.
N	Number of data points in the dataset
n_{tr}	Index of training data point from a training dataset with size N_{tr}
N_{tr}	Number of data points for training from the dataset with size N
N_{te}	Number of data points for testing from the dataset with size N
$N_{nT,te}^{te}$	Number of testing data points from testing devices of nontarget class
$N_{nT,te}^{te}$	Number of testing data points from testing devices of target class
$N_{nT,te,l}^{te}$	Number of testing data points from testing device l of nontarget class
$N_{nT,te,M_{nT,te}}^{te}$	Number of testing data points from testing device $M_{nT,te}$ of nontarget class
$N_{T,te}^{te}$	Number of testing data points from testing devices of target class
$N_{T,te}^{te}$	Number of testing data points from testing devices of target class
$N_{T,te,z}^{te}$	Number of testing data points from testing device z of target class
$N_{T,te,M_{T,te}}^{te}$	Number of testing data points from testing device $M_{T,te}$ of target class
N_{unit}	Number of the units in an RNN cell
O_{n}^{RNN}	n^{th} output matrix from an RNN cell
n_{val}	Index of validation data point from a validation dataset with size N_{val}
N_{val}	Number of data points for validation from the dataset with size N
$\mathbf{o}_{ex_{n,k}}^{RNN}$	Ideal output vector from an RNN cell at time k for $unth$ input data point $\mathbf{x}_{n}^{RNN}$
$\mathbf{o}_{n,k}^{RNN}$	Output vector from an RNN cell at time k for n^{th} input data point $\mathbf{x}_{n}^{RNN}$
p_{FA}	False alarm probability
Q_{Comp}	Compensated quadrature part of preamble
$\Re$	Real value
$\mathrm{R}_{\mathbf{x}_n}(k_1, k_2)$	Correlation of a signal $\mathbf{x}_n$ (at time-step k_1) with its delayed version (at time-step k_2)
rec_{pr}	Received preamble

ref_{pr}	Reference preamble
S	Parameter set
S_i	i^{th} member of parameter set
U_f	Forget gate weighting matrix for current input ($\mathbf{x}_n$)
U_g	Input branch weighting matrix for current input ($\mathbf{x}_n$)
U_i	Input gate weighting matrix for current input ($\mathbf{x}_n$)
U_o	Output gate weighting matrix for current input ($\mathbf{x}_n$)
V_f	Forget gate weighting matrix for previous output ($\hat{\mathbf{x}}_n$)
V_g	Input branch weighting matrix for previous output ($\hat{\mathbf{x}}_n$)
V_h	Output weighting matrix
V_i	Input gate weighting matrix for previous output ($\hat{\mathbf{x}}_n$)
V_o	Output gate weighting matrix for previous output ($\hat{\mathbf{x}}_n$)
W_{neg}	Negative class
W_{pos}	Positive class
$\mathbf{x}_n$	n^{th} input vector data point from the dataset with size N
$\mathbf{x}_n^{RNN}$	n^{th} input vector data point to an RNN cell from the dataset with size N
$\mathbf{x}_{n_F}$	Extracted feature vector from the input vector data point ($\mathbf{x}_n$)
$\mathbf{x}_{n,k}$	A subset of n^{th} input vector data point ($\mathbf{x}_n$) containing n beginning samples
$\hat{\mathbf{x}}_n$	Reconstructed output vector data pint ($\mathbf{x}_n$)
$\hat{\mathbf{x}}_{n,k}$	A subset of n^{th} input vector data point ($\hat{\mathbf{x}}_n$) containing k beginning samples
$x_{n,k}$	k^{th} sample from n^{th} input vector data point ($\mathbf{x}_n$) from the dataset with size N
$x_{n,k}^{RNN}$	k^{th} input sample to an RNN cell from n^{th} input vector data point ($\mathbf{x}_n^{RNN}$)
$\hat{x}_{n,k}$	k^{th} sample from n^{th} reconstructed output vector data point ($\hat{\mathbf{x}}_n$)
y or $y(\mathbf{x}_n)$	Predicted model label for a single data point $\mathbf{x}_n$
γ	Skewness
δ	Standard deviation
δ^2	Variance
$\Delta\varphi_{linear}$	Linear phase difference needed for the correction of phase in each data point
κ	Kurtosis
τ or τ_0	Threshold for likelihood ratio
$\tau_{p_{FA}}$	τ for a specific amount of p_{FA}

φ or $\varphi(t)$	Phase
φ_{corr}	Corrected phase of the received preamble
φ_{error}	Phase error of the received preamble
φ_{error_1}	First sample in-phase error
φ_{error_2}	Last sample in-phase error
ϕ_n	Phase of n^{th} sample from received the preamble
φ_n^*	Initial phase of n^{th} sample from the received preamble
$\psi\,[k]$	DDWT window
Ω	Weight matrix for extraction of feature vector ($\mathbf{x}_{u_F}$)
Ω'	Weight matrix for reconstruction of input vector ($\hat{\mathbf{x}}_u$)

To my Family

Remerciements

Tout d'abord, je voudrais remercier tout particulièrement mon Professeur au cours de ce doctorat, le Professeur Jean-Yves Chouinard, pour ses conseils utiles au cours des dernières années qui m'ont orienté dans la bonne direction pour surmonter tous les défis auxquels j'ai fait face dans ce projet, et sans l'aide de ce dernier, ce projet était si loin de la dernière station.

Deuxièmement, merci tout particulièrement au Professeur Paul Fortier, qui a collaboré à ce projet en tant que co-superviseur et nous avons pu profiter de son aide dans le façon d'aborder la solution de ce projet.

L'auteur tient à remercier M. L.-N. Bélanger, M. M. Lévesque, M. G. Godbout, et M. S. Brin-Marquis pour leur aide dans le développement de la mise en place du système d'acquisition de données.

Ces travaux de recherche sont financés scientifiquement et financièrement par le CRSNG (Conseil de recherches en sciences naturelles et en génie du Canada), Thales Research and Technology, Canada et Mitacs. Je tiens à remercier M. Bernard Lebel de Thales Research and Technology, Canada, qui a collaboré avec notre groupe sur le même projet, et nous avons pu utiliser son aide autant. De plus, je voudrais exprimer ma gratitude à M. Sébastien Carrier pour ses suggestions utiles sur l'analyse des données. En attendant, un merci spécial à M. Frédéric Audet, Mme Gabrielle Teyssier-Roberge, Mme Sarah Savage, et à toutes les autres personnes de Thales et de l'UMR-SU qui m'ont tant aidé pendant ce stage.

Enfin, je dois remercier tous mes collègues de bureau, M. Masoud Rezaei, M. Esmaeel Maghsoudloo, M. Mousa Karimi, M. Vahid Khojasteh, Mme Seyedeh Samira Moosavi, Mme S. Nazila Hosseini, Mme Zahra Rezaei et Mme Anita Ebrahimian, pour un environnement agréable et convivial qui m'a été d'une grande aide dans mes recherches.

Acknowledgement

First of all, I would like to present my special thanks to my Professor during this Ph.D., Professor Jean-Yves Chouinard, for his helpful guidance during the last years which directed me in the right direction to overcome all the challenges that I faced in this project, and without his help, this project was so far away from the last station.

Second, special thanks to Professor Paul Fortier, who collaborated in this project as co-supervisor and we could take the benefit of his help in the way of approaching the solution for this project.

The author wishes to thank Mr. L.-N. Bélanger, Mr. M. Lévesque, Mr. G. Godbout, and Mr. S. Brin-Marquis for their help with the data acquisition system set-up development.

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. I really like to thank Mr. Bernard Lebel from Thales Research and Technology, Canada, who collaborated with our group on the same project, and we could use his help so much. Moreover, I would like to present my gratitude to Mr. Sébastien Carrier for his helpful suggestions on the data analysis. Meanwhile, special thanks to Mr. Frederic Audet, Ms. Gabrielle Teyssier-Roberge, Ms. Sarah Savage, and all other people from Thales and UMR-SU who helped me so much during this internship.

Finally, I should mention my gratefulness to all my office colleagues, Mr. Masoud Rezaei, Mr. Esmaeel Maghsoudloo, Mr. Mousa Karimi, Mr. Vahid Khojasteh, Ms. Seyedeh Samira Moosavi, Ms. S. Nazila Hosseini, Ms. Zahra Rezaei, and Ms. Anita Ebrahimian, for a nice and friendly environment which was a great help for me in my research.

Introduction

Nowadays, wireless communications are becoming more vital in today's modern life and access to information is of crucial importance in the modern world. The most common wireless technologies, such as deep-space radio communications, GPS, garage door controllers, wireless computer mice, and keyboards use radio communications. Wireless communications allow the long-range signal transmission that is not possible with the wired communications. The growth of wireless networks has given us the opportunity to use personal devices anywhere and anytime. Providing access to the Internet anywhere and anytime without physical contact, medical health monitoring for inaccessible or hardly accessible areas, urgent situations alerting for rapid support of medical and rescue plans in natural disasters are some important advantages of this kind of network.

Besides all the advantages, wireless communication has let us face many threats such as easy access of hackers to wireless signals, resulting in eavesdropping of transferred data. Providing a security system for detection and rejection of unauthorized access in such a growing network is of a great importance to reduce the risk of information leakage. Nowadays, different standards for wireless communication systems exist such as radio Frequency (RF), light, visible and infrared (IR), ultrasonic short range communications, Wi-Fi, ZigBee, and Bluetooth.

Proposing a comprehensive strategy for securing the wireless network is dependent on the protocol which is used. Different protocols have different features, such as modulation and data structure, which affect the performance in the suggested methodology. Due to the low-cost and low-power performance, ZigBee (IEEE 802.15.4) is a popular technology in the industry of smart homes and IoT devices, as an alternative for other wireless protocols, such as Wi-Fi and Bluetooth [1]. Based on these features, and also due to the long battery life, ZigBee is a good candidate to be widely used in different applications such as smart home low-rate and short-range data transfer sensor networks, public bus transportation query systems, traffic management, and real time air pollution monitoring networks [2]. Therefore, the ZigBee technology is selected for the research in this thesis. In this manner, using a group of ZigBee devices, it is intended to provide a rogue device detection system which grants access to authorized devices and rejects unauthorized or rogue ones.

Problem definition

As discussed above, hacking the wireless signals is possible for the hackers in real world communication systems. Every day, a large amount of wireless signals are transferred and received by different devices in different parts of the world. Some of these devices include such important informations making security as one of the most crucial requirements in wireless communication systems. The nature of wireless networks, in such a way that they can be adopted easily with this technology, may result in various security threats. Researches like [3] include the place in the network, where attacks happen, as an important component into the target-based threat model. Referring to [4] (which focuses on ZigBee networks), different attacks to wireless communication systems can be classified into six types:

- Eavesdropping: in this type of attack, the information extraction is done through listening to the channel without permission.
- Denial of service: during the denial of service (DoS) attack, attackers to the wireless networks, such as ZigBee, emit the signals to interfere with the transmitted data in the network. This kind of attack can be detected at the data link. The end goal of this kind of attack is to disturb communication protocols, such as ZigBee [5].
- Node compromise: in this kind of attack, the attacker obtains the control over a legitimate node in the network, through different methods like reprogramming [5].
- Sinkhole attacks: through broadcasting the false routing identification signals, the attacker achieves access to the network and also to the transmitted information packets.
- Wormhole attacks: the attacker tries to confuse one or more nodes in the network through receiving and transmitting the different packets in the network [5].
- Physical attack: the attacker earns access to the devices in the network, through the wired connection. For example, in a Man-in-the-Middle (MitM) attack, the goal is to receive the leakage information. Then, after changing the information, it is transmitted back to the system. The changed information will fool the network and jeopardize its performance [6].

In recent decades, different protocols have been introduced to provide the wireless networks with security. For instance, strong security protocols such as Wi-Fi Protected Access (WPA), WPA2 [7], and WPA3 [8] have been created to protect and secure wireless signals. Among the different types of attacks, physical layer attacking by rogue devices are considered from those kinds that affect the strength of underlying protocols considerably [9]. A rogue device is a node which was never been seen by the network previously, and attempts to access to the wireless network by cloning the bit-level credentials of one of the known/authorized devices [10]. Physical layer attacks have a wide range from complicated to very simple, low-cost techniques targeting unintentional information leakage [9]. In such a case, the longer a system remains in operation, the greater the chance for unauthorized access, and the robustness of stand-alone bit-level security remains a concern [11]. Nowadays, production of smart low cost, short-range wireless devices expand to facilitate the realization of the internet of things (IoT) in modern life. Therefore, security of transmitted information in these networks relies on using the cryptographic keys. Then, the transmitted information by these devices is as secure as those keys [12].

In addition to the vulnerability of these growing low-cost IoT wireless systems to the complicated attacks, there is another problem here; the traditional cybersecurity protocols are not very efficient in blocking the complex attacks [13]. Therefore, a physical layer attack in these networks can really put the confidentiality of the transmitted signals in danger. In such a case, an accurate and high quality system is needed to discriminate the sophisticated malfunctioning and block the access. To reach such a goal, some cybersecurity systems rely on the application of artificial intelligence to detect the unathorized access of the attacking devices to increase the security level. Thus, this is the same approach which is selected in this thesis for providing security in the physical layer.

Motivations

The mechanism of physical structure attacks on secure electronic systems has changed during the last years, which makes it difficult for security systems to keep pace. The intended type of threats in such a scenario can be done by the devices with the different manufacturer or even the devices from the same family as the authorized devices. Therefore, proposing an effective and accurate methodology for distinguishing and rejecting devices with close enough physical characteristics to the authorized members of the network is a critical problem of a crucial importance. As mentioned above, the main idea in this thesis is to provide a wireless network with the high level security in the physical layer level, to prevent the unauthorized access to the network by rogue devices which copy the credential identity of authorized ones in order to falsify the network.

In this thesis, ZigBee network is selected as the target system. Also, the methodology proposed is based on the application of artificial intelligence and deep learning models fed by physical layer fingerprints for increasing the physical layer security in these communication systems. Here, three main elements are ZigBee networks, physical layer fingerprinting, and artificial intelligence. The reason to select each of these elements as a main part of research subject in the thesis are described here.

ZigBee networks

During the last years, different wireless protocols have been introduced, which differ in power consumption, data transmission range, speed, etc. To have a better comparison between the ZigBee and other protocols, Bluetooth and Wi-Fi are discussed here.

Bluetooth technology has evolved over the years. This technology transfers the data in low-band frequency and over short distances. Because of the level of security in Bluetooth networks, services provided in this kind of network are easy to use [14]. Despite all its advantages, due to the high bandwidth which results in relatively power-hungry applications (which makes it not the best choice for many IoT applications), new technologies like Bluetooth mesh has been introduced in recent years which take benefit from low energy use and good network security. But the disadvantage of all Bluetooth networks (even Bluetooth mesh structures) is their high latency [15].

The second protocol which is taken into account here is Wireless Fidelity (Wi-Fi). Wi-Fi is a type of technology for wireless local area communications. In such a network, and due to the mobility provided by Wi-Fi networks for communication systems, the requirement for extending the security is essential. So, Wi-Fi Protected Access protocols WPA, WPA2, and WAP3 [8] are considered as one of the most utilized protocols in wireless networks [16]. Other advantages of the Wi-Fi networks such as no need for a hub, low implementation cost, and low power consumption, makes it a good choice for wireless communication systems.

The third type, ZigBee protocol, is reviewed here. ZigBee development started in 1998, but not until December 2004, when the ZigBee Alliance published its first ratified specification, did it become so popular. Fast communication, low interference, high potential for scaling, and the ability to cover up to 65,000 nodes in a ZigBee network makes them good choices for small range IoT connections like smart homes [17]. Beside these points, dynamically reconfigurable mesh to repair or replace missing nodes, routing tables, address resolution, security, and up to two miles of ideal line-of-sight outdoor range are enumerated as other outstanding features of ZigBee networks. Moreover, different ZigBee devices from the same or various manufacturers can communicate which makes this protocol suitable for home and industrial IoT applications [15]. Because of the low cost and low complexity aspects of ZigBee networks, this kind of wireless communication systems constitutes an attractive alternative for both commercial and military applications [18].

ZigBee in IoT Networks

ZigBee is a popular technology in the industry of smart homes and IoT. This wireless protocol was designed to transfer data in a short range, using the most connected low-power devices. Due to these features, it became one of the most fascinating technologies in industrial application of IoT, such as smart homes and smart phones [19]. For instance, Hydro-Québec uses the 2.4 GHz ZigBee communications devices through the Honeywell REXUniversal meter platform to realize the smart grid of the future [1], Philips Hue connects its bulbs with ZigBee devices, and Amazon implements the smart AI based Alexa for controlling the smart home through application of ZigBee in Echo Plus [20]. These days, ZigBee is a hot technology in smart meters to control the energy usage, local sensors for controlling the temperature and moisture in toxic areas, personal shopping assistance and smart digital carts. In addition, using the "dotdot" universal programming language of IoT devices, developed by Alliance [21], ZigBee devices have the potential to communicate together in a network and this characteristic, makes this technology able to grow faster.

To have a better comparison between Bluetooth, WiFi, and ZigBee wireless protocols, Table 1 is presented here [17, 22–25]. In this table, these three technologies are compared in terms of application, frequency bandwidth, battery Life, number of cell nodes in the network, data rate, transmission range, topology, standby current, and memory.

Table 1: Comparison of Wi-Fi, ZigBee, and Bluetooth technologies [17, 22–25].

Features	**Wi-Fi IEEE 802.11**	**Bluetooth IEEE 802.15.1**	**ZigBee IEEE 802.15.4**
Application	Wireless LAN	Cable Replacement	Control and Monitor
Frequency Bands	2.4 GHz and 5.0 GHz	2.4 GHz	2.4 GHz, 868 MHz, 915 MHz
Battery Life (Days)	0.1-5	1-7	100-7,000
Nodes Per Network	30	7	65,000
Data Rate	2-100 Mbps	1 Mbps	20-250 kbps
Range (Metres)	1-100	1-10	1-75 and more
Topology	Tree	Tree	Star, Tree, Cluster Tree, and Mesh
Standby Current	$20 \cdot 10^{-3}$ amps	$200 \cdot 10^{-6}$ amps	$3 \cdot 10^{-6}$ amps
Memory	100 kB	100 kB	32-60 kB

Transposing the proposed methodology in this thesis to other wireless technologies

As indicated in this chapter, the focus in this thesis is on ZigBee devices. Based on this fact, the discussion in this chapter is concentrated on the data acquisition from Wireless ZigBee devices. The proposed approach is expandable to all other wireless technologies, such as Wi-Fi, WiMAX, Bluetooth, etc. The main point is that the dataset acquisition methodology, challenges, and solution presented in Chapter 3 focuses on ZigBee networks, but the main idea is generalized to all other wireless technologies.

Physical layer fingerprinting

The authentication used in wireless systems include high-level authentication mechanisms. Universal subscriber identity module (USIM) [26], message authentication code (MAC) addresses [27], or service set identifiers (SSID) [28] are some examples of this kind of authentication. However, because of the high possibility of copying the MAC and SSID by hackers, other complicated cryptography-based algorithms, including elliptic curve cryptography (ECC) [29] or one-way hash functions-based approaches [30] have been introduced in recent years. In spite of high level of security provided with these algorithms, due to their high complexity and cost, they are not suitable for generally low-cost networks of IoT devices. In such a case, the radio frequency (RF) signal used by these devices to connect to the wireless network contains unique features in the transmitted signal originated from the manufacturing deviations during the production. These features, referred to as RF distinct and native attributes (RF-DNA), are physical layer features which are unique, secure, less expensive, and hard to copy, which makes RF signal a good candidate for providing security in the wireless communication networks [31].

Artificial Intelligence

In recent years, the reports provided by the Information Technology (IT) companies shows an increase of cyberattacks [32]; for instance, in the 2019 SIM IT Trends [33] and the Cisco Cybersecurity [34] reports, cybersecurity is known as the most challenging IT management problem. Due to the vulnerability of IoT devices to the complicated growing attacks, any IT environment needs an effective security system which is more effective than traditional cybersecurity systems which have failed to detect complex attacks. Consequently, to overcome the security approaches (such as cryptographic key-based identitification) problems, some cybersecurity systems rely on the application of artificial intelligence and machine learning classifiers [10] and [35], because they provide the systems with more accurate attack detentions from the attackers with complicated technologies. Therefore, in this thesis, motivated by the success of deep learning methods at complex classification tasks (such as [36]), this approach is selected for detection and rejection of the rogue devices.

Contributions of the thesis

In this thesis, we propose a mechanism for the classification of features extracted from ZigBee devices, produced by the same or by different manufacturers. The main contributions of our thesis are:

1. **Model design using deep neural networks** The special kind of deep learning model, based on autoencoders, leads to promising results for device classification. In this thesis, using an automated feature extraction structure implemented by an autoencoder deep neural network, essential features from each preamble are extracted. Later on, the extracted features are fed to a fully connected classifier to distinguish between the authorized and rogue devices. Also, adding a long-short term memory layer to the autoencoder structure, the time-dependency of the signal is extracted which is used as an important feature for increasing the accuracy in the neural networks classifier. In addition, with simultaneous training of the decoder and the classifier parts, it is guarantied that the extracted features are meaningful enough for the classifier.

2. **Designing a model for each device** We present a basic structure for the separation of one device from all other devices. This means that the goal of each model is to distinguish one device from all others, known as *one-vs-all* strategy. Consequently, the number of models in such a methodology is equal to the number of authorized devices. Through this strategy, instead of multi-class classification, we can assign a model to each authorized device. Therefore, the mission for each model is to discriminate a specific device, from all other devices in the world which try to copy the identity of this device. In such a case, if a model is responsible to learn a single and unique device, the accuracy will be higher compard to the multi-class case. Then, the selected strategy in this thesis is *one-vs-all*.

3. **Domain transformation** The analysis of classifiers is adapted from the time to the time-scale domain using wavelet transform. In such a case, the wavelet transformation can map the signal from the time domain to the time-scale domain. Using such a transformation, it is possible to extract some features from the signal that can be used for classification. In this thesis, using the Haar wavelet transformation to extract the detail coefficients, the abrupt local changes of generated preambles (data points) by each device are detected to be used in rogue and authorized device discrimination.

4. **Testing the model with unseen devices** We test the model with a subset of devices which has never been seen by the model at the time of training. For such a purpose, before training, the dataset is divided into three subsets, training, validation, and testing. None of these three subsets have any data points in common. This is a main aspect at the time of testing the model with new unseen data points, which is not present in some of the works in the literature. Besides, the dataset which is fed to these models is divided into three parts, including devices which are seen/unseen in the training phase. Some of these unseen devices are from the family of training devices or completely new brands.

5. **Focusing on the time-dependency of data points** The time-dependency of preambles in the dataset is an important factor which can be used for extraction of high accuracy

features, resulting in a high classification rate. After adding a long-short term memory layer to the autoencoder, which is responsible to extract the time-dependency features of the preamble, if we compare the results of the correct classification rates between the autoencoder structures with and without long-short term memory, we can see that the area under the curve for the worst classification case has increased remarkably, after adding this layer, which is a good factor which justifies its efficiency.

Thesis outline

The organization of this thesis is as follows. In Chapter 1, we review the different strategies presented in the literature. By reviewing these methodologies, we can have a better picture of the advantages and disadvantages of each approach.

In Chapter 2, the structure of the deep learning model used in this thesis is presented. Feature extraction is an important step for delivering the maximum quality to the classification process. For such a purpose, the application of autoencoder and recursive neural network structures are proposed and described in detail.

Chapter 3 describes the methodology used to acquire responses from ZigBee devices. During the data acquisition, different distortions such as environmental noise, data transmission delay, and receiver demodulation frequency mismatch are possible to affect the signal quality. The quality of the captured signals in the real environment from the wireless devices is not high enough to make efficient databases. Therefore, the first step after capturing the data is to do a preprocessing on the captured signal. Because the data acquisition and cleaning procedure is not described in detail in the literature, this procedure is described in this chapter.

In Chapter 4, we propose a device classification method based on the autoencoders, as a deep learning model for feature extraction. Using this type of feature extraction model, we deliver a feature dataset with an acceptable level of accuracy in device discrimination, resulting in high level classification rates. The main strategies of this chapter are the *one-vs-all* device selection and the device allocation for the testing.

Next, in Chapter 5, the effect of adding a specific kind of deep layers to the autoencoder is verified, which is suitable for feature extraction from signals with long-term time dependencies, called long short-term memory (LSTM) layer. In addition, the strategies of *one-vs-all* and the testing device allocation are repeated in this chapter.

Finally, a discussion and a conclusion are presented at the end of the thesis. In addition, suggestions for future works are presented which can be used by researchers.

Appendices A and B provide additional information about the model evaluation using confusion matrix and receiver operating characteristic, and LSTM based autoencoders.

Chapter 1

Literature Review on ZigBee Wireless Network Security

In this chapter a review on ZigBee wireless networks security is presented. We will review the methods which have been proposed for the application of security requirements to ZigBee devices in previous years. Network security is defined as "assurance of a PC network and its administrations from unauthorized modification, destruction, or disclosure" [5]. Wireless technologies such as ZigBee devices, as a type of IoTs, focus on three main factors, transmitting data, receiving data, and processing received data [37].

In such an environment, the different types of attacks can jeopardize the transmitted information security, due to the lightweight security protocols used in IoT devices [13]. As mentioned earlier, physical layer attacks jeopardize the strength of underlying security protocols, effectively [9], physical layer security is the most important type of security protocols, since losing the security in other levels is tolerable, but failing at physical layer exposes the information completely [38]. Attacks on physical layer signals existed before the invention of computer networks, and evolved during the last years. Facing the different types of attacks from simple information leakage to complicated eavesdropping approaches in the modern communication systems, security professionals require to recognize the potential thread and present relatively inexpensive and effective approaches such as locks to delay the attacks.

In smart home technologies, private and sensitive information produced by ZigBee networks can be conveyed to the outside world by means of different wired or wireless technologies. Therefore, this integration will definitely increase the requirements in ZigBee networks. Meanwhile, the sensor nodes of the network are often placed in the accessible range of other devices, and in such a manner, the network may be more vulnerable to potential attacks [4]. Security objectives of electronic information systems are determined with respect to the type of threats and vulnerabilities that the system encounters. Here is a list of important security requirements that need to be adopted by the system [5]:

- Confidentiality - Preventing the disclosure of data transmitted through the network to eavesdroppers.
- Integrity – Preventing edition of transmitted data in the network by rogue devices.
- Freshness – Guaranteeing the freshness of the data. By this concept, we ensure that the node that the information delivered is the updated version.
- Availability – Guaranteeing that unauthorized persons or systems cannot apply the disturbance during the access of authorized persons to the assets of the network.
- Authenticity - Affirming the genuine identity of system utilization.

Nowadays, IoT networks use the different wireless technologies such as ZigBee protocols. IoT networks, as a hot topic in industrial networking applications, like AI-based smart devices at homes and vehicles, ease life in the modern era, but high dependency on device-based security of transmitted information put us at risk. Technical report [26] shows that IoT networks have become a vulnerable point for information leakage because of their low efficiency of security protocols. In recent years, a combination of hardware-based and encryption-based approaches has delivered a good performance for increasing the security level in wireless networks [28], but a deeper overview on presented security strategies is presented in this chapter, which clarifies the approach selected for research in this thesis. A summary review is shown in Table 1.1 [39]- [40].

1.1 Review on ZigBee security enhancements

During the last years, in conjunction with the evolution of physical attacks over this period, security strategies have emphasized the use of the uniqueness of inter-device process variations. Previously, the physical-layer device detection methodologies used the physical unclonable functions (PUFs) [41], [42], RF certificates of authenticity [43], and the RF fingerprinting of unique signal within intentional emissions produced by the wireless networking [44], [45], [46] and RFID-based security enhancement technologies [47]. In the following, we will present a short overview of some of the most important methologies among all the proposed ones in previous related works.

1.1.1 Physical unclonable functions

Physical unclonable functions (PUF) techniques are divided into two main groups. Each group employs a different approach for providing the access to the devices entering the network. The first focuses on an integrated circuit (IC) using an internal measurement circuit.

Table 1.1: Summary of security technologies, threats, requirements, and countermeasures [39,40].

Techniques	Security Threats	Security Requirements	Countermeasure Methods
Balanced Security Protocol	Access control attacks – to penetrate a network Authentication attacks – to steal legitimate user identities and credentials to access otherwise private networks and services	Confidentiality & authenticity	Propose WZ-lcp protocol/ scheme Block the malicious attack by new authentication encryption method using XOR calculation 2 times and keys updating based on time synchronization
Spoofing Prevention Using Received Signal Strength	Authentication attacks Integrity attacks – send forged control, management or data frames over wireless to mislead the recipient or facilitate another type of attack (e.g., DoS)	Availability, confidentiality & integrity	Propose spoofing prevention using RSS (Received Signal Strength) Have attack detection module installed on security center and attack prevention module installed on network nodes
Security Enhanced Key Distribution Scheme for AODVjr Routing Protocol	Access control attacks i.e., Man-in-the middle attacks during network formation War driving attacks – false routing information during route discovery	Integrity & authenticity	Introduce new key distribution scheme for secure AODVjr protocol Key distribution is based on the ZigBee handshake protocol and improved Diffie-Hellman algorithm with higher security

The duty of this circuit is on computing a function from the number of glitches or propagation delays [41, 48]. To have an overview on the different PUF based strategies, [1] proposes a silicon based Ring-Oscillator PUF with 3 gates, to prevent any hardware trojan. [3] suggests many advanced PUF based security protocols. The second method is the combination of integrated sensors into the top metal layer of coating dopped dielectric layers of the ICs. This coating needs the internal measurement of the response by an internal circuit [41].

1.1.2 RF certificates of authenticity

The RF certificates of authenticity (RF-COA) technique has a lot in common with the PUF, with the exception that the measurement component is an external modified RFID reader which extracts a fingerprint for computing a COA. In the RF-COA technique, the principle is to attach a small shaped conductive or dielectric into a radio-frequency identification (RFID) device [43].

[49] suggests an ultra-lightweight RFID security protocol. The application of this protocol is for realization of security in block chain systems. In such a network, combining different security protocols with RFID increases the level of information confidentiality, considerably. Due to the low computation complexity, suggested approach in [49] supposed to deliver a high performance.

These days, RFID is a key tool for identification of remote object or people. Due to the application of tags in such a structure, the reader can search through the database of different registered tags and comparison of received tag within the RFID signal, the identity if connecting device is extracted. Although the mentioned tag database provides a relative security for such a network, the requirement to the storage is a challenge [49]. Over the last years, different works such as [50] proposed lightweight security protocols, to be used in low computational power wireless transmission IoT devices. A back end type of security protocol focusing on the mutual RFID authentication protocol is presented in the works such as [51]. The level of accuracy provided by such a protocol can be customized by the user. Through this procedure, the level of computational power and complexity needed will be adjusted to the required security level.

1.1.3 RF-DNA fingerprinting

RF fingerprinting which represents the main foundation of the security verification mechanism of this thesis, has been proposed as a physical-layer technique for increasing the security level in wireless communications (e.g., RFID [47], 802.11 Wi-Fi [52], 802.15 WPAN (wireless personal area network) [45, 46], 802.16 WiMAX (Worldwide Interoperability for Microwave Access) [53], GSM (Global System for Mobile communications) [54]). Beside the reported methods, another common approach is to use the RF fingerprinting focusing on the transient response of a device. In [55], a classification research approach is proposed which uses the RF fingerprinting extracted from the amplitude of the transient parts of the Wi-Fi signal, received from 8 IEEE 802.11b Wi-Fi cards. The device fingerprinting methodology developed herein is mainly based on previous RF-DNA (Radio Frequency DNA) works in [44, 56]. The results in [57] shows a good security response in detection of authorized devices/ICs. Comparing RF-DNA fingerprinting with PUF/RF-COA methodologies shows its ability to absorb the device response without the use of any additional transmitter or authenticate any IC without modification of the internal circuitry [57].

A common feature extraction approach for ZigBee devices is to obtain the statistical parameters such as the mean, variance, skewness, and kurtosis from the physical signal characteristics (amplitude, phase, and frequency) [58–61]. Among all statistical features, phase is the most appropriate physical characteristic for the classification purposes. Another approach for feature extraction is to analyze a fixed length header called preamble [10,62]. This is the mechanism used in this thesis for feature extraction.

In contrast to the methods which focus on adding physically traceable components to devices [63], [64], RF fingerprinting searches for the physical characteristic features of the device [46], [65], [66], [56]. Four special features are numerated for any kinds of RF fingerprinting in communications devices; these features are universality, distinctiveness, permanence, and collectability [65], [66]. Different RF fingerprinting procedures in the literature include transient and steady state methods [56]. Steady-state approaches, focusing on the preambles of the device, are of interest because of predefined device specified preamble standards. The procedure of extraction of RF fingerprinting can be categorized as [67]:

1. Extracting the signal region of interest (ROI).
2. Extracting features from the ROI.
3. Computing fingerprints from these features.
4. Developing classifiers on these features.

In RF fingerprinting, some approaches use instantaneous amplitude only [67], while others beside the amplitude, use instantaneous frequency and/or phase [65]. RF-DNA has been used in the discrimination of devices from different and also the same manufacturers. In the latter case, the devices are with the same model, but different serial numbers [68]. However, RF-DNA considers many fingerprint features such as kurtosis, variance, and skewness of amplitude, phase, and frequency.

1.1.4 Deep learning classification methods

Computational hardware improvements during the last few years, have allowed the application of artificial intelligence in different devices. In [69], a high performance classification scheme using the convolutional neural networks (CNN) is presented which relies on the time-domain signals. Moreover, in [70], a CNN for classification purposes in the IEEE 802.11, 802.15.4, and 802.15.1 protocols, is proposed. In the model presented by [70], the model is fed by the channel frequency and the type of wireless technology.

Although the strategies focusing on a fixed length preamble, [58–61] and [71], presented acceptable classification accuracies, one of the recent approaches for feature extraction presented by [35] used a deep learning model fed by the steady state component of the initial transmitted data points.

The method presented in [35] does not focus on *a priori* knowledge about the transmitted signal or the preamble. The extracted features are completely independent from the received signal such as in [58–61] and [72]. This allows the network to learn the features that best discriminate different devices from each other, without a priori knowledge about these devices. In [35], for the first time, the authors used the frequency compensation to increase the quality of the dataset for better classification accuracy in RF fingerprinting. Removing device-dependent carrier frequency offsets which may appear in low signal-to-noise ratio (SNR) transmissions will enhance the accuracy of detection/rejection of unauthorized/rogue/spoofing nodes, and decrease the probability of frequency variations at baseband. While the results from [35] were 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 thesis addresses this issue. The idea behind the proposed scenario of this work is *one-vs-all*, which contains the work of [73], [10], and [62].

Why do we use *one-vs-all* instead of multi-class classification? Is the one-vs-all strategy artificial for this work? Like the methodologies in [73], [10], and [62], due to the high complexity of the dataset used in this thesis, using a multi-class classification approach is not as efficient as needed. Therefore, designing a model for each device whose duty is to distinguish between that specific device and all other devices is a good workaround. In such a case, training a model with enough data points from a specific device as the positive device and the data points from some other devices playing the role of rogue or attacking units trying to falsifying the network by cloning the physical characteristics of the positive device in the real world, makes us able to design an efficient enough network for rejection of all different attacking devices from having access to the network cloning the identity of mentioned positive device. Through this procedure, we can increase the security level in wireless communication systems.

1.1.5 Wavelet-based classification methods

Nowadays, the application of wavelet transform for signal processing is very popular in medicine, communications, engineering, etc. For instance, [74] presents an approach using wavelet transform for risk analysis in financial systems. Classification methods based on wavelet transform [75] have shown successful RF fingerprinting performances in this field, focusing on neural networks to design models of nonlinear power amplifiers and to perform pre-distortion [76, 77]. Other works such as [65, 66] proposed a time-scale based wavelet transform method for RF-DNA fingerprinting.

The main idea behind the transformation in this work is to provide a kind of feature space providing the required separated enough classes in the dataset for the classification network. Using this strategy, the classification is more successful in discrimination of the different classes which leads in higher accuracy of the model. Using the wavelet transform, the local properties of the signal are characterized, and the robust fingerprint features of non-transient preamble are extracted [66], which can be used as an effective feature set for the classifier.

Another methodology for security in physical layer was to propose a wavelet transformed form of the randomly modulated signal [78]. The result of this combination was a low bit error (BER) in the transmitted signal. In [79,80], an approach based on wavelet transform for detecting false data injection attacks (FDIA), as a specific kind of cyber attacks, is presented. In this work, using the wavelet singular entropy (WSE) of the current and voltage signals, the data correlation is extracted to calculate the error signal, with respect to the base signal. The selected wavelet window in this work is the Haar window to extract the detailed coefficients, due to its sensitivity to abrupt and high frequency local changes [81] of the signal. Using such an approach presented a detection accuracy of over 96.5% through simulations.

Other examples that focused on the application of the wavelet transform in recent years are [56, 82] which focus on fault detection, [83] that presents the wavelet multi-resolution analysis (MRA) method for fault detection, and [84] combining the fuzzy analysis and wavelet transformation to extract the sharp local changes.

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

1.1.6 Other proposed approaches to improve security

In [39], the authors suggested a new set of scheme called WZ-lcp to increase the security against attacks in wireless communication networks. The suggested scheme basically uses a new method of encryption using an XOR calculation. In addition, this scheme also uses the updating keys based on synchronization of time as a variable. By this mechanism, the data obtained through eavesdrop will be useless. In [85], the authors proposed to use ZigBee's received signal strength (RSS) to prevent the ZigBee-based home networks from facing the spoofing attacks for cloning the identity of an authorized device and experiencing interruption in the normal operation of the network. In [40], the increment of ZigBee networks is realized though using the Ad hoc on-demand distance vector junior (AODVjr) routing protocol. AODVjr is one of the most widely adopted routing algorithms in ZigBee protocol [86]. In this case, the part of routing information must be protected through the use of new distribution schemes. By doing this, they claimed that the improvement can prohibit a man-in-the-middle attack and also false routing information attacks during the routing mechanism of networks. Later, [39] and [40] revised the mentioned scheme to improve the security, while [85] used physical layer PHY information of the RSS ZigBee network to detect and filter any malicious attack. Although the mentioned mechanisms load more tasks on the nodes of the network, the rate of increment is still within an acceptable range and it can be implemented in wireless networks such as ZigBee. Besides the mentioned studies, [39] and [40] focused on security at the nodes which requires efficiency and accuracy in determining the presence of attacks. In other words, a pro-active mechanism before an attack occurs while [85] is more passive in which it would only act if the attack happens. However, [39], [85], [40] did specify another security need in ZigBee networks, the forward security, which is also important in handling security of the device leaving the network. Using this idea, in [87], the authors focused on the problem that each device may leave the network when it has either reached its aim, or when it is sent for maintenance, or when is compelled to leave the network. Regardless of the case, a device that has left the network should not have the capacity to get to any further information exchange within the interior devices. Therefore, a forward security requisite can increase the security level in these systems.

1.2 Conclusion

In this chapter, we have presented a review of the current literature on the physical layer security of wireless communication devices. As discussed, some of the approaches used in the literature for device discrimination in ZigBee networks are the PUF techniques, using the RF-COA, extracting the RF-DNA features, focusing on the deep learning models, etc.

The strategy which is selected in this thesis is to use the *one-vs-all* concept for device time domain and time-scale domain datasets allocation in training, validation, and testing. The allocated dataset is fed to a deep neural network model for device discrimination purposes. The dataset which is used for this purpose is captured from real ZigBee devices, and the resulting performance of the model is evaluated through machine learning algorithms.

Chapter 2

Deep Learning Models Structures and Evaluation

In recent years, due to the improvement of hardware computational power, the application of deeper learning models has increased. Modern hardware capabilities give us the ability to design and use more complicated and deep models resulting in higher performance accuracy. For this reason, the application of two deep learning models are suggested here, autoencoder and recurrent neural networks [88]. To have a better idea about the performance of these systems, the data structure and the theoretical concept behind these networks is described in the next three sections of this chapter. Finally, last two sections of this chapter are allocated to model training and validation, and conclusion.

2.1 Dataset structure

Research in this thesis starts with data acquisition from real devices. However, signal acquisition from physical environment requires impairment mitigation techniques to reduce the effects of environmental noise and distortion, data transmission delay, receiver demodulation frequency mismatch, and imperfect signal bursts synchronization. Therefore, the experimental data need to be preprocessed and cleaned before being applied to the actual proposed classifiers. This preprocessing procedure is explained in Chapter 3. After preprocessing the acquired data, it is called the dataset and ready to be fed to the classifier. The structure of the dataset is shown in Table 2.1. As seen, each of M devices (D_m for $m \in \{1, \ldots, M\}$) has N_m data points, and the summation of the numbers of data points from all devices is equal to dataset size (N). Also, each data point is a vector with K samples $x_{n,k}$ for $n \in \{1, \ldots, q, \ldots, l, \ldots, N\}$.

Table 2.1: Dataset structure.

	Data point			Device
$(\mathbf{x}_n)_{n=1}$	$(x_{n,k})_{\substack{n=1\\k=1}}$	$\dots$	$(x_{n,k})_{\substack{n=1\\k=K}}$	D_1
$\vdots$		$\vdots$		$\vdots$
$(\mathbf{x}_n)_{n=N_1}$	$(x_{n,k})_{\substack{n=N_1\\k=1}}$	$\vdots$	$(x_{n,k})_{\substack{n=N_1\\k=K}}$	D_1
$\vdots$		$\vdots$		$\vdots$
$(\mathbf{x}_n)_{n=q}$	$(x_{n,k})_{\substack{n=q\\k=1}}$	$\dots$	$(x_{n,k})_{\substack{n=q\\k=K}}$	D_m
$\vdots$		$\vdots$		$\vdots$
$(\mathbf{x}_n)_{n=q-1+N_m}$	$(x_{n,k})_{\substack{n=q-1+N_m\\k=1}}$	$\dots$	$(x_{n,k})_{\substack{n=q-1+N_m\\k=K}}$	D_m
$\vdots$		$\vdots$		$\vdots$
$(\mathbf{x}_n)_{n=l}$	$(x_{n,k})_{\substack{n=l\\k=1}}$	$\dots$	$(x_{n,k})_{\substack{n=l\\k=K}}$	D_M
$\vdots$		$\vdots$		$\vdots$
$(\mathbf{x}_n)_{n=l-1+N_M}$	$(x_{n,k})_{\substack{n=l-1+N_M\\k=1}}$	$\dots$	$(x_{n,k})_{\substack{n=l-1+N_M\\k=K}}$	D_M
$N_1 + \cdots + N_m + \cdots + N_M = N$				

2.2 Feature extraction model

In this thesis, a binary classification system is used to provide a mechanism of device discrimination into 2 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 or not. If it is an authorized one, it will be granted access to the network. If not, it will be rejected by the network. For this aim, the model adopted is an *autoencoder (AE) combined with a fully connected classifier*. The model structure is shown in Fig. 2.1.

The question that arises here is that why using an autoencoder for device classification? Methods focusing on known RF-DNA features such as statistical parameters (mean, variance, skewness, and kurtosis) of amplitude, phase, and frequency [58–61], try to extract the best features from the measured data, resulting in the maximum possible classification rate. Among all RF-DNA features, phase is assigned as the most effective one for the classification purposes in the literature. Based on this, the question that arises is about the feature selection mechanism [10]. 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 used in this thesis focuses on the utilization of an autoencoder to

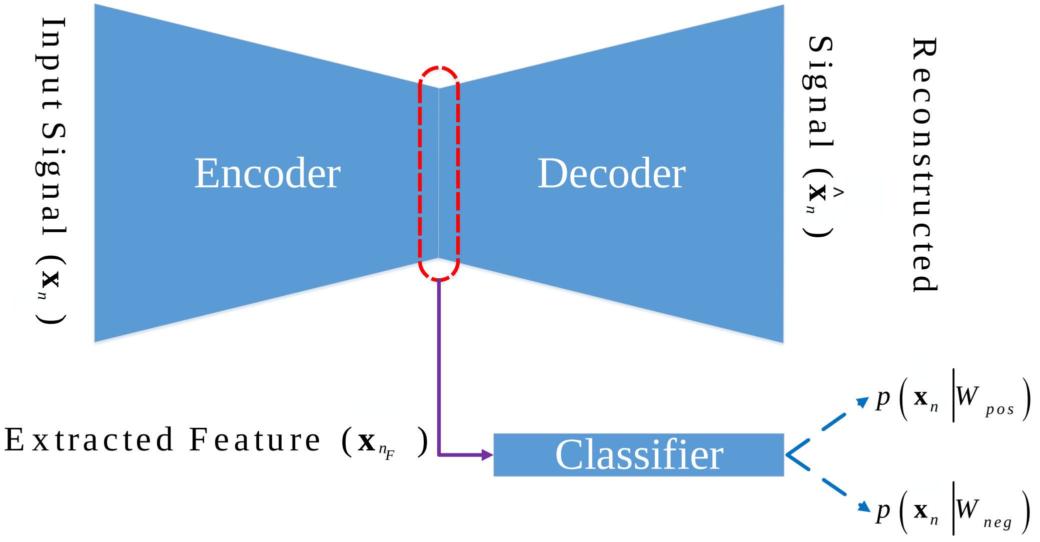

Input Signal ($\mathbf{x}_n$)
Encoder
Decoder
Signal ($\hat{\mathbf{x}}_n$)
Reconstructed
Extracted Feature ($\mathbf{x}_{n_F}$)
Classifier
$p\left(\mathbf{x}_n \middle| W_{pos}\right)$
$p\left(\mathbf{x}_n \middle| W_{neg}\right)$

Figure 2.1: Model symbolic representation.

extract the most dominant features of the input data, which give the maximum inter-class and minimum intra-class distances [89].

Autoencoder

An autoencoder consists of two parts, an encoder and a decoder, as shown in Fig. 2.1. The input vector, $\mathbf{x}_n$ represents a data point:

$$\mathbf{x}_n \in \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \tag{2.1}$$

where N is the total number of data points in the dataset. Vector $\mathbf{x}_n$ contains K samples:

$$\mathbf{x}_n = [x_{n,1}, x_{n,2}, \cdots, x_{n,k}, \cdots, x_{n,K}]^\intercal \tag{2.2}$$

The output of the encoder, $\mathbf{x}_{n_F}$, is a simplified representation of $\mathbf{x}_n$. The decoder is designed so that its output, $\hat{\mathbf{x}}_n$, reproduces the original dataset, $\mathbf{x}_n$, from the encoder's representation, $\mathbf{x}_{n_F}$, by minimizing the difference between $\mathbf{x}_n$ and $\hat{\mathbf{x}}_n$ as illustrated in Fig. 2.1. The mechanism for decreasing this difference is through the mean squared error (MSE). Eq. (2.3) presents the output data point $\hat{\mathbf{x}}_n$.

$$\hat{\mathbf{x}}_n = [\hat{x}_{n,1}, \hat{x}_{n,2}, \cdots, \hat{x}_{n,k}, \cdots, \hat{x}_{n,K}]^\intercal \tag{2.3}$$

Feature extraction by the encoder Feature extraction maps the high dimensional data to a simplified low-dimensional space [90]. This transformation can be either linear or nonlinear. Specifically, considering a given data point $\mathbf{x}_n$, feature extraction generates a new feature

$\mathbf{x}_{n_F}$. The encoder can be described as a function $f(.)$ that maps an input $\mathbf{x}_n$ to a hidden representation $\mathbf{x}_{n_F}$:

$$\mathbf{x}_{n_F} = f\left(\mathbf{x}_n\right) = s_f\left(\Omega \mathbf{x}_n + \mathbf{b}_{\mathbf{x}_n}\right) \tag{2.4}$$

where $s_f(.)$ is a linear or a nonlinear activation function. The encoder is parameterized by a weight matrix Ω and a bias vector $\mathbf{b}_{\mathbf{x}_n} \in \mathbb{R}^n$.

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

$$\hat{\mathbf{x}}_n = g\left(\mathbf{x}_{n_F}\right) = s_g\left(\Omega' \mathbf{x}_{n_F} + \mathbf{b}'_{\mathbf{x}_{n_F}}\right) \tag{2.5}$$

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

Training an autoencoder involves finding parameter $\theta = (\Omega, \Omega', \mathbf{b}_{\mathbf{x}_n}, \mathbf{b}'_{\mathbf{x}_{n_F}})$ using a loss function that minimizes the difference between the original space $\mathbf{x}_n$ and the reconstruction space $\hat{\mathbf{x}}_n$.

It is worth mentioning that the importance of the decoder is at the training stage, due to the fact that simultaneous training of the decoder and classifier outputs guaranties that the encoder part provides the classifier with the meaningful input features resulting in the high classification accuracy. Therefore, at the testing stage, there is no need for the decoder part, anymore, and it can be removed from the model to reduce the required computational power for using the proposed model in the time of testing.

Classification

After extraction of the features from the input dataset, these extracted features are fed to the classifier section depicted in Fig. 2.1. The typical classification structure used in the literature involves two connected layers. However, such a structure may overfit the training data, unless the training dataset is very large [91].

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

$$p\left(\mathbf{x}_n \left| W_{pos}\right.\right) = 1 - p\left(\mathbf{x}_n \left| W_{neg}\right.\right) \tag{2.6}$$

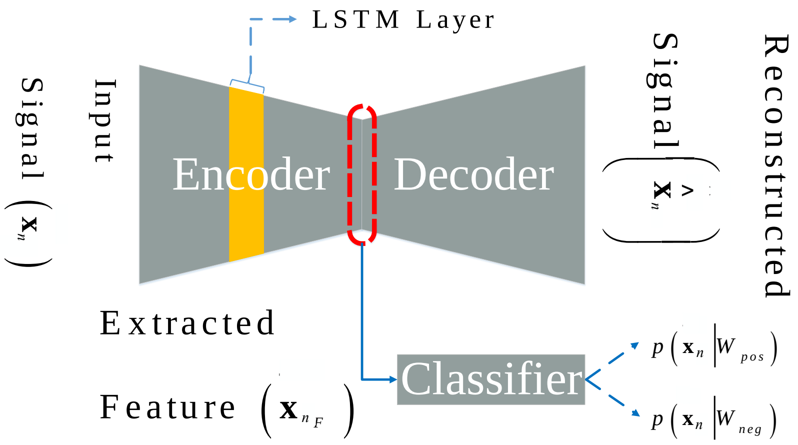

LSTM Layer
Input Signal $(\mathbf{x}_n)$
Encoder
Decoder
Reconstructed Signal $(\hat{\mathbf{x}}_n)$
Extracted Feature $(\mathbf{x}_{n_F})$
Classifier
$p(\mathbf{x}_n|W_{pos})$
$p(\mathbf{x}_n|W_{neg})$

Figure 2.2: Model symbolic representation with LSTM layer.

2.3 Recurrent neural networks

An RNN (recurrent neural network) is a neural network that is specialized for processing a sequence of values [88]. Each preamble that is captured from a ZigBee device is a signal which has a time-dependency between the samples. An RNN has the ability to extract this time-dependency and keeping this idea in mind, an RNN based network can scale to much longer sequences than those used in the simple feedforward ones.

In Chapter 5 we will use a specific type of RNNs called long short-term memory (LSTM) in the encoder part of the autoencoder, as shown in Fig. 2.2. The structure of the RNNs and LSTM layers are described in detail in the next section. Through analyzing an RNN layer using the mathematical equations, the real merit of addition of an LSTM layer for time-dependency extraction, which is the main topic of Chapter 5, will be clarified.

2.3.1 RNN structures

In some sequences, the time-dependency of elements of a data point from a dataset is crucial. For instance, in a single input data point vector $\mathbf{x}_n$ consists of K samples, as shown in Eqs. (2.2) and (2.3). Corresponding to each input sample, there is an output sample $\hat{x}_{n,k}$. It is worth mentioning that the samples $x_{n,k}$ and $\hat{x}_{n,k}$ $(1 \leq k \leq K)$ are scalars. Referring to Fig. 2.4 [88], showing an RNN cell feedback loop, the input and output of the RNN, $x_{n,k}^{RNN}$ and $\mathbf{o}_{n,k}^{RNN}$, can be interpreted as the input sample (scalar value) and output vector of the RNN cell at time k, for $1 \leq k \leq K_{RNN} \leq K$, as shown in Eq. (2.7). As can be seen, the input and output of RNN cell after K_{RNN} time steps are $\mathbf{x}_n^{RNN}$ and O_n^{RNN} $(1 \leq n \leq N)$, respectively.

Based on the design of the autoencoder in Fig. (2.3), which will be used and described in more detail in this chapter, $\mathbf{x}_n^{RNN}$ and O_n^{RNN} are the mapped versions of the input and

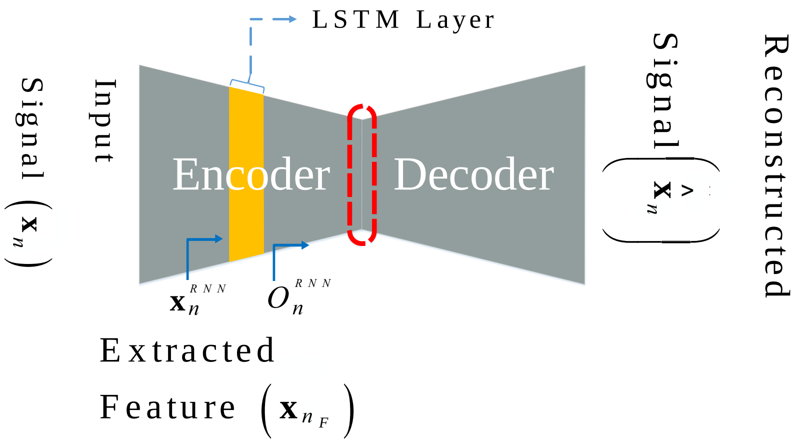

LSTM Layer
Input Signal $(\mathbf{x}_n)$
Encoder
Decoder
Reconstructed Signal $(\hat{\mathbf{x}}_n)$
$\mathbf{x}_n^{RNN}$
O_n^{RNN}
Extracted Feature $(\mathbf{x}_{n_F})$

Figure 2.3: Input and output for the RNN structure in an autoencoder.

output in Eq. (2.3), $\mathbf{x}_n$ and $\hat{\mathbf{x}}_n$ $(1 \leq n \leq N)$, respectively. As can be seen, the samples $x_{n,k}^{RNN}$ and b_n^j $(1 \leq n \leq N,\ 1 \leq k \leq K_{RNN} \leq K$ and $1 \leq j \leq N_{unit})$ are scalars, and the size of an output from an RNN cell after K_{RNN} time steps is dependent on a factor called N_{unit}, which is the number of units in a cell.

$$\mathbf{x}_n^{RNN} = \left[x_{n,1}^{RNN}, x_{n,2}^{RNN}, \cdots, x_{n,k}^{RNN}, \cdots, x_{n,K_{RNN}}^{RNN}\right]^\mathsf{T} = [a_1, a_2, \cdots, a_k, \cdots, a_{K_{RNN}}]^\mathsf{T}$$
$$O_n^{RNN} = \begin{bmatrix} \mathbf{o}_{n,1}^{RNN} \\ \vdots \\ \mathbf{o}_{n,k}^{RNN} \\ \vdots \\ \mathbf{o}_{n,K_{RNN}}^{RNN} \end{bmatrix} = \begin{bmatrix} [b_1^1, ..., b_1^{N_{unit}}] \\ \vdots \\ [b_k^1, ..., b_k^{N_{unit}}] \\ \vdots \\ [b_{K_{RNN}}^1, ..., b_{K_{RNN}}^{N_{unit}}] \end{bmatrix} \tag{2.7}$$

In the structure of Fig. 2.4, the Z block represents a delay block, $\mathbf{h}_k$ is the current state of the system at time k, which is a fixed length vector [88]. The size of this vector is dependent on the structure of an RNN cell. As depicted, a feedback loop feeds the output of the previous time to the network, resulting in the current output. An expanded form of this model is illustrated in Fig. 2.5 [88]. In each step, the input of the model is composed of the current input and the previous output, as shown in Eq. (2.8).

$$\mathbf{o}_{n,k}^{RNN} = f(x_{n,k}^{RNN}, \mathbf{o}_{n,k-1}^{RNN}) \tag{2.8}$$

where $x_{n,k}^{RNN}$ is the input scalar to the network at time k, $\mathbf{o}_{n,k-1}^{RNN}$ is the output vector of the network at time $k-1$, and $f(.)$ represents the network function.

The structures shown in Fig. 2.6 illustrates the simplest structure for an RNN network. Different and more complicated structures are introduced in Chapter 10 of [88]. For instance,

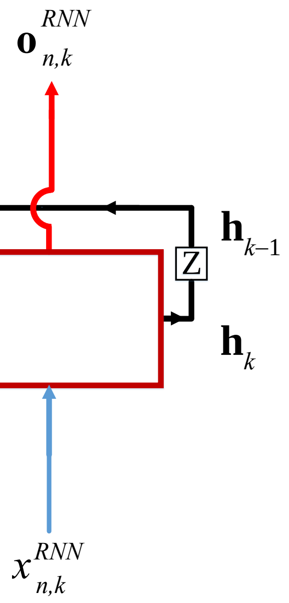


Figure 2.4: Self state loop feedback of an RNN network [88].

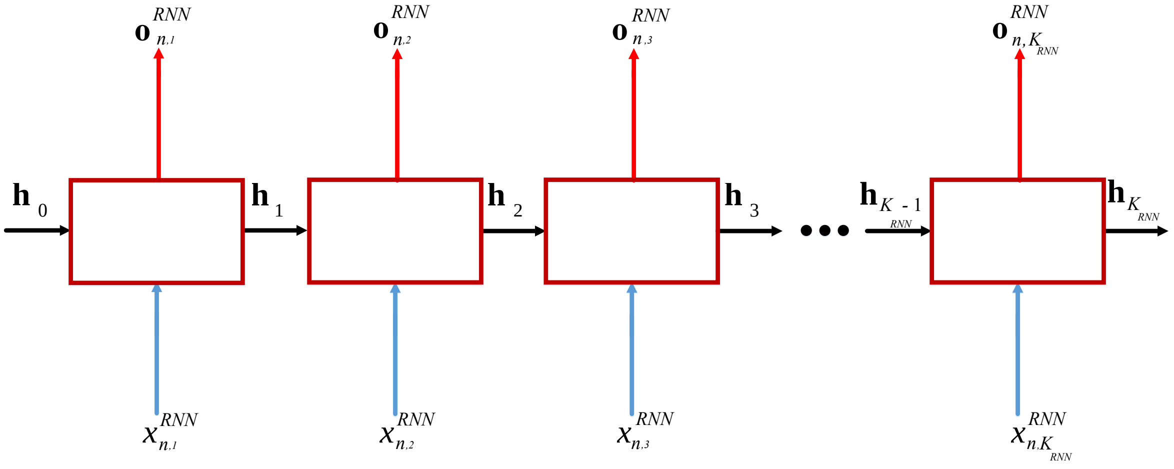


Figure 2.5: Expanded form of RNN state feedback loop [88].

the structures shown in Figs. 2.4 and 2.5 are special kinds of networks which use the feedback of $\mathbf{h}_{k-1}$:

$$\mathbf{o}_{n,k}^{RNN} = f(x_{n,k}^{RNN}, \mathbf{h}_{k-1}) \tag{2.9}$$

where $\mathbf{h}_{k-1}$ is the state of the system at time $k-1$, $f(.)$ is the network function, $x_{n,k}^{RNN}$ and $\mathbf{o}_{n,k}^{RNN}$ are the input sample and the output vector of the system at time n, respectively.

Simple RNN

The detailed structure of the cell in Fig. 2.4 is shown in Fig. 2.6 [92].

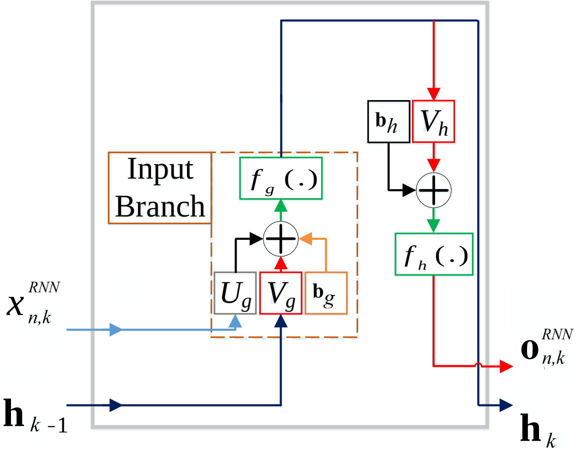


Figure 2.6: Cell unit structure for a simple RNN at the time-step k [92].

The output vector state of all cell units of the network for the time-step k is [88]:

$$\mathbf{h}_k = f_g(U_g x_{n,k}^{RNN} + V_g \mathbf{h}_{k-1} + \mathbf{b}_g) \tag{2.10}$$

where U_g and V_g are the weight matrices, $\mathbf{b}_g$ is the bias vector, and $f_g(.)$ is the function which is applied to obtain the current state $\mathbf{h}_k$. As can bee seen in Eq. (2.10), the feedback from the output vector state at time-step $\mathbf{h}_{k-1}$ to the output vector state at time-step $\mathbf{h}_k$ provides the loop feedback of the network.

The output vector $\mathbf{o}_{n,k}^{RNN}$ of each step in the model is calculated as follows.

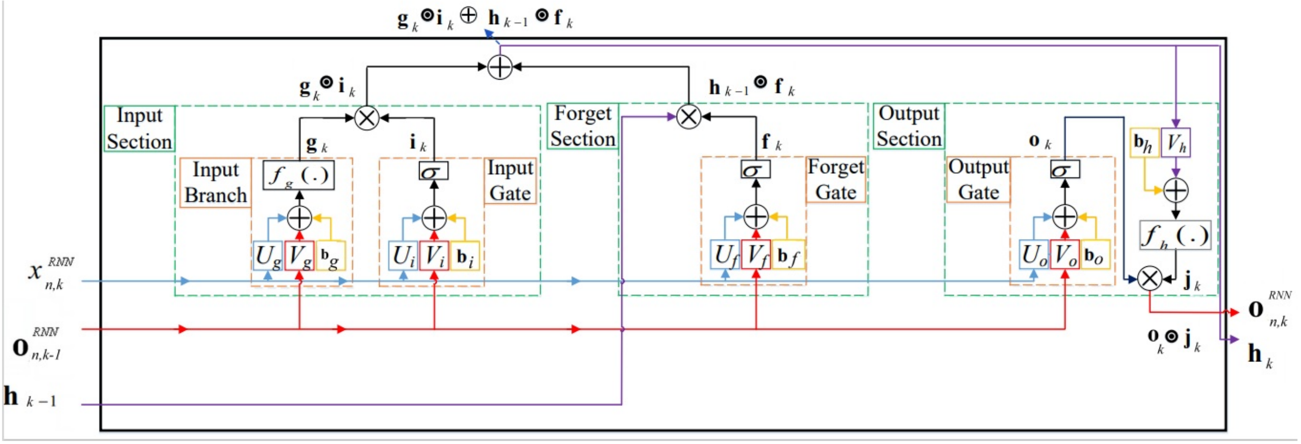
$$\mathbf{o}_{n,k}^{RNN} = f_h(V_h \mathbf{h}_k + \mathbf{b}_h) \tag{2.11}$$

where V_h is the weight matrix, $\mathbf{b}_h$ is called the bias vector, and $f_h(.)$ is the system output function.

The most important point about Eqs. (2.10) and (2.11) is about the set of parameters $S = \left[U_g, V_g, V_h, \mathbf{b}_g, \mathbf{b}_h\right]$. Recurrent networks share parameters through time, meaning that each member of the output $\mathbf{o}_{n,k}^{RNN}$ for $1 \leq n \leq N$ and $1 \leq k \leq K_{RNN}$ is produced using the same parameter set S applied to the previous outputs.

2.3.2 Long short-term memory

Compared to the cell unit structure of a simple RNN at the time-step k shown in Fig. 2.6, it is preferred that the output of each unit cell ($\mathbf{o}_{n,k}^{RNN}$) be a function of the current input (x_n^{RNN}), the previous output ($\mathbf{o}_{n,k-1}^{RNN}$), and the previous state ($\mathbf{h}_{k-1}$). Adding a gate named "Output Gate", we can control what information encoded in the cell state is sent to the network as the input in the following time step. This is done via the output vector $\mathbf{o}_{n,k}^{RNN}$. The revised version of Fig. 2.6 is shown in Fig. 2.7



$\mathbf{g}_k \odot \mathbf{i}_k \oplus \mathbf{h}_{k-1} \odot \mathbf{f}_k$
$\mathbf{g}_k \odot \mathbf{i}_k$
$\mathbf{h}_{k-1} \odot \mathbf{f}_k$
Input Section
Forget Section
Output Section
Input Branch
Input Gate
Forget Gate
Output Gate
$\mathbf{g}_k$
$\mathbf{i}_k$
$\mathbf{f}_k$
$\mathbf{o}_k$
$f_g(.)$
σ
σ
σ
U_g V_g $\mathbf{b}_g$
U_i V_i $\mathbf{b}_i$
U_f V_f $\mathbf{b}_f$
U_o V_o $\mathbf{b}_o$
$\mathbf{b}_h$ V_h
$f_h(.)$
$\mathbf{j}_k$
$x_{n,k}^{RNN}$
$\mathbf{o}_{n,k-1}^{RNN}$
$\mathbf{h}_{k-1}$
$\mathbf{o}_{n,k}^{RNN}$
$\mathbf{o}_k \odot \mathbf{j}_k$
$\mathbf{h}_k$

Figure 2.7: Effect of the Output Gate on the cell units of an LSTM layer [93].

In this new structure, the current output ($\mathbf{o}_{n,k}^{RNN}$) at time-step k of all cell units of the LSTM layer is as follows [93]:

$$\begin{aligned} \mathbf{o}_k &= \sigma\left(U_o x_{n,k}^{RNN} + V_o \mathbf{o}_{n,k-1}^{RNN} + \mathbf{b}_o\right) \\ \mathbf{j}_k &= f_h\left(V_h \mathbf{h}_k + \mathbf{b}_h\right) \end{aligned} \tag{2.12}$$

$$\begin{aligned} \mathbf{o}_{n,k}^{RNN} &= \mathbf{o}_k \odot \mathbf{j}_k \\ &= f_h\left(V_h \mathbf{h}_k + \mathbf{b}_h\right) \odot \sigma\left(U_o x_{n,k}^{RNN} + V_o \mathbf{o}_{n,k-1}^{RNN} + \mathbf{b}_o\right) \end{aligned} \tag{2.13}$$

where $\odot$ is the inner product of two vectors. U_o, V_o, and V_h are weight matrices, $\mathbf{b}_o$ and $\mathbf{b}_h$ are the bias vectors.

Vanishing Moment Problem

In the forward pass, a set of weight matrices (U_g, V_g, and V_h) are used to pass the input to the output of the model. As described, the whole parameter set S is shared through time. For simplicity, let us ignore the bias vectors ($\mathbf{b}_g$ and $\mathbf{b}_h$) and redefine parameters set S as:

$$S = \left[U_g, V_g, V_h\right] \tag{2.14}$$

In Eq. (2.14), each weighting matrices U_g, V_g, and V_h are referred to as S_i for $1 \leq i \leq 3$. The output vector should minimize the error between the expected ($\mathbf{o}_{ex_{n,k}}^{RNN}$) and the real ($\mathbf{o}_{n,k}^{RNN}$) outputs:

$$Error = (\mathbf{o}_{ex_{n,k}}^{RNN} - \mathbf{o}_{n,k}^{RNN})^2 \tag{2.15}$$

Keeping in mind that the set of weights which are selected in the first place are random and need to be trained, a log likelihood cost function is defined as follows, which will be used for updating the weights [88]:

$$L_{n,k} = -\log\left(p\left(\mathbf{o}_{n,k}^{RNN}|x_{n,1}^{RNN}, ..., x_{n,k}^{RNN}\right)\right) \tag{2.16}$$

The output vector error of the model at time $k = 1, \cdots, K_{RNN}$ in Fig. 2.5 is:

$$L_n = \sum_{k=1}^{K_{RNN}} L_{n,k} = -\sum_{k=1}^{K_{RNN}} \log\left(p\left(\mathbf{o}_{n,k}^{RNN}|x_{n,1}^{RNN}, \ldots, x_{n,k}^{RNN}\right)\right) \tag{2.17}$$

Here, the overall error for all N data points from the dataset is:

$$L = \sum_{n=1}^{N} L_n \tag{2.18}$$

After this step, the gradient of the weights for minimizing the cost function error is [94]:

$$\frac{\partial L}{\partial S_i} = \sum_{n=1}^{N} \sum_{k=1}^{K_{RNN}} \frac{\partial L_{n,k}}{\partial S_i} \tag{2.19}$$

The weights are updated through the backpropagation procedure.

$$S_i = S_i - \alpha \frac{\partial L}{\partial S_i} \tag{2.20}$$

Vanishing moment of LSTM The structure of an LSTM cell unit is shown in Fig. 2.7. The output of "Forget Gate" in Fig. 2.7 controls the part of the information in the cell state which should be forgotten, given the new input to the network. In this structure, the new output of the system is [93]:

$$\mathbf{f}_k = \quad \sigma\left(U_f x^{RNN}_{n,k} + V_f \mathbf{o}^{RNN}_{n,k-1} + \mathbf{b}_f\right) \tag{2.21}$$

Therefore, the new state ($\mathbf{h}_k$) in this structure is:

$$\begin{aligned}\mathbf{h}_k &= \mathbf{g}_k \odot \mathbf{i}_k \oplus \mathbf{h}_{k-1} \odot \mathbf{f}_k \\ &= f_g\left(U_g x^{RNN}_{n,k} + V_g \mathbf{o}^{RNN}_{n,k-1} + \mathbf{b}_g\right) \odot \sigma\left(U_i x^{RNN}_{n,k} + V_i \mathbf{o}^{RNN}_{n,k-1} + \mathbf{b}_i\right) \\ &\oplus \mathbf{h}_{k-1} \odot \sigma\left(U_f x^{RNN}_{n,k} + V_f \mathbf{o}^{RNN}_{n,k-1} + \mathbf{b}_f\right)\end{aligned} \tag{2.22}$$

where U_i, U_f, U_o, V_i, V_f, and V_o, are weight matrices, $\mathbf{b}_i$, $\mathbf{b}_f$, and $\mathbf{b}_o$ are the bias vectors. Using Eq. (2.19) and the chain rule, we have:

$$\begin{aligned}\frac{\partial L_{n,k}}{\partial S_i} &= \frac{\partial L_{n,k}}{\partial \mathbf{o}^{RNN}_{n,k}} \times \frac{\partial \mathbf{o}^{RNN}_{n,k}}{\partial \mathbf{h}_k} \times \frac{\partial \mathbf{h}_k}{\partial \mathbf{h}_{k-1}} \times \cdots \times \frac{\partial \mathbf{h}_2}{\partial \mathbf{h}_1} \times \frac{\partial \mathbf{h}_1}{\partial S_i} \\ &= \frac{\partial L_{n,k}}{\partial \mathbf{o}^{RNN}_{n,k}} \times \frac{\partial \mathbf{o}^{RNN}_{n,k}}{\partial \mathbf{h}_k} \times \prod_{j=2}^{k}\left(\frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}}\right) \times \frac{\partial \mathbf{h}_1}{\partial S_i}\end{aligned} \tag{2.23}$$

Now,

$$\begin{aligned}\frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}} &= \sigma\left(U_f x^{RNN}_{n,j} + V_f \mathbf{o}^{RNN}_{n,j-1} + \mathbf{b}_f\right) \\ &\oplus \frac{\partial}{\partial \mathbf{h}_{j-1}}\left[f_g\left(U_g x^{RNN}_{u,j} + V_g \mathbf{o}^{RNN}_{u,j-1} + \mathbf{b}_g\right) \odot \sigma\left(U_i x^{RNN}_{u,j} + V_i \mathbf{o}^{RNN}_{u,j-1} + \mathbf{b}_i\right)\right]\end{aligned} \tag{2.24}$$

which if $\sigma\left(U_f x^{RNN}_{n,j} + V_f \mathbf{o}^{RNN}_{n,J-1} + \mathbf{b}_f\right)$ (as the output of the "Forget Gate") is greater than 0, guaranties that even for the specific type of the hyperbolic tangent function $\left(f_g(.) = \tanh(.)\right)$ [93], which is often used, with the limits of the derivative of this function:

$$0 \leqslant \tanh'(x) \leqslant 1 \tag{2.25}$$

where $\tanh'(x) = \frac{\partial \tanh(x)}{\partial x}$, for $k \gg 1$ Eq. 2.24 does not converge to zero. Therefore, replacing Eq. (2.24) in Eq. (2.23) results in:

$$\begin{aligned}\frac{\partial L_{n,k}}{\partial S_i} &= \frac{\partial L_{n,k}}{\partial \mathbf{o}^{RNN}_{n,k}} \times \frac{\partial \mathbf{o}^{RNN}_{n,k}}{\partial \mathbf{h}_k} \times \\ &\prod_{j=2}^{k}\left(\sigma\left(U_f x^{RNN}_{n,j} + V_f \mathbf{o}^{RNN}_{n,j-1} + \mathbf{b}_f\right)\right. \\ &\left.\oplus \frac{\partial}{\partial \mathbf{h}_{k-1}}\left[f_g\left(U_g x^{RNN}_{n,k} + V_g \mathbf{o}^{RNN}_{n,k-1} + \mathbf{b}_g\right) \odot \sigma\left(U_i x^{RNN}_{n,k} + V_i \mathbf{o}^{RNN}_{n,k-1} + \mathbf{b}_i\right)\right]\right) \\ &\quad \times \frac{\partial \mathbf{h}_1}{\partial S_i}\end{aligned} \tag{2.26}$$

$\frac{\partial \mathbf{h}_k}{\partial \mathbf{h}_{k-1}}$ does not vanish, even for deep RNN networks (equivalent to $k \gg 1$):

$$\frac{\partial L_{n,k}}{\partial S_i} \not\to 0 \tag{2.27}$$

As shown in Eq. (2.26), because of the fact that $\sigma\left(U_f x_{n,j}^{RNN} + V_f \mathbf{o}_{n,j-1}^{RNN} + \mathbf{b}_f\right)$ will not converge to 0 for any values of k, regardless of the function selected as $f_g(.)$, or despite the factor $\frac{\partial}{\partial \mathbf{h}_{j-1}}\left[f_g\left(U_g x_{n,j}^{RNN} + V_g \mathbf{o}_{n,j-1}^{RNN} + \mathbf{b}_g\right) \odot \sigma\left(U_i x_{n,j}^{RNN} + V_i \mathbf{o}_{n,j-1}^{RNN} + \mathbf{b}_i\right)\right]$, the vanishing moment phenomenon in Eq. (2.27) will never happen for the structure of the LSTM shown in Fig. 2.7.

Finally, Eq. (2.20) will update the coefficients. Since $\sigma\left(U_f x_{n,k}^{RNN} + V_f \mathbf{o}_{n,k-1}^{RNN} + \mathbf{b}_f\right)$ is the most important term, Eq. (2.26) means that the gradient behaves similarly to the "Forget Gate", and if the "Forget Gate" decides that a certain piece of information should be remembered, it will be open and have values closer to 1 to allow for information flow.

2.4 Model training and validation procedure

After model definition, the model is trained using the extracted dataset. During training of the model, the part of the dataset of the devices assigned to training is fed to the classifier. The output values of of the decoder and classifier should 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 the reconstruction shown in Fig. 2.1 during the training or the validation processes. The mean squared error (MSE) between the input and the reconstructed data points, $\mathbf{x}_n$ and $\hat{\mathbf{x}}_n$, at the decoder output, and also the MSE for the training/validation phases ($MSE_{tr/val}$) are expressed as:

$$\begin{aligned} MSE_{(\mathbf{x}_n,\hat{\mathbf{x}}_n)} &= \frac{1}{N}\sum_{k=0}^{K-1}\left(x_{n,k} - \hat{x}_{n,k}\right)^2 \\ MSE_{tr/val} &= \frac{1}{n_{tr/val}}\sum_{n_{tr/val}} MSE_{(\mathbf{x}_n,\hat{\mathbf{x}}_n)} \end{aligned} \tag{2.28}$$

where n is the sample index of a single data point (either $\mathbf{x}_n$ or $\hat{\mathbf{x}}_n$), K is the number of samples in a data point, and $n_{tr/val}$ and $N_{tr/val}$ are the index and number of the training/validation data points, respectively.

The MSE function should decrease at each iteration. At the same time, after training the model at each iteration, the same should happen during the validation of the dataset, and the decoder and classifier outputs should show a similar decrease in the MSE values.

After feeding the training/validation dataset to the model, it classifies these data points and assigns the positive class label (+1) or the negative class label (−1) to the each of these data points. In an ideal case, the extracted labels by the model and the real labels of these data points must be the same, but in the real world, there is a difference between these two sets of labels. The binary cross-entropy (BCE) $H(y,p)$ between the distribution of the extracted labels $y(\mathbf{x}_n)$ and the distribution of the input data point $p(\mathbf{x}_n)$ from a dataset with size $N_{tr/val}$ is used to assess the accuracy of classifier output during the training process [88]:

$$\begin{aligned} H(y,p) &= \mathbb{E}_y\left[-\log p\right] \\ &= -\frac{1}{N_{tr/val}} \\ &\times \sum_{\mathbf{x}_n \in \{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_{N_{tr/val}-1}\}} \left[y(\mathbf{x}_n)\log\left(p(y(\mathbf{x}_n))\right) + (1-y(\mathbf{x}_n))\log\left(1-p(y(\mathbf{x}_n))\right)\right] \end{aligned} \tag{2.29}$$

Eq. (2.29) is called the binary cross-entropy (BCE) for the $y(\mathbf{x}_n) \in \{0,1\}$. If the values other than 0 and 1 are selected for the $y(\mathbf{x}_n)$, it may produce negative cross-entropy (BCE) values, but it will not essentially break down the convergence of loss function. The question arising here is that why do we need training of the decoder? Can we just focus on the classification training fed by the encoder output? The answer to this question is that the mean square error in Eq. (2.28) and the binary cross-entropy (BCE) in Eq. (2.29) should evolve simultaneously during training, to ensure that the accuracy of the decoder output improves in such a way that it provides meaningful features for the classifier at the output of the encoder and makes the label allocation more accurate.

Another aspect to consider is the decoder part required by the autoencoder model in training phase. After training, during testing, the classification procedure just needs the encoder and classifier parts, since the weights of the encoder and classifier are already set.

At the end of the training process, among all models, the one which has the minimum validation loss function value (BCE) of the classifier's output, is selected as the best model for testing purposes.

2.5 Conclusion

In this chapter, the theoretical concept behind the selected deep learning models such as autoencoder, RNN, LSTM, and the reason for selecting them for this thesis were explained. In addition, the training approach for the autoencoder and classification parts in the thesis are described in detail, clarifying the methodology for feeding the dataset to the model and training the weights. On the other hand, the updating approach through using the mean squared error (MSE) and binary cross-entropy (BCE) cost functions for model training in decoder and classifier parts are presented in detail, respectively.

Chapter 3

Dataset Generation for Authorized and Rogue Device Discrimination for Security Purposes in Wireless Communication Systems

3.1 Introduction

Designing a secure network which rejects the access of rogue devices and accepts authorized devices to enter the network begins with the capture of datasets from real devices. As discussed, the main focus in this thesis is on the ZigBee protocol (IEEE 802.15.4). For such a goal, a set of ZigBee devices are collected and stimulated to produce responses. Based on the IEEE 802.15.4, these responses includes a part called preamble which is later used in the model training for each device in Chapters 4 and 5.

To generate a model with high classification accuracy, different factors are important. These factors include a large enough high quality dataset, a model with enough complexity, carefully selected model parameters for training the model, and testing the trained network. In this chapter, we start with the first step, which is the dataset generation from ZigBee devices. As mentioned, feeding the model with a high quality dataset is of high importance.

However, because of different factors such as distortion and environment noise signals, the quality of the captured dataset from used devices decreases. On the other hand, the received signal from a ZigBee device includes the transmitted messages. In this chapter, first, the structure of a ZigBee signal is discussed in detail and the part which is appropriate for model training is assigned. Next, the detailed mechanism of dataset generation is presented, and the different challenges and their solutions are reviewed in detail.

The selected devices for data transmission, from Texas Instruments (2 devices), RZUSBstick (5 devices), and Digi XBee (1 device), are used in dataset generation. Signal capturing is done using a Zynq XC7Z020 FPGA.

3.1.1 Why do we need to explain the data acquisition procedure in detail?

As mentioned, data acquisition and dataset making is one of the most important steps in the selected methodology. In the environment, there are different negative factors such as distortion and environment noise which make the dataset acquisition a big challenge. On the other hand, even without these external resources, the received data from a ZigBee device suffers from some internal noise and distortions caused by the different factors such as the defects generated during the production, due to the environmental or other conditions such as temperature, humidity, etc. [95]. Some of these internal distortions are unique and helpful in distinguishing the different devices from each other, but some have negative effects such as delayed start of the preamble. Before doing the analysis, it is needed to clean the captured signal to remove the environmental noise and distortions, and also unwilling distortions caused by the device itself. Although the data capturing from ZigBee devices has been investigated in the literature (e.g. [96]) in recent years, however, it is a challenging procedure due to the diversity of the problems in this area, and in some cases, neither these problems (such as the distortions caused by the data transmission and reception) nor their solutions (e.g. extraction of the real starting point of a preamble or phase or frequency compensation) are presented in the literature, completely. In this chapter, all these possible problems and their solutions for the data acquisition and cleaning are addressed in full detail.

The structure of this chapter is as follows. In Section 3.2, a detailed description of the ZigBee burst (received signals from different devices which are going to be used for RF-DNA extraction [97]) structure is given. In Sections 3.3 and 3.4, the procedures of data transmission and reception are explained, respectively. Section 3.5 is devoted to preamble extraction, which is the basis of RF-DNA generation from different devices. Section 3.6 presents how convolution is used to find the beginning of a preamble. Finally, Section 3.7 concludes the chapter.

3.2 ZigBee burst structure

Fig. 3.1 shows a sequence of successive bursts using ZigBee protocol IEEE 802.15.4. To design a good model with high classification accuracy, it is important to feed it with a high quality dataset, with a fixed repetitive pattern. By fixed repetitive pattern we mean that the model must be fed with a dataset which is generated with the same pattern in real cases by

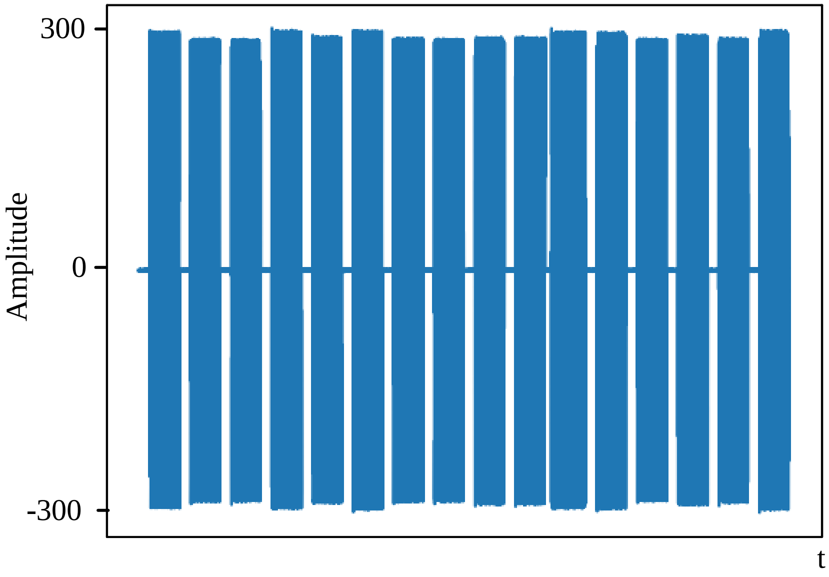
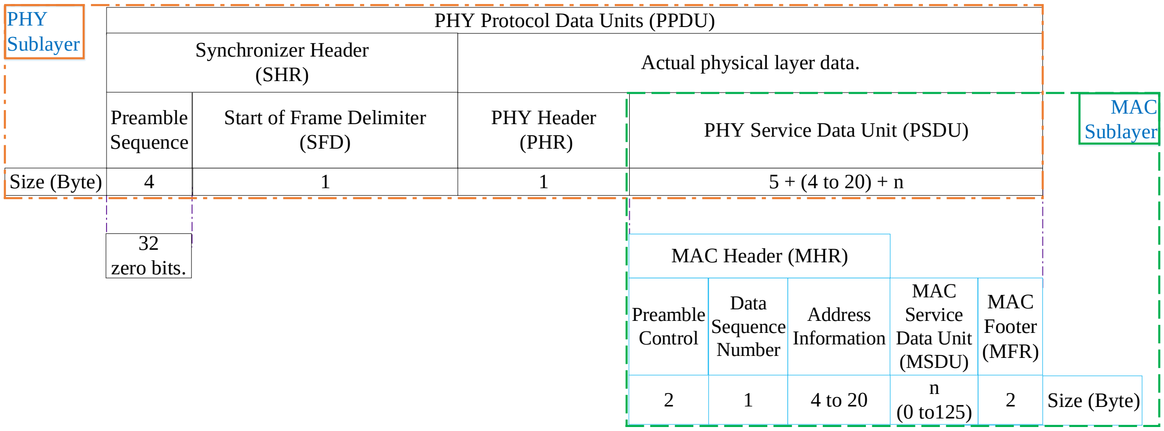


Figure 3.1: Sequence of successive bursts (ZigBee protocol IEEE 802.15.4).

real devices. Thus, it is important to detect the structure of the transmitted signal.

A ZigBee device generates and transmits a train of successive bursts, containing the transmitted information. Each burst includes a sequence of different kinds of headers and ends with a data part. The structure of a burst is shown in Fig. 3.2. As illustrated, each burst starts with a preamble. This preamble is an essential part for feature extraction from the ZigBee device, which will be used in Chapters 4 and 5 for feeding the presented models.



PHY Sublayer	PHY Protocol Data Units (PPDU)			
	Synchronizer Header (SHR)		Actual physical layer data.	
	Preamble Sequence	Start of Frame Delimiter (SFD)	PHY Header (PHR)	PHY Service Data Unit (PSDU)
Size (Byte)	4	1	1	5 + (4 to 20) + n
	32 zero bits.			

MAC Header (MHR)					MAC Sublayer
Preamble Control	Data Sequence Number	Address Information	MAC Service Data Unit (MSDU)	MAC Footer (MFR)	
2	1	4 to 20	n (0 to125)	2	Size (Byte)

Figure 3.2: Structure of a single burst.

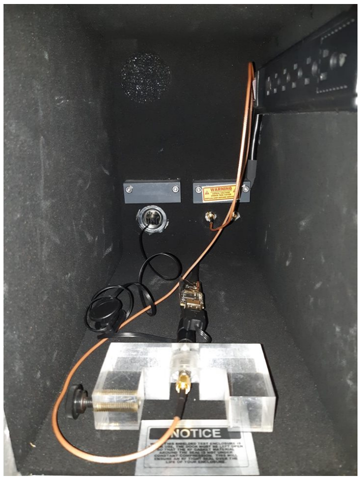


Figure 3.3: Device and antenna in the shielded box (anechoic chamber).

3.3 Data transmission

3.3.1 Empirical data transmission procedure

In this section, the procedure for data transmission is described. This procedure can be summarized as stimulating the device and capturing the response. However, it needs some special requisites, which are as follows:

1. Programming the device
   First, it should be noted that the transmitter device needs to be programmed. Therefore, the appropriate driver should be uploaded on the device.

2. Shielding the device
   Next is the data transmission procedure. In the case of data acquisition for dataset generation in model training, it is optimistic to think that no environmental signal contaminates the transmitted signal. Therefore, it is better to do the capturing in a shielded box. For this aim, the antenna and the device (Fig. 3.3) can be both put inside the shielded box, such as the anechoic chamber shown in Fig. 3.4. Obviously, shielding the device is a case that is far from a real life scenario, but it is nevertheless a good strategy for the purpose of this research.

3. Feeding the device
   In the next phase, there should be a host which feeds the selected data to the device to be transmitted. After connecting the device to the computer through a USB cable, a software should manage the transmission of bursts through the connected port of the computer to the device.

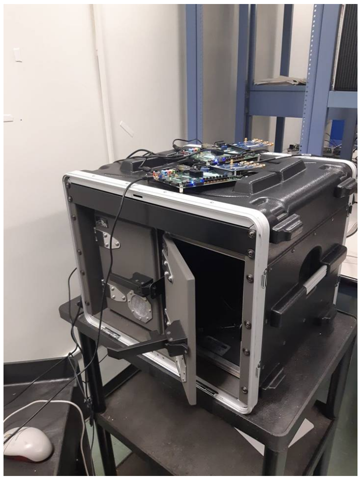


Figure 3.4: Shielded box.

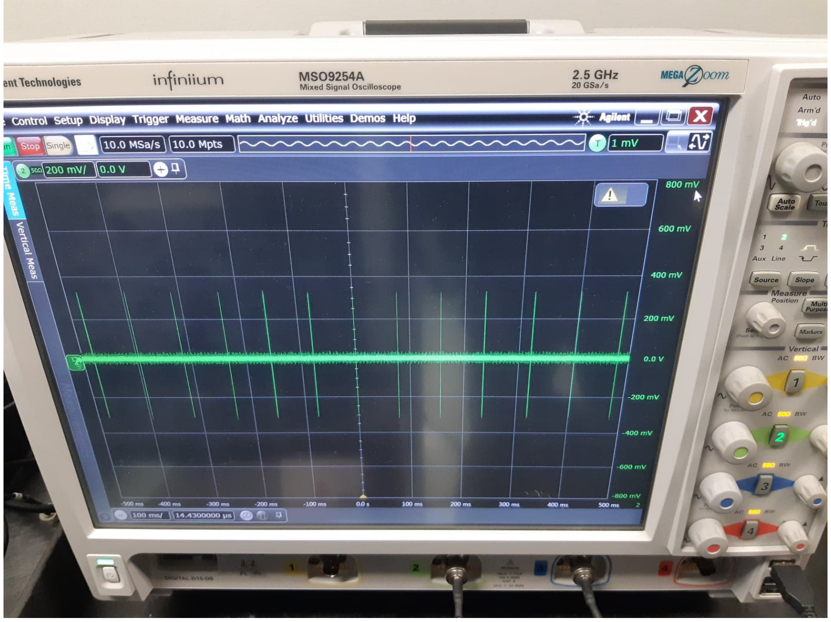


Figure 3.5: Real image of successive bursts from ZigBee device TI_1 captured from the oscilloscope.

4. Bursts measurements

   Now, the question that should be answered here is about the real form of the bursts. If the transmitted bursts of the device are shown on an oscilloscope, the data train should look like Fig. 3.5 (received signal from a Texas Instrument device). As can be seen, the train of the bursts is similar to Fig. 3.1.

   The successive bursts have almost equal amplitudes. In this experience the minimum acceptable amplitude contributing to the quality of the signal in the model training part is about 40 mV. Fig. 3.6 shows an enlarged part of each of the bursts depicted in Fig. 3.5. As can be seen, the amplitude for this burst is almost 300 mV which is higher

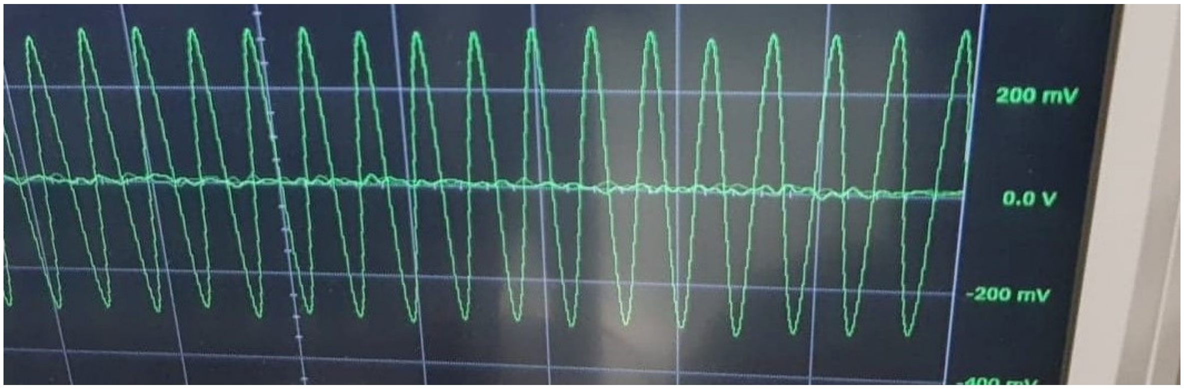


Figure 3.6: Enlarged bursts from the ZigBee device TI_1 with amplitude higher than 40 mV.

than the minimum acceptable value (40 mV). Below that minimum amplitude, it is possible to have problems in the feature extraction part.

3.3.2 Possible distortions caused by the transmission procedure

In some situations, because of different kinds of impairments which may have happened in the procedure of signal transmission, the received signal's quality may be altered. Data mining that can be used to mitigate these imperfections is needed. These damages can be enumerated in three different groups.

1. Distortions caused by the environment
   This group of errors is mainly caused by working devices in the environment. In this case, the received signal will suffer from serious distortions in the shape of the I and Q components. As an example, it is worth pointing at environmental noise as one of the most important causes. Shielding the receiver and the transmitter antennas is a good solution, but sometimes in real cases, this is not a feasible strategy.

2. Distortions caused by the transmission procedure
   As explained in Section 3.2 and depicted in Fig. 3.2, the structure of a burst in the IEEE 802.15.4 protocol is composed of different parts such as the preamble, the start of frame delimiter (SFD), the PHY header (PHR), and the PHY service data unit (PSDU). It may seem that each preamble starts with a well shaped successive pattern exactly at the beginning of each burst, but that is not always the case. The starting symbol of the preamble in the burst can get damaged for different reasons. The two most important cases are:

   a) Delayed data transmission
      In some cases, because of the delay in the performance of the operating system in

the host which feeds the transmission devices (discussed in part 3) of Section 3.3.1, the transmission of the preamble suffers from a delay in time. In this condition, the starting samples at the beginning may seem to be a part of the preamble, but they are not and the whole packet of the signal has a shift in time. If the duration of packet transmission by the device is fixed, it may happen that part of the data is lost from the end of the burst. On the other hand, if the amount of this shift is too large, then it is highly probable not only to lose the data but also to miss a part or the whole preamble.

The preamble extraction approach should be very efficient about finding the beginning of the preamble (discussed in Section 3.5.1). In Fig. 3.7, the starting index of the depicted preamble is about 30, which indicates the delay in the received signal.

b) Transmission of signal during the device transition state
The transmitter device has a transition time at the beginning of each burst and based on the device, the duration, type, and effect of this transition may differ. Then, a specific kind of distortion arises as a result of this transition to the shape of the signal emitted from the transmitter. As an example, for this kind of distortion, a received signal from the ZigBee device RZUSBSTICK 1 (RZ_1) is shown in Fig. 3.7. The most important thing about this distortion is that the shape deformation at the beginning of each preamble is related to the transition characteristics of the device, and it can be used as a unique signature in the procedure of feature extraction from the device for classification. At the beginning of the preamble, there is a part which is almost zero.
The extraction at the beginning of the preamble is discussed in more detail in Section 3.5.1.

3.4 Data reception

The procedure of receiving the transmitted signal in the receiver is not straightforward in practice. In other words, during data reception, various steps such as demodulation, preamble extraction, and data discrimination should be done. Meanwhile, in each if these steps, some different types of error may happen which need to be detected and compensated.

3.4.1 Data demodulation

As discussed in the previous section, when a device enters the network, it is stimulated to broadcast a special response. The received signal is modulated by a PN/chip sequence. This kind of response is received by the network and recorded for later computations and feature

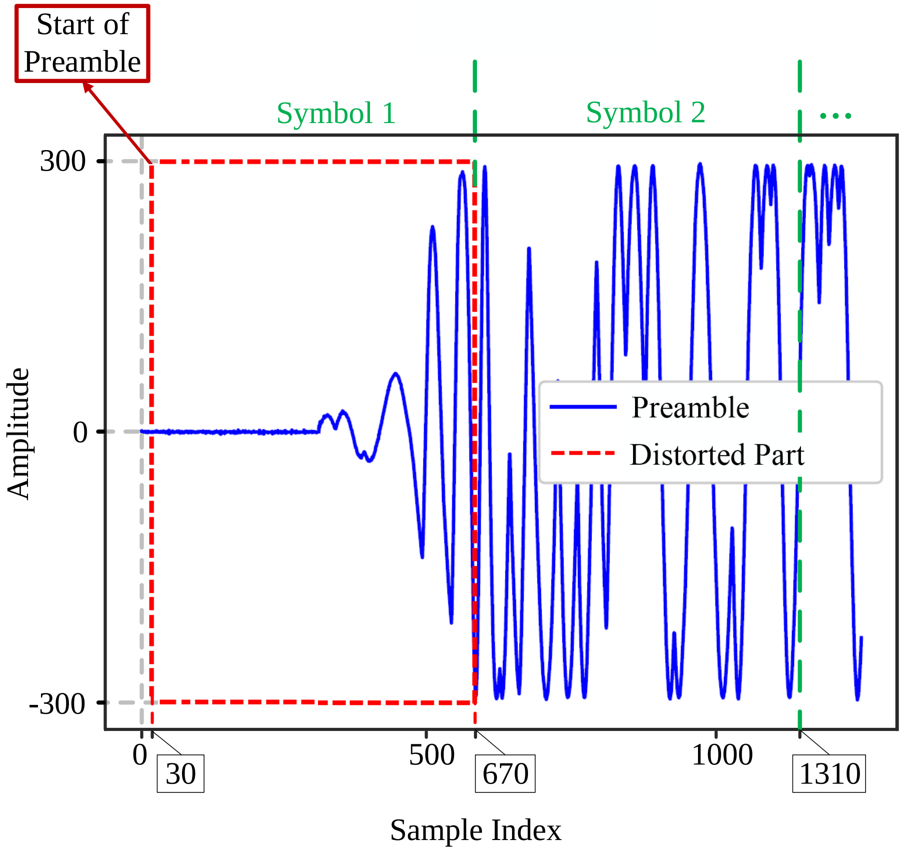


Figure 3.7: An example of data distortion at the beginning of the in-phase (I) component of the transmitted preamble, from device RZ_1.

extraction. Fig. 3.8 shows the signal of one burst received from device RZ_1. As seen, the signal is in a complex form and includes I and Q components. Also, the amplitude range of the received signal is $[-1860, 1860]$ for I, and $[-1914, 1914]$ for Q, which are different from the corresponding I and Q components in the transmitted one (in the range of $[-300, 300]$ for I and Q in Figs. 3.6 and 3.7). This difference is the result of applying the FPGA gain to the received signal, described in more detail in Section 3.4.2 and Part 5 of 3.4.3. The first step for making the data ready for feature extraction is demodulation.

3.4.2 Empirical data receiving procedure

The main element in the data receiving procedure is done in an FPGA module (Zynq XC7Z020). This device is shown in Fig. 3.9.

For using the FPGA as a receiver, some steps should be followed:

1. Programming the FPGA

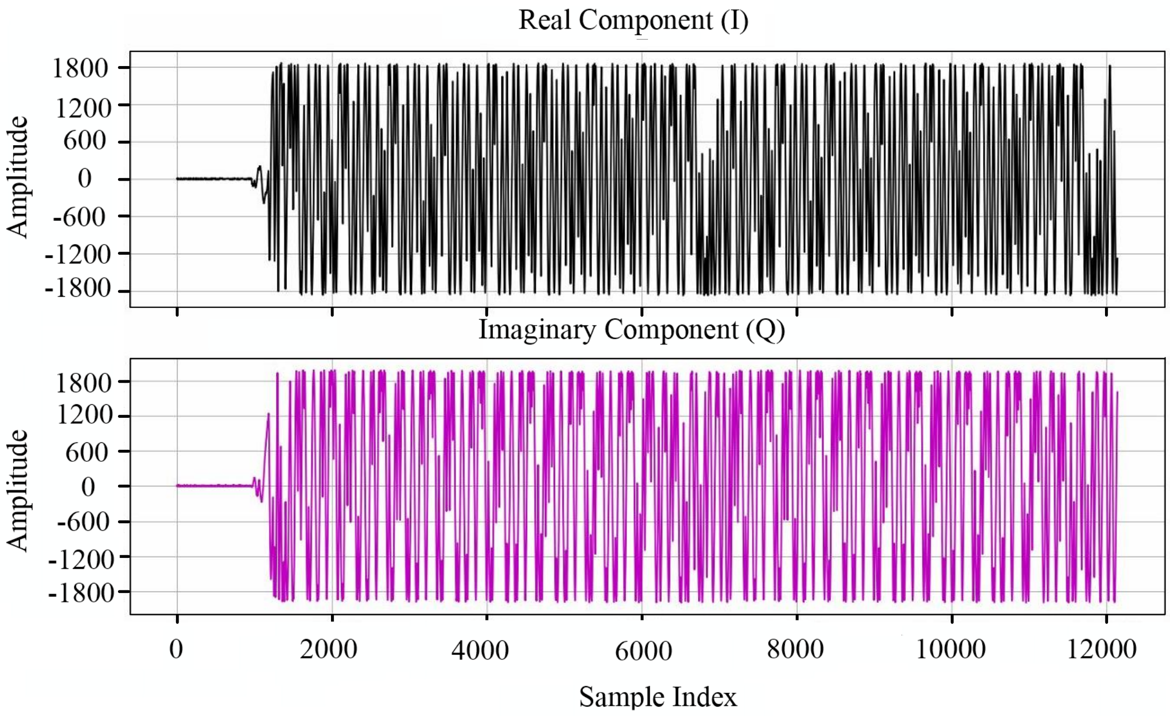


Figure 3.8: Real and imaginary parts of RZ_1 device signal.

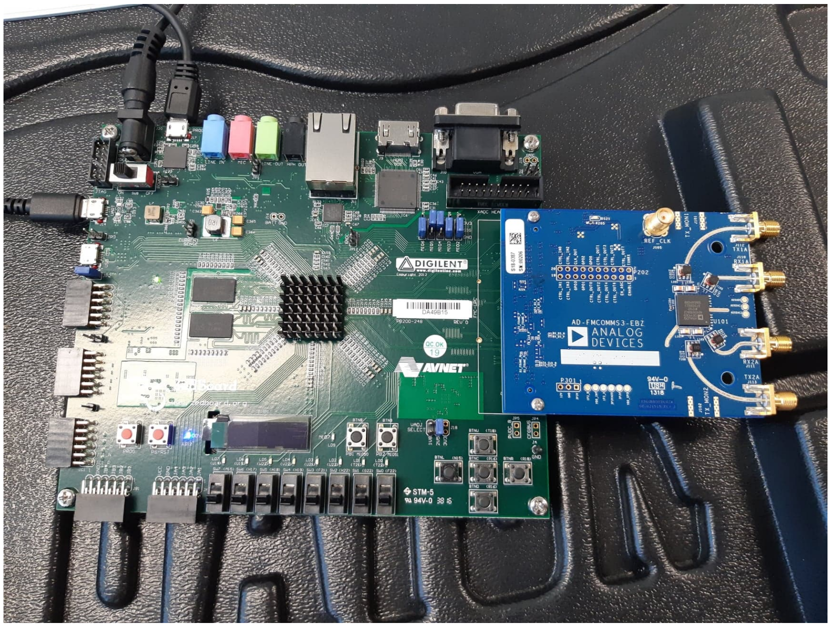
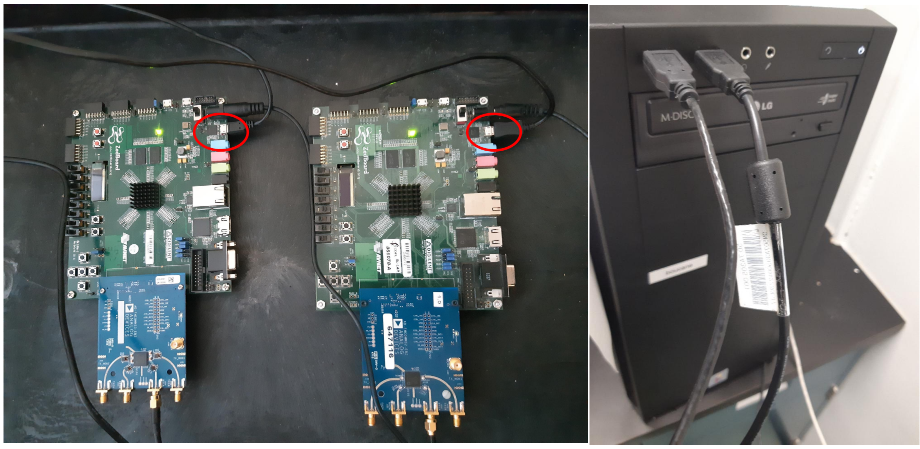


Figure 3.9: FPGA used for receiving bursts.

First of all the FPGA should be programmed. For this, the FPGA should be connected to the computer through a USB cable and the programming port, as shown in Fig. 3.10. Next, a program should manage the ports and memory allocated to the data acquisition procedure. The programming code can be written on the FPGA using a software development kit (SDK) program. Next, the programmed FPGA (confirmed by the blue light in Fig. 3.11), is ready to capture the transmitted signals from the



(a) (b)

Figure 3.10: Connection of the FPGA to the computer for programming through USB. The programming ports of the FPGA are illustrated with circles in (a).

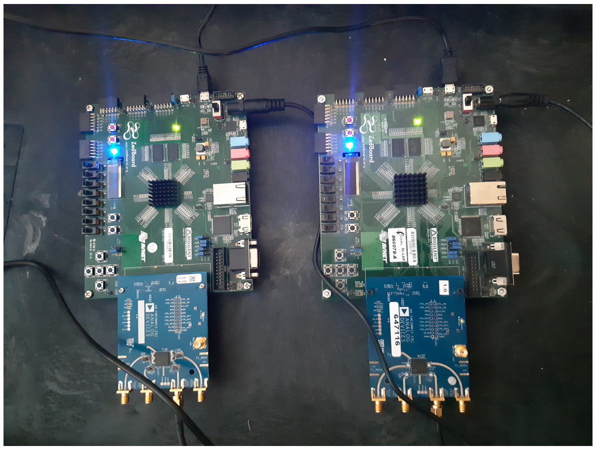


Figure 3.11: The blue light on the FPGA shows that it is programmed correctly.

device.

2. Down-conversion

   The generated base-band bursts, before transmission, are up-converted to mid-band in the modulation section. Then, the first step is to down-convert the bursts to the base-band frequency. Also, as mentioned, different devices (such as RZUSBSTICKs, Texas Instruments, XBEE Digi, etc.) focus on different channels defined in the IEEE 802.15.4 protocol. Therefore, during the down-conversion, this frequency difference should be taken into account.

   The last point is that in the case of down-conversion, since the clock of the demodulator is slightly different from the modulator upconversion frequency, this frequency difference can lead to a serious phase and frequency distortion of the received base-band burst. This case of distortion will be discussed in more detail in part 4 of Section 3.4.3.

3. Burst attenuation/amplification
   After down-conversion, the quality of the received bursts in terms of amplitude is verified using the SDK software. If the received burst is attenuated/amplified too much during the transmission procedure of the generated signals, it can be compensated through increasing/decreasing the FPGA gain (the gain which is applied to the received bursts by the FPGA). On the other hand, even though the amplitude of the transmitted signal has remained unchanged during the transmission, amplifying the signal using the FPGA is useful for feature extraction. However, it should be kept in mind that increasing the gain should be done carefully, since strong receiving gains on the FPGA can result in saturation of the bursts, which leads to damage in the received data. This effect of saturation is going to be introduced in Part 5 of Section 3.4.3 in more detail.

4. Saving the bursts
   After making sure that the quality of the received bursts is acceptable, they can be stored in memory for later use. The reception and storage of the bursts in the hard drive from corresponding ports are done using MATLAB/Python codes. But before running the MATLAB/Python programs, we have to make sure that no other program is connected to the programming port of the computer.

5. Burst acquisition using MATLAB/Python
   First, the MATLAB/Python program should be connected to the assigned communication port. Next, the code saves the bursts transmitted by the device one by one into the hard drive.

3.4.3 Possible distortions caused by the receiving procedure

1. Zero record capture
   As mentioned, during signal emission, a train of bursts is transmitted from the transmitter and received by the receiver. These trains of bursts are saved in separate files, during the collection of data from the devices. In some cases, because of the problems in the receiving procedure, a part of the transmitted signal is missed and the corresponding recording is just a flat zero signal.

2. Non-loadable records
   This is a rare phenomenon. In some cases, because of problems in the procedure of saving the recorded signal, despite the existence of data, the file is not loadable to the

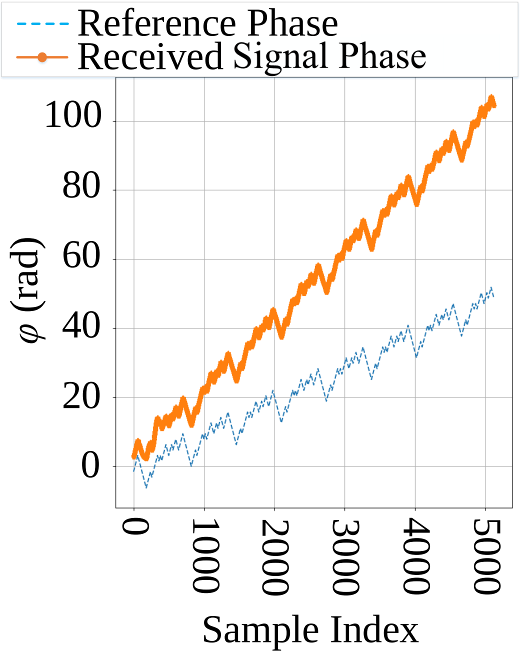


Figure 3.12: Example of phase shift in a preamble, from device RZ_1.

memory for feature extraction. The receiving program should be aware of these faulty recordings and eliminate them from the rest of the procedure.

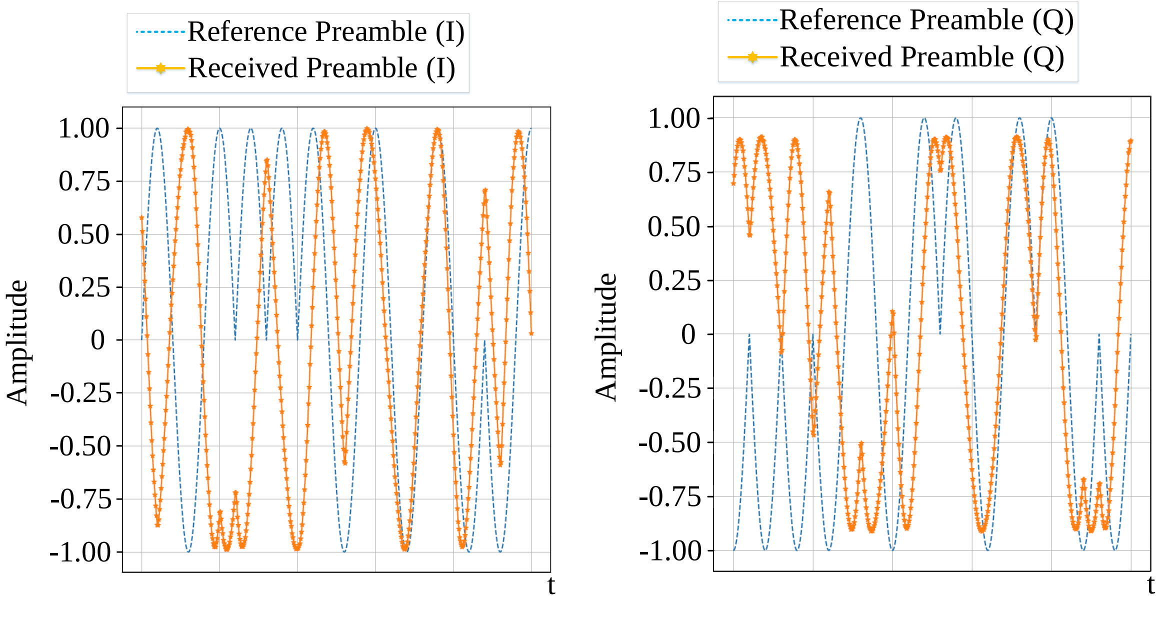
3. Empty recording

   In contrast to the previous cases, here the file does not contain any data, and the whole received burst in the recording is missed.

4. Phase and frequency mismatch

   In the case of a carrier frequency mismatch between the demodulator and the modulator, there will be a linear shift in the phase of the received preamble. In this case, the phase of received preambles has a shift which increases in time, linearly, as shown in Fig. 3.12. In this figure, the result of the phase shift on a burst from RZ_1 is illustrated. It is possible to fit a regression line to the phase of preambles and compare it with the phase of the reference preamble. As can be seen, there is a bias and a shift in the slopes of the two regression lines (representing the reference phase and the received signal phase). Through a compensation procedure, this slope should be eliminated from the phase of the preamble to make it as close as possible to the phase of the reference preamble. The mechanism of phase compensation is described in Section 3.5.2.

   On the other hand, the effect of this phase shift on the preamble should be the same as in Fig. 3.13. As can be seen, in some cases, the phase shift affecting the phase of the preamble is up to 180 degrees which inverses completely the related *PN* sequence. In this figure, the amplitude of the received signal is different from what is shown in Figs. 3.7 and 3.8, since the amplitude of the reference preamble is in the range of



(a)

(b)

Figure 3.13: Effect of phase shift deduced from modulation and demodulation frequency mismatch on enlarged (a) I and (b) Q components of a received preamble, from device RZ_1.

$[-1, 1]$. Therefore, the received preamble's amplitude is normalized to provide a good comparison in Fig. 3.13.

5. Burst saturation

   The amplitude of the bursts transmitted by different devices may differ from one manufacturer to another (e.g. the signals transmitted by Texas Instruments ZigBee devices are stronger than what is transmitted by RZUSBSTICK devices). It is probable that the appropriate FPGA gain for one device does not work for the others. If the used gain is too high, a specific distortion will be applied to the peak values of the received preamble's symbols, as shown in Fig. 3.14. The illustrated in-phase (I) and quadrature (Q) are measured from a burst of RZ_1 device. Decreasing the transmitter or FPGA gain can solve this problem.

   Meanwhile, focusing on Fig. 3.14 and comparing it with Fig. 3.8, the difference of amplitudes of both figures (from 1860 in the I and 1914 in the Q components, to almost 2000 for both components) is a clue to show the higher FPGA gain in the latter case, resulting in saturation of the received signals.

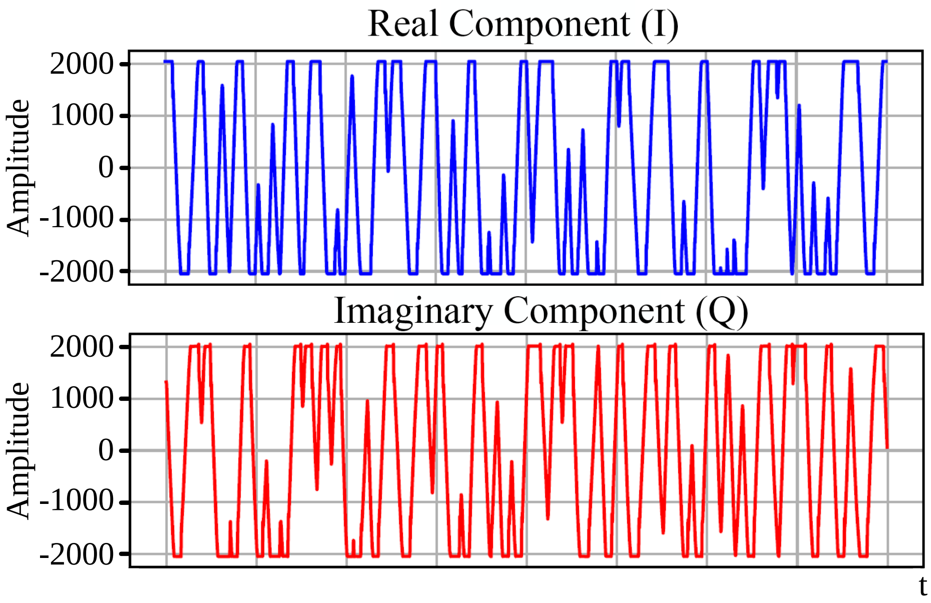


Figure 3.14: Burst saturation as an effect of extra FPGA gain for device RZ_1.

3.5 Preamble extraction

During the procedure of signal recording, if none of the errors mentioned above happens, this means that the transmitted signal has been correctly captured by the receiver. In such a case, the received bursts are ready for preamble extraction. However, if one or more of the mentioned imperfections decrease the quality of the received burst, taking into account the following steps is essential. As mentioned in part 2a of Section 3.3.2 and illustrated in Fig. 3.7, it is probable that the starting point of the burst (and consequently the preamble) in the received signal is shifted in time. First, the starting index of the preamble should be extracted and after that, one can see how much of the preamble is received correctly. According to this information, one can decide if a burst is legitimate (and therefore should be kept) or not (so, will be eliminated from the dataset).

1. Preamble starting index extraction
   For extraction of the beginning of a preamble, the mechanism which is used is as follows.

   a) Production of a single reference symbol and extracting its related phase
      As described in Section 3.2, at the beginning of each burst, theoretically there should be 8 successive symbols forming the preamble together (32 zero bits).
      Based on [96], the *PN* sequence for each zero symbol (0000) is:

      11011001110000110101001000101110

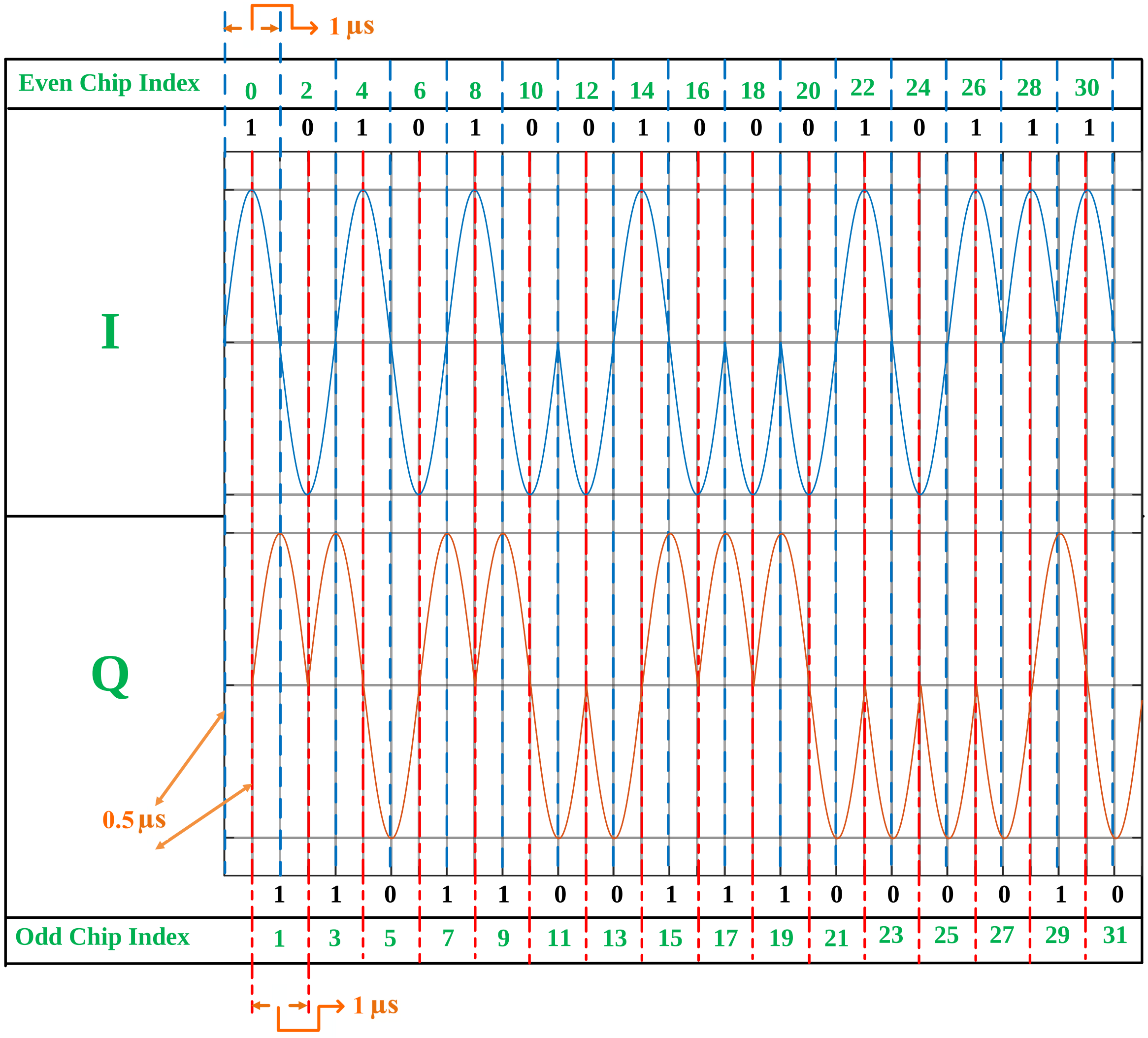


Figure 3.15: Modulated I and Q components corresponding to the zero symbol.

and the corresponding modulated I and Q components are shown in Fig. 3.15. This generated symbol can be used in determining the beginning of the preamble.

b) Finding the beginning of the preamble
After generating the reference symbol, the beginning of the received preamble is extracted from the burst by convolving its phase and the phase of the reference symbol. In other words, through sweeping the reference symbol's phase along the phase of the extracted burst and calculating the convolution, the points which are the real beginning of the symbols (since there are 8 successive symbols in each preamble) should lead to 8 equal maxima, located apart from each other with a distance of exactly one symbol. In such a case, the first peak can be used as a guide to find the starting index of the preamble. However, 2 points must be taken into account.

i. Effect of transition on successive maximum convolution peaks

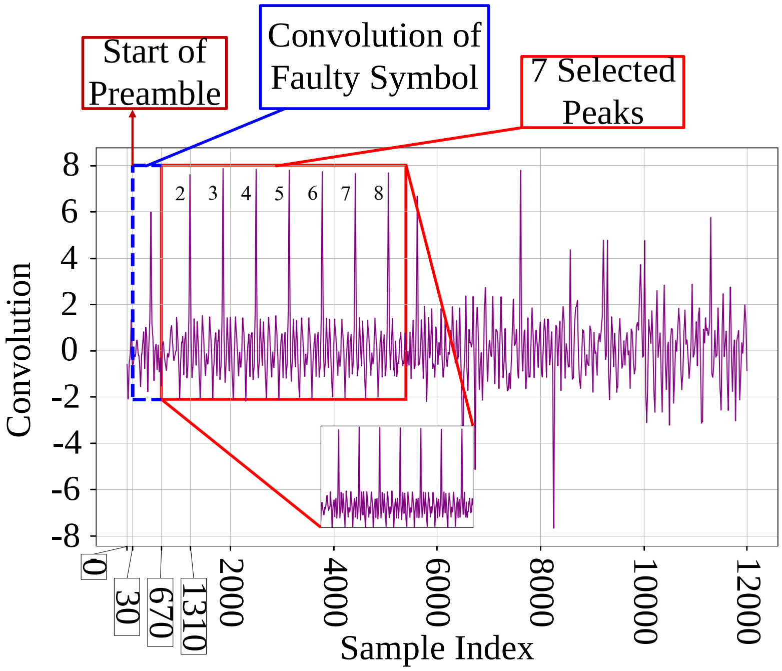


Figure 3.16: Convolution of the reference symbol and the received burst. The first peak in the convolution is distorted, compared to the other 7 remaining peaks. The region containing the first peak is indicated as a 'convolution of faulty symbol'.

As mentioned in Part 2b of Section 3.3.2, and illustrated in Fig. 3.7, in most of the cases, the first symbol is corrupted because of the transition response of the device. Then, in most of the cases it is better to find 7 successive equal maximum peaks instead of 8, and thus, the second peak will indicate the starting index of the second symbol. As an example, the output resulting from the convolution with 7 successive peaks is shown in Fig. 3.16, obtained from device RZ_1.

ii. Relationship between the maximum peaks and starting index of preambles
Before determining the most appropriate approach for finding the start of a preamble, it is better to consider the concept of convolution, as presented in Section 3.6. First, let us suppose that there are 8 good quality peaks in the convolution result. As described in Section 3.6, the time of occurrence of the first peak in the procedure of convolution calculation is always one window (in this work, equal to the length of one symbol) away from the actual starting moment of first symbol (and therefore the corresponding preamble). Also, 7 peaks are considered, thus the actual starting time of the preamble will be 2 symbols away from the occurrence of the second peak.
Keeping the above in mind, as shown in Fig. 3.16, if for example the second peak happens at index 1310, this means that the starting index of the first

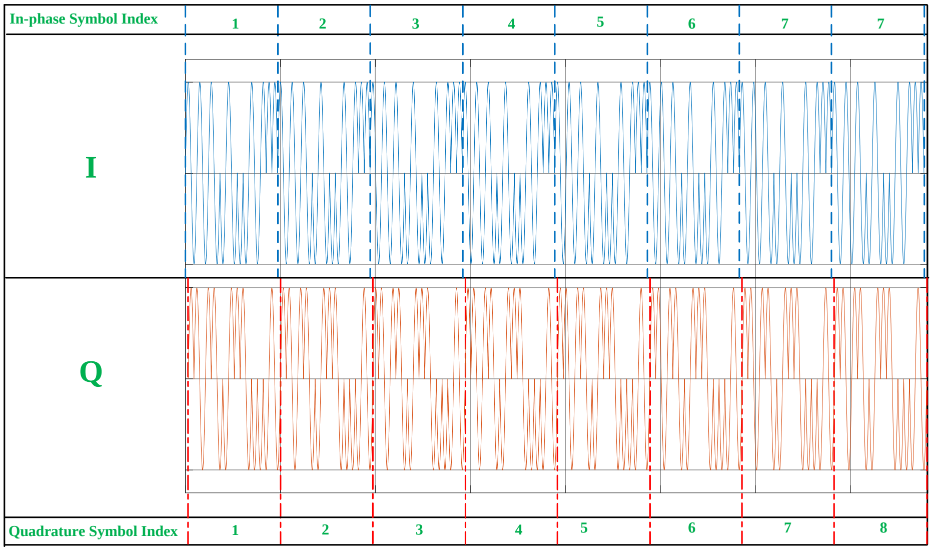


Figure 3.17: Modulated I and Q components of the reference preamble.

symbol is:

$$\begin{aligned}\text{Symbol length} &= 640\\ \text{First symbol starting index} &= 1310 - 2 \times 640\\ &= 30\end{aligned}$$

If the focus is on 8 successive bursts, and the first peak happens at index 670, then the start of the preamble is just a symbol away from the first convolution peak.

$$\text{Preamble starting index} = 670 - 640 = 30$$

Now, it is possible to extract the preamble. But, as mentioned in part 4 of Section 3.4.3, because of the mismatch in the down-converting frequency of the demodulator and in the up-converting frequency of the modulator, there will be a frequency and a phase shift, as shown in Fig. 3.12, which needs to be compensated.

2. Phase and frequency compensation

   Each preamble contains a repetitive pattern of one symbol. A preamble is made up of 8 successive modulated I and Q zero symbols. Then, based on what was discussed in Part 1a of Section 3.5 and shown in Fig. 3.15, the reference preamble is as depicted in Fig. 3.17.

   Fig. 3.18 shows the real and imaginary components, phase, and constellation diagrams of a received preamble from ZigBee device RZ_1. Comparing this figure with the I, Q,

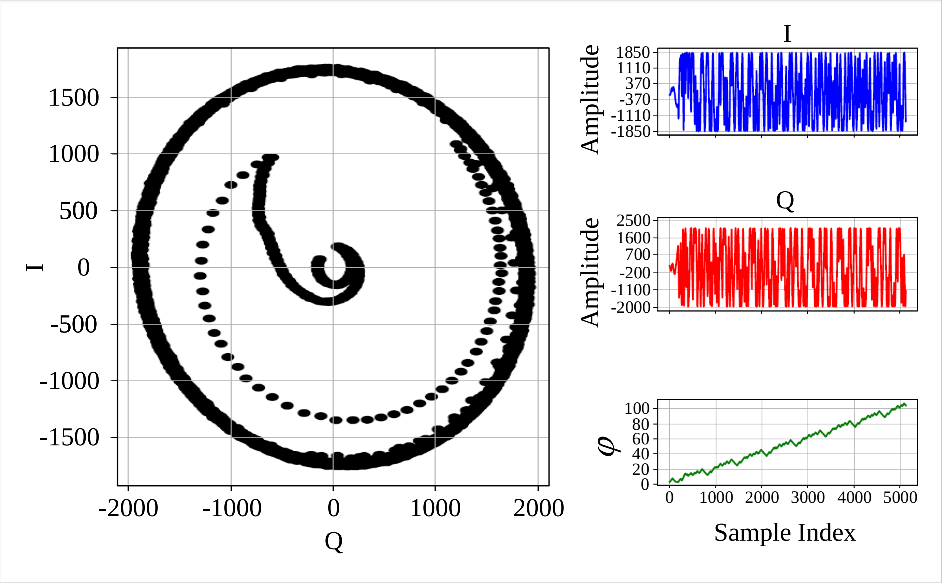


Figure 3.18: Received preamble I and Q components, phase, and constellation, from device RZ_1.

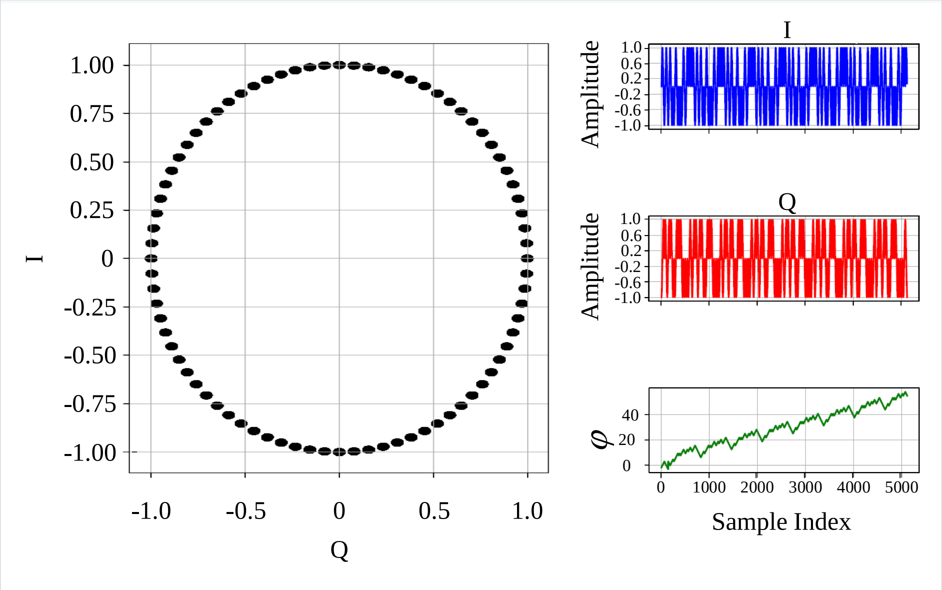


Figure 3.19: Reference preamble I, and Q components, phase, and constellation.

phase, and constellation of a reference preamble, as shown in Fig. 3.18, a reference symbol (and its corresponding preamble) covers all 2π radians of the unit polar coordinate circle. This phase is the reference for all computations in the phase compensation.

As discussed in Part 4 of Section 3.4.3, working with the initial captured signals, because of differences in the modulation and demodulation frequencies, there will be a shift in the slope of the captured signal phase compared with the reference preamble's, as shown in Fig. 3.20. The procedure of phase compensation consists of reducing the slope difference of two graphs. Consequently, the mechanism is as follows.

a) First a reference symbol is generated as shown in Fig. 3.15.

b) Based on the fact mentioned in Part 1a of Section 3.5, repeating the symbol makes a reference preamble, as shown in Fig. 3.17.

c) Calculation of the phase of the extracted preamble from the burst and also the generated reference preamble. The phase of a complex number (C) is:

$$\varphi = \arctan\left(\frac{\Im(C)}{\Re(C)}\right) \tag{3.1}$$

where $\Re(C)$ and $\Im(C)$ are the real and imaginary values of C, respectively.
But as discussed in Part 1a of Section 3.5 and shown in Fig. 3.17, each modulated preamble signal is the result of repeating a zero symbol 8 times. On the other hand, as shown in Fig. 3.19, each symbol covers all 2π radians of the complex plane. Thus, repeating a symbol in a preamble results in the addition of 2π radians to the phases of the repeated symbols in the preamble. Therefore, the calculation of data points in a preamble obey the rule in the following equation [98].

$$\begin{aligned}\varphi_n &= \left\lfloor \frac{\varphi_{n-1} - \varphi_n^*}{2\pi} + .5 \right\rfloor \times 2\pi + \varphi_n^* \\ \varphi_n^* &= \arctan\left(\frac{\Im(pr_n)}{\Re(pr_n)}\right) \\ \varphi_0 &= \varphi_0^*\end{aligned} \tag{3.2}$$

where pr_n is the n^{th} sample from the received preamble, for $n = 0, 1, \cdots, N-1$ and N is the preamble length.

d) Calculation of the error using the following equation, as shown in Fig. 3.20:

$$\varphi_{err} = \varphi(ref_{pr}) - \varphi(rec_{pr}) \tag{3.3}$$

where $\varphi(ref_{pr})$ and $\varphi(rec_{pr})$ are the phase of the reference and the received (non-compensated) preambles, respectively.

e) As seen in Fig. 3.20, the error is almost linear, after about 500 measured samples. Therefore, by fitting a first degree polynomial to the error through the extraction of the linear regression parameters a and b in the following equation, the linear phase difference needed for the correction of phase in each data point can be calculated as follows:

$$\begin{aligned}\Delta\varphi_{linear} &= a \times n + b \\ a &= \frac{\varphi_{err_2} - \varphi_{err_1}}{N}\end{aligned} \tag{3.4}$$

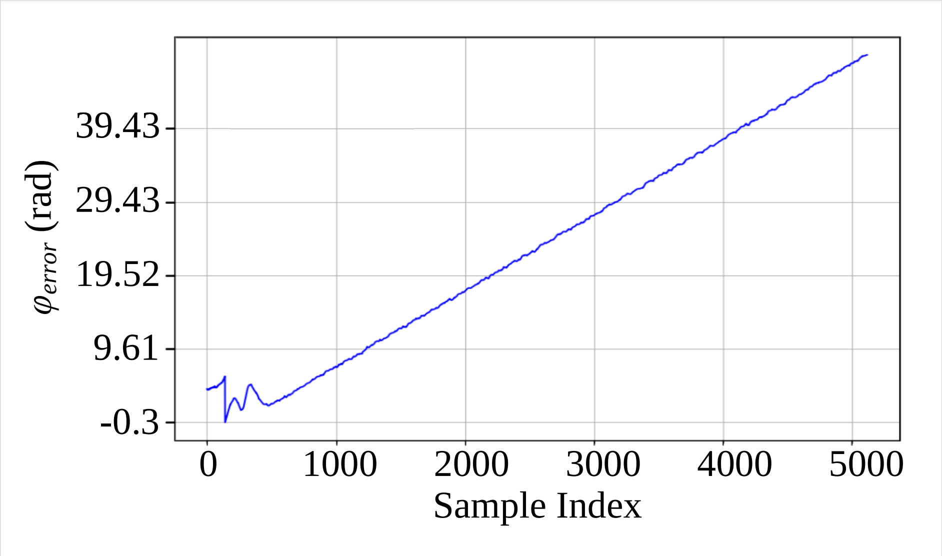


Figure 3.20: Phase error between the reference and the received preambles of device RZ$_1$, using Eq. 3.3.

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

$$\varphi_{corr} = \varphi(rec_{pr}) - \Delta\varphi_{linear} \tag{3.5}$$

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

f) Using Eq. 3.5, the compensated I and Q components of the preamble can be obtained using the following equation:

$$\begin{aligned} I_{comp} &= A(\cos(\varphi_{corr})) \\ Q_{comp} &= A(\sin(\varphi_{corr})) \end{aligned} \tag{3.6}$$

where A is the amplitude of received preamble:

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

After the phase compensation procedure, the preamble is extracted, phase compensated, and ready for feature extraction and signal analysis. An example of the compensated phase of a received preamble of ZigBee device RZ$_1$ is presented in Fig. 3.21. As shown, the comparison of this phase with the reference preamble's results in a good match compared to the non-compensated preamble's phase. Also, the corresponding preamble for each of these three mentioned cases are shown in Fig. 3.22. As can be seen, the compensated I and Q components of the received preamble match very well with the corresponding components of the reference preamble. It is worth mentioning that like in Fig. 3.13, the amplitude of the I and Q components in the received and compensated preambles are normalized in Fig. 3.22 to facilitate the comparison.

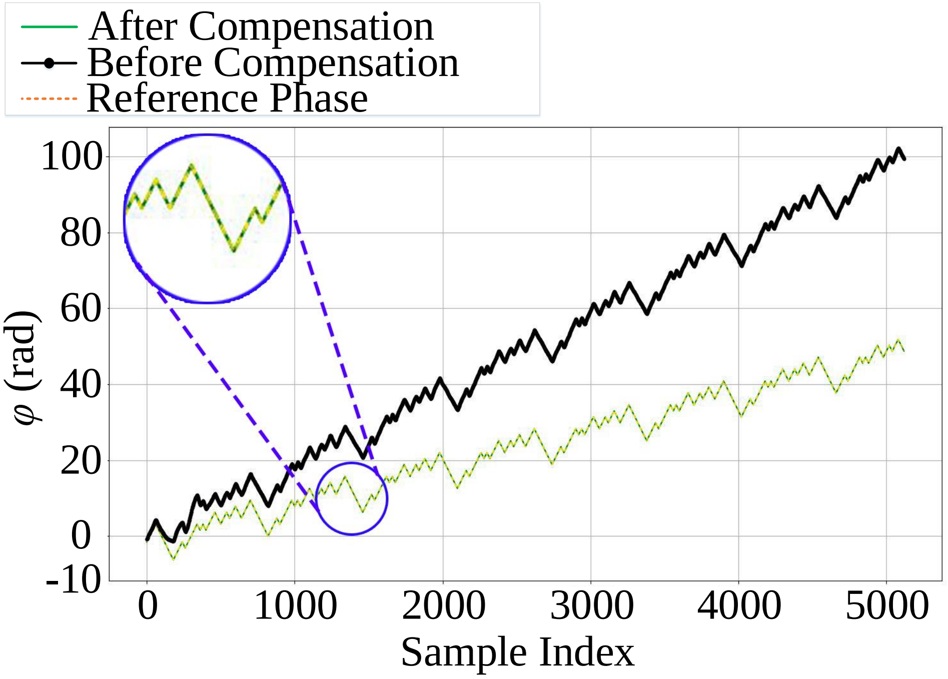


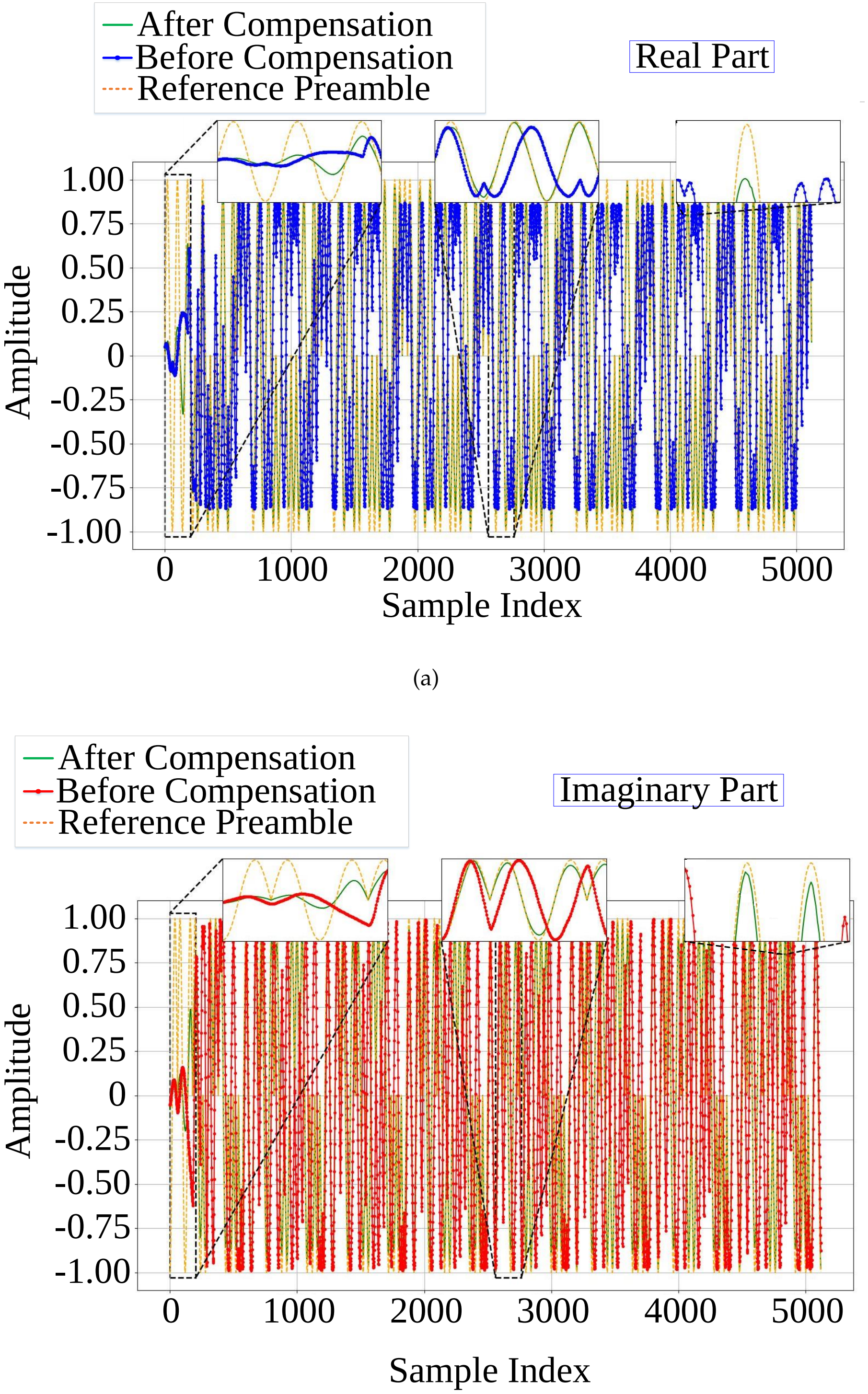
Figure 3.21: Phases of the received, reference, and compensated preambles of device RZ_1.

3.6 Determination of the beginning of a preamble using a convolution

Consider the problem of finding the beginning of a preamble through convolving the phases of a symbol and the whole burst. As explained earlier, the beginning of the preamble has a shift compared to the beginning of the received signal. Therefore, through convolving the phases of a symbol and the whole burst, the beginning of the preamble is achievable. After finding the beginning of a preamble, knowing its length, we are able to extract it for the processing purposes. It is worth mentioning that the real beginning of the preamble is one symbol before the moment shown by the convolution results.

3.7 Conclusion

In this chapter, the mechanism of dataset acquisition and cleaning was described in detail. As mentioned earlier, designing an effective device discrimination model starts from the generation of a large enough high quality clean dataset. In the environment, different distortion factors such as noise, wireless signal, Wi-Fi access points, and other resources interference decrease the quality of the captured signal. After cleaning the dataset, the preamble part of the ZigBee signal based on IEEE protocol 802.15.4 must be extracted for training, validation,



(a)

(b)

Figure 3.22: Received, reference and compensated preambles of ZigBee device RZ$_1$.

and testing of the model. After these two steps, the dataset is ready for device discrimination purposes. These two steps were described in detail in this chapter, and the extracted dataset is ready to be used in the next chapters.

Designing a secure network which rejects the access of rogue devices and accepts the authorized devices to enter the network begins by capturing a dataset from real devices. As discussed, the main focus in this thesis is on the ZigBee protocol (IEEE 802.15.4). For such a goal, a set of ZigBee devices are collected and stimulated to produce a response. Based on the IEEE 802.15.4, this response includes a part called preamble which is later used in the model training for each device in chapters 4 and 5.

Chapter 4

Rogue Device Discrimination in ZigBee Networks Using Wavelet Transform and Autoencoders

4.1 Introduction

In recent decades, the development of wireless communication systems and networks has led to the use of portable devices anytime and anywhere. Consequently, different security protocols such as Wi-Fi Protected Access (WPA), WPA2, and WPA3 have provided a higher degree of security for short or long range radio communication systems over the last years [7, 8]. One of these communication protocols, ZigBee, introduced in 1999 [99], is considered as an attractive wireless system for commercial and military applications, because of its low cost and low complexity [97]. Despite the advantages in security protocols and systems in the last decade, the 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 [9].

Physical layer attacks include a vast range of attack types, typically using very sophisticated methods and devices to extract the information from the network by falsifying the identity of the main members (legitimate nodes), during normal data transmission. Based on this, the longer a communication system transmits information, the higher the risk for information leakage or unauthorized access of rogue devices.

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) [100], which are inherently unique for a given device [101]. In this thesis, these RF-DNA features are analyzed and processed for the discrimination and rejection of spoofing devices. The methodology presented in this chapter is published in [10].

The structure of this chapter is as follows. First, Section 4.2 introduces the methodology adopted in our work for security classification purpose. In Section 4.3, the outcome of the proposed method on real data is explained. Section 4.4 presents the comparison of the proposed methodology in this chapter with some results presented in the literature. Finally, Section 4.5 summarizes different findings of this chapter.

4.2 Proposed classification method

The main objective of this chapter is to propose a binomial (or binary) classification system which provides a mechanism for device discrimination into two classes: legitimate devices and (unauthorized) rogue devices. The purpose of this binomial classification is the ability to reject the access of rogue devices to a ZigBee network. 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. Assuming that a model is made for a specific device, device$_m$, when any new device enters the network, this model tries to detect if the anonymous device is an authorized device, i.e. device$_m$, which will be granted access to the network. If not, it will then be rejected by the network.

4.2.1 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 11 channels from 2.4 GHz to 2.48355 GHz, each with a 2 MHz bandwidth [102]. Different manufacturers set the central carrier frequency of their ZigBee devices 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.

The structure of the bursts, following the IEEE 802.15.4 protocol, was shown in Fig. 3.2. A burst begins with a known preamble [102], followed by 8 successive modulated zero symbols, consisting of I and Q components as shown in Fig. 3.18. Each preamble contains a repetitive pattern of a single symbol. The duration of each symbol is 16 µs, and thus the length of the whole preamble is:

$$\text{preamble length} = 8 \times 16\ \mu\text{s} = 128\ \mu\text{s} \tag{4.1}$$

4.2.2 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 in Fig. 3.17 is convolved with the received signal. Thus, 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 the preamble, and knowing its length, extraction of the preamble itself is easy.

4.2.3 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 captured and the original preambles: phase compensation consists of reducing the slope difference between these two. The dataset phase and frequency compensation is described in detail in Section 3.5.2.

An example of a phase compensated signal is presented in Section 4.3.2. Once phase compensation is done, the extracted preambles can be processed for feature extraction, signal analysis, etc.

4.2.4 Dataset transformation

In this chapter, the discrete wavelet transform is investigated as a means to improve the discrimination process between 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 coefficient of the received ZigBee vector signal, $\mathbf{x}_n = [x_{n,1}, x_{n,2}, \cdots, x_{n,k'}, \cdots, x_{n,K}]^\mathsf{T}$ and ($1 \leq k' \leq K$ with K samples), is given by [103, 104]:

$$c_{j,k} = \frac{1}{\sqrt{2^j}} \sum_{k'=-\infty}^{\infty} x_{n,k'} \psi^* \left[\frac{k' - k2^j}{2^j} \right] = \frac{1}{\sqrt{2^j}} \sum_{k'=1}^{K-1} x_{n,k'} \psi^* \left[\frac{k' - k2^j}{2^j} \right] \tag{4.2}$$

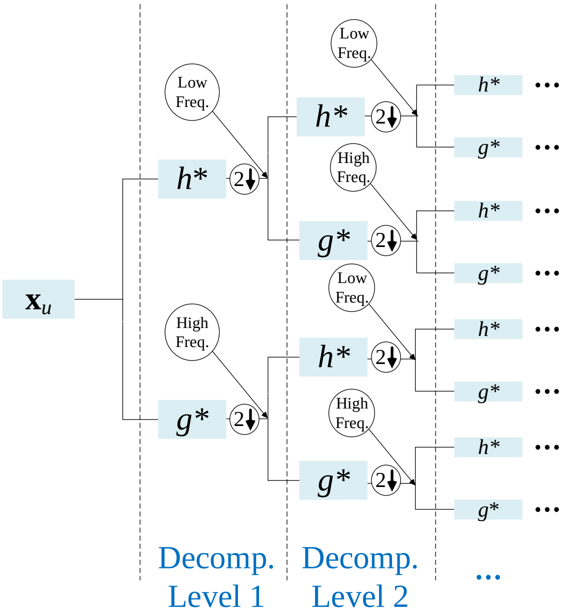


Figure 4.1: Dyadic discrete wavelet transform decomposition diagram.

where $j = 0, 1, 2, ...$ and $k' = 1, 2, ..., K$, and $\psi^*[k']$ is the conjugate of the wavelet window, $\psi[k']$. Changing j (called the dyadic scale factor) results in changing the width of $\psi[k']$. The reason for calling this scale coefficient a dyadic factor is that the width of the wavelet window $\psi[k']$ changes with this variable, dyadically (as a power of 2). The result of the calculation of $c_{j,k}$ for different values of j is referred to as wavelet decomposition. The decomposition level of the wavelet coefficients is determined by the wavelet parameter j. As the size of the wavelet window changes, the number of features extracted will change, too, as shown in Fig. 4.1.

Let us have a detailed look at Fig. 4.1. In this figure, the diagram is divided into different parts, called decomposition levels. These are the same levels assigned by the j factor in Eq. (4.2). In each decomposition level, two different types of filters can be seen, a low-pass filter, $h[k']$, and a high-pass filter, $g[k']$. Each of these two filters can be used as $\psi[k']$ in Eq. (4.2). Using the low-pass filter $h[k']$, the calculated coefficients $c_{j,k}$ are called low-frequency or approximation coefficients $a_{j,k}$. On the other hand, replacing $\psi[k']$ by $g[k']$ results in the extraction of the detail coefficients $d_{j,k}$. Eq. (4.3) [103,104] presents the same rule, in a clearer way.

$$\begin{aligned}
a_{j,k} &= \frac{1}{\sqrt{2^j}} \sum_{k'=-\infty}^{\infty} x_{n,k'} h^* \left[\frac{n - k2^j}{2^j}\right] = \frac{1}{\sqrt{2^j}} \sum_{k'=0}^{N-1} x_{n,k'} h^* \left[\frac{n - k2^j}{2^j}\right] \\
d_{j,k} &= \frac{1}{\sqrt{2^j}} \sum_{k'=-\infty}^{\infty} x_{n,k'} g^* \left[\frac{k' - k2^j}{2^j}\right] = \frac{1}{\sqrt{2^j}} \sum_{k'=0}^{N-1} x_{n,k'} g^* \left[\frac{k' - k2^j}{2^j}\right]
\end{aligned} \tag{4.3}$$

The selected wavelet window for this chapter is the Haar wavelet. The low-pass and high-pass filters for the Haar wavelet are as presented in Eq. (4.4), and as shown in Fig 4.2 [105].

$$
\begin{aligned}
h[k] &= \begin{cases} 1, & \text{for } k=0,1.0 \\ 0, & \text{otherwise} \end{cases} \\
g[k] &= \begin{cases} 1, & \text{for } k=0 \\ -1, & \text{for } k=1.0 \\ 0, & \text{otherwise} \end{cases}
\end{aligned}
\tag{4.4}
$$

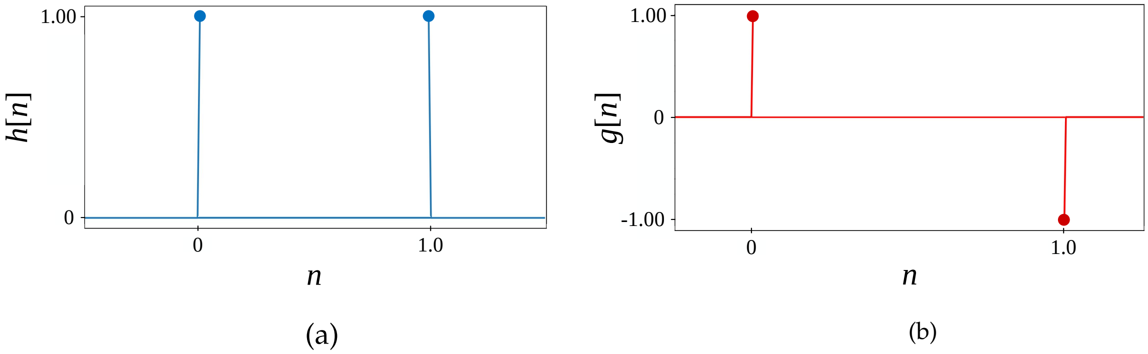


Figure 4.2: (a) low-frequency and (b) high-frequency Haar wavelet windows [105].

All 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 4.2.1, 4.2.2, and 4.2.3) will be referred to as the *RAW dataset* in the remaining of the chapter.

4.2.5 Model definition

As described in Chapter 2, the structure used for feature extraction in this chapter is the autoencoder. After feature extraction by the autoencoder, the extracted features are fed to a classifier for device discrimination as an authorized/rogue device. The whole procedure of the definition of the model used in this chapter (depicted in Fig. 2.1) is described in detail in Section 2.2.

4.2.6 Dataset subdivision into training, validation, and testing datasets

The classifier model selection requires three tasks: a training phase, a validation phase, and a testing phase [106]. During the training and validation phases, a new model is generated for the discrimination of one specific device from the other devices. Then a testing phase with at least a new device is done to assess the reliability of the classifier model. Before training the model, the dataset is thus subdivided into three different datasets: a training dataset, a validation dataset, and a testing dataset. The step-by-step dataset subdivision procedure is as follows.

Selection of positive and negative devices As mentioned before, the classification strategy adopted for this work 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 devices 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.

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. Training
   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 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 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 this device should be kept in the training set. Then, after allocation of the devices for training, validation, and testing, the

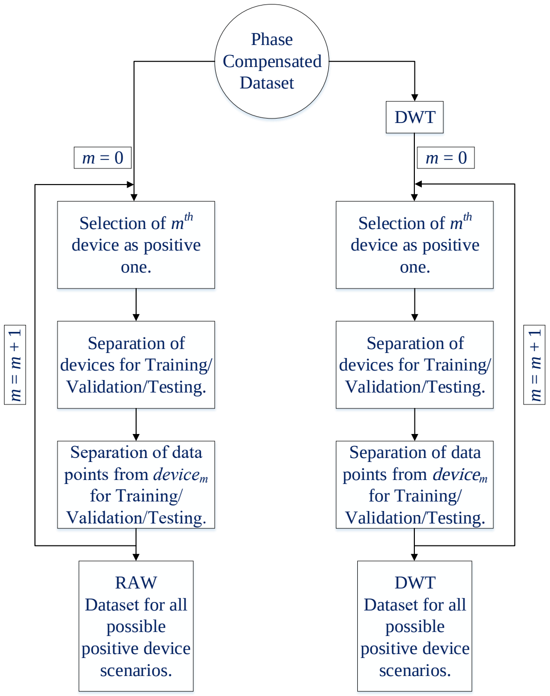

Phase Compensated Dataset
DWT
m = 0
m = 0
Selection of m^th device as positive one.
Selection of m^th device as positive one.
m = m + 1
m = m + 1
Separation of devices for Training/ Validation/Testing.
Separation of devices for Training/ Validation/Testing.
Separation of data points from device_m for Training/ Validation/Testing.
Separation of data points from device_m for Training/ Validation/Testing.
RAW Dataset for all possible positive device scenarios.
DWT Dataset for all possible positive device scenarios.

Figure 4.3: Training, validation, and testing dataset generation.

data points themselves are assigned for each of them. Therefore, almost 60% of the data points from training devices are used in training. Next, 10% of the validation devices are allocated to it, and finally, the remaining data points from all devices should be used as testing points. In this work, none of the devices used in training, validation, and testing have data points in common: each data point from each device is used only in one of these steps. In other words, the number of training (N_{tr}), validation (N_{val}), and testing (N_{te}) data points are as follows.

$$\begin{aligned} N_{tr} &= 0.6N \\ N_{val} &= 0.1N \\ N_{te} &= 0.3N \end{aligned} \tag{4.5}$$

Fig. 4.3 illustrates the block diagram for the data points separation. As shown, the dataset generation procedure can be applied to either the RAW or the DDWT datasets. It is worth mentioning that in this figure, the goal is to make a dataset for each device through stratified sampling. In such a case, although the sampling is done randomly, it can be assured that the model is not going to be biased during the training.

4.2.7 Model training and validation procedure

As mentioned in Chapter 2, the training procedure of the model includes feeding the dataset to the model, minimizing the distance between the input data points ($\mathbf{x}_n$) and reconstructed ones ($\hat{\mathbf{x}}_n$), and training of the classifier output. The whole procedure of training of the autoencoder and classifier is described in Section 2.4.

4.2.8 Model testing

The last step in the procedure of device discrimination is testing. After designing the model, it is very important to test the model performance to investigate the probability of overfitting or underfitting. Such a purpose makes us to use the *Confusion Matrix* (CM) and the *Receiver Operating Characteristics* (ROC) curves, as fully explained in Appendix A.

4.3 Experimental results

In this section, we validate the proposed methodology for dataset generation, model training, validation, and testing, in real world cases. For this purpose, after extraction, the dataset is fed to a deep learning model, and after the training section, the output performance of the model is evaluated through the different machine learning approaches. This procedure is described in more detail in the following.

4.3.1 Experimental equipment setup

Figs. 3.3, 3.4, 3.9 and 3.10 show the laboratory setup designed for the signal measurements and dataset acquisition. Also, the selected devices for data acquisition are shown in Fig. 4.4. 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 XBEE Digi module (AR_1), and two (2) Texas Instruments devices (TI_1 and TI_2).

4.3.2 Preprocessing of acquired signals

Preambles extraction from the received signal bursts

As explained in Section 4.2.2, in practical situations the beginning of the preamble may not be located exactly at the beginning of a burst. Therefore, the first step in processing the sampled

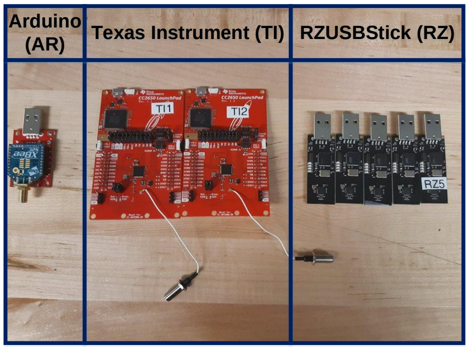


Figure 4.4: Selected devices for data acquisition.

signals is to determine the beginning of the preamble by convolving the received signal with a single known symbol, that is for each of 8 successive reference O-QPSK symbols.

After determining the starting point, the preamble can be extracted from the received signal. As mentioned in Section 4.2.1, the length of a preamble is 128 µs, 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.

Phase and frequency compensation

As discussed in Section 4.2.3, 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), (3.5) and (3.6). Fig. 4.5 depicts the extracted preamble phases of devices AR_1 before and after phase and frequency compensation. The effect of phase compensation on the received signals themselves is illustrated in Fig. 4.6 for the same three devices. As shown, the comparison of the received signals with the reference preamble signals demonstrates the efficiency of the phase and frequency compensation approach. The compensated signal phase (green line) is superimposed to the reference phase (dotted red line).

4.3.3 Datasets processing

DDWT dataset generation

After preprocessing the data, with respect to Sections 4.2.2 and 4.2.3, the DDWT is applied to the resulting data points as described in Section 4.2.4.

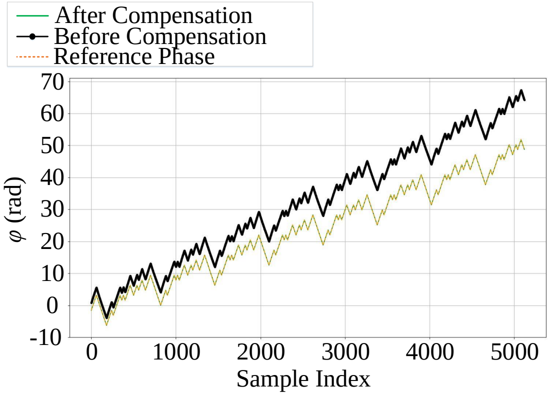
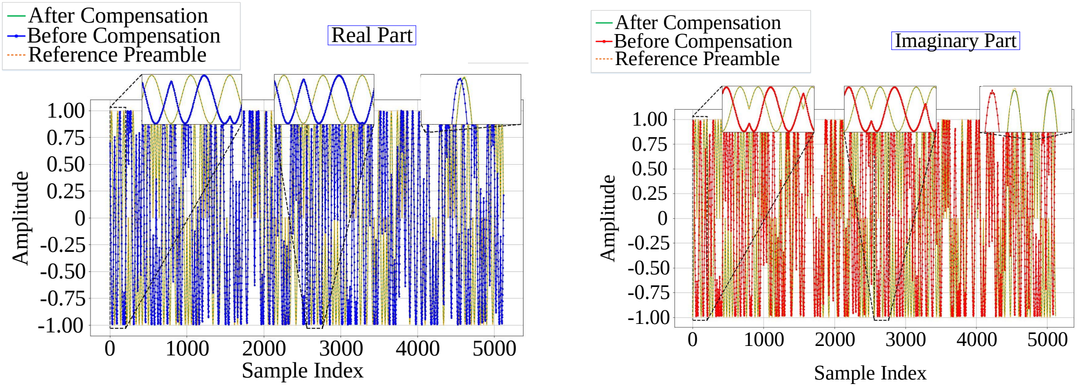


Figure 4.5: Phases of received, reference, and compensated preambles from devices AR_1.



(a)

(b)

Figure 4.6: (a) Real and (b) imaginary components of received, reference, and compensated preambles obtained from a burst of AR_1.

Model generation

The model is shown in Fig. 4.7 and summarized in Table. 4.1. The model is constructed with respect to the classifier model of Fig. 2.1 and has 708 003 trainable parameters.

The model training, validation, and testing, were performed using a 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 fed to the classifier batch by batch. The batch size was set to 20.

As mentioned in Section 2.2, the devices feature extraction and classification are based on deep learning, and more specifically on autoencoders. The autoencoder is shown in Fig. 4.7 (En.layer_l ($l \in \{0, \cdots, 9\}$) and Dec.layer_j ($j \in \{0, \cdots, 8\}$) and summarized in the corresponding rows of Table 4.1.

10 successive *Maxpooling/Conv.* layers, referred to as the encoder, extract the features

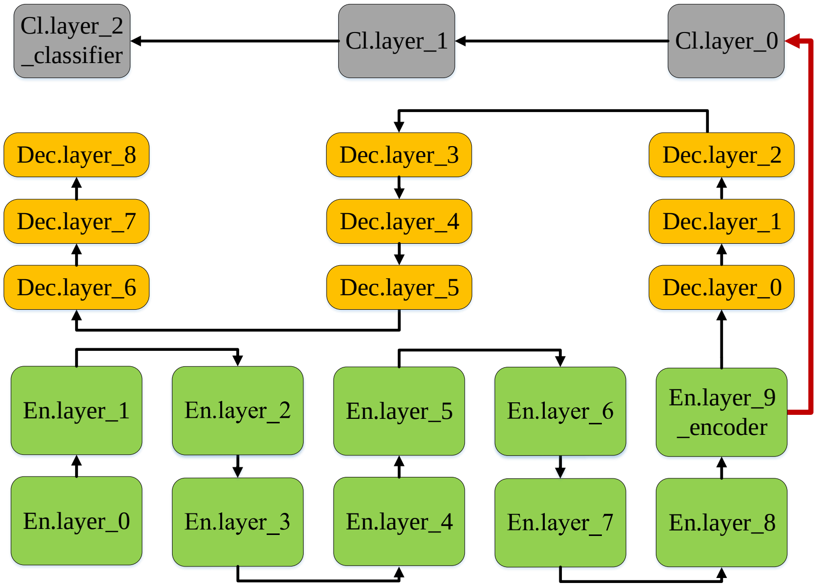


Figure 4.7: Trained classifier model with deep learning (autoencoder).

Table 4.1: Model summary.

Index	Layer Name	Layer Type	Output Shape	Active 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			

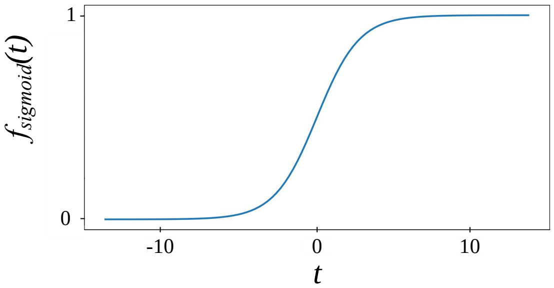


Figure 4.8: Sigmoid function [88].

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 parts En.layer_*l* ($l \in \{0, \cdots, 9\}$) in Fig. 4.7 and the corresponding rows (0 to 9) in Table 4.1. In this structure, no dropout or batch normalization layer is used. In the decoder part for the autoencoder, 9 successive *Upsampling/Deconv.* layers (parts Dec.layer_*l* ($l \in \{0, \cdots, 8\}$) of Fig. 4.7 and the corresponding rows (10 to 17) in Table 4.1) are used to reconstruct the dataset at the output feature layer with the same size as the input layer. As for the encoder, no dropout or batch normalization layer is used.

The classification layers consist of 2 successive fully connected (dense) layers (Cl.layer_*k* ($k \in \{1, 2\}$). Rows 20 and 21 of Table 4.1 identify these layers, using a *Sigmoid* as an active function for binary (or binomial) discrimination, shown in the following equation and Fig. 4.8 [88].

$$f_{sigmoid}(t) = \frac{1}{1 + e^{-t}} = \frac{e^t}{1 + e^t} \tag{4.6}$$

4.3.4 Dataset processing

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* [88, 107] with a learning rate value $lr = 0.0001$. This value assigns how fast the model moves toward the optimized solution during training [88]. If this value is very small, the model will move toward the solution, slowly and it will search all possible local solutions, but at the same time, it may get stuck in the local solutions. On the other hand, if it is selected to be large, the model will move fast toward the optimal solution during the training, but it may miss some of the optimized solutions. During this project, based on the trial and error, a value of 0.0001 for the learning rate showed the best result for converging the model. Meanwhile, the number of epochs for training is selected to be 100 without early stopping to make sure that the number of epochs are large enough to allow the model to converge to the final solution.

Data points allocation for training, validation and testing There are 2 datasets: RAW and DDWT. Therefore, the following strategy for training, validation and testing is applied to both datasets. Different *scenarios* for training, validation, and testing, based on Section 4.2.6 are provided in Table 4.2. Referring to Chapter 2, a target or nontarget device for training is shown by $D_{T,tr}$ or $D_{nT,tr}$, respectively. $D_{T,val}$ refers to a target validation device and $D_{nT,val}$ relates to a nontarget validation device. In addition, target or nontarget testing devices are labeled by $D_{T,te}$ and $D_{nT,te}$. RZ_m, TI_m, and AR_m refer to m^{th} RZUBSTICK, Texas Instrument, and XBEE Digi devices, respectively. Based on this assumption, and with respect to Table 4.2, 4 devices are used for training, 5 devices for validation, and all 8 devices 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 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 4.2.6, 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 Table 4.3. In this table, $D_{tr,m}$, $D_{val,m}$, and $D_{te,m}$ refer to the m^{th} training, validation, and testing device. $N^{tr}_{tr,m}$ is the number of training data points from the m^{th} target or nontarget training device. $N^{val}_{val,m}$ points at the number of validation data points from the m^{th} target or nontarget validation device, and $Nte_{te,m}$ refers to the the number of testing data points from the m^{th} target or nontarget testing device. In addition, N_{tr}, N_{val}, and N_{te} are the total number of training, validation, and testing data points. As can be seen, the total number of data points for each device in the dataset is 11 000 data points. This number is the summation of $N^{tr}_{tr,m}$ (equal to 6000), $N^{val}_{val,m}$ (equal to 1000), and $N^{te}_{te,m}$ (equal to 4000), for m^{th} device.

Shuffling the dataset before feeding into the classifier After splitting the dataset into training, testing, and validation, the dataset for each part should be shuffled before feeding to the classifier. The reason for this randomization is to prevent the model from learning

Table 4.2: Different scenarios for label allocation.

	Positive Device	Training		Validation	
		$D_{T,tr}$	$D_{nT,tr}$	$D_{T,val}$	$D_{nT,val}$
1	AR_1	AR_1	RZ_1, RZ_2, TI_2	AR_1	RZ_1, RZ_2, TI_1, TI_2
2	RZ_1	RZ_1	RZ_2, RZ_3, TI_2	RZ_1	RZ_2, RZ_3, TI_1, TI_2
3	RZ_2	RZ_2	RZ_1, RZ_3, TI_2	RZ_2	RZ_1, RZ_3, TI_1, TI_2
4	RZ_3	RZ_3	RZ_1, RZ_2, TI_2	RZ_3	RZ_1, RZ_2, TI_1, TI_2
5	RZ_4	RZ_4	RZ_1, RZ_3, TI_2	RZ_4	RZ_1, RZ_3, TI_1, TI_2
6	RZ_5	RZ_5	RZ_1, RZ_3, TI_2	RZ_5	RZ_1, RZ_3, TI_1, TI_2
7	TI_1	TI_1	RZ_1, RZ_2, RZ_3	TI_1	RZ_1, RZ_2, RZ_3, TI_2
8	TI_2	TI_2	RZ_1, RZ_2, RZ_3	TI_2	RZ_1, RZ_2, RZ_3, TI_1

	Positive Device	Testing			
		A		B	C
		$D_{T,te}$	$D_{nT,te}$	$D_{nT,te}$	$D_{nT,te}$
1	AR_1	AR_1	RZ_1, RZ_2, TI_2	RZ_3, RZ_4, RZ_5, TI_1	-
2	RZ_1	RZ_1	RZ_2, RZ_3, TI_2	RZ_4, RZ_5, TI_1	AR_1
3	RZ_2	RZ_2	RZ_1, RZ_3, TI_2	RZ_4, RZ_5, TI_1	AR_1
4	RZ_3	RZ_3	RZ_1, RZ_2, TI_2	RZ_4, RZ_5, TI_1	AR_1
5	RZ_4	RZ_4	RZ_1, RZ_3, TI_2	RZ_2, RZ_5, TI_1	AR_1
6	RZ_5	RZ_5	RZ_1, RZ_3, TI_2	RZ_2, RZ_4, TI_1	AR_1
7	TI_1	TI_1	RZ_1, RZ_2, RZ_3	RZ_4, RZ_5, TI_2	AR_1
8	TI_2	TI_2	RZ_1, RZ_2, RZ_3	RZ_4, RZ_5, TI_1	AR_1

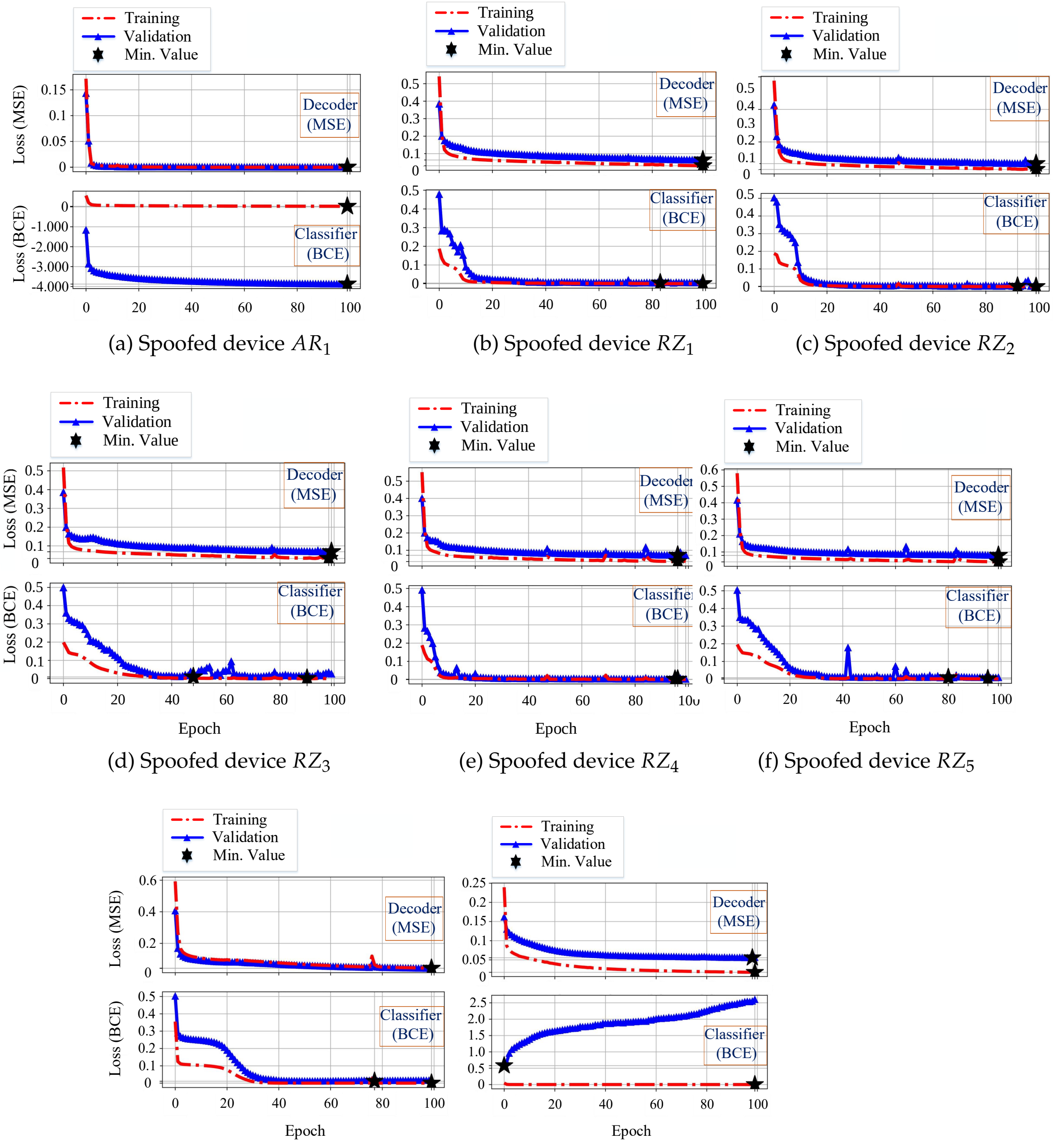
Table 4.3: Number of data points allocated for training, validation, and testing.

Section	Devices	No. of Data Points/Device	Total
Training	$D_{tr,1}$, $D_{tr,2}$, $D_{tr,3}$, $D_{tr,4}$	$N^{tr}_{tr,m}$ $, m = 1, \cdots, 4$	N_{tr}
		6000	24000
Validation	$D_{val,1}$, $D_{val,2}$, $D_{val,3}$, $D_{val,4}$, $D_{val,5}$	$N^{val}_{val,m}$ $, m = 1, \cdots, 5$	N_{val}
		1000	5000
Testing	$D_{te,1}$, $D_{te,2}$, $D_{te,3}$, $D_{te,4}$, $D_{te,5}$, $D_{te,6}$, $D_{te,7}$, $D_{te,8}$	$N^{te}_{te,m}$ $, m = 1, \cdots, 8$	N_{te}
		4000	32000

any possible patterns which exist in the dataset. In such a case, the order of the data points will not be learned by the model.

Decoding and classification convergence During the training of the model, a decoder loss function, such as the mean squared error (MSE) described in Section 2.4, is measured to ensure that the reconstructed data points at the decoder output ($\hat{\mathbf{x}}_n$) are as close as possible to the input data points ($\mathbf{x}_n$).

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 the model. The training and validation losses of the model for decoding (MSE) and classification (BCE) of each of the 8 scenarios are shown in Figs. 4.9 and 4.10. The training and validation minimum errors for decoding



(a) Spoofed device AR_1

(b) Spoofed device RZ_1

(c) Spoofed device RZ_2

(d) Spoofed device RZ_3

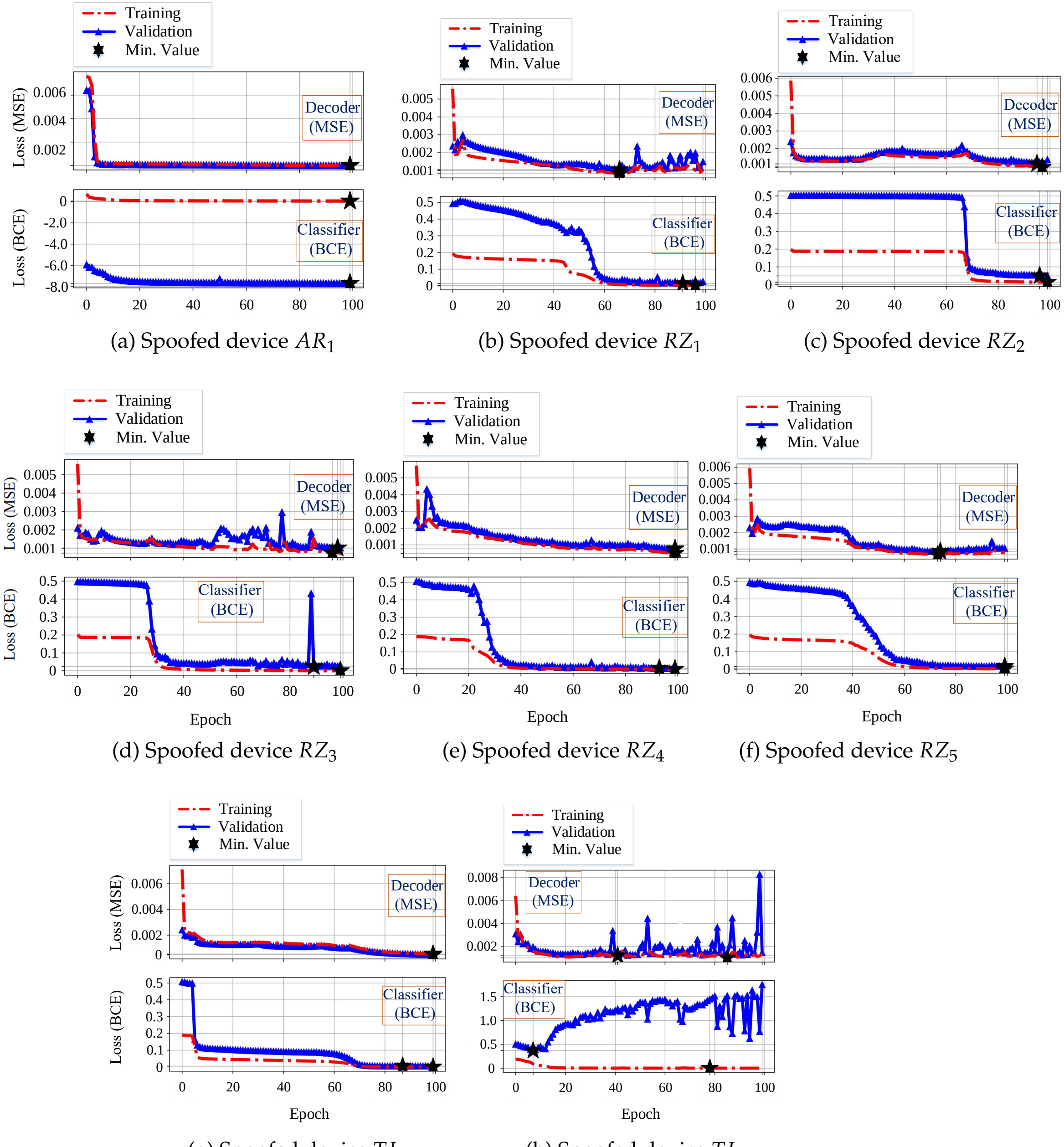
(e) Spoofed device RZ_4

(f) Spoofed device RZ_5

(g) Spoofed device TI_1

(h) Spoofed device TI_2

Figure 4.9: Training and validation losses for all eight scenarios listed in Table 4.2 for the RAW dataset.



(a) Spoofed device AR_1

(b) Spoofed device RZ_1

(c) Spoofed device RZ_2

(d) Spoofed device RZ_3

(e) Spoofed device RZ_4

(f) Spoofed device RZ_5

(g) Spoofed device TI_1

(h) Spoofed device TI_2

Figure 4.10: Training and validation losses for all eight scenarios listed in Table 4.2 for the DDWT dataset.

and classification are depicted as stars in each plot of Figs. 4.9 and 4.10. The iteration where the minimum validation error of classifier happens determines the best model to be used for testing.

As shown, for both the RAW and DDWT datasets, in both cases of training and validation, the converged MSE value for the decoding of the trained model is low enough for 7 out of 8 scenarios in Table 4.2. This means that the model has been able to reconstruct the input signal at the output of the decoder for these cases, with an acceptable approximation. Convergence of the training MSE to a low value will guarantee that the classifier is fed with features with an acceptable quality resulting in the training of the classifier with a high accuracy. On the other hand, achieving a low reconstruction (or decoding) validation MSE value will prove that the model is not overfitted the training dataset. In such a case, like training, the input feature applied to the classifier during the validation is accurate enough to deliver high classification rates (and low BCE values). The low BCE error values, for both training and validation, using the RAW and DDWT datasets, in Figs. 4.9 and 4.10 is a proof for the efficiency of provided input features for a high accuracy classification.

In one case, (device TI_2) for both RAW and DDWT datasets, although the training and validation MSE and the training BCE converge to low values, the validation BCE loss value diverges. In this case, the model has overfitted the training dataset, and increasing the size or quality of the captured dataset from the device TI_2 can help the model to distinguish this device from all other devices with a higher accuracy.

4.3.5 Evaluation of the classification method

After training and evaluation, the classification model is tested against new devices previously unseen by the classifier. A detailed explanation of evaluation procedure using the Confusion Matrix (CM) and the Receiver Operating Characteristic (ROC) curves is presented in Appendix A. As stated in Section 4.3.4 , 30% of data points in the dataset are allocated for testing. The resulting percentages of correct and false allocations of data points for each device are discussed in terms of confusion matrices and receiver operating characteristic plots.

The strategy for the demonstration of the results

Although the presented methodology of this chapter is based on $one - vs - all$ strategy, but following the strategy mentioned in Table 4.2, the results for each rogue device are shown separately. Based on this idea, the confusion matrices in Figs. 4.11, 4.12, 4.13, and the ROC plots in Figs. 4.15, 4.16, 4.17, and 4.18 present a different confusion matrix rate or ROC plot graph for each attacking device. But, why not combining the results from all rogue devices as a single one, labeled as others? As shown in Table 4.2, we try to examine the

Table 4.4: Correct classification rate range for the RAW and DDWT datasets.

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

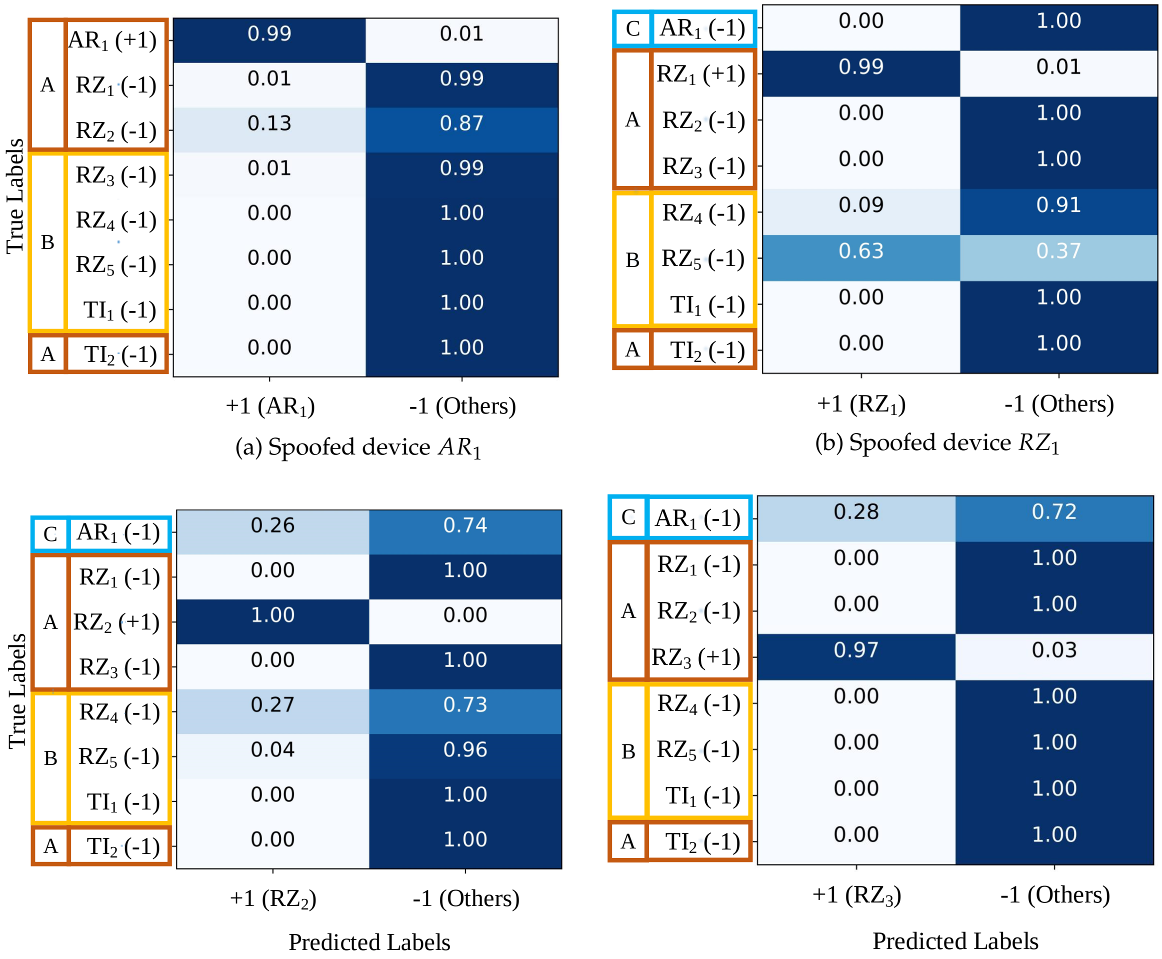
effect of different types of devices in falsifying the trained model. As shown in Table 4.2, the dataset is divided into groups A, B, and C. Using this strategy, the effect of different types of devices is investigated here. Initially it is assumed that group C (the devices which neither them nor their family members have been seen during the training or validation) is the most difficult problem for model performance. The real effect of all these groups on the model performance will be reviewed in the rest of this chapter.

Confusion matrix

Figs. 4.11, 4.12, 4.13, and 4.14 show the confusion matrix for each of the 8 scenarios of Table 4.2 for both the RAW and the DDWT datasets. The selected devices associated to group A in Table 4.2 are assigned with label A, devices in group B are depicted with label B, and devices belonging to group C are shown with label C.

The minimum classification rates for groups of devices A, B, and C of Table 4.2 are reported in Table 4.4. The classification rates in Table 4.4 represent the worst cases of classification of the RAW and DDWT datasets.

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



(a) Spoofed device AR_1

(b) Spoofed device RZ_1

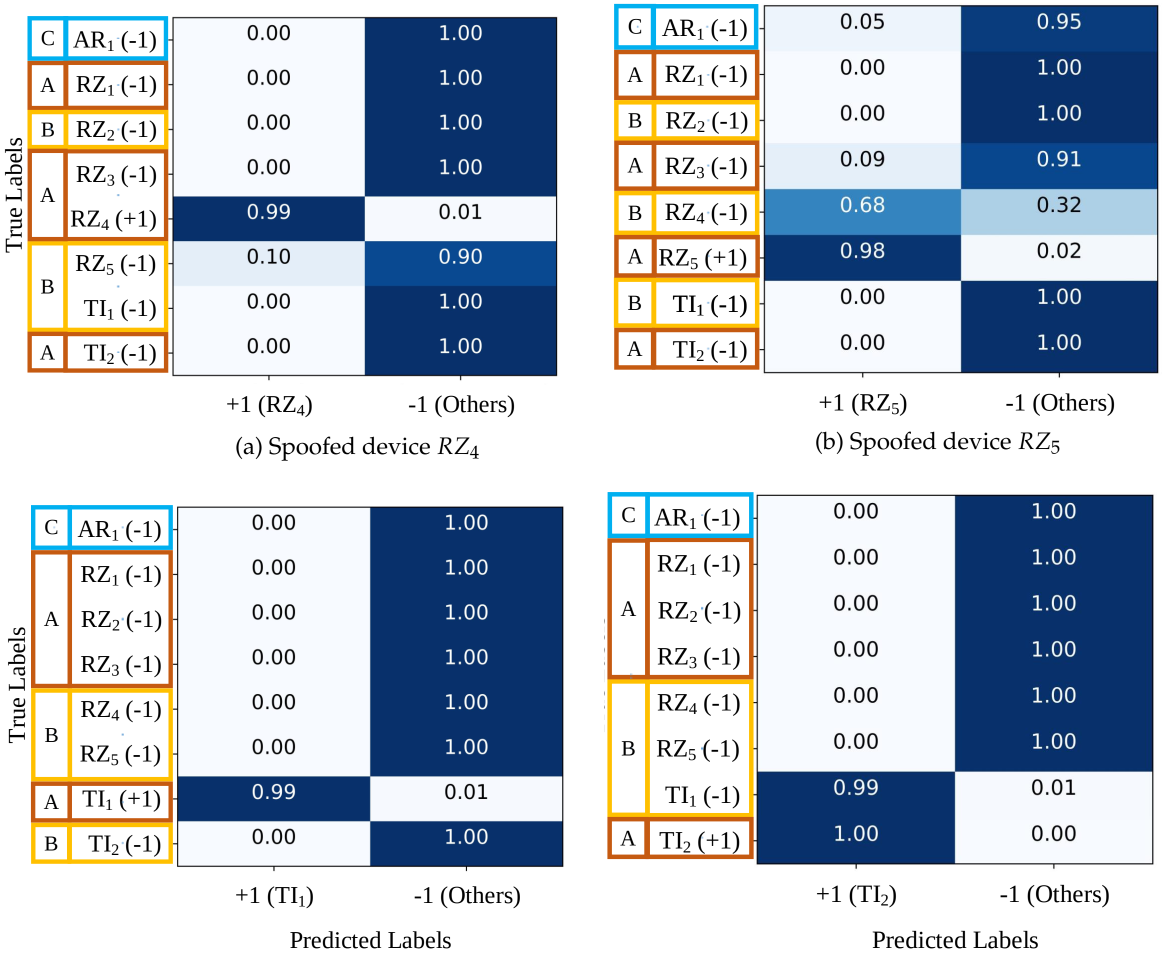
(c) Spoofed device RZ_2

(d) Spoofed device RZ_3

Figure 4.11: Confusion matrix for the first group of four scenarios of Table 4.2 for the RAW dataset.

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, 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. A similar analysis can be applied to all rows of classification rates for the RAW and DDWT datasets in Table 4.4, for groups A, B, and C. The minimum correct classification rate using the RAW dataset in Table 4.4 belongs to the device TI_1 (0.01) obtained from the model trained based on the positive device TI_2.

The comparison between the model performance for the RAW and DDWT datasets Although the classification rate for the same rogue device and the same model (the rogue de-



(a) Spoofed device RZ_4

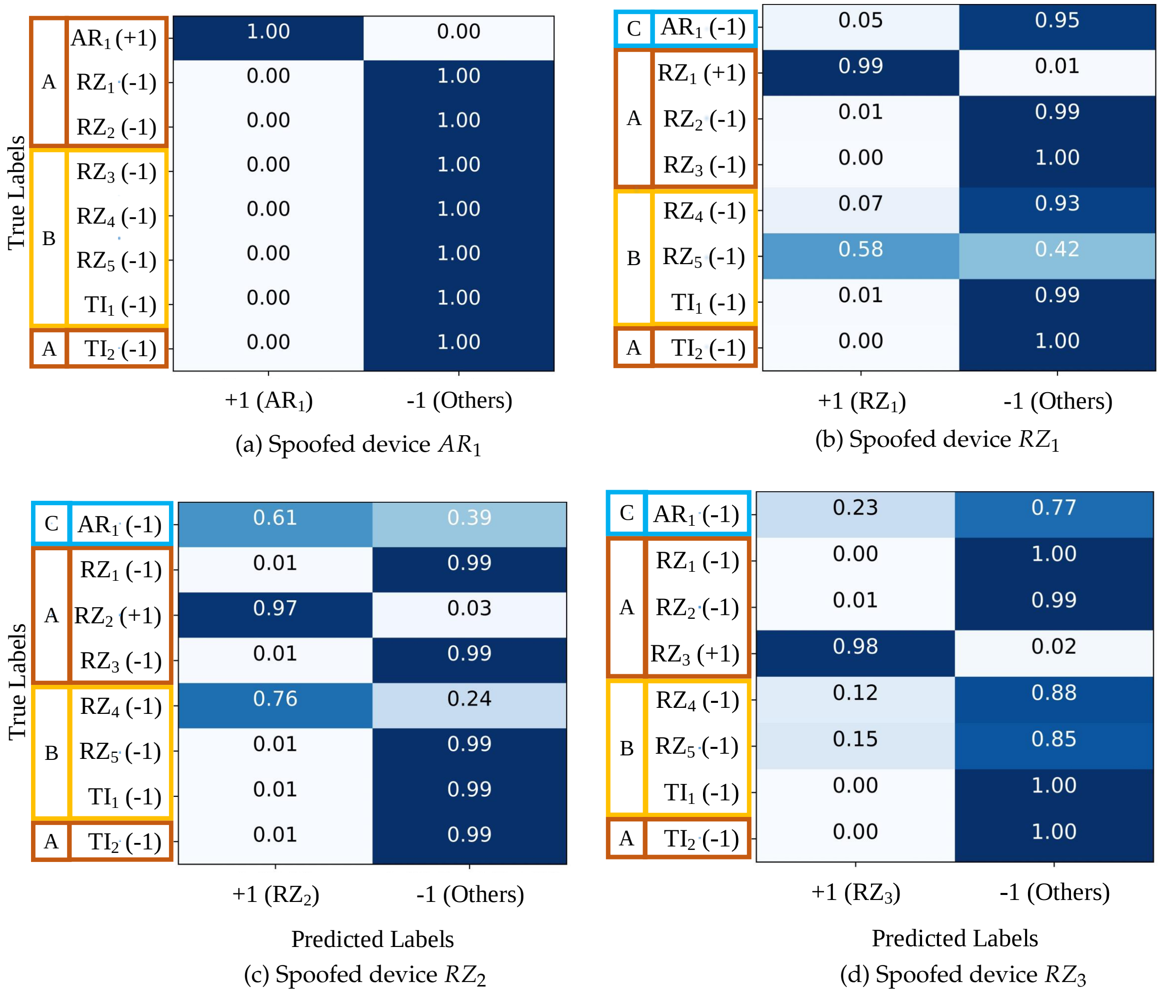
(b) Spoofed device RZ_5

(c) Spoofed device TI_1

(d) Spoofed device TI_2

Figure 4.12: Confusion matrix for the second group of four scenarios of Table 4.2 for the RAW dataset.

vice TI_1 and the model of positive device TI_2) did not improve using the DDWT dataset, the minimum classification rate for some other cases could increase significantly. For instance, using the DDWT dataset, for the rogue device RZ_4, the correct classification rate of the model of positive device RZ_5 (0.64), has doubled compared to using the RAW dataset (0.32). Therefore, comparing the results for the RAW and DDWT datasets in Table 4.4, using the DDWT dataset improves some of the classification rates of the models presented in this chapter, such as the minimum classification rate for group A and positive device RZ_5 has increased from 0.91 to 0.98, using the DDWT dataset, shown in Table 4.4. Using the DDWT dataset can improve the worst case classification rates of the models presented in this chapter. In such a case, using the DDWT dataset, or a combination of both datasets is a good solution to improve the suggested models' worst classification rates.



(a) Spoofed device AR_1

(b) Spoofed device RZ_1

(c) Spoofed device RZ_2

(d) Spoofed device RZ_3

Figure 4.13: Confusion matrix for the first group of four scenarios of Table 4.2 for the DDWT dataset.

Receiver operating characteristics

The resulting receiver operating characteristic (ROC) plots are shown in Figs. 4.15, 4.16, 4.17, and 4.18. For each ROC plot, the area under the curve (AUC), is given in the legend. As can be seen, the devices selected as groups A, B, and C, are labeled with A, B, and C, respectively.

A summary of the ROC plots for device groups A, B, and C is given in Tables 4.5 and 4.6 for both the RAW and DDWT datasets.

The correct classification rate of target/authorized/spoofed device, p_d (detection probability), and its corresponding misclassification rate of rogue/unauthorized/spoofing device, p_{FA} (false alarm probability) in this table, are related to the worst cases using the RAW and DDWT datasets.

Table 4.5: ROC plots summary for the RAW dataset.

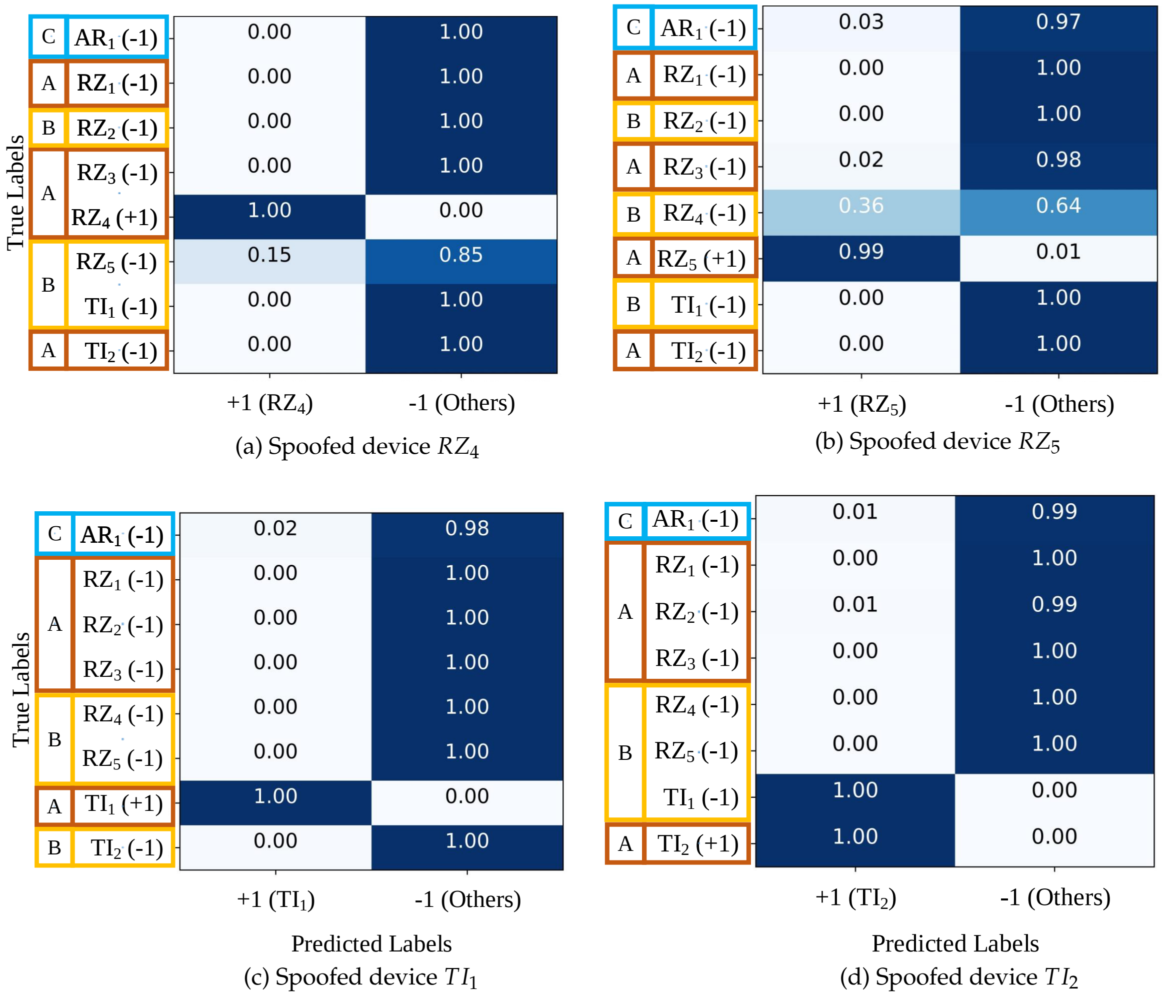
Scenarios	Positive Dev.	***A***				***B***			
		$\boldsymbol{p_d}$	$\boldsymbol{AUC}_{\mathbf{min}}$	$\boldsymbol{AUC}_{\mathbf{max}}$	$\boldsymbol{AUC}_{\mathbf{avg}}$	$\boldsymbol{p_d}$	$\boldsymbol{AUC}_{\mathbf{min}}$	$\boldsymbol{AUC}_{\mathbf{max}}$	$\boldsymbol{AUC}_{\mathbf{avg}}$
1	AR_1	$\sim 1.0\ for\ p_{fa} \in [0,1]$	1.0000	1.0000	1.0000	$\geq 0.95\ for\ p_{fa} \in [0,1]$	1.0000	1.0000	1.0000
2	RZ_1	$\sim 1.0\ for\ p_{fa} \in [0,1]$	0.9993	0.9999	0.9996	$\geq 0.9\ for\ p_{fa} \in [0.2,1]$	0.9544	0.9997	0.9835
3	RZ_2	$\sim 1.0\ for\ p_{fa} \in [0,1]$	1.0000	1.0000	1.0000	$\geq 0.95\ for\ p_{fa} \in [0,1]$	0.9980	1.0000	0.9992
4	RZ_3	$\sim 1.0\ for\ p_{fa} \in [0,1]$	0.9993	0.9999	0.9996	$\geq 0.95\ for\ p_{fa} \in [0,1]$	0.9981	1.0000	0.9991
5	RZ_4	$\sim 1.0\ for\ p_{fa} \in [0,1]$	0.9998	0.9999	0.9999	$\geq 0.9\ for\ p_{fa} \in [0.01,1]$	0.9971	0.9998	0.9988
6	RZ_5	$\geq 0.9\ for\ p_{fa} \in [0.01,1]$	0.9947	0.9999	0.9980	$\geq 0.8\ for\ p_{fa} \in [0.2,1]$	0.8815	0.9992	0.9599
7	TI_1	$\sim 1.0\ for\ p_{fa} \in [0,1]$	1.0000	1.0000	1.0000	$\geq 0.9\ for\ p_{fa} \in [0.01,1]$	0.9999	1.0000	0.9999
8	TI_2	$\sim 1.0\ for\ p_{fa} \in [0,1]$	1.0000	1.0000	1.0000	$\geq 0.8\ for\ p_{fa} \in [0.65,1]$	0.3926	1.0000	0.7975

Scenarios	Positive Dev.	***C***			
		$\boldsymbol{p_d}$	$\boldsymbol{AUC}_{\mathbf{min}}$	$\boldsymbol{AUC}_{\mathbf{max}}$	$\boldsymbol{AUC}_{\mathbf{avg}}$
1	AR_1	-	-	-	-
2	RZ_1	$\sim 1.0\ for\ p_{fa} \in [0,1]$	0.9988	0.9988	0.9988
3	RZ_2	$\geq 0.9\ for\ p_{fa} \in [0.01,1]$	0.9953	0.9953	0.9953
4	RZ_3	$\geq 0.9\ for\ p_{fa} \in [0.2,1]$	0.9430	0.9430	0.9430
5	RZ_4	$\sim 1.0\ for\ p_{fa} \in [0,1]$	0.9997	0.9997	0.9997
6	RZ_5	$\geq 0.9\ for\ p_{fa} \in [0.01,1]$	0.9441	0.9441	0.9441
7	TI_1	$\sim 1.0\ for\ p_{fa} \in [0,1]$	1.0000	1.0000	1.0000
8	TI_2	$\sim 1.0\ for\ p_{fa} \in [0,1]$	1.0000	1.0000	1.0000

Table 4.6: ROC plots summary for the DDWT dataset.

Scenarios	Positive Dev.	***A***				***B***			
		$\boldsymbol{p_d}$	$\boldsymbol{AUC}_{\mathbf{min}}$	$\boldsymbol{AUC}_{\mathbf{max}}$	$\boldsymbol{AUC}_{\mathbf{avg}}$	$\boldsymbol{p_d}$	$\boldsymbol{AUC}_{\mathbf{min}}$	$\boldsymbol{AUC}_{\mathbf{max}}$	$\boldsymbol{AUC}_{\mathbf{avg}}$
1	AR_1	$\sim 1.0\ for\ p_{fa} \in [0,1]$	1.0000	1.0000	1.0000	$\sim 1.0\ for\ p_{fa} \in [0,1]$	1.0000	1.0000	1.0000
2	RZ_1	$\geq 0.9\ for\ p_{fa} \in [0.01,1]$	0.9979	0.9989	0.9984	$\geq 0.9\ for\ p_{fa} \in [0.2,1]$	0.9313	0.9986	0.9752
3	RZ_2	$\geq 0.9\ for\ p_{fa} \in [0.01,1]$	0.9901	0.9978	0.9952	$\geq 0.8\ for\ p_{fa} \in [0.3,1]$	|0.8337	0.9981	0.9428
4	RZ_3	$\geq 0.9\ for\ p_{fa} \in [0.01,1]$	0.9990	0.9996	0.9994	$\geq 0.8\ for\ p_{fa} \in [0.05,1]$	0.9741	0.9995	0.9843
5	RZ_4	$\sim 1.0\ for\ p_{fa} \in [0,1]$	0.9998	0.9999	0.9998	$\geq 0.9\ for\ p_{fa} \in [0.01,1]$	0.9965	0.9998	0.9987
6	RZ_5	$\geq 0.9\ for\ p_{fa} \in [0.01,1]$	0.9979	0.9991	0.9986	$\geq 0.9\ for\ p_{fa} \in [0.1,1]$	0.9684	0.9990	0.9883
7	TI_1	$\sim 1.0\ for\ p_{fa} \in [0,1]$	1.0000	1.0000	1.0000	$\geq 0.95\ for\ p_{fa} \in [0.01,1]$	1.0000	1.0000	1.0000
8	TI_2	$\sim 1.0\ for\ p_{fa} \in [0,1]$	1.0000	1.0000	1.0000	$\geq 0.8\ for\ p_{fa} \in [0.4,1]$	0.8115	1.0000	0.9372

Scenarios	Positive Dev.	***C***			
		$\boldsymbol{p_d}$	$\boldsymbol{AUC}_{\mathbf{min}}$	$\boldsymbol{AUC}_{\mathbf{max}}$	$\boldsymbol{AUC}_{\mathbf{avg}}$
1	AR_1	-	-	-	-
2	RZ_1	$\sim 1.0\ for\ p_{fa} \in [0,1]$	0.9978	0.9978	0.9978
3	RZ_2	$\geq 0.85\ for\ p_{fa} \in [0.2,1]$	0.9062	0.9062	0.9062
4	RZ_3	$\geq 0.8\ for\ p_{fa} \in [0.1,1]$	0.8997	0.8997	0.8997
5	RZ_4	$\geq 0.95\ for\ p_{fa} \in [0.01,1]$	0.9997	0.9997	0.9997
6	RZ_5	$\geq 0.9\ for\ p_{fa} \in [0.01,1]$	0.9945	0.9945	0.9945
7	TI_1	$\sim 1.0\ for\ p_{fa} \in [0,1]$	0. 9999	0.9999	0.9999
8	TI_2	$\geq 0.95\ for\ p_{fa} \in [0.01,1]$	1.0000	1.0000	1.0000



(a) Spoofed device RZ_4

(b) Spoofed device RZ_5

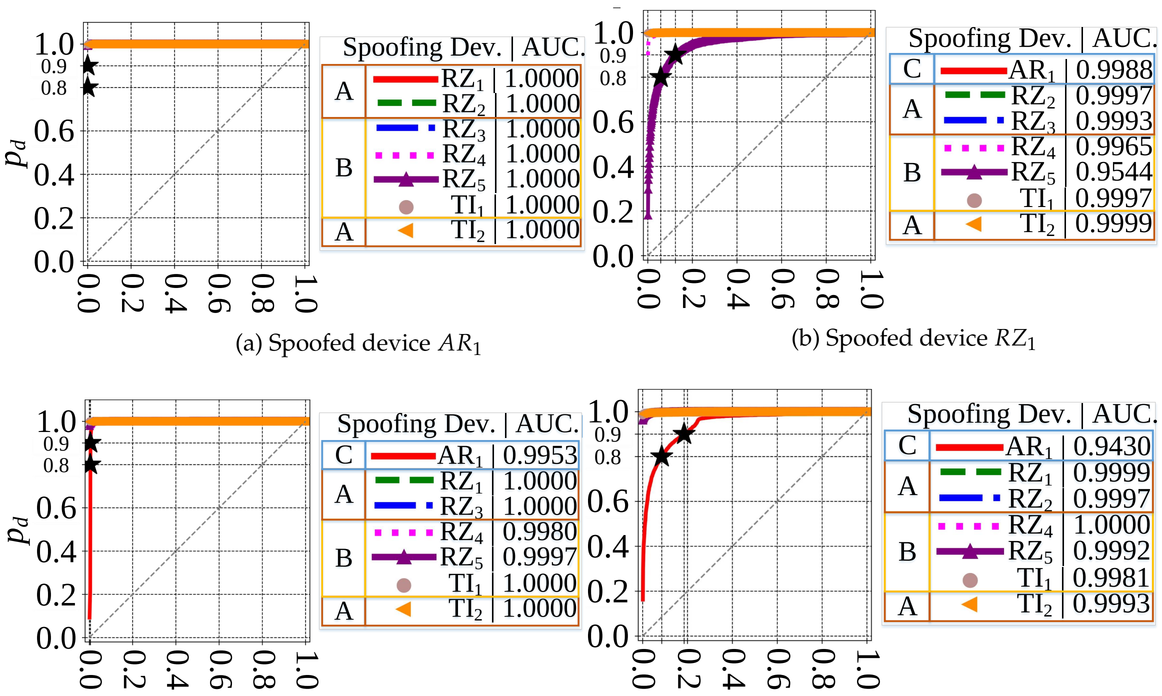
(c) Spoofed device TI_1

(d) Spoofed device TI_2

Figure 4.14: Confusion matrix for the second group of four scenarios of Table 4.2 for the DDWT dataset.

For instance, in the second scenario, feeding the model trained for positive device RZ_1 with the RAW dataset shown in Fig. 4.15 (b) and summarized in Table 4.5, the range of correct classification rate (p_d) for the values of p_{FA} in $[0, 1.0]$ is almost equal to 1.0, corresponding to group A of testing devices. As mentioned in Table 4.2, this group of devices are those used in training, too. The lowest value of AUC_{min} belongs to the sixth scenario (0.9947) and the minimum AUC_{max} is equal to 0.9999. For group A and the RAW dataset, the average area under the curve (AUC_{avg}) for the sixth scenario is 0.9980, which is the minimum AUC_{avg}.

Looking at the same plot or the second scenario in Table 4.5, group B gives p_d values higher than 0.9 for p_{FA} in the range $[0.2, 1.0]$ for the same RAW dataset. The minimum value among all values of AUC_{min} and the minimum value among all vales of AUC_{max} for this group are equal to 0.3926 and 0.9992, respectively. AUC_{avg} is in the range $[0.7975, 1.0000]$ for all scenarios. As shown in Fig. 4.16 (d) and summarized in the eighth scenario of Table 4.5,



(a) Spoofed device AR_1

(b) Spoofed device RZ_1

(c) Spoofed device RZ_2

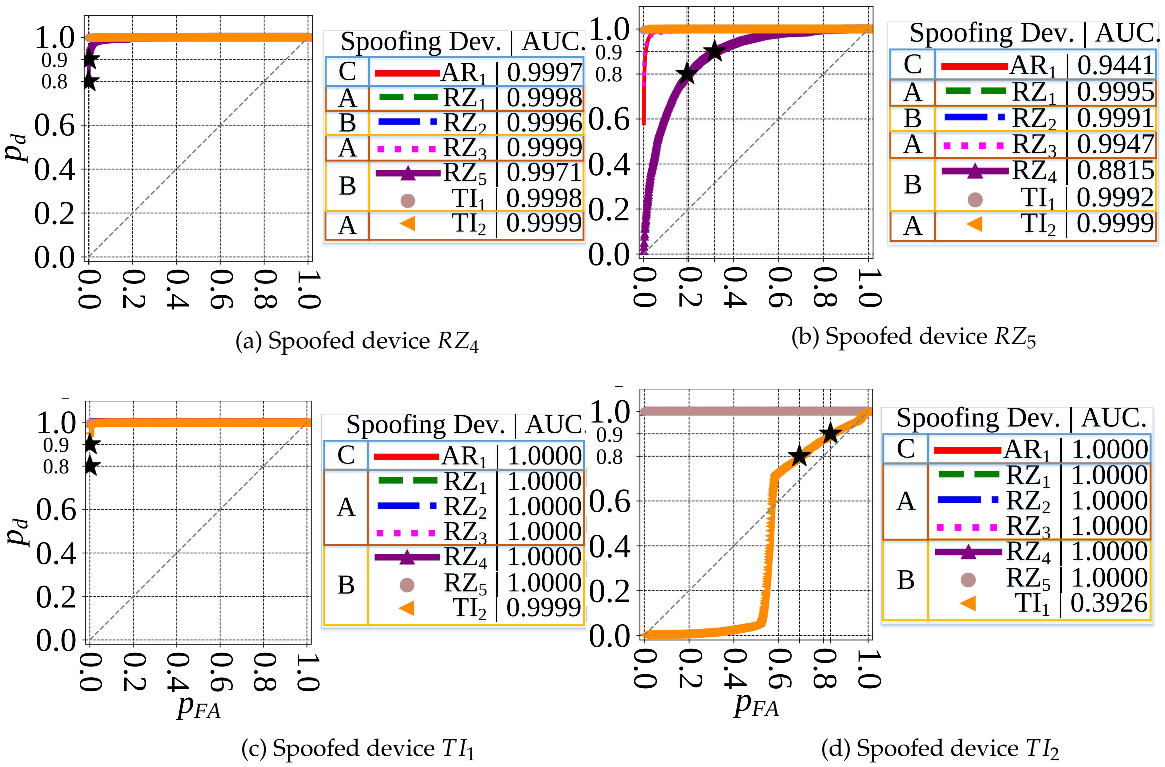
(d) Spoofed device RZ_3

Figure 4.15: ROC plot for first group of four scenarios of Table 4.2 for the RAW dataset.

the model trained based on the authorized device TI$_2$ has the lowest AUC (that is 0.3926) for group B, among all scenarios of all three groups A, B, and C. As can bee seen, based on Fig. 4.9 (h), although the training BCE loss decreases and converges to values near 0, the validation BCE value diverges which means that the model is overfitted to the training dataset and is not able to classify the validation datasets, correctly. Therefore, in Fig. 4.16 (d), due to the overfitting, the authorized class conditional probabilities of testing data points from the authorized device (TI$_2$) are so low that no matter how much we change the threshold for the different values of false alarm probabilities (p_{FA}) lower than 0.4, none of the likelihood ratios of the authorized class data points are affected, and the authorized device correct classification probabilities (p_d) are close to 0. For values of (p_{FA}) higher than 0.4, the p_d values start to increase, abruptly, and p_d reaches values higher than 0.8 for $p_{FA} \geq 0.7$.

Finally, referring to the same plot or the second scenario in Table 4.5, for the RAW dataset, group C (as a new device which does not have any family member in the training) presents a correct classification rate (p_d) almost equal to 1.0 for all possible values of p_{FA}. For all scenarios in group C, for the area under the curve we have $0.9430 \leq AUC_{min} \leq 1.0000$ and $0.9430 \leq AUC_{max} \leq 1.0000$. In addition, the average area under the curve (AUC_{avg}) is in the range $[0.9430, 1.0000]$.

As can be seen, for the RAW dataset, the best performance for this model relates to



(a) Spoofed device RZ_4

(b) Spoofed device RZ_5

(c) Spoofed device TI_1

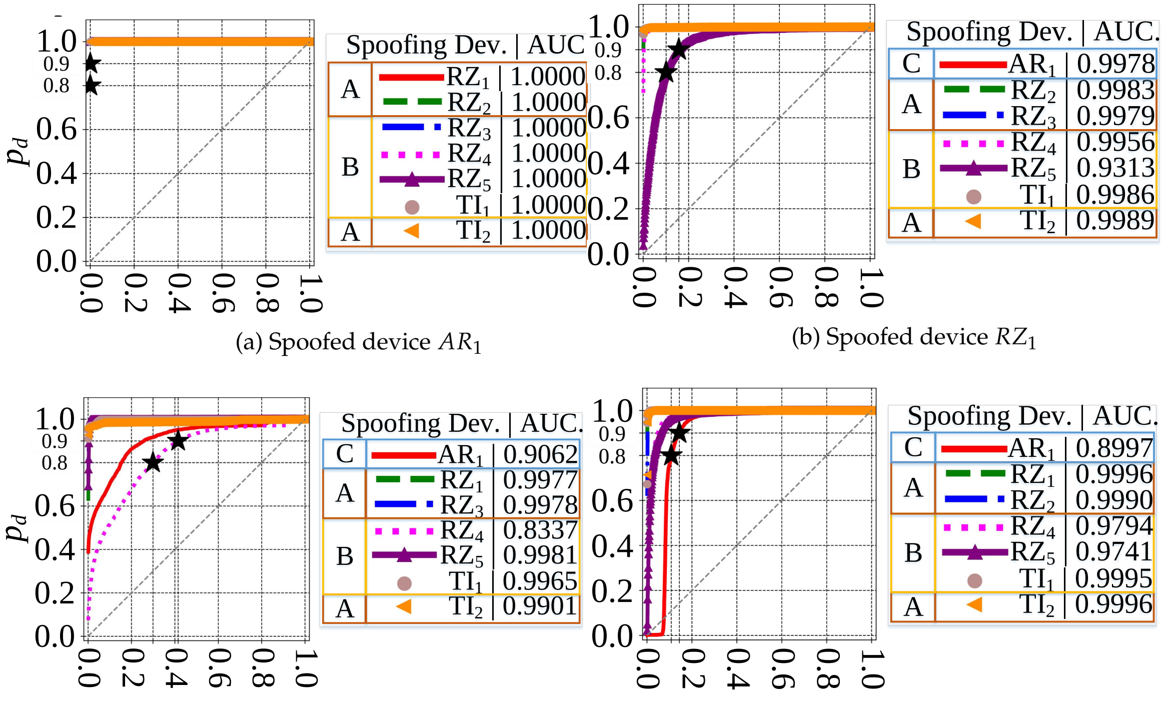
(d) Spoofed device TI_2

Figure 4.16: ROC plot for second group of four scenarios of Table 4.2 for the RAW dataset.

group A with the minimum $AUC_{avg} = 0.9980$, and the worst case scenario (resulting in the worst model performance) for this type of dataset is related to group B with the minimum $AUC_{avg} = 0.7975$.

A similar explanation can be presented about all other rows of Tables 4.5 and 4.6 for both RAW and DDWT datasets. It is worth mentioning that for the DDWT dataset, the best performance for this model goes to group A with the minimum $AUC_{avg} = 0.9952$, and the minimum AUC_{avg} for this type of dataset is corresponding to group C with the minimum $AUC_{avg} = 0.8997$.

Comparison between the model performance for the 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 part of Table 4.5 is better than the corresponding group of devices in the DDWT dataset part of Table 4.6, the ROC plots for device 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 device group B; that is from 0.65 to 1 for the RAW dataset, and from 0.4 to 1 for the DDWT dataset. Therefore, decreasing the lower threshold of p_{FA} will result in better performance of the model through achieving higher values of cor-



(a) Spoofed device AR_1

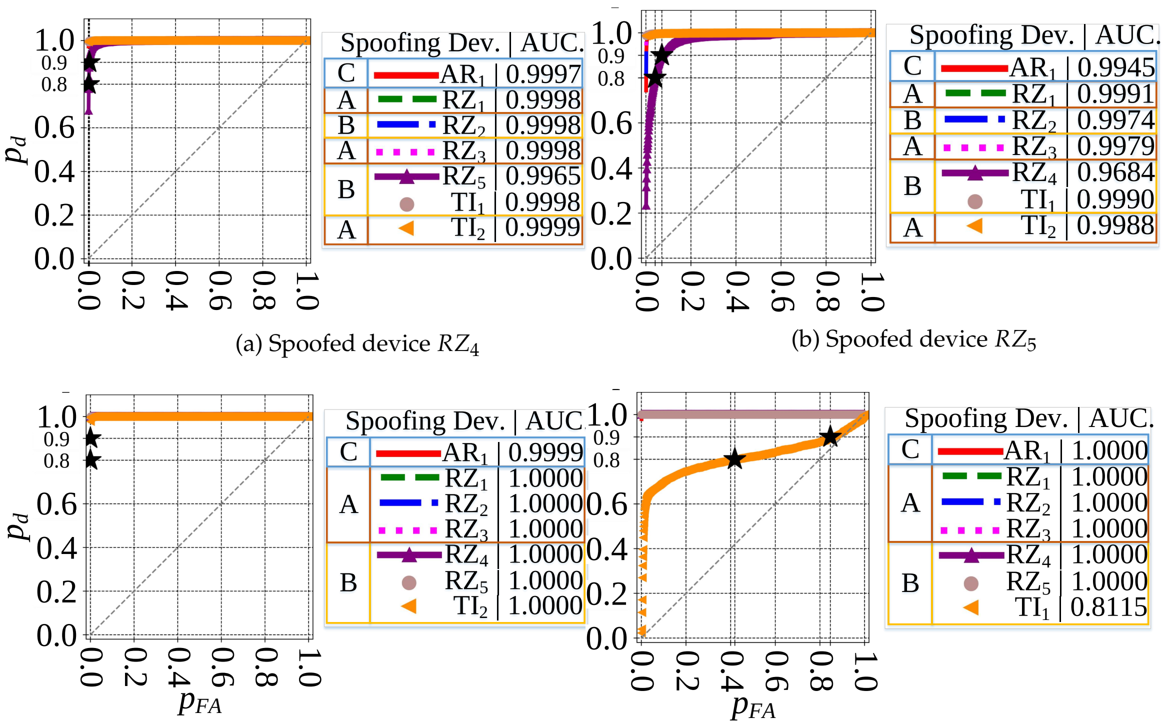
(b) Spoofed device RZ_1

(c) Spoofed device RZ_2

(d) Spoofed device RZ_3

Figure 4.17: ROC plot for first group of four scenarios of Table 4.2 for the DDWT dataset.

rect classification rates (p_d) by tolerating lower values of false alarm probability (p_{FA}). For the comparison of the minimum AUC_{avg} for all scenarios related to group A, this value for the RAW and DDWT datasets are equal to 0.9980 and 0.9952, respectively. For the same variable (minimum AUC_{avg}), the DDWT dataset shows a better performance ($0.9372 > 0.7975$) for group B, and the RAW dataset presents an improvement over the RAW dataset for group C ($0.9430 > 0.8997$). Based on this, it may be concluded that using the RAW dataset the better performance can be achieved by the model. But, if the focus is on the worst case scenarios, the minimum AUC_{min} for the DDWT dataset is 0.8115 which is more than twice the same factor for the RAW dataset (0.3926). Therefore, although the comparison of confusion matrix results shows a better classification rate using the RAW dataset, the ROC plots of Figs. 4.17 and 4.18 compared to Figs. 4.15 and 4.16 determine the better performances using the DDWT dataset for the cases where the classifier *tolerates* larger false alarm values p_{FA}. Therefore, combining both datasets, we can achieve the higher AUC values and also, improve the model performance for the worst case scenarios.



(a) Spoofed device RZ_4

(b) Spoofed device RZ_5

(c) Spoofed device TI_1

(d) Spoofed device TI_2

Figure 4.18: ROC plot for second group of four scenarios of Table 4.2 for the DDWT dataset.

4.4 Classifier performance comparison

In this section, the performance of the proposed classification method is compared with other classifiers recently presented in the literature. Before comparison, the following points regarding the strategy adopted in this chapter are summarized.

(a) First, device discrimination is repeated for two types of datasets:

  (i) RAW dataset.

  (ii) DDWT dataset.

(b) As indicated in Table 4.2, focusing on each devices allows to consider different scenarios for device allocation for training, validation, and testing. Each scenario results in the training of a different model. Each model represents a specific device as an authorized/spoofed unit.

(c) The discrimination of a device per model requires that beside the selected device (as authorized/spoofed unit), at least another device plays the role of a rogue/unauthorized/ spoofing one at the time of training.

(d) The devices used in the testing phase are separated into three groups:

  (i) Group A:
  Devices which are used in the training phase.

  (ii) Group B:
  Devices which are not used in the training, but for which one or more members of their family are.

  (iii) Group C:
  Devices for which neither them nor their family members are used in the training phase.

Referring to this short summarization, three cases of comparison are done with respect to their approach to each mentioned criteria above.

AUC performance evaluation Why do we use AUC for the comparison? What does it mean if a methodology presents a higher value of AUC? Area under the curve (AUC) is used for comparing the different classifiers or methodologies. When a classifier presents a higher value of AUC compared to another one, it can be inferred that it has a better performance in discrimination of rogue and authorized classes. The summary of the AUC results presented in this section are reported in Tables 4.8 and 4.9, respectively.

4.4.1 Comparison with reference [35]

In [35], Merchant et al. present a framework for training a convolutional neural network for the verification of the rogue and authoried devices. Using the model designed in this work, and shown in Table 4.7, comparison with [35] is done based on the 8 strategies of Table 4.2 with the RAW and DDWT datasets. For the comparison purposes, the model presented in Table 4.7 is implemented, and the RAW and DDWT datasets used in this thesis are fed to the model.

Referring to the summary shown in Tables 4.8 and 4.9, the AUC_{max} for both methodologies in this chapter for our proposed autoencoder classifier and [35] is equal to 1.0000. This value is the same for groups A, B, and C, related to the RAW and DDWT datasets. On the other hand, the AUC_{min} and AUC_{avg} for the proposed autoencoder classifier shows a better performance. For example, let us focus on the RAW dataset. For group A, AUC_{min} and AUC_{avg} have increased from 0.6216 to 0.9947, and from 0.9840 to 0.9996, respectively. Focusing on group B, AUC_{min} and AUC_{avg} have increased from 0.0580 to 0.3926, and from 0.8830 to 0.9686, respectively. Finally, in the case of group C, AUC_{min} and AUC_{avg} related to [35] show a better performance, but the performance of the approach presented in this

Table 4.7: Model summary based on [35].

Index	Layer Type	Dimension	Output Shape	Active Func.	Parameters No.
0	InputLayer	-	(None, 5120, 2)	-	0
1	Conv1D	32×19	(None, 5102, 32)	ELU	1248
2	MaxPooling	2	(None, 2551, 32)	-	0
3	Conv1D	128×19	(None, 2533, 128)	ELU	77952
4	MaxPooling	2	(None, 1266, 128)	-	0
5	Conv1D	32×15	(None, 1252, 32)	ELU	61472
6	MaxPooling	2	(None, 626, 32)	-	0
7	Conv1D	16×11	(None, 616, 16)	ELU	5648
8	MaxPooling	2	(None, 308, 16)	-	0
9	Flatten	-	(None, 4928)	-	0
10	Dense	128	(None, 128)	ELU	630912
11	Dropout (0.5)	-	(None, 128)	-	0
12	Dense	16	(None, 16)	ELU	2064
13	Dropout (0.5)	-	(None, 16)	-	0
14	Dense	2	(None, 2)	Softmax	34

chapter for the proposed classifier is again not far away from the method in [35]. In [35], AUC_{min} and AUC_{avg} are 0.9532 and 0.9919, respectively, whereas in our work AUC_{min} and AUC_{avg} are 0.9430 and 0.9830, respectively.

Using the DDWT dataset, for group A, AUC_{min} and AUC_{avg} have increased from 0.3964 to 0.9901, and from 0.9745 to 0.9989, respectively. Focusing on group B, AUC_{min} and AUC_{avg} have increased from 0.0602 to 0.8115, and from 0.8470 to 0.9792, respectively. Finally, in the case of group C, AUC_{min} and AUC_{avg} related to [35] show a better performance, but the performance of the approach presented in this work is not far away from the method in [35]. In our work, the AUC_{min} is 0.8997, whereas in [35], the AUC_{min} is 0.9184. Also, AUC_{avg} in the proposed approach is 0.9711 while this factor in [35] is equal to 0.9879.

4.4.2 Comparison with reference [108]

In the work of Ramsey et al. [108], the dataset consists of the 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. This dataset is fed to the multi discriminant analysis (MDA) model, and the results are analyzed with the ROC plot. For the comparison of the results in [108], the statistical features (σ^2, γ, and κ) of $a(t)$, $\varphi(t)$, and $f(t)$ of the RAW dataset in this thesis are fed to the MDA and the results for all 8 strategies of Table 4.2 are summarized in Table 4.8. Referring to the summary shown in Table 4.8, the AUC_{max} for both methodologies in this chapter for the proposed classifier and that presented in [108], for all the groups A, B, and C of RAW dataset is 1.0000. The AUC_{min} and AUC_{avg} of our work shows an improvement for the RAW dataset for group A, from 0.2404 to 0.9947, and from 0.9634 to 0.9996, respectively. In the case of group B, AUC_{min} and AUC_{avg} have changed from 0.0108 to 0.3926, and from

0.9355 to 0.9686, respectively. For group C, [108] showed a bad performance of AUC_{min} equal to 0.1976, but the methodology presented in this chapter improved this value to 0.9430. On the other hand, the AUC_{avg} in this chapter has a higher value (0.9830) compared to that of [108] (0.8803).

As can bee seen, the performance of the methodology presented is very dependant on the complexity of the dataset. Due to the complexity of the dataset in this thesis, the approach in [108] presents a bad performance for the lowest value of AUC_{min} (0.0108 for group B). Using our proposed autoencoder method in this chapter, we could increase the worst performance to 0.3926, using the RAW dataset.

4.4.3 Comparison with reference [97]

In [97], Dubendorfer et al. employed multiple discriminant analysis (MDA) to classify ZigBee devices from their RF-DNA (Radio Frequency Distinct Native Attributes). The used dataset consists of the wavelet transform of statistical features (variance as σ^2, skewness as γ, and kurthosis as κ of physical signal characteristics like amplitude ($a(t)$), phase ($\varphi(t)$), and instantaneous frequency ($f(t)$)) of the recorded signals. For the comparison of the results in [97], the wavelet transform of the statistical features (σ^2, γ, and κ) of $a(t)$,$\varphi(t)$, and $f(t)$ of the RAW dataset in this thesis are fed to the MDA for all 8 strategies of Table 4.2. A summary of the results is depicted in Table 4.9.

Based on Table 4.9, the AUC_{max} for both methodologies proposed in this chapter and that in [97], for all the groups in the DDWT dataset is 1.0000. The AUC_{min} and AUC_{avg} of our proposed classifier illustrates the change of result for the DDWT dataset for group A, from 0.0000 to 0.9901, and from 0.7474 to 0.9989, respectively. Let us focus on group B, where AUC_{min} and AUC_{avg} have increased from 0.0000 to 0.8115, and from 0.7356 to 0.9792, respectively. Ultimately, for group C, [97] showed a bad performance of AUC_{min} equal to 0.0000, but the proposed methodology gives a value equal to 0.8997. On the other hand, the AUC_{avg} in this chapter has a better value (0.9711) compared to that of [97] (0.6992).

4.5 Conclusion

In this chapter, a rogue device discrimination method in a vulnerable network channel at the physical layer is presented. The main strategy presented in this chapter relies on discriminating the 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

Table 4.8: Comparison of methodology presented in Chapter 4 and approaches in references [35] and [108], using the RAW dataset.

	RAW Dataset					
	A			*B*		
	AUC_{min}	AUC_{max}	AUC_{avg}	AUC_{min}	AUC_{max}	AUC_{avg}
Autoencoder (Ch. 4)	0.9947	1.0000	0.9996	0.3926	1.0000	0.9686
Merchant et al. [35]	0.6216	1.0000	0.9840	0.0580	1.0000	0.8830
Ramsey et al. [108]	0.2404	1.0000	0.9634	0.0108	1.0000	0.9355
	C					
	AUC_{min}		AUC_{max}		AUC_{avg}	
Autoencoder (Ch. 4)	0.9430		1.0000		0.9830	
Merchant et al. [35]	0.9532		1.0000		0.9919	
Ramsey et al. [108]	0.1976		1.0000		0.8803	

Table 4.9: Comparison of methodology presented in this chapter and approaches in references [35] and [97], using the DDWT dataset.

	DDWT Dataset					
	A			*B*		
	AUC_{min}	AUC_{max}	AUC_{avg}	AUC_{min}	AUC_{max}	AUC_{avg}
Autoencoder (Ch. 4)	0.9901	1.0000	0.9989	0.8115	1.0000	0.9792
Merchant et al. [35]	0.3964	1.0000	0.9745	0.0602	1.0000	0.8470
Dubendorfer et al. [97]	0.0000	1.0000	0.7474	0.0000	1.0000	0.7356
	C					
	AUC_{min}		AUC_{max}		AUC_{avg}	
Autoencoder (Ch. 4)	0.8997		1.0000		0.9711	
Merchant et al. [35]	0.9184		1.0000		0.9879	
Dubendorfer et al. [97]	0.0000		1.0000		0.6992	

the feature extraction process. Feature extraction is investigated for the RAW (time domain) and DDWT (time-scale domain) 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 results reported in the literature presented in Section 4.4.

Chapter 5

Unauthorized Device Rejection in Wireless Communication Systems using Wavelet Transform and LSTM Based Autoencoders

5.1 Introduction

In modern wireless communication systems, different new and effective approaches have been presented recently to increase security. Recent developments in machine learning have led to several methods for efficient classifiers, such as that proposed by Merchant et al. in [35] where a deep learning model is fed with the steady state component of the initial data points. The proposed strategy of this chapter is to add a long short-term memory (LSTM) layer to the structure presented in Chapter 4. As will be discussed in more details in this chapter, it can be seen that there is a long time-dependency in the captured signal of wireless ZigBee devices. On the other hand, LSTM layers are suitable to extract features from the signals with time-dependency. Then, adding an LSTM layer to the autoencoder based classifier (as presented in Fig. 2.2) can result in the increase of the classification rate of the classifier. The methodology presented in this chapter is published in [62].

The structure of this chapter is as follows. Section 5.2 describes the proposed classification method. The implementation of the proposed classifier is described in Section 5.3. Section 5.4 addresses the training of the designed model of Section 5.3. Experimental results obtained with the proposed device classification system are reported in Section 5.5. Finally, Section 5.7 discusses the findings and outcomes of the chapter.

5.2 Proposed classification method

Our goal is to propose an approach such that when an unknown device, for instance device$_{\mathrm{i}}$, enters a network, the network model tries to detect if this anonymous device is an authorized device, identified as device$_{\mathrm{m}}$, or is an unauthorized (rogue) device. The proposed approach employs a binomial (binary) classifier to distinguish the authorized members of the network from the rest of the world, that is, a *one-vs-all* strategy. As explained in Chapter 4, before feeding the data to the model for feature extraction and classification, first, we need to increase the quality of the dataset to an acceptable level. For such a purpose, the following steps are needed.

1. Dataset acquisition
2. Received preamble extraction
3. Dataset phase and frequency compensation
4. Dataset generation

Each of these steps are described in detail in Sections 4.2.1, 4.2.2, 4.2.3, and 4.2.4, respectively. After the generation of high quality RAW and DDWT datasets, the next step is to design a feature extraction structure which provides the classifier with high quality features, for classification.

5.2.1 Time-dependency extraction

The main purpose of adding an LSTM layer to the autoencoder in this chapter is to extract the time-dependency of the signal, as a key feature for classification purpose. As mentioned in Section 2.3, each preamble is not just a simple vector, but a sequence of successive values with a meaningful relationship between the adjacent values, called the time-dependency, and the order of the values is important [109]. Time-dependency of the successive samples is calculated with an autocorrelation, defined as the correlation of a signal $\mathbf{x}_n$ (at time-step k_1) with its delayed version (at time-step k_2), as shown in Eq. (5.1) [110].

$$\mathrm{R}_{\mathbf{x}_n}\left(k_1, k_2\right) = \mathrm{E}\left[\mathbf{x}_n\left(k_1\right), \mathbf{x}_n^*\left(k_2\right)\right] \tag{5.1}$$

where E is the expected value, * shows the complex conjugate, and $\mathbf{x}_n^*\left(k_i\right)$ is equal to the complex conjugate of sample x_{n,k_i} at time-step k_i ($i \in \{1,2\}$), shown in Eq. (2.2). For the acquired signal from ZigBee devices in this work, the autocorrelation in Eq. (5.1) is higher than 0 for successive samples ($\mathrm{R}_{\mathbf{x}_n}\left(k_1, k_2\right) > 0$), which means that there is a time-dependency between samples at times steps k_1 and k_2.

Extraction of this time-dependency is the responsibility of an RNN layer [88]. Also, as described in Section 5.3.4, using an LSTM layer (as a type of RNN layer) in the encoder part for time-dependency extraction, it is possible to feed the decoder and classifier with this key feature, for a better classifier accuracy [111]. Now, as explained in Section 2.3.2, an RNN layer is able to extract the time-dependency of each input preamble signal by the output vector state of all cell units of the network ($\mathbf{h}_k$) at time-step k in Eq. (2.10), and the loop feedback from the output vector state at time-step $k-1$ (by $\mathbf{h}_{k-1}$) to the output vector state at time-step k (by $\mathbf{h}_k$) is shown in this equation. This feedback can be used as a key feature for improving the classification accuracy.

5.2.2 Model definition

Autoencoder

An LSTM layer which was described in more detail in Section 2.3.2, can be added to the encoder part, increasing the classification accuracy, as shown in Fig. 2.2, and evaluated in Section 5.5.

Feature extraction at the encoder Feature extraction maps high dimensional data to a reduced low-dimensional space. New feature $\mathbf{x}_{n_F}$ is obtained as the result of either a linear or a nonlinear transformation of $\mathbf{x}_n$ into $\mathbf{x}_{n_F}$ at the encoder based on Eq. (2.4).

Input reconstruction at the decoder The reconstruction in the decoder is done based on Eq. (2.5), where function $s_g(.)$ maps the hidden representation $\mathbf{x}_{n_F}$ back to the reproduction vector $\hat{\mathbf{x}}_n$.

The training of the autoencoder extracts $\theta = (\Omega, \Omega', \mathbf{b}_{\mathbf{x}_n}, \mathbf{b}'_{\mathbf{x}_{n_F}})$, by decreasing the distance between the original input data point $\mathbf{x}_n$ and the reconstruction space $\hat{\mathbf{x}}_n$ using the appropriate loss function. The full description of encoder and decoder parts is presented in Section 2.2.

Classification

Since the strategy adopted in this thesis is *one-vs-all*, there are two classifier outputs, each giving the conditional probability of the data points belonging to either the *positive* or the *negative* class (see Fig. 2.2), where W_{pos} and W_{neg} represent these positive and negative (mutually exclusive) classes, respectively. Training the classifier is done by feeding the allocated features ($\mathbf{x}_{n_F}$) to the classifier in Fig. 2.2. In the implemented version of this structure, two dense and two dropout layers are used in the classifier which decreases the probability of overfitting the model for small datasets.

5.2.3 Model training/validation/testing

After making the dataset ready, and defining the model structure, it is now time to train, validate, and test the defined model to evaluate the performance of the proposed structure. The steps needed for such a purpose are as follows.

1. Dataset partitioning into training, validation, and testing datasets
2. Model training and validation
3. Model testing

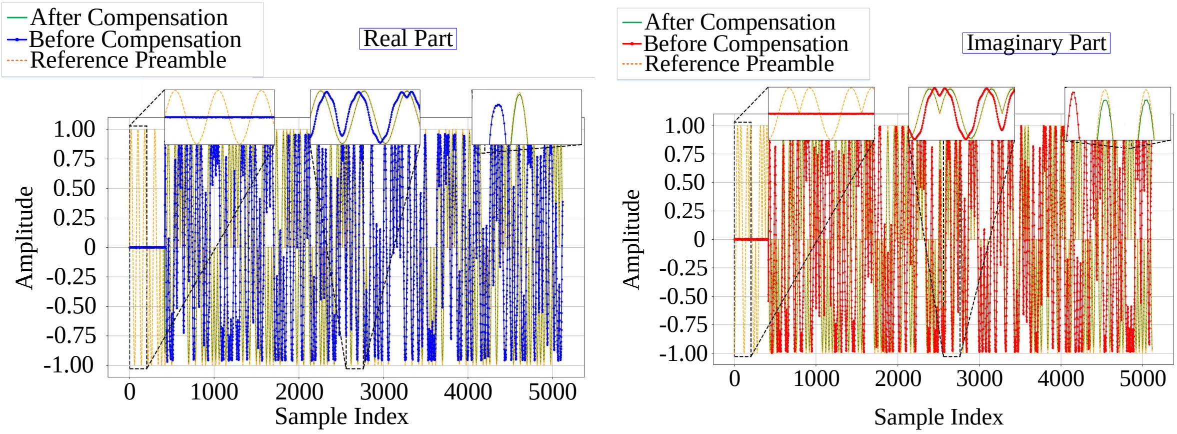
Steps 1 and 2 are essential for training the selected model and are explained in Sections 4.2.6 and 2.4, respectively. On the other hand, step 3 is the last and the most important part in the determination of the efficiency of the trained model by evaluating the confusion matrices and the ROC plots. This part is described in detail in Sections A.1 and A.2.

5.3 Classifier model implementation

As discussed, the trained model for device separation and classification is to reject the access of rogue devices. Then, the introduced model of Fig. 2.2 is tested for real cases. In this section, the result of the model classification fed by the real dataset is discussed.

5.3.1 Experimental setup

As explained in Chapter 4, Figs. 3.3, 3.4, 3.9 and 3.10 depict the required laboratory setup designed for the signal measurements. Fig. 4.4 illustrates the devices used for the data acquisition in this chapter. A Zynq XC7Z020 SoC FPGA is used as the ZigBee signal receiver. A set of 8 different ZigBee wireless devices, including 5 RZUSBSticks (RZ_1, RZ_2, RZ_3, RZ_4, and RZ_5), one XBee Digi (AR_1), and two Texas Instruments devices (TI_1 and TI_2) are used for the data acquisition. An extracted signal burst obtained from the experimental setup is depicted in Fig. 3.8. The extracted signal bursts need preprocessing to increase their quality prior to training the classifier model.



(a)

(b)

Figure 5.1: (a) Real and (b) imaginary components of received, reference, and compensated preambles obtained from a burst of TI_2.

5.3.2 Preprocessing of acquired signals

Preambles extraction from the received signal bursts

As pointed out in Section 5.2, the first task in processing of the sampled signals consists of preamble extraction. This is done by convolving the received signal with a single known reference symbol (each of the 8 successive reference QPSK symbols). The convolution results in 8 successive peaks as shown for the ZigBee device RZ_1 in Fig. 3.16.

Section 5.2 refers to Section 4.2.1 for estimating the preamble length, that is 5120 samples, obtained by sampling a preamble of 128 µs duration at a 40 MHz sampling rate.

Phase and frequency compensation

As addressed in Section 5.2, the preamble phase and frequencies of the received and the reference preambles should be as close as possible. Phase and frequency compensation is realized using Eqs. (3.4), (3.5) and (3.6). Comparison between extracted and compensated signals and theirs phases, for device TI_2 are illustrated in Figs. 5.1 and 5.2, respectively. As can be seen, the comparison of the received and reference preambles shows the efficiency of the suggested approach for compensation.

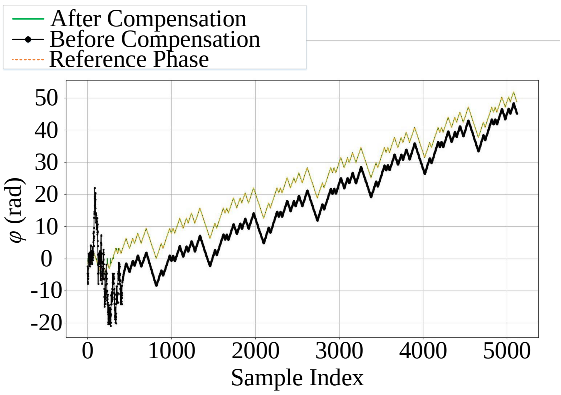


Figure 5.2: Phases of received, reference and compensated preambles from TI_2.

5.3.3 Dataset generation

Two different types of datasets are generated: a RAW dataset and the DDWT dataset. For the latter dataset, the dyadic wavelet transform is applied on the extracted preambles. The procedure is addressed in Section 5.2.

5.3.4 Classifier model generation

The classifier model is trained using the autoencoder deep learning method. The autoencoder model is depicted in Fig. 5.3 and its different layers are summarized in Table 5.1. Its structure is based on the model shown in Fig. 2.2 and consists of 1 023 718 trainable parameters. This is the summation of trainable parameters for each layer in Table. 5.1.

Comparison of the LSTM based autoencoder model in this chapter and the autoencoder in Chapter 4 As discussed earlier, both models in Figs. 4.7 and 5.3 are taking benefit from an autoencoder based classifier structure. But they have three important differences. First, the autoencoder introduced in Fig. 5.3 takes benefit from using an LSTM layer for time dependency extraction. Second, besides using an LSTM layer, if the autoencoder in both structures of Figs. 4.7 and 5.3 are compared, it can be seen that they are completely different. The autoencoder in Fig. 4.7 has 19 layers, but the number of layers in the autoencoder part of Fig. 5.3 is 41. Third, referring to Tables 4.1 and 5.1, the type of layers which are used in both models is slightly different, since the structure in Table 5.1 takes benefit from using dropout layers. Consequently, the design of the models in both chapters are completely different.

The model training, validation, and testing phases were performed using a *gpu NVIDIA Quadro K620* hardware platform. The dataset processing was done using *Python* 3.6, *Ten-*

sorFlow 1.0.8 and *Keras* 2.2.0 software tools. The whole set of data points were fed to the classifier, batch by batch, with the batch size set to 20 data points.

The autoencoder part of the classifier (Section 5.2.2) is used for feature extraction; the corresponding autoencoder layers (En.layer_*l* ($l \in \{0, \cdots, 19\}$) and Dec.layer_*j* (for $j \in \{0, \cdots, 14\}$)) in Fig. 5.3 are detailed in the corresponding rows of Table 5.1. The encoding layers (En.layer_*l* ($l \in \{0, \cdots, 19\}$) of Table 5.1) reduce the array size from (5120, 2) at the input layer to (10, 1) at the encoding output layer. The decoding section (Dec.layer_*j* ($j \in \{0, \cdots, 14\}$)) decodes the preamble back to the input array size (5120, 2).

In the classification part (Cl.layer_*k* ($k \in \{0, \cdots, 5\}$) of Table 5.1), fully connected (dense) layers (Cl.layer_*k* ($k \in \{2, 4\}$)) assign the classifier conditional probabilities of the extracted features. The active function used for these layers are *Sigmoid* (as shown in Eq. (4.6) and Fig. 4.8) or *Relu*, shown in Eq. (5.2) and Fig. 5.4 [88].

LSTM layer performance As can be seen in Fig. 5.3, En.layer_3 is an LSTM layer. As mentioned in Chapters 2 and 5, the background idea for the presence of LSTM layer in the encoder part is to extract the time-dependency of the signal, as a key feature for feeding the decoder and classification parts. But there is a question about the performance of this layer, how it is fed, the input size to this layer, the output size of it, and its approach for extracting the time-dependency. In Appendix B, an example of the LSTM layer design using Keras [111] is presented. The input and output shapes of En.layer_3 in Table 5.1 are described, in detail in this appendix. On the other hand, the mechanism of an LSTM layer for extracting the time dependency of a preamble is explained in detail in Section 2.3.2

$$f_{relu}(t) = \begin{cases} 0, & \text{for } t < 0 \\ t, & \text{otherwise} \end{cases} \tag{5.2}$$

5.4 Classifier model training and validation

The designed autoencoder and classification parts need to be trained with the RAW and DDWT datasets. During the training, the weights of the designed model will update to deliver the maximum possible classification accuracy at the output of the classifier.

5.4.1 Training and validation of acquired datatsets

After defining the classifier model, the model training phase starts with assigning some initial parameters. One important parameter is the learning rate of the optimization algorithm

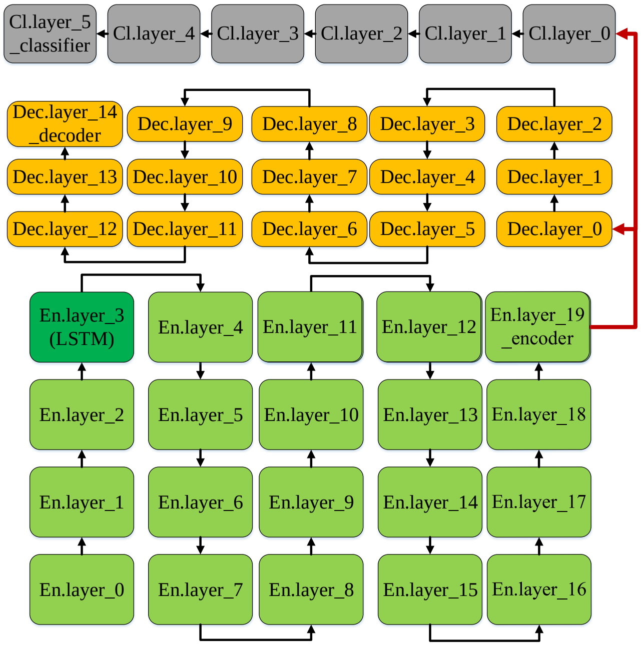


Figure 5.3: Trained classifier model with deep learning (LSTM based autoencoder and classifier).

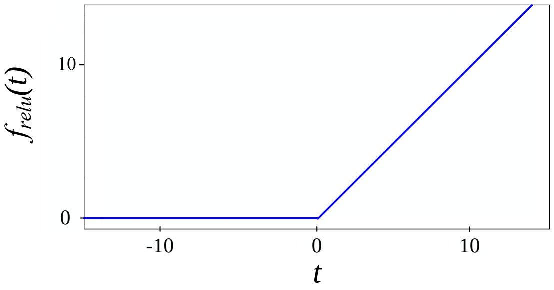


Figure 5.4: Relu function [88].

Adam [88,107]. Adam is an optimization algorithm for stochastic gradient descent which is used in the training phase of the deep learning models. Based on what explained in detail in Section 4.3.4, the learning rate of the optimization and the number of epochs for training the model are selected equal to 0.0001 and 100, respectively.

5.4.2 Data points allocation for training, validation and testing

In the training, validation, and testing phases, depending on which of the eight devices is selected, eight possible *scenarios* are defined as in Table 4.2. The detailed procedure of "Data points allocation for training, validation and testing" is explained in Section 4.3.4. As

Table 5.1: LSTM based autoencoder and classifier model summary.

Index	Layer Name	Layer Type	Output Shape	Active Func.	Parameters No.
0	En.layer_0	InputLayer	(None, 5120, 2, 1)	-	0
1	En.layer_1	Conv2D	(None, 4800, 1, 1)	Relu	643
2	En.layer_2	Reshape	(None, 4800, 1)	-	0
3	En.layer_3	CuDNNLSTM	(None, 4800, 320)	-	413440
4	En.layer_4	Dropout	(None, 4800, 320)	-	0
5	En.layer_5	Reshape	(None, 4800, 320, 1)	-	0
6	En.layer_6	MaxPooling1D	(None, 960, 1, 1)	-	0
7	En.layer_7	Dropout	(None, 960, 1, 1)	-	0
8	En.layer_8	Conv1D	(None, 800, 1, 32)	Relu	5184
9	En.layer_9	Dropout	(None, 800, 1, 32)	-	0
10	En.layer_10	MaxPooling1D	(None, 160, 1, 32)	-	0
11	En.layer_11	Dropout	(None, 160, 1, 32)	-	0
12	En.layer_12	Conv1D	(None, 80, 1, 32)	Relu	82976
13	En.layer_13	Dropout	(None, 80, 1, 32)	-	0
14	En.layer_14	Conv1D	(None, 40, 1, 32)	Relu	42016
15	En.layer_15	Dropout	(None, 40, 1, 32)	-	0
16	En.layer_16	Conv1D	(None, 20, 1, 32)	Relu	21536
17	En.layer_17	Dropout	(None, 20, 1, 32)	-	0
18	En.layer_18	Conv1D	(None, 10, 1, 32)	Relu	11296
19	En.layer_19_encoder	Dropout	(None, 10, 1, 32)	-	0
20	Dec.layer_0	Conv1DTranspose	(None, 10, 1, 32)	Relu	11296
21	Dec.layer_1	Dropout	(None, 20, 1, 32)	-	0
22	Dec.layer_2	Conv1DTranspose	(None, 40, 1, 32)	Relu	21536
23	Dec.layer_3	Conv1DTranspose	(None, 80, 1, 32)	Relu	42016
24	Dec.layer_4	Dropout	(None, 80, 1, 32)	-	0
25	Dec.layer_5	Conv1DTranspose	(None, 160, 1, 32)	Relu	82976
26	Dec.layer_6	Dropout	(None, 160, 1, 32)	-	0
27	Dec.layer_7	UpSampling1D	(None, 800, 1, 32)	-	0
28	Dec.layer_8	Dropout	(None, 800, 1, 32)	-	0
29	Dec.layer_9	Conv1DTranspose	(None, 960, 1, 32)	Relu	164896
30	Dec.layer_10	Dropout	(None, 960, 1, 32)	-	0
31	Dec.layer_11	UpSampling1D	(None, 4800, 1, 32)	-	0
32	Dec.layer_12	Dropout	(None, 4800, 1, 32)	-	0
33	Dec.layer_13	Conv2DTranspose	(None, 5120, 2, 1)	Relu	20545
34	Dec.layer_14_decoder	Dropout	(None, 5120, 2, 1)	-	0
35	Cl.layer_0	Flatten	(None, 320)	-	0
36	Cl.layer_1	Dropout	(None, 320)	-	0
37	Cl.layer_2	Dense	(None, 320)	Relu	102720
38	Cl.layer_3	Dropout	(None, 320)	-	0
39	Cl.layer_4	Dense	(None, 266666)	Sigmoid	642
40	Cl.layer_5_classifier	Dropout	(None, 2)	-	0
Total parameters:		1,023,718			
Trainable parameters:		1,023,718			

explained, the allocated percentages of the data points from the dataset for each group are 60% for training, 10% for validation, and 30% for testing. Based on this assumption, and with respect to Table 4.2, four different devices are used for training, five devices for validation and eight devices for testing. The number of allocated data points from each device for each phase is given in Table 4.3. Note that the total number of data points for each device in the dataset is higher than 11 000. Thus, there are enough data points from each device, which prevents overfitting of the model to a specific device.

Table 5.2: Dropout percentage allocated for the encoding, decoding, and classification layers of LSTM based autoencoder model in training for scenario 8 of Table 4.2.

Model Section	Dropout Percentage	Row Number
Encoder	0.5	4, 7, 9, 11, 13, 15, 17, 19
Decoder	0.5	21, 24, 26, 28, 30, 32, 24
Classifier	0.3	36, 38, 40

In addition, as discussed in Section 4.3.4, the dataset should be shuffled before feeding to the classifier to prevent the model from learning the order of the data points.

5.4.3 Decoding and classification convergence

During the training of the model, the *dropout percentage* used for encoding, decoding, and classification, depends on the type of scenario of Table 4.2 that the model is trained for. In seven out of 8 scenarios (scenarios 1 to 7), the dropout is eliminated for all three sections (by assigning the $dropout\ percentage = 0$ for all model layers). For scenario 8, the *dropout percentage* for specific layers of the model in Table 5.1 are as shown in Table 5.2.

As mentioned in Section 4.3.4, to assess the *distance* between the input data points and those at the output layer of the decoder and thus the training process accuracy, the loss function used is the mean squared error (MSE) between the two sets of data points. The loss function selected for the classification output is the binary cross-entropy (BCE). On the other hand, for the evaluation of the model performance at the training phase, the loss function value of the validation dataset is an accurate criterion. The loss values and the minimum errors for decoding and classification parts in training and validation are illustrated in Figs. 5.5 and 5.6. The iteration where the minimum validation error of classifier happens determines the best model to be used for testing.

As illustrated, for both the RAW (Fig. 5.5) and DDWT (Fig. 5.6) datasets, in both cases of training and validation, the converged MSE for the decoding of the trained model is low enough for 7 out of 8 scenarios of Table 4.2, which illustrates that the model has been able to reconstruct the input signal at the output for these cases, with an acceptable accuracy. On the other hand, low training MSE values provide the classifier with a good quality input. Meanwhile, convergence of the validation MSE shows that the model is not overfitting the training dataset.

Let us have a look at BCE values for both training and validation, using the RAW and DDWT datasets. In Figs. 5.5 and 5.6, the BCE error values for both training and validation, using the RAW and DDWT datasets, are shown. As can be seen, similar to the MSE, for 7 out of 8 scenarios in Table 4.2, both training and validation BCE values converge to a low value, as a sign of good classification performance. The classifier high accuracy is a proof that the quality of the input features at the output of the encoder has an acceptable level.

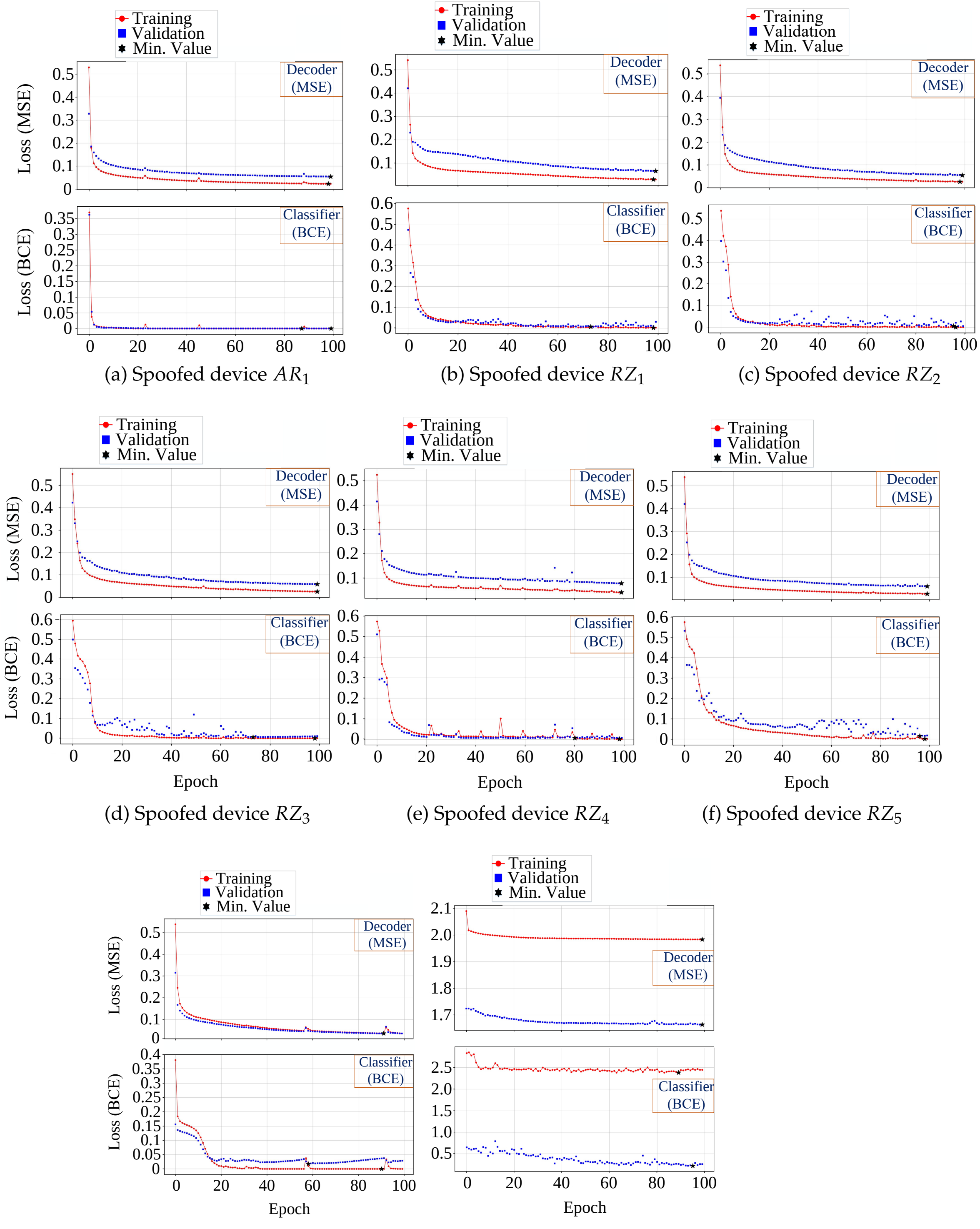
For device TI_2, using the RAW datasets, due to the $dropout\ percentage = 0.5$ for the encoding and decoding, and the $dropout\ percentage = 0.3$ for the classification parts (as shown in Table 5.2), the converged MSE loss value for training (2.0) is higher than the validation MSE loss value (1.7). Similarly, the BCE value in training converges to a higher value (2.5) than that of validation (which is less than 0.5). For the DDWT dataset, the MSE for both training and validation converges to a value lower than 0.2. However, although the BCE in training converges to 0, the minimum validation BCE loss value is lower than 0.5. As can be seen, although the performance of the model for scenario 8 is not as good as the other 7 cases in Table 4.2, feeding the model with the DDWT dataset improves the model performance at the output of the decoder (MSE), during the training and validation stages, compared to the RAW dataset. In such a case, increasing the size or quality of the captured dataset from device TI_2 can help the model to distinguish this device from all other devices with a higher accuracy.

Validation comparison with autoencoder model presented in Chapter 4 As can be seen in Figs. 4.9, 4.10, 5.5, and 5.6, the most challenging scenario is the case 8 from Table 4.2. On the other hand, since the output of the classifier is the part which is evaluated in Sections 4.3.5 and 5.5, the performance of the autoencoder model in Chapter 4 and the LSTM based autoencoder in Chapter 5 during the validation are compared using the BCE values of the validation. As can be seen in Figs. 4.9 (h) and 4.10 (h), the minimum BCE value in validation for scenario 8 in Table 4.2, for both RAW and DDWT datasets, is close to 0.5, and it diverges due to the fact that the model is overfitting the training dataset. But referring to the 5.5 (h) and 5.6 (h), for both the RAW and DDWT datasets, the validation BCE value does not diverge and the minimum value is close to 0.5, which shows the advantage of the LSTM based autoencoder proposed in this chapter compared to the autoencoder model analyzed in Chapter 4, at the validation stage, for the worst case scenario.

5.5 Experimental results

5.5.1 Proposed classification testing

In this section, the emphasis is on testing and validation of the model, using the LSTM based autoencoder. This is done by allocating 30% of the dataset to model testing as referred to in Section 5.4.2. The evaluation of the model is based on the confusion matrices and the ROC plots. As indicated in Table 4.2 and explained in detail in Section 4.3.5, the strategy for the demonstration of the results is based on dividing the whole set of data points into 3 subsets:



(a) Spoofed device AR_1

(b) Spoofed device RZ_1

(c) Spoofed device RZ_2

(d) Spoofed device RZ_3

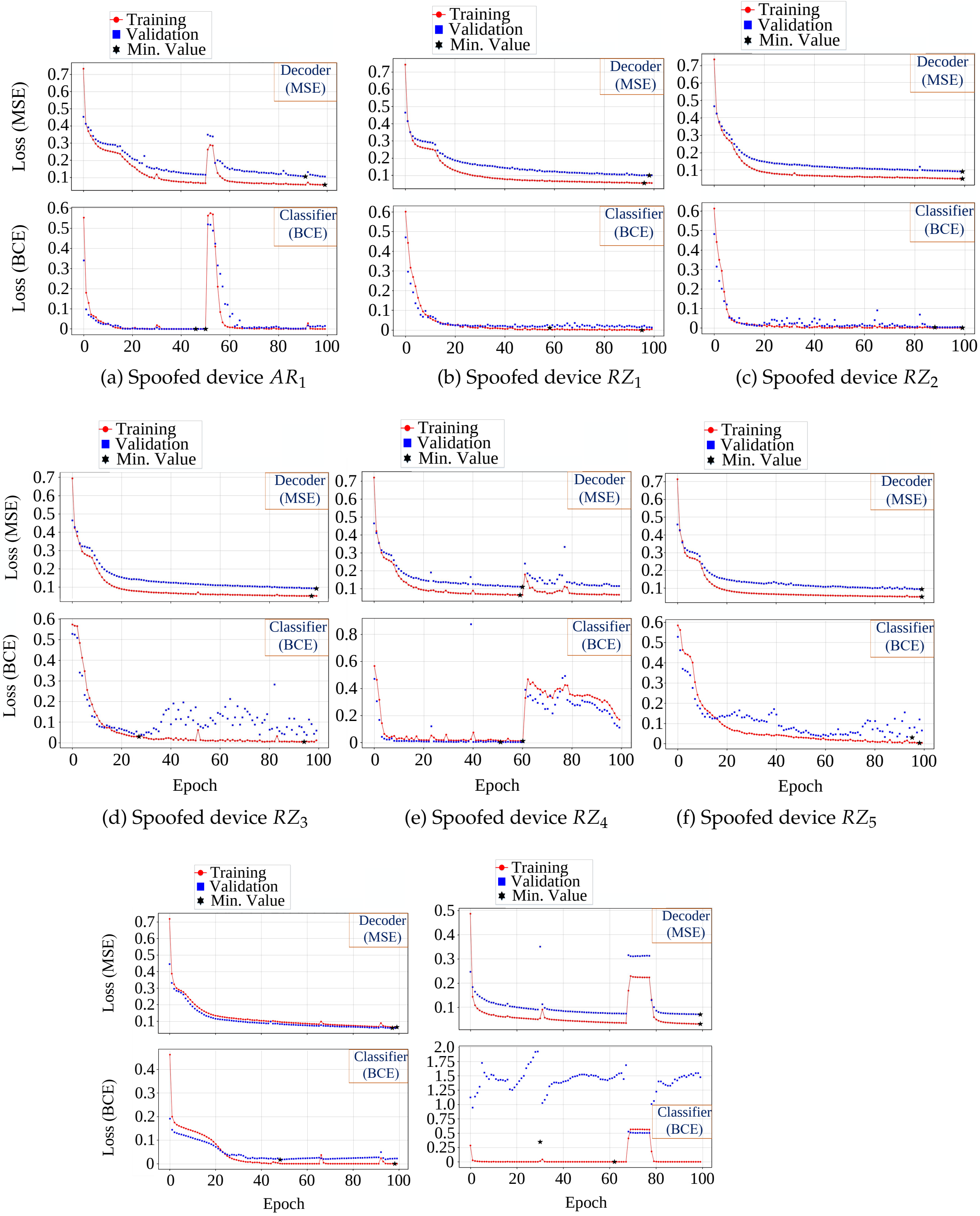
(e) Spoofed device RZ_4

(f) Spoofed device RZ_5

(g) Spoofed device TI_1

(h) Spoofed device TI_2

Figure 5.5: Training and validation losses for all eight scenarios listed in Table 4.2 for the RAW dataset fed to the LSTM based autoencoder model.



(a) Spoofed device AR_1

(b) Spoofed device RZ_1

(c) Spoofed device RZ_2

(d) Spoofed device RZ_3

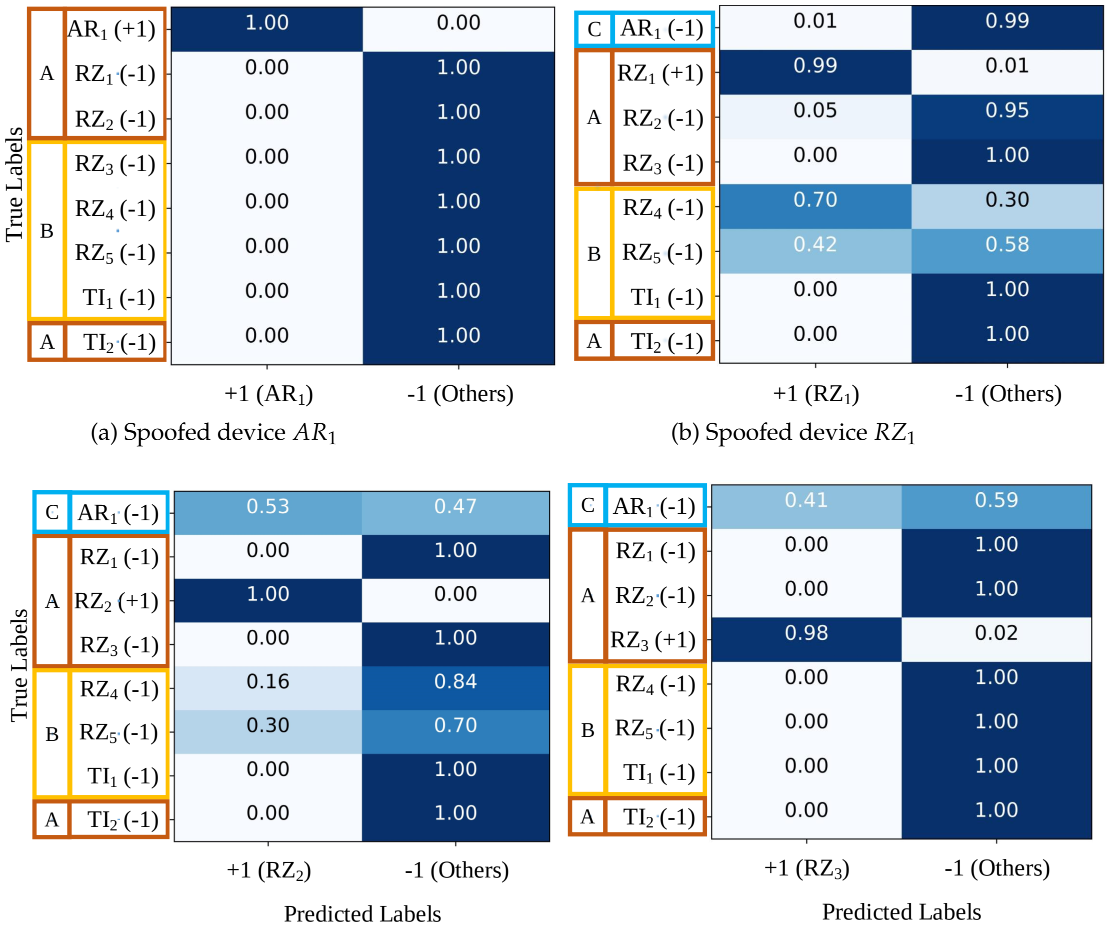
(e) Spoofed device RZ_4

(f) Spoofed device RZ_5

(g) Spoofed device TI_1

(h) Spoofed device TI_2

Figure 5.6: Training and validation losses for all eight scenarios listed in Table 4.2 for the DDWT dataset fed to the LSTM based autoencoder model.



(a) Spoofed device AR_1

(b) Spoofed device RZ_1

(c) Spoofed device RZ_2

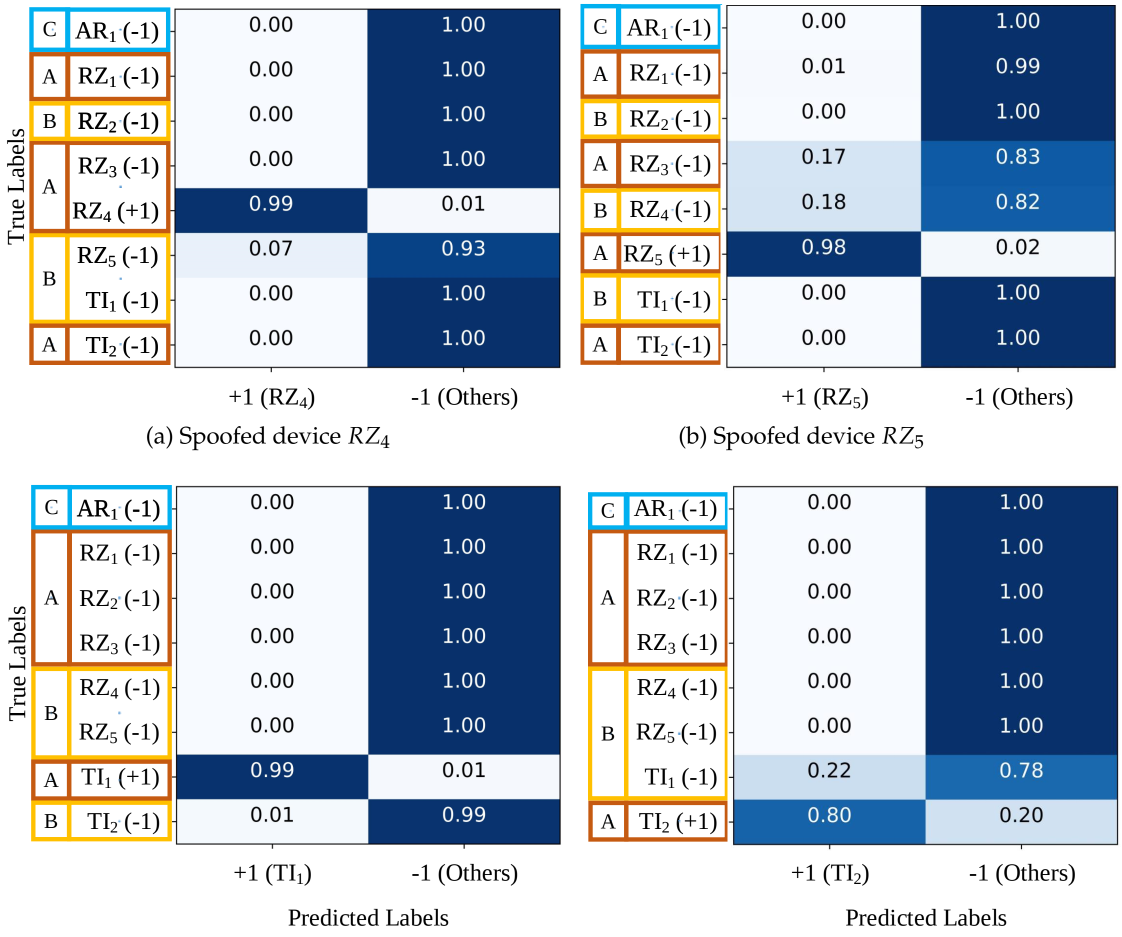
(d) Spoofed device RZ_3

Figure 5.7: Confusion matrix for the first group of four scenarios of Table 4.2 for the RAW dataset fed to the LSTM based autoencoder model.

subset A, subset B, and subset C to evaluate the effect of different types of devices on the model performance.

Confusion matrices

As the first step for the evaluation of the proposed classifier, the confusion matrices of the testing data points are computed and shown in Figs. 5.7, 5.8, 5.9, and 5.10 for the LSTM based autoencoder model. It is important to note that before testing, the classifier model is trained for each of the 8 scenarios of Table 4.2 for both the RAW (original) and the DDWT (wavelet-transformed) datasets. As shown, the results are compared from two points of views: first, on the type of dataset (RAW or DDWT) and second, on the device group (A, B , or C).



(a) Spoofed device RZ_4

(b) Spoofed device RZ_5

(c) Spoofed device TI_1

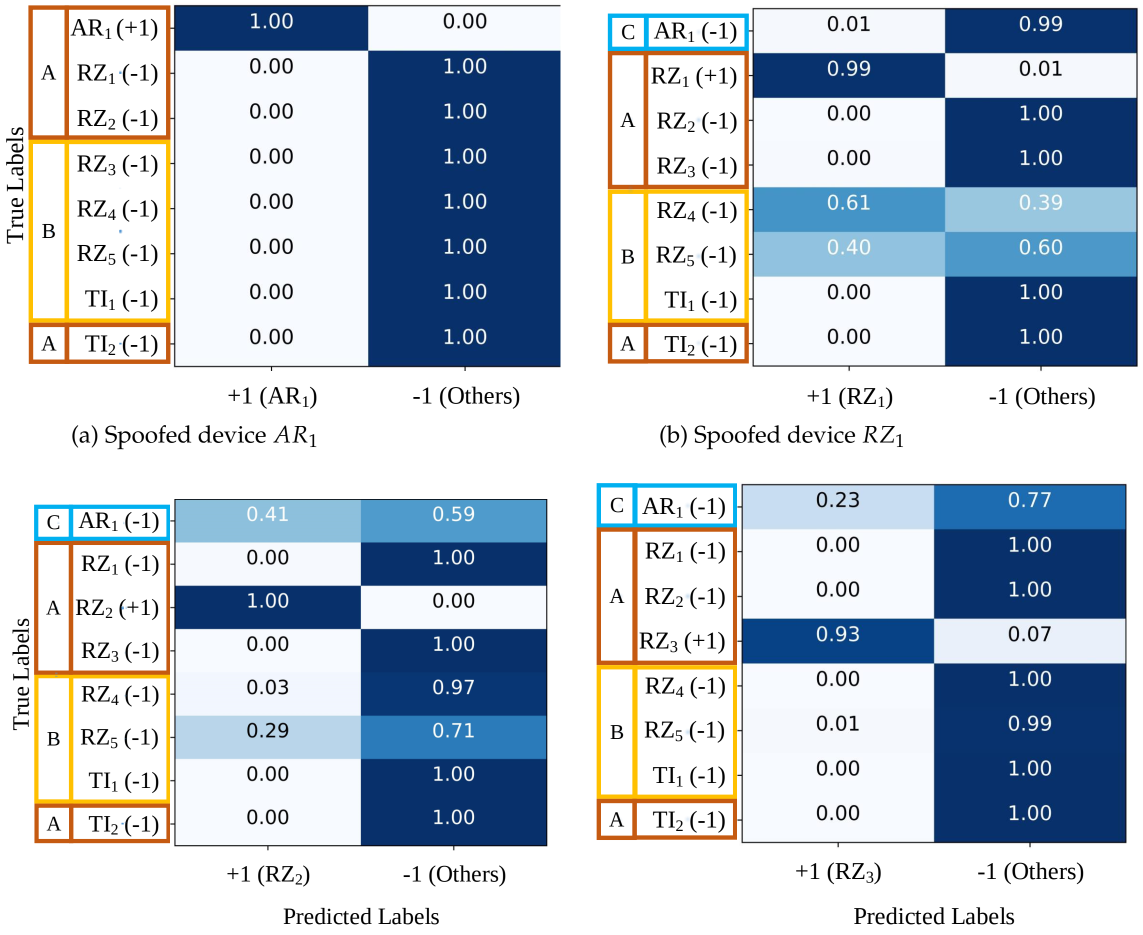
(d) Spoofed device TI_2

Figure 5.8: Confusion matrix for the second group of four scenarios of Table 4.2 for the RAW dataset fed to the LSTM based autoencoder model.

The classification rates for groups A, B, and C of Table 4.2 shown in Figs. 5.7, 5.8, 5.9, and 5.10, are summarized for each scenario in Table 5.3 with the RAW and DDWT datasets.

For instance, the classification rate in scenario 2 and group A of Table 5.3 (a) is larger than or equal to 0.95, which means that the model trained for RZ_1, using the RAW dataset, leads to a classification rate higher than 0.95 for the testing data points of group A.

As seen in Table 5.3, the minimum classification rates for group A are 0.80 and 0.84, for the RAW and DDWT datasets, respectively. The minimum classification rates for group B are 0.30 and 0.39, which are much lower than for group A, for corresponding datasets. This means that the model has a better performance for discrimination of the devices from group A than group B, due to the fact that it has seen them during the training. For group C, the minimum classification rates of the RAW and DDWT datasets are equal to 0.47 and 0.59, respectively. A similar analysis can be applied to all other rows of classification rates



(a) Spoofed device AR_1

(b) Spoofed device RZ_1

(c) Spoofed device RZ_2

(d) Spoofed device RZ_3

Figure 5.9: Confusion matrix for the first group of four scenarios of Table 4.2 for the DDWT dataset fed to the LSTM based autoencoder model.

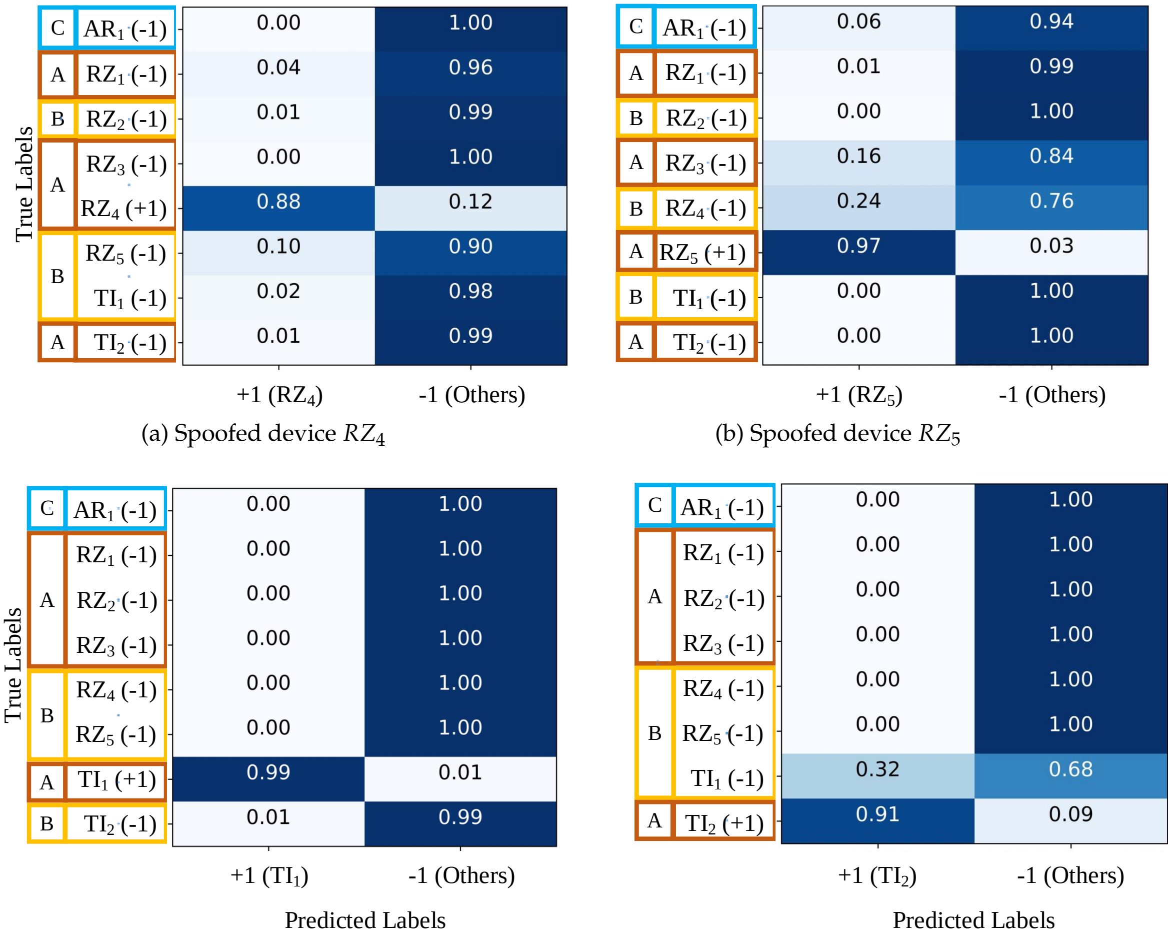
Table 5.3: Confusion matrix summary for the LSTM based autoencoder model using the (a) RAW and (b) DDWT datasets fed to the LSTM based autoencoder model.

(a)

	Positive Device	**Members Classification Rate**		
		A	***B***	***C***
1	AR_1	1.00	1.00	-
2	RZ_1	≥ 0.95	≥ 0.30	0.99
3	RZ_2	1.00	≥ 0.70	0.47
4	RZ_3	≥ 0.98	1.00	0.59
5	RZ_4	≥ 0.99	≥ 0.93	1.00
6	RZ_5	≥ 0.83	≥ 0.82	1.00
7	TI_1	≥ 0.99	0.99	1.00
8	TI_2	≥ 0.80	≥ 0.78	1.00

(b)

	Positive Device	**Members Classification Rate**		
		A	***B***	***C***
1	AR_1	1.00	1.00	-
2	RZ_1	≥ 0.99	≥ 0.39	0.95
3	RZ_2	1.00	≥ 0.71	0.59
4	RZ_3	≥ 0.93	≥ 0.99	0.77
5	RZ_4	≥ 0.88	≥ 0.90	1.00
6	RZ_5	≥ 0.84	≥ 0.76	0.94
7	TI_1	≥ 0.99	≥ 0.99	1.00
8	TI_2	≥ 0.91	≥ 0.68	1.00



(a) Spoofed device RZ_4

(b) Spoofed device RZ_5

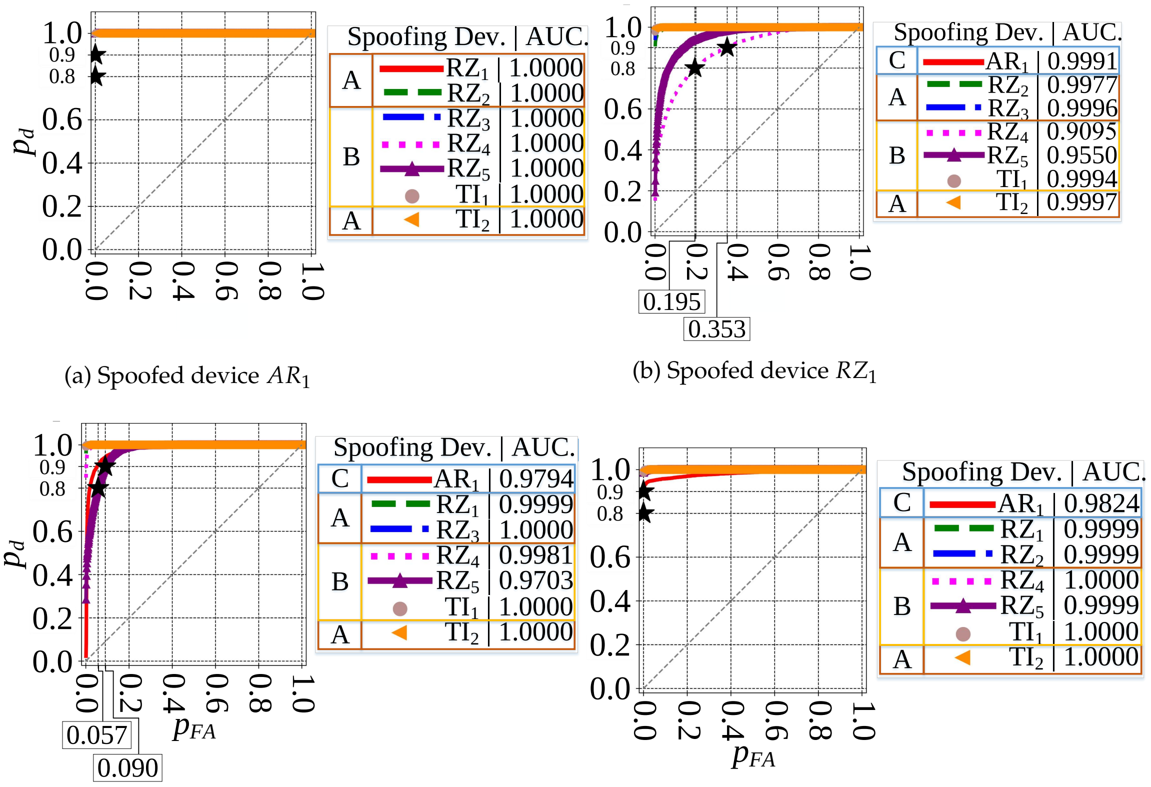
(c) Spoofed device TI_1

(d) Spoofed device TI_2

Figure 5.10: Confusion matrix for the second group of four scenarios of Table 4.2 for the DDWT dataset fed to the LSTM based autoencoder model.

for the RAW and DDWT datasets in Table 5.3, for groups A, B, and C. The minimum correct classification rate using the RAW dataset in Table 5.3 (a) and Fig. 5.7 (b) belongs to device RZ_4 (0.30) obtained from the model trained based on the positive device RZ_1. For the DDWT dataset, the minimum correct classification rate corresponds to device RZ_4 (0.39) obtained from the model trained based on positive device RZ_1, shown in Table 5.3 (b) and Fig. 5.9 (b).

Comparison between the model performance for the RAW and DDWT datasets Although the classification rate for rogue device TI_1 and the model of positive device TI_2 did not improve using the DDWT dataset, by feeding the mode with this dataset we could increase the classification rate of the worst case scenario (for group B and positive device RZ_1 in Table 5.3) from 0.3 to 0.39, using the DDWT dataset. Therefore, comparing Tables 5.3 (a) and 5.3 (b), in some cases, using the DDWT dataset can improve the minimum classification rates of the models presented in this chapter. For instance, the minimum classification rate for group



(a) Spoofed device AR_1

(b) Spoofed device RZ_1

(c) Spoofed device RZ_2

(d) Spoofed device RZ_3

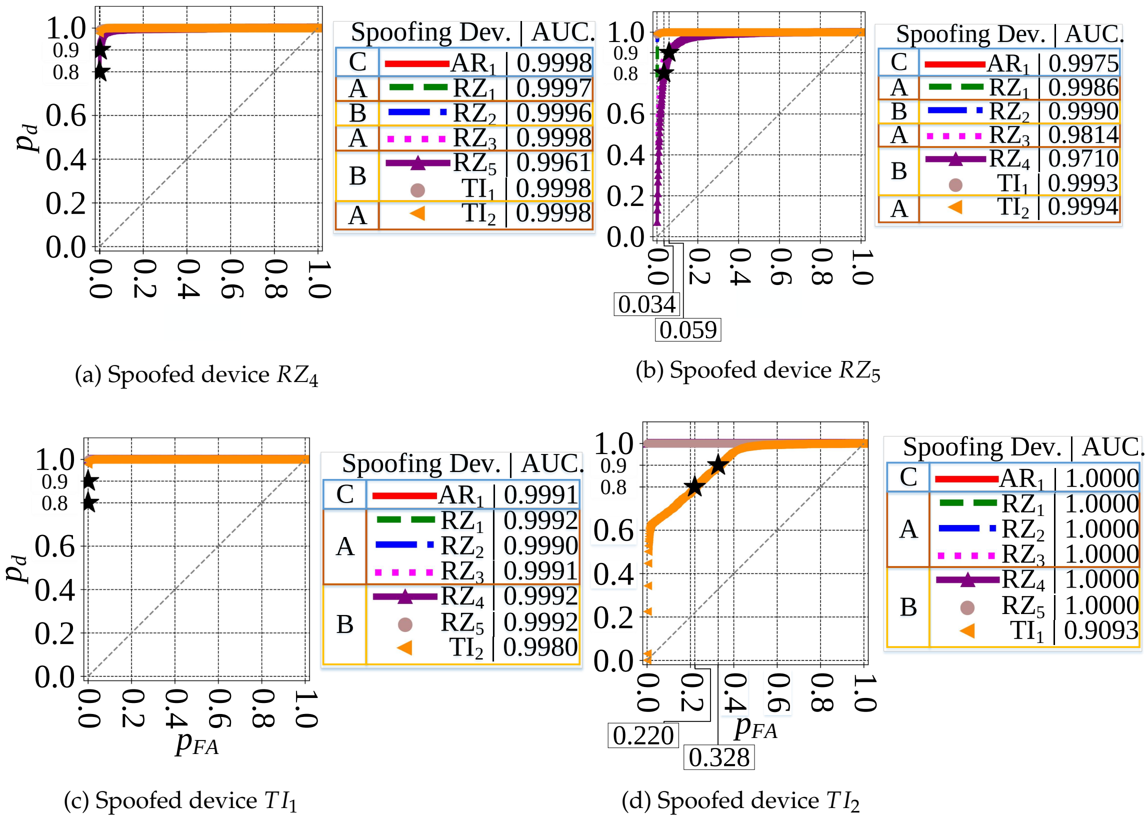
Figure 5.11: ROC plot for first group of four scenarios of Table 4.2 for the RAW dataset fed to the LSTM based autoencoder model.

C and positive device RZ_2 has increased from 0.47 to 0.59, using the DDWT dataset, shown in Tables 5.3 (a) and 5.3 (b). Using the DDWT dataset, or a combination of both datasets is a good solution to improve the suggested models' worst classification rates.

Receiver operating characteristics

As shown in Section 4.3.5, the resulting ROC plots for the LSTM based autoencoder model are shown in Figs. 5.11, 5.12, 5.13, and 5.14. For each ROC plot, the area under the curve (AUC), is given in the legend. Higher values closer to 1.0 illustrate better performances of the trained models in the case of acceptance of an authorized device and rejection of a rogue device [106]. The correct classification rate of a target/authorized/spoofed device, is characterized by the detection probability p_d, and its corresponding misclassification rate of rogue/unauthorized/spoofing device by the false alarm probability p_{FA}.

Tables 5.4 and 5.5 summarize the worst results obtained with the A, B, and C sets of devices using the RAW and DDWT datasets. For instance, in the second scenario, feeding



(a) Spoofed device RZ_4

(b) Spoofed device RZ_5

(c) Spoofed device TI_1

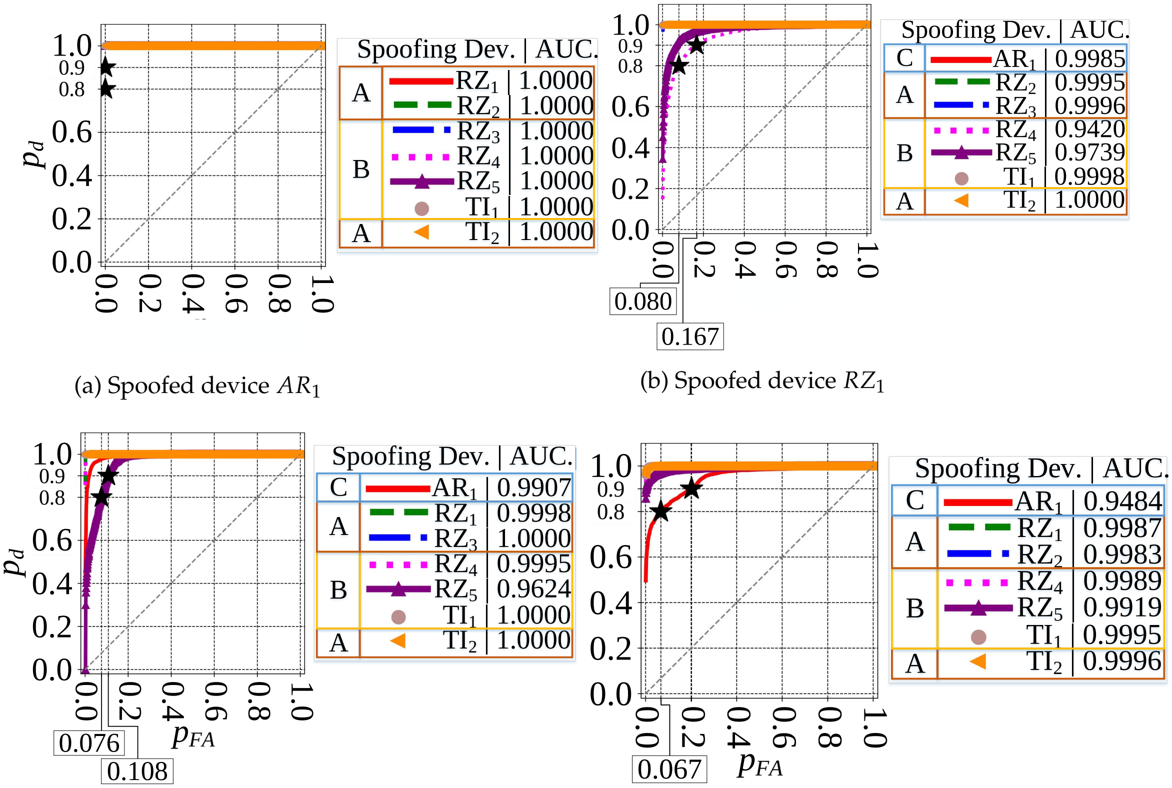
(d) Spoofed device TI_2

Figure 5.12: ROC plot for second group of four scenarios of Table 4.2 for the RAW dataset fed to the LSTM based autoencoder model.

the model trained for positive device RZ_1 with the RAW dataset in Fig. 5.11 (b) and Table 5.4, the range of correct classification rate (p_d) for the values of p_{FA} in $[0.01, 1.0]$ is almost equal to 1.0, corresponding to group A of testing devices. As mentioned in Table 4.2, this group of devices are also those used in training. For group A and the RAW dataset, the minimum AUC_{min} belongs to the sixth scenario (0.9814) and the minimum AUC_{max} is equal to 0.9992. Focusing on the same group, the average area under the curve (AUC_{avg}) for the first scenario and the RAW dataset is equal to 1.0000, the minimum AUC_{avg} for all scenarios is 0.9931.

Looking at the same plot (Fig. 5.11 (b)) and the same scenario in Table 5.4, group B gives p_d values higher than 0.9 for p_{FA} in the range $[0.353, 1.0]$ for the same RAW dataset. The values of minimum AUC_{min} and minimum AUC_{max} for group B are 0.9093 for the scenario 8 and 0.9992 for scenario 7, respectively. AUC_{avg} is higher than 0.9546 for all scenarios.

Finally, for group C, the p_d is equal to 1.0 for values of p_{FA} higher than 0.057 in all scenarios of Table 5.4. For this group, for the area under the curve we have $0.9794 \leq AUC_{min} \leq 1.0000$ and $0.9794 \leq AUC_{max} \leq 1.0000$. For such a group, the average area under the curve has the same range as AUC_{min} and AUC_{max}.



(a) Spoofed device AR_1

(b) Spoofed device RZ_1

(c) Spoofed device RZ_2

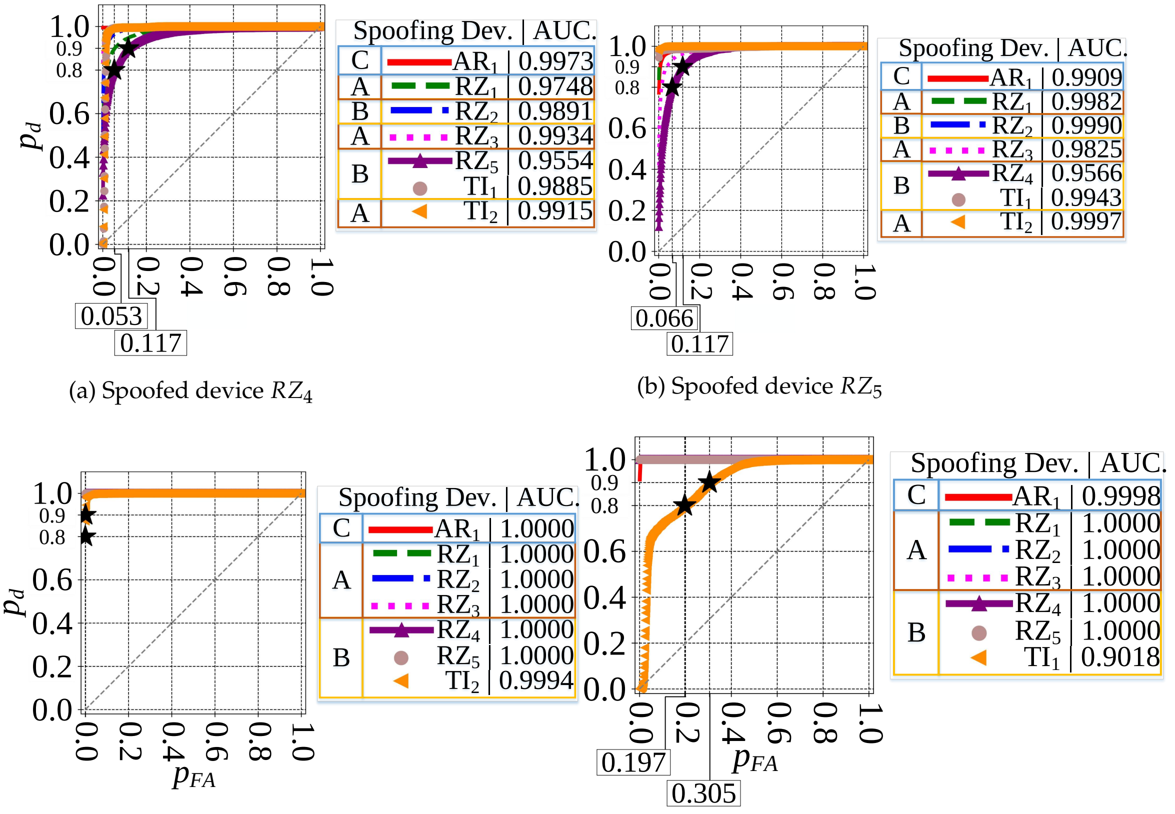
(d) Spoofed device RZ_3

Figure 5.13: ROC plot for first group of four scenarios of Table 4.2 for the DDWT dataset fed to the LSTM based autoencoder model.

The best AUC_{avg} (and the best model performance) corresponding to the RAW dataset relates to group A with the minimum $AUC_{avg} = 0.9931$, and the worst case scenario (resulting in the worst model performance) for this type of dataset is related to group B with the minimum $AUC_{avg} = 0.9546$.

The summary of all resulted for the RAW and DDWT, and groups A, B, and C are presented in Tables 5.4 and 5.5. Focusing on the DDWT dataset, the best performance for this model goes to group A with the minimum $AUC_{avg} = 0.9866$, and the worst model performance is corresponding to group B with the minimum $AUC_{avg} = 0.9673$.

Comparison between the model performance for the RAW and DDWT datasets Although the p_d for a specific range of p_{FA} for devices from group A in Table 5.4 for the RAW dataset shows a better performance compared to the same scenarios in Table 5.5 for the DDWT dataset, the ROC plots for groups B and C show the advantage of the DDWT dataset over the RAW dataset. For example, in the second scenario (device RZ_1), using the DDWT dataset improves the worst p_{FA} range from $[0.353, 1.0]$ to $[0.167, 1.0]$ for the same correct classifica-



(a) Spoofed device RZ_4

(b) Spoofed device RZ_5

(c) Spoofed device TI_1

(d) Spoofed device TI_2

Figure 5.14: ROC plot for second group of four scenarios of Table 4.2 for the DDWT dataset fed to the LSTM based autoencoder model.

tion rate p_d in device group B; and this improvement in the case of group C is from $[0.057, 1.0]$ related to the RAW dataset to $[0.02, 1.0]$ in DDWT dataset. Therefore, decreasing the lower threshold of p_{FA} will result in better performance of the model through achieving higher values of correct classification rates (p_d) by tolerating lower values of false alarm probability (p_{FA}). For the comparison of the minimum AUC_{avg} corresponding to all scenarios of group A, it is 0.9931 for the RAW dataset and 0.9935 for the DDWT dataset, respectively. For group B, the DDWT dataset shows a better performance ($0.9546 < 0.9673$), and the RAW dataset presents an advantage over the DDWT dataset for group C ($0.9794 > 0.9484$). Based on this fact, it may be concluded that using the RAW dataset better performance can be achieved by the model. But, if the focus is on the worst case scenarios, as mentioned before, in the second scenario (device RZ_1), using the DDWT dataset improves the worst p_{FA} range from $[0.3531.0]$ to $[0.167, 1.0]$ for $p_d \geq 0.9$, and AUC_{avg} from 0.9546 to 0.9719, in device group B. Therefore, although the comparison of confusion matrix results shows a better classification rate using the RAW dataset, the ROC plots of Figs. 5.13 and 5.14 compared to Figs. 5.11 and 5.12 determine the better performances using the DDWT dataset for the cases where the classifier *tolerates* larger false alarm probability values p_{FA}. Therefore, combining both datasets,

Table 5.4: ROC plots summary for the RAW dataset fed to the LSTM based autoencoder model.

Scenarios	Positive Dev.	A				B			
		p_d	$AUC_{\min}$	$AUC_{\max}$	AUC_{avg}	p_d	$AUC_{\min}$	$AUC_{\max}$	AUC_{avg}
1	AR_1	$\sim 1.0\ for\ p_{fa} \in [0, 1]$	1.0000	1.0000	1.0000	$\sim 1.0\ for\ p_{fa} \in [0, 1]$	1.0000	1.0000	1.0000
2	RZ_1	$\geq 0.9\ for\ p_{fa} \in [0.01, 1]$	0.9977	0.9997	0.9990	$\geq 0.9\ for\ p_{fa} \in [0.353, 1]$	0.9095	0.9994	0.9546
3	RZ_2	$\sim 1.0\ for\ p_{fa} \in [0, 1]$	0.9999	1.0000	0.9999	$\geq 0.9\ for\ p_{fa} \in [0.09, 1]$	0.9703	1.0000	0.9895
4	RZ_3	$\geq 0.95\ for\ p_{fa} \in [0.01, 1]$	0.9999	1.0000	0.9999	$\geq 0.9\ for\ p_{fa} \in [0.09, 1]$	0.9999	1.0000	0.9999
5	RZ_4	$\geq 0.95\ for\ p_{fa} \in [0.01, 1]$	0.9997	0.9998	0.9998	$\geq 0.9\ for\ p_{fa} \in [0.01, 1]$	0.9961	0.9998	0.9985
6	RZ_5	$\geq 0.95\ for\ p_{fa} \in [0.01, 1]$	0.9814	0.9994	0.9931	$\geq 0.9\ for\ p_{fa} \in [0.059, 1]$	0.9710	0.9993	0.9898
7	TI_1	$\geq 0.95\ for\ p_{fa} \in [0.01, 1]$	0.9990	0.9992	0.9991	$\geq 0.95\ for\ p_{fa} \in [0.01, 1]$	0.9980	0.9992	0.9992
8	TI_2	$\sim 1.0\ for\ p_{fa} \in [0, 1]$	1.0000	1.0000	1.0000	$\geq 0.9\ for\ p_{fa} \in [0.328, 1]$	0.9093	1.0000	0.9988

Scenarios	Positive Dev.	C			
		p_d	$AUC_{\min}$	$AUC_{\max}$	AUC_{avg}
1	AR_1	-	-	-	-
2	RZ_1	$\sim 1.0\ for\ p_{fa} \in [0, 1]$	0.9991	0.9991	0.9991
3	RZ_2	$\geq 0.9\ for\ p_{fa} \in [0.057, 1]$	0.9794	0.9794	0.9794
4	RZ_3	$\geq 0.9\ for\ p_{fa} \in [0.01, 1]$	0.9824	0.9824	0.9824
5	RZ_4	$\geq 0.95\ for\ p_{fa} \in [0.01, 1]$	0.9998	0.9998	0.9998
6	RZ_5	$\geq 0.9\ for\ p_{fa} \in [0.01, 1]$	0.9975	0.9975	0.9975
7	TI_1	$\sim 1.0\ for\ p_{fa} \in [0, 1]$	0.9991	0.9991	0.9991
8	TI_2	$\sim 1.0\ for\ p_{fa} \in [0, 1]$	1.0000	1.0000	1.0000

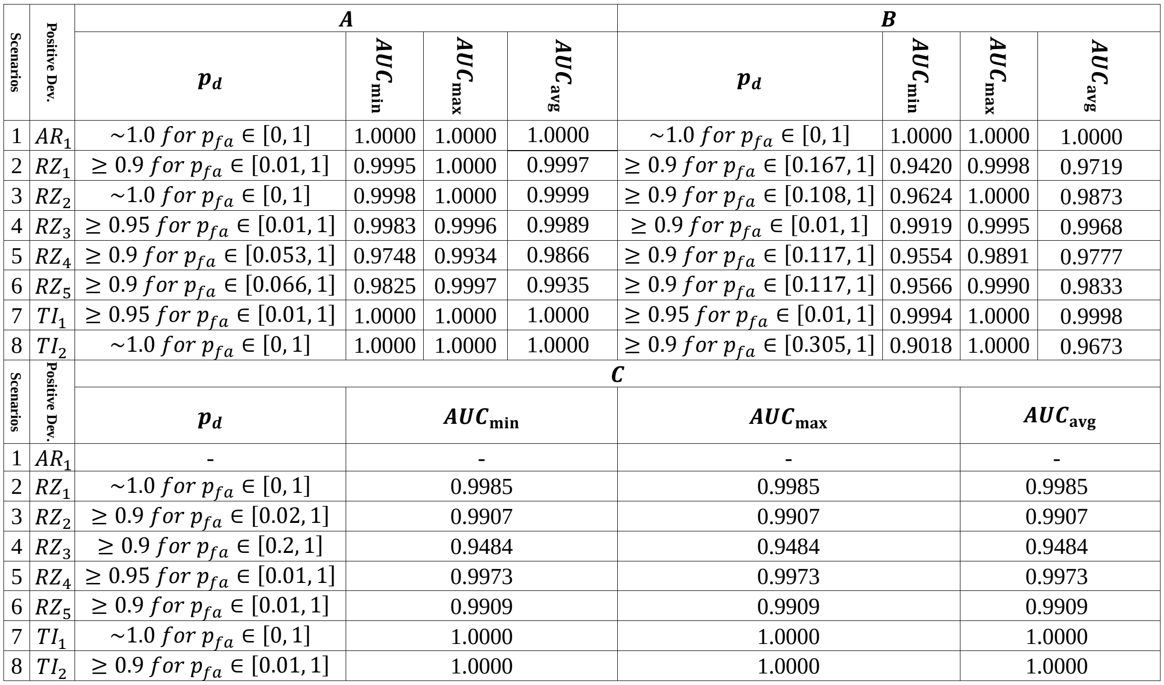
we can achieve higher AUC values, lower p_{FA}, and also, improve the model performance for the worst case scenarios.

5.6 Classifier performance comparison

In this section, after testing the performance of the proposed methodology on real datasets, the performance of the proposed classification method is compared with other classifiers recently presented in the literature. As some special considerations are taken into account, it is worth mentioning them.

- As in the testing process, all testing steps are repeated on the RAW and DDWT datasets.
- The main strategy of this work is based on the separation of devices based on Table 4.2, resulting in training the model for a different authorized/spoofed unit.
- During the training of the model, two sets of devices are needed. One set (including a device) plays the role of a spoofed device and the other set (consisting of at least one device) is used as a spoofing set.
- As noticed in the testing section, the device dataset is composed of three groups: group A (devices which are used in the training phase), group B (devices which are not used in the training, but for which one or more members of their family are), and group C

Table 5.5: ROC plots summary for the DDWT dataset fed to the LSTM based autoencoder model.



Scenarios	Positive Dev.	A				B			
		$\boldsymbol{p_d}$	$\boldsymbol{AUC}_{\mathbf{min}}$	$\boldsymbol{AUC}_{\mathbf{max}}$	$\boldsymbol{AUC}_{\mathbf{avg}}$	$\boldsymbol{p_d}$	$\boldsymbol{AUC}_{\mathbf{min}}$	$\boldsymbol{AUC}_{\mathbf{max}}$	$\boldsymbol{AUC}_{\mathbf{avg}}$
1	AR_1	$\sim 1.0\ for\ p_{fa} \in [0, 1]$	1.0000	1.0000	1.0000	$\sim 1.0\ for\ p_{fa} \in [0, 1]$	1.0000	1.0000	1.0000
2	RZ_1	$\geq 0.9\ for\ p_{fa} \in [0.01, 1]$	0.9995	1.0000	0.9997	$\geq 0.9\ for\ p_{fa} \in [0.167, 1]$	0.9420	0.9998	0.9719
3	RZ_2	$\sim 1.0\ for\ p_{fa} \in [0, 1]$	0.9998	1.0000	0.9999	$\geq 0.9\ for\ p_{fa} \in [0.108, 1]$	0.9624	1.0000	0.9873
4	RZ_3	$\geq 0.95\ for\ p_{fa} \in [0.01, 1]$	0.9983	0.9996	0.9989	$\geq 0.9\ for\ p_{fa} \in [0.01, 1]$	0.9919	0.9995	0.9968
5	RZ_4	$\geq 0.9\ for\ p_{fa} \in [0.053, 1]$	0.9748	0.9934	0.9866	$\geq 0.9\ for\ p_{fa} \in [0.117, 1]$	0.9554	0.9891	0.9777
6	RZ_5	$\geq 0.9\ for\ p_{fa} \in [0.066, 1]$	0.9825	0.9997	0.9935	$\geq 0.9\ for\ p_{fa} \in [0.117, 1]$	0.9566	0.9990	0.9833
7	TI_1	$\geq 0.95\ for\ p_{fa} \in [0.01, 1]$	1.0000	1.0000	1.0000	$\geq 0.95\ for\ p_{fa} \in [0.01, 1]$	0.9994	1.0000	0.9998
8	TI_2	$\sim 1.0\ for\ p_{fa} \in [0, 1]$	1.0000	1.0000	1.0000	$\geq 0.9\ for\ p_{fa} \in [0.305, 1]$	0.9018	1.0000	0.9673

Scenarios	Positive Dev.	C			
		$\boldsymbol{p_d}$	$\boldsymbol{AUC}_{\mathbf{min}}$	$\boldsymbol{AUC}_{\mathbf{max}}$	$\boldsymbol{AUC}_{\mathbf{avg}}$
1	AR_1	-	-	-	-
2	RZ_1	$\sim 1.0\ for\ p_{fa} \in [0, 1]$	0.9985	0.9985	0.9985
3	RZ_2	$\geq 0.9\ for\ p_{fa} \in [0.02, 1]$	0.9907	0.9907	0.9907
4	RZ_3	$\geq 0.9\ for\ p_{fa} \in [0.2, 1]$	0.9484	0.9484	0.9484
5	RZ_4	$\geq 0.95\ for\ p_{fa} \in [0.01, 1]$	0.9973	0.9973	0.9973
6	RZ_5	$\geq 0.9\ for\ p_{fa} \in [0.01, 1]$	0.9909	0.9909	0.9909
7	TI_1	$\sim 1.0\ for\ p_{fa} \in [0, 1]$	1.0000	1.0000	1.0000
8	TI_2	$\geq 0.9\ for\ p_{fa} \in [0.01, 1]$	1.0000	1.0000	1.0000

Table 5.6: Comparison of methodology presented in Chapters 4 (autoencoder model) and 5 (LSTM based autoencoder model), with approaches in [35] and [97], using the RAW dataset.

	RAW Dataset					
	A			B		
	AUC_{min}	AUC_{max}	AUC_{avg}	AUC_{min}	AUC_{max}	AUC_{avg}
Autoencoder (Ch. 4)	0.9947	1.0000	0.9996	0.3926	1.0000	0.9686
LSTM autoencoder (Ch. 5)	0.9814	1.0000	0.9989	0.9093	1.0000	0.9916
Merchant et al. [35]	0.6216	1.0000	0.9840	0.0580	1.0000	0.8830
Ramsey et al. [108]	0.2404	1.0000	0.9634	0.0108	1.0000	0.9355

	C		
	AUC_{min}	AUC_{max}	AUC_{avg}
Autoencoder (Ch. 4)	0.9430	1.0000	0.9830
LSTM autoencoder (Ch. 5)	0.9794	1.0000	0.9939
Merchant et al. [35]	0.9532	1.0000	0.9919
Ramsey et al. [108]	0.1976	1.0000	0.8803

Table 5.7: Comparison of methodology presented in Chapters 4 (autoencoder model) and 5 (LSTM based autoencoder model), with approaches in [35] and [97], using the DDWT dataset.

	DDWT Dataset					
	A			B		
	AUC_{min}	AUC_{max}	AUC_{avg}	AUC_{min}	AUC_{max}	AUC_{avg}
Autoencoder (Ch. 4)	0.9901	1.0000	0.9989	0.8115	1.0000	0.9792
LSTM autoencoder (Ch. 5)	0.9748	1.0000	0.9973	0.9018	1.0000	0.9705
Merchant et al. [35]	0.3964	1.0000	0.9745	0.0602	1.0000	0.8470
Dubendorfer et al. [97]	0.0000	1.0000	0.7474	0.0000	1.0000	0.7356

	C		
	AUC_{min}	AUC_{max}	AUC_{avg}
Autoencoder (Ch. 4)	0.8997	1.0000	0.9711
LSTM autoencoder (Ch. 5)	0.9484	1.0000	0.9894
Merchant et al. [35]	0.9184	1.0000	0.9879
Dubendorfer et al. [97]	0.0000	1.0000	0.6992

(devices for which neither they nor their other family members are used in the training phase).

Referring to this short summarization, three cases of comparison are done with respect to their approach to each mentioned criteria above.

AUC performance evaluation Why do we use AUC for the comparison? What does it mean if a methodology presents a higher value of AUC? As in Chapter 4 (autoencoder model), the area under the curve (AUC) is used for comparing the different classifiers or methodologies, presenting the area under the ROC curve. When a classifier presents a higher value of AUC compared to another one, it can be inferred it has a better performance in discrimination of rogue and authorized classes. The summary of the AUC results presented in this section are reported in Tables 5.6 and 5.7, respectively.

5.6.1 Comparison with autoencoder model presented in Chapter 4

In Chapter 4 (autoencoder model), an autoencoder based classifier for discrimination of rogue and authorized devices is proposed. Using the model designed in Fig. 4.7 and shown in Table 4.1, comparison with Chapter 4 (autoencoder model) is done based on the 8 strategies of Table 4.2 with the RAW and DDWT datasets. The results obtained from this model for the RAW and DDWT datasets are shown in Tables 5.6 and 5.7, the AUC_{max} for both methodologies in Chapters 4 (autoencoder model) and 5 (LSTM based autoencoder model) is equal to 1.0000. This value is the same for groups A, B, and C, related to the RAW and DDWT dasets. On the other hand, the AUC_{min} and AUC_{avg} corresponding to Chapter 4 (autoencoder model) shows a better performance for group A. For example, let us focus on the RAW dataset. Although the performance of the model for group A is a little bit better than Chapter 4 (autoencoder model), in Chapter 5 (LSTM based autoencoder model), for group B, AUC_{min} has almost doubled (increased from 0.3926 to 0.9093), and AUC_{avg} has increased from 0.9686 to 0.9916. Finally, in the case of group C, AUC_{min} and AUC_{avg} related to Chapter 5 (LSTM based autoencoder model) show a better performance, since they could increase from 0.9430 to 0.9794. In brief, for the RAW dataset, the AUC_{min} values for Chapters 4 (autoencoder model) and 5 (LSTM based autoencoder model) are 0.3926 and 0.9093, respectively, which shows the better performance of the presented method in Chapter 5 (LSTM based autoencoder model).

Using the DDWT dataset, we obtain similar results as for the RAW dataset. Although the performance of the model for group A is slightly better for Chapter 4 (autoencoder model), in Chapter 5 (LSTM based autoencoder model), for group B, AUC_{min} has increased from 0.8115 to 0.9018, compared to Chapter 4 (autoencoder model). Focusing on group C, AUC_{min} for Chapter 5 (LSTM based autoencoder model) shows a better performance ($AUC_{min} = 0.9484$) compared to Chapter 4 (autoencoder model) ($AUC_{min} = 0.8997$). As a summary, the minimum AUC_{min} in Chapter 5 (LSTM based autoencoder model) has increased from 0.8115 to 0.9018, compared to Chapter 4 (autoencoder model), which is the proof for better performance of the LSTM based autoencoder model presented in Chapter 5.

5.6.2 Comparison with reference [35]

In [35], Merchant et al. introduced a discrimination system (a convolutional based neural network) to distinguish between the different devices, as shown in Table 4.7. Implementing the model in this table, the comparison with [35] is done using the RAW and DDWT datasets with focus on all 8 strategies of Table 4.2.

In Tables 5.6 and 5.7, the $AUC_{max} = 1.0000$ for Chapter 5 (LSTM based autoencoder model) and also the results obtained from [35], related to groups A, B, and C of both the RAW and DDWT dasets. The AUC_{min} and AUC_{avg} for Chapter 5 (LSTM based autoencoder model) for the RAW dataset and group A have increased from 0.6216 to 0.9814, and from 0.9840 to 0.9989, respectively. In the case of group B from the RAW dataset, Chapter 5 (LSTM based autoencoder model) could improve AUC_{min} and AUC_{avg} from 0.0580 to 0.9093 (more than doubled compared to Chapter 4 (autoencoder model)), and from 0.8830 to 0.9916, respectively. Finally, AUC_{min} and AUC_{avg} related to [35] (0.9532 and 0.9919) are not as high as those in Chapter 5 (LSTM based autoencoder model) (0.9794 and 0.9939).

Let us now focus on the DDWT dataset. In the case of group A, AUC_{min} and AUC_{avg}

improved from 0.3964 to 0.9748 (but not as much as Chapter 4 (autoencoder model)), and from 0.9745 to 0.9973, respectively. The same factors for group B increased from 0.0602 to 0.9018 (higher than Chapter 4 (autoencoder model)), and from 0.8470 to 0.9705, respectively. Eventually, group C proves that AUC_{min} and AUC_{avg} related to Chapter 5 (LSTM based autoencoder model) (0.9484 and 0.9879) has a better condition compared to [35] (0.9184 and 0.9879).

5.6.3 Comparison with reference [108]

For the third comparison, the method in [108] is compared to that of Chapter 5 (LSTM based autoencoder model). In this work, Ramsey et al. used the time domain features of the ZigBee devices, such as $a(t)$, $\varphi(t)$, and $f(t)$ and their statistical features: σ^2, γ, and κ. Using the multi discriminant analysis (MDA) model, the results for all 8 strategies of Table 4.2 are summarized in Table 5.6. Table 5.6 shows the AUC_{max} for both methods in Chapter 5 (LSTM based autoencoder model) and [108], for all the groups A, B, and C of RAW dataset, equal to 1.0000. In Chapter 5 (LSTM based autoencoder model), the AUC_{min} and AUC_{avg} for group A has increased from 0.2404 to 0.9814, and from 0.9634 to 0.9989, respectively. Focusing on group B, the same factors have changed from 0.0108 to 0.9093 (which is a remarkable improvement compared to Chapter 4 (autoencoder model)), and from 0.9355 to 0.9916, respectively. For group C, AUC_{min} in Chapter 5 (LSTM based autoencoder model) and [108] are 0.9794 and 0.1976, respectively. As seen, the performance of an approach can easily be compromised by a complex dataset like one which is used in this thesis. In such a case, [108] could not reach an AUC_{min} higher than 0.0108 for group B.

5.6.4 Comparison with reference [97]

For the last comparison, the focus is on [97]. Dubendorfer et al. fed MDA by Radio Frequency Distinct Native Attributes (RF-DNA) dataset for discrimination purposes. In such a case, the wavelet transform of statistical features (variance as σ^2, skewness as γ, and kurthosis as κ of physical signal characteristics like amplitude ($a(t)$), phase ($\varphi(t)$), and instantaneous frequency ($f(t)$)) was used. In this section, the wavelet transform of the statistical features (σ^2, γ, and κ) of $a(t)$,$\varphi(t)$, and $f(t)$ of the RAW dataset in this thesis are fed to the MDA for all 8 strategies of Table 4.2. The summary of the results is depicted in Table 5.7. Based on this table, the AUC_{max} for both methodologies in Chapter 5 (LSTM based autoencoder model) and [97], for all the groups in the DDWT dataset is 1.0000. In Chapter 5 (LSTM based autoencoder model), using the DDWT dataset and group A, the AUC_{min} and AUC_{avg} have increased from 0.0000 to 0.9748, and from 0.7474 to 0.9973, respectively. For group B, where AUC_{min} and AUC_{avg} have increased from 0.0000 to 0.9018, and from 0.7356 to 0.9705, respectively. In the case of group C, the AUC_{min} in [97] is 0.0000, which is due to the complexity of the dataset, and it was improved to 0.9484 in Chapter 5 (LSTM based autoencoder model).

5.7 Conclusion

In this chapter, an approach based on deep learning has been presented for the rejection of unauthorized devices and to prevent them from having access to the network. The physical devices features are extracted from (time domain) RAW or from (time-scale domain) DDWT datasets and fed to a deep learning network using an LSTM layer. The main strategy of this work relies on the separation of target/authorized/spoofed devices from the group of rogue/unauthorized/spoofing ones, using the mechanism of training the model for each device and evaluating the trained model using seen/unseen devices. As observed, the classification rates for the rejection of rogue devices in worst and best cases for the area under the curve (AUC) of ROC plots are 0.9093 and 1.0000 when using the RAW dataset, and 0.9018 and 1.0000 using the DDWT datasets, respectively. These results are comparable to those reported in the literature, as shown in Section 5.6.

Conclusion

Summary of the thesis

The main goal of this Ph.D. is to propose a system for discrimination of authorized devices from rogue ones in ZigBee systems based on the IEEE 802.15.4 protocol. In other words, the network should be able to reject unauthorized/rogue devices from having access to the network, and at the same time, provide access for authorized ones with a high accuracy. Different researches have been done in this field to provide access granting systems in the past. However, the main two factors which were missed in the majority of these researches were high accuracy and the ability of the system to detect unseen devices. In this thesis, the main goal is on presenting a design which covers both factors. The proposed system has the ability to accept/reject seen or unseen devices (at the time of training of the model) with high accuracy.

After a problem definition in the introduction chapter, and a literature review of works done in Chapter 1, Chapter 2 presented the theoretical background on device classifiers investigated in this thesis (autoencoder, RNN, and LSTM), training procedure of the autoencoder and classifier models. In addition, Chapter 3 went over the different steps of dataset generation from the selected devices. As discussed in this chapter, the quality of captured signals was low due to the distortion and noise in the environment. Therefore, before using this kind of signal, an essential preprocessing procedure was done on the dataset to make it ready for classification purposes.

Chapter 4 presented the performance results for rogue device discrimination using the proposed autoencoder based classifier. With this autoencoder based model, the classification rate lead to an acceptable accuracy for new (unseen) devices. The proposed rogue device discrimination method compared favorably with other methods reported in the literature.

Chapter 5 addressed unauthorized device rejection using this time the long short-term memory (LSTM) based autoencoder. Using LSTM, the extracted time-dependency of input data points can be extracted for classification purposes. In such a manner, similar results as those presented in Chapter 4 were reported, for the original (RAW) and wavelet-transformed

(DDWT) datasets. These results are comparable to the results reported in the literature. However, it should be kept in mind that for these tests, the combination of seen and unseen devices by the model has been used which makes these results more reliable in real cases. As a conclusion, it is worth mentioning that the model complexity of the LSTM based autoencoder proposed in Chapter 5 is more than that of the autoencoder presented in Chapter 4. Therefore, there should always be a trade-off between the implementation complexity and performance accuracy of the model with respect to the available computational power of the system.

Contributions

The contributions that were presented in our thesis are as follows.

1. **Model design using deep neural networks** The special kind of deep learning model, based on autoencoders, leads to promising results for device classification. In this thesis, using an automated feature extraction structure implemented by an autoencoder deep neural network, essential features from each preamble are extracted. Later on, the extracted features are fed to a fully connected classifier to distinguish between the authorized and rogue devices. Also, adding a long-short term memory layer to the autoencoder structure, the time-dependency of the signal is extracted which is used as an important feature for increasing the accuracy in the neural networks classifier. In addition, with simultaneous training of the decoder and the classifier parts, it is guarantied that the extracted features are meaningful enough for the classifier.

2. **Designing a model for each device** We present a basic structure for the separation of one device from all other devices. This means that the goal of each model is to distinguish one device from all others, known as *one-vs-all* strategy. Consequently, the number of models in such a methodology is equal to the number of authorized devices. Through this strategy, instead of multi-class classification, we can assign a model to each authorized device. Therefore, the mission for each model is to discriminate a specific device, from all other devices in the world which try to copy the identity of this device. In such a case, if a model is responsible to learn a single and unique device, the accuracy will be higher compard to the multi-class case. Then, the selected strategy in this thesis is *one-vs-all*.

3. **Domain transformation** The analysis of classifiers is adapted from the time to the time-scale domain using wavelet transform. In such a case, the wavelet transformation can map the signal from the time domain to the time-scale domain. Using such a transformation, it is possible to extract some features from the signal that can be used

for classification. In this thesis, using the Haar wavelet transformation to extract the detail coefficients, the abrupt local changes of generated preambles (data points) by each device are detected to be used in rogue and authorized device discrimination.

4. **Testing the model with unseen devices** We test the model with a subset of devices which has never been seen by the model at the time of training. For such a purpose, before training, the dataset is divided into three subsets, training, validation, and testing. None of these three subsets have any data points in common. This is a main aspect at the time of testing the model with new unseen data points, which is not present in some of the works in the literature. Besides, the dataset which is fed to these models is divided into three parts, including devices which are seen/unseen in the training phase. Some of these unseen devices are from the family of training devices or completely new brands.

5. **Focusing on the time-dependency of data points** The time-dependency of preambles in the dataset is an important factor which can be used for extraction of high accuracy features, resulting in a high classification rate. After adding a long-short term memory layer to the autoencoder, which is responsible to extract the time-dependency features of the preamble, if we compare the results of the correct classification rates between the autoencoder structures with and without long-short term memory, we can see that the area under the curve for the worst classification case has increased remarkably, after adding this layer, which is a good factor which justifies its efficiency.

In [73], the strategy of *one-vs-all* is used to design an efficient convolutional neural network (CNN) for discrimination of authorized and unauthorized devices. The set of contributions 3 to 4 is the basis of the work presented in [10] and Chapter 4. In addition to the mentioned contributions, Chapter 5 and the work presented in [62] presents the benefit of adding an LSTM layer to the structure of the autoencoder. Referring to the results of Chapters 4 and 5, these contributions lead to an efficient approach for the detection and rejection of seen and also unseen devices from the same or different manufacturers.

At the time of writing this thesis, based on the LSTM based autoencoder design presented for classification purposes in Chapter 5, a conference paper with the title of "Wireless Device Authentication using LSTM based Autoencoders" has been presented at the 30th Biennial Symposium on Communications 2021 [62], and a journal paper entitled "Unauthorized Device Rejection in Wireless Communication Systems using Wavelet Transform and LSTM Based Autoencoders" is in progress.

Suggestions for future works

The complex structure of neural networks is a key barrier in the realization of a model for fast and high accuracy networks for the detection of devices in very large datasets. Moreover, failure of the model for detecting some special devices is an important factor which should be investigated more. Also, all procedures for signal acquisition in this project have been done in an isolated environment and the effect of the external noise was not taken into account. For future works, the following ideas are suggested:

1. The rogue device rejection in this thesis is done for ZigBee standard protocol (802.15.4) in 5G wireless communication systems. However, there is a potential for investigating an appropriate rogue device rejection approach for other wireless protocols (Wi-Fi, Bluetooth, Z-Wave, etc.) or sixth generation standard for wireless communications technologies (6G).

2. The data acquisition done in this work was in an anechoic chamber. A next step could be to bring the devices outside of the anechoic chamber, which will add noise to the received signal. This noisy environment results in distortion, reflection, and receiving signals from other working devices in the area which decreases the quality of the received signal for feature extraction and classification.

3. Increasing the dataset size by adding new devices from the same families of devices used in this thesis (RZUSBSTICK, TI, and XBee) or devices from other families is another suggestion. On the other hand, the data receiving setup in this thesis includes a Zynq XC7Z020 SoC FPGA sampling the captured signal 40 MHz. It is possible to test different receivers and different sampling frequencies in the next researches.

4. Another suggestion for future research work is to investigate other classifier structures with different types and number of layers at the time of designing the deep models. Decreasing the size of the model, while keeping the accuracy is a good way of reducing the computational power needed.

5. This project focused on the ZigBee protocol. However, through subtraction of reference preambles from the phase and frequency compensated received preambles before processing, we can eliminate the dependency of the methodology of this thesis from the IEEE 802.15.4 protocol, since the remaining signal is noise. This noise comes from different sources including the device itself. This specific noise can be used as the fingerprints of the device and can be used for device detection.

Appendix A

Model testing evaluation using confusion matrix and receiver operating characteristic curve

Model training and validation were discussed in Section 2.4. The last step in the procedure of device discrimination is testing. Although the model is evaluated during the validation process with a 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 Characteristic* (ROC) curves. In the next two sections, the procedures for generating the confusion matrix [112] and receiver operating characteristics [113] are described in more detail.

A.1 Confusion matrix for model testing and verification of the classifier

Verification is defined as the probability of correct classification of authorized (target) devices, with respect to the possibility of misclassification of rogue (nontarget) devices [106]. During the evaluation of the results obtained from a classifier, the rate of the correct classification of data points from the target class is called true positive rate (TPR). On the other hand, the rate of misclassification of the data points from a target class is called the false negative rate (FNR). Similarly, if the data points from the nontarget class are discriminated as data points from the target class, the rate of this misclassification is referred to as false positive

Table A.1: Confusion matrix.

Confusion matrix		
	Target (predicted)	Nontarget (predicted)
Target (actual)	TPR (true positive rate)	FNR (false negative rate)
Nontarget (actual)	FPR (false positive rate)	TNR (true negative rate)

rate (FPR). Finally, correct classification of the data points from nontarget class is referred to as true negative and the rate of this classification is called the true negative rate (TNR). The summary of all mentioned above is shown in Table A.1, called the *confusion matrix* (CM).

A.2 ROC plot for model testing and verification of the classifier

Through changing the threshold value (τ) for calculation of TPR, FPR, TNR, and FNR, it can be seen that the different values of these variables can be extracted according to the selected value of τ. Using the different values of TPR and FPR, a special plot, referred to as ROC plot, can be drawn pointing at the rate of correct classification of target class members known as a true positive rate (TPR, also called the probability of detection) vs nontarget data points misclassification or false positive rate (FPR, known as false alarm probability). The resulting ROC plot is depicted in Fig. A.1.

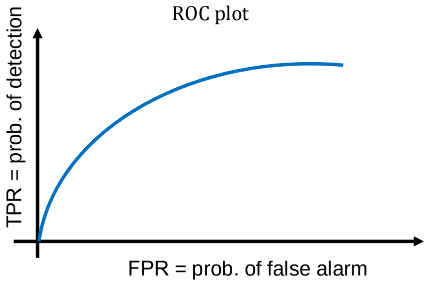


Figure A.1: ROC plot graph.

Step-by-step procedure of the model training to extract the ROC plot is as follows.

A.2.1 Model training

An example of device allocation with 10 devices for model training is shown in Fig. A.2. As can be seen, one can train the model using different scenarios. One possible scenario is using

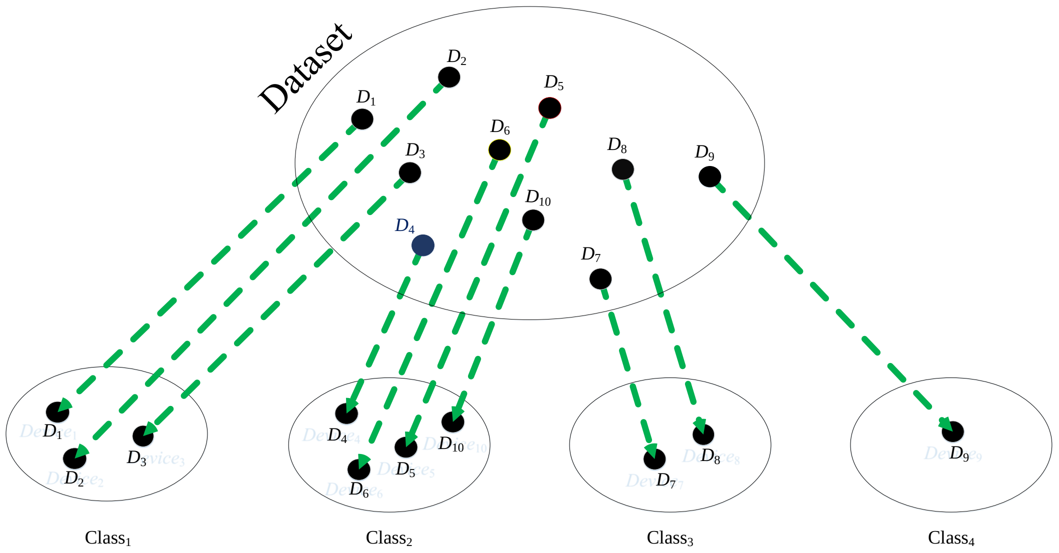


Figure A.2: Example of a dataset device allocation.

all devices with different labels like D_1, D_2, D_3,..., D_{10}. The model will be able to classify between different devices existing in a dataset. Another scenario is to divide the dataset into some sets or groups where in each group, one or more devices are allocated. One example of this case is to separate the dataset into 2 sets: an authorized set and a rogue set. Therefore, a large number of scenarios with different number of groups for device allocation are possible. Fig. A.2 shows an example of a device dataset partitioning into 4 possible groups.

Having allocated different devices into N_g groups, training the model is the next step in this procedure. After training the model, the output of the model will attribute a probability of belonging to any of these N_g groups to each data point from each device. As an example, if we consider 10 devices and $N_g = 4$ groups as in Fig. A.2, the output of the model is as shown in Eq. A.1.

$$p\left(D_j|W_i\right) = \begin{cases} 1.0 & \text{if } j = 1,2,3 \text{ and } i = 1 \\ 1.0 & \text{if } j = 4,5,6,10 \text{ and } i = 2 \\ 1.0 & \text{if } j = 7,8 \text{ and } i = 3 \\ 1.0 & \text{if } j = 9 \text{ and } i = 4 \\ 0 & \text{otherwise} \end{cases} \tag{A.1}$$

(a) Model training for identification

For identification [106], the model should be able to distinguish between the different devices. As discussed before, the model just knows the classes, and this mean that each class should include one device only. Such class allocation is illustrated in Fig. A.3, and summarized in Eq. (A.2).

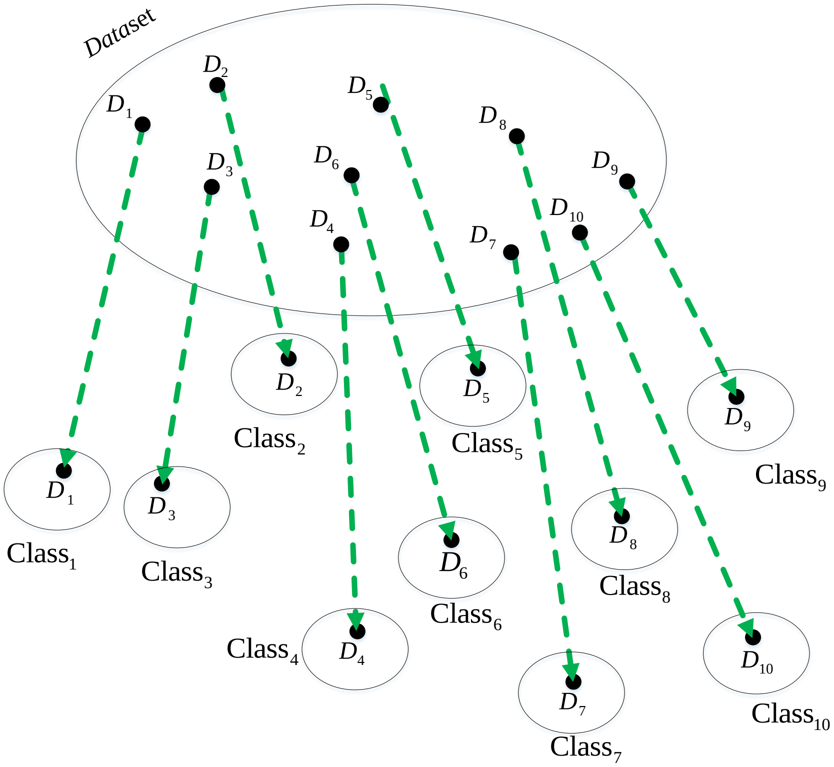


Figure A.3: Identification dataset device allocation.

$$p\left(D_j|W_i\right) = \begin{cases} 1.0 & \text{if } j = i \\ 0 & \text{otherwise} \end{cases} \tag{A.2}$$

(b) Model training for verification

The main strategy is a *one-vs-all* verification, and the main classification strategy during the training is slightly different from the simple mechanism (known as identification) described in part (a). Each model is trained based on a specific device. In other words, during the training, a single model is allocated to distinguish between the data points of a specific device from all other data points from any other devices. Therefore, during the training, all data points from the target device are labeled as $+1$ and all other data points (regardless of their real device label) are labeled as -1. Then, for example, for the separation of data points of D_1 from all the other devices, the new form of data points labeling is as shown in Fig. A.4.

After the training phase, the model should be able to distinguish between data points of D_1 from all other classes. Keeping this in mind, the model is not able to differentiate any of the remaining devices $D_2, D_3, D_4, ..., D_{10}$ from each other. The resulting data points labeling in this mode is as shown in Eq. A.3.

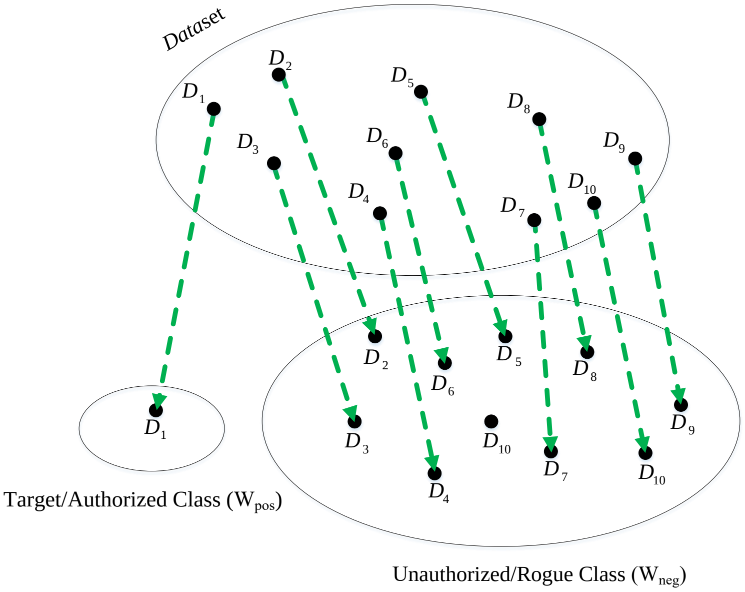


Figure A.4: Verification dataset device allocation.

$$p\left(D_j | W_{type}\right) = \begin{cases} 1.0 & \text{if } j = 1 \text{ and type=pos} \\ 1.0 & \text{if } j = 2, ..., 10 \text{ and type=neg} \\ 0 & \text{otherwise} \end{cases} \tag{A.3}$$

In the remaining of this section, the main training strategy which is selected for ROC plot extraction is the same as the verification training strategy.

A.2.2 ROC plot extraction

We now explain the mechanism for the extraction of the ROC plot based on practical cases. After the calculation of the conditional probabilities of each of the data points, the first step in classification is the determination of the threshold τ. On the other hand, by varying the value of the threshold τ, we can draw the ROC plot. First, the target class has to be assigned. Referring to [61, 100, 114], the main scenario is to divide the set of classes into two regions, such that the region of the authorized device(s) is called the target class, and the other set of the device(s) is referred to as the nontarget class. Through this definition, all data points of the target class are labeled as $+1$, and all other data points of the nontarget class are labeled as -1. These are the true classification labels of the dataset. In this step, let us imagine that we have M devices $D_1, D_2, D_3, ..., D_M$. For the division of a dataset into two sets, lets say that $[D_{i_1}, ..., D_{i_o}]_{1 \leqslant i_1,...,i_o \leqslant M}$ are o selected units as the authorized devices. Besides, $[D_{j_1}, ..., D_{j_p}]_{1 \leqslant j_1,...,j_p \leqslant M, j_1,...,j_p \neq i_1,...,i_o}$ are p devices which will be allocated to rogue devices set.

A good example for this case is the separation of devices into two groups of authorized and rogue ones, as shown in Fig. A.4.

Now, with respect to Fig. A.1, the definition of the ROC plot relates to the TPR vs the FPR. Here, the main step is to make a decision about the target and nontarget classes. Let us say that based on Fig. A.4, the authorized class (D_1) is selected as the target class, and the data points from this class are labeled as $+1$. In this case, other devices belong to the unauthorized/rogue class, and the data points of these devices are labeled as -1.

After training/validation of the model with the training/validation dataset, it is time to manage the probabilities of the trained model for target/nontarget classes to extract the exact percentage of target/nontarget testing data points classified as the target ones. To reach this goal, the strategy is as follows.

Steps to extract the thresholds for (p_{FA})

1. Extraction of the conditional target/nontarget class probabilities from the trained model for each of the testing data points of the nontarget class (-1) fed to the trained model. As discussed, each device can attend the training, validation, and/or testing. The selection of a device for one/two/all of the mentioned stages is dependent on the training strategy. Therefore, if a device is selected for training, validation, and testing, its data points are divided into 3 parts.
   If the number of testing devices is M_{te}, then we will have:

   $$M_{te} = M_{T,te} + M_{nT,te} \tag{A.4}$$

   where $1 \leq M_{te} \leq M$, and $M_{T,te}$ and $M_{nT,te}$ are the number of target and nontarget testing devices, respectively. For example, one possible testing set (S_{te}) from all possible testing subsets related to the specific case of Fig. A.4 , shown in Table A.2, is:

   $$S_{te} = \{D_1, D_2, D_4, D_8, D_9\} \tag{A.5}$$

   where $M = 8$, $M_{te} = 5$, $M_{T,te} = 1$ (since we have just D_1 as the target device in S_{te}), and $M_{nT,te} = 4$.

   $D_{nT,te,l}$ for $1 \leq l \leq M_{T,te}$ is defined as the l^{th} nontarget testing device. For Eq. (A.5), the $D_{nT,te,l}$ is any of the devices for testing, except D_1, since D_1 is a target device:

   $$(D_{nT,te,l})_{l=1,2,3,4} = \{D_2, D_4, D_8, D_9\} \tag{A.6}$$

   $\mathbf{x}^{te,n}_{nT,te,l}$ refers to the n^{th} ($1 \leq n \leq N^{te}_{nT,te,l}$) testing data point from the l^{th} nontarget testing device ($D_{nT,te,l}$). $N^{te}_{nT,te,l}$ is the number of testing data points, from l^{th} nontarget testing device ($D_{nT,te,l}$). For example, for $(D_{nT,te,l})_{l=1}$ which is equal to D_2 in Eq. (A.6), and

Table A.2: An example for dataset division to training, validation, and testing.

Devices	Target/Nontarget	Training No.	Validation No.	Testing No.
D_1	Target	6000	1000	4000
D_2	Nontarget	6000	1000	4000
D_3	Nontarget	6000	1000	-
D_4	Nontarget	6000	1000	4000
D_5	Nontarget	-	1000	-
D_6	Nontarget	-	-	-
D_7	Nontarget	-	-	-
D_8	Nontarget	-	-	4000
D_9	Nontarget	-	-	4000
D_{10}	Nontarget	-	-	-

based on Table A.2, $\left(N^{te}_{nT,te,l}\right)_{l=1} = 4000$.

Let us have a deeper look at Table A.2. Besides the testing set in Eq. (A.6), we have the following training and validation subsets.

$$\begin{aligned} S_{tr} &= \{D_1, D_2, D_3, D_4\} \\ S_{val} &= \{D_1, D_2, D_3, D_4, D_5\} \end{aligned} \tag{A.7}$$

Referring to Eq. (A.7), the definition of nontarget devices from these two subsets are as shown in Eq. (A.8).

$$\begin{aligned} \left(D_{nT,tr,o}\right)_{o=1,2,3} &= \{D_2, D_3, D_4\} \\ \left(D_{nT,val,q}\right)_{q=1,2,3,4} &= \{D_2, D_3, D_4, D_5\} \end{aligned} \tag{A.8}$$

In Eqs. (A.6) and (A.8), we can divide the devices into some different groups. Among the devices which are used, some devices just belong to testing ($\{D_8, D_9\}$). In such a case, the dataset from these devices are completely allocated to the testing phase, and $\left(N^{te}_{nT,te,l}\right)_{l=3,4}$ (referring to Eq. (A.6)) is equal to 4000. Another group of the used devices only belong to the validation phase. For instance, D_5 is a validation device, and all 1000 data points from this device are used in validation. In addition, some devices are used in more than one phase. For example, D_3 is used in training and validation, or D_4 is used in all phases of training, validation, and testing. Then, for the specific device D_4 we have:

$$\begin{aligned} \left(N^{tr}_{nT,tr,o}\right)_{o=3} &= 6000 \\ \left(N^{val}_{nT,val,q}\right)_{q=3} &= 1000 \\ \left(N^{te}_{nT,te,l}\right)_{l=2} &= 4000 \end{aligned} \tag{A.9}$$

where $\left(N^{tr}_{nT,tr,o}\right)_{o=3}$ is the number of training data points, from the 3^{th} nontarget training device $\left(D_{nT,tr,o}\right)_{o=3}$. $\left(N^{val}_{nT,val,q}\right)_{q=3}$ is the number of validation data points from 3^{th}

Table A.3: Conditional probabilities of class $+1$ and -1 for nontarget testing dataset.

	Unauthorized/Rogue/Nontarget testing dataset		
	$\left(D_{nT,te,l}\right)_{l=1}$		
Positive Class	$p\left(\mathbf{x}^{te,n}_{nT,te,l}\middle\vert W_{pos}\right)_{\substack{n=1\\l=1}}$	$\cdots$	$p\left(\mathbf{x}^{te,n}_{nT,te,l}\middle\vert W_{pos}\right)_{\substack{n=N^{te}_{nT,te,l}\\l=1}}$
Negative Class	$p\left(\mathbf{x}^{te,n}_{nT,te,l}\middle\vert W_{neg}\right)_{\substack{n=1\\l=1}}$	$\cdots$	$p\left(\mathbf{x}^{te,n}_{nT,te,l}\middle\vert W_{neg}\right)_{\substack{n=N^{te}_{nT,te,l}\\l=1}}$
$\vdots$	$\vdots$		
	$\left(D_{nT,te,l}\right)_{l=M_{nT,te}}$		
Positive Class	$p\left(\mathbf{x}^{te,n}_{nT,te,l}\middle\vert W_{pos}\right)_{\substack{n=1\\l=M_{nT,te}}}$	$\cdots$	$p\left(\mathbf{x}^{te,n}_{nT,te,l}\middle\vert W_{pos}\right)_{\substack{n=N^{te}_{nT,te,l}\\l=M_{nT,te}}}$
Negative Class	$p\left(\mathbf{x}^{te,n}_{nT,te,l}\middle\vert W_{neg}\right)_{\substack{n=1\\l=M_{nT,te}}}$	$\cdots$	$p\left(\mathbf{x}^{te,n}_{nT,te,l}\middle\vert W_{neg}\right)_{\substack{n=N^{te}_{nT,te,l}\\l=M_{nT,te}}}$
	$\left(N^{te}_{nT,te,l}\right)_{l=1} + \cdots + \left(N^{te}_{nT,te,l}\right)_{l=M_{nT,te}} = N^{te}_{nT,te}$		

nontarget validation device $\left(D_{nT,val,q}\right)_{q=3}$.

As mentioned, the summation of the number of all testing data points from all nontarget testing devices is equal to the number of all testing data points from nontarget devices:

$$\left(N^{te}_{nT,te,l}\right)_{l=1} + \cdots + \left(N^{te}_{nT,te,l}\right)_{l=M_{nT,te}} = N^{te}_{nT,te} \tag{A.10}$$

For instance, based on Eq. (A.6) and Table A.2, $\left(N^{te}_{nT,te,l}\right)_{l=1,2,3,4} = 4000$, and $N^{te}_{nT,te} = 16000$.

Then, the resulting conditional probabilities are as illustrated in Table A.3.

2. Calculation of the likelihood ratios (LR) of the nontarget data points obtained as:

$$LR\left(\mathbf{x}^{te,n}_{nT,te,l}\right)_{\substack{1\leq l\leq M_{nT,te}\\1\leq n\leq N^{te}_{nT,te,l}}} = \frac{p\left(\mathbf{x}^{te,n}_{nT,te,l}\middle\vert W_{pos}\right)}{p\left(\mathbf{x}^{te,n}_{nT,te,l}\middle\vert W_{neg}\right)} \tag{A.11}$$

3. Sorting the likelihood ratios of the nontarget class data points in ascending order.

$$\begin{gathered}
LR^{\min}_{nT} \leq \cdots \leq LR^{\max}_{nT} \\
LR^{\min}_{nT} = \min\left\{ LR\left(\mathbf{x}^{te,n}_{nT,te,l}\right)_{\substack{1\leq l\leq M_{nT,te}\\1\leq n\leq N^{te}_{nT,te,l}}} \right\} \\
LR^{\max}_{nT} = \max\left\{ LR\left(\mathbf{x}^{te,n}_{nT,te,l}\right)_{\substack{1\leq l\leq M_{nT,te}\\1\leq n\leq N^{te}_{nT,te,l}}} \right\}
\end{gathered} \tag{A.12}$$

In the next step, the extraction of essential thresholds for each false alarm probability is done. Then, generate a vector of false alarm probabilities from 0 to 1:

$$p_{FA} = [0.000, 0.001, \cdots, 1.000] \tag{A.13}$$

4. Determination of the corresponding threshold value (τ) for each false positive rate or FPR (referred to as false alarm probability or p_{FA}) is needed in this step. Each false alarm probability relates to the percentage of all cases where the data points from nontarget/unauthorized/rogue devices are misclassified. For example, $p_{FA} = 0.6$ means that 60% of the nontarget/unauthorized/rogue data points have LR values higher than the selected threshold τ, as shown in Eq. (A.14).

   $$LR\left(\mathbf{x}_{nT,te,l}^{te,n}\right)_{\substack{1\leq l\leq M_{nT,te} \\ 1\leq n\leq N_{nT,te,l}^{te}}} \geq \tau_{p_{FA}=0.6} \tag{A.14}$$

   In other words, the role of the relating threshold for $p_{FA} = 0.6$ ($\tau_{p_{FA}=0.6}$) is as follows.

   To obtain the related threshold τ for each corresponding p_{FA}, the following steps should be made:

   - Attributing an index to each LRs in a sorted order, as shown in Eq. (A.15).

   $$\left(LR_j\right)_{j=1} = LR_{nT}^{\min} \leq \cdots \leq \left(LR_j\right)_{j=N_{nT,te}^{te}} = LR_{nT}^{\max} \tag{A.15}$$

   - For the calculation of τ, the following equation is used:

   $$\begin{gathered} a = N_{nT,te}^{te} \times (1 - p_{FA}) \\ b = \lfloor a \rfloor \\ c = a - b \\ \tau_{p_{FA}} = LR_b + c \times (LR_{b+1} - LR_b) \end{gathered} \tag{A.16}$$

5. Repeating the previous step for the extraction of related thresholds (τ) for all values of p_{FA} in Eq. (A.13).

Steps to extract the detection probabilities (p_d)

1. Extraction of the conditional target/nontarget class probabilities of the model for each target class data points fed to the model, as shown in Table A.4. In this table, $D_{T,te,z}$ for $1 \leq z \leq M_{T,te}$ corresponds to the z^{th} target testing device. For instance, in the specific case of Fig. A.4, $D_{T,te,z}$ can be none of the devices for testing, except D_1, since D_1 is a target device:

   $$D_{T,te,z} \in \{D_1\} \tag{A.17}$$

   As discussed, each device can be used during the training, validation, and/or testing phases. The selection of a device for one/two/all of the mentioned stages is dependent on the training strategy. $N_{T,te,z}^{te}$ is the number of testing data points from z^{th} target testing device ($D_{T,te,z}$).

Table A.4: Conditional probabilities of class $+1$ and -1 for the target testing dataset.

	Authorized/Target testing dataset		
	$(D_{T,te,z})_{z=1}$		
Positive Class	$p\left(\mathbf{x}^{te,n}_{T,te,z}\middle\|W_{pos}\right)_{\substack{n=1\\z=1}}$	$\cdots$	$p\left(\mathbf{x}^{te,n}_{T,te,z}\middle\|W_{pos}\right)_{\substack{n=N^{te}_{T,te,z}\\z=1}}$
Negative Class	$p\left(\mathbf{x}^{te,n}_{T,te,z}\middle\|W_{neg}\right)_{\substack{n=1\\z=1}}$	$\cdots$	$p\left(\mathbf{x}^{te,n}_{T,te,z}\middle\|W_{neg}\right)_{\substack{n=N^{te}_{T,te,z}\\z=1}}$
$\vdots$	$\vdots$		
	$(D_{T,te,z})_{z=M_{T,te}}$		
Positive Class	$p\left(\mathbf{x}^{te,n}_{T,te,z}\middle\|W_{pos}\right)_{\substack{n=1\\z=M_{T,te}}}$	$\cdots$	$p\left(\mathbf{x}^{te,n}_{T,te,z}\middle\|W_{pos}\right)_{\substack{n=N^{te}_{T,te,z}\\z=M_{T,te}}}$
Negative Class	$p\left(\mathbf{x}^{te,n}_{T,te,z}\middle\|W_{neg}\right)_{\substack{n=1\\z=M_{T,te}}}$	$\cdots$	$p\left(\mathbf{x}^{te,n}_{T,te,z}\middle\|W_{neg}\right)_{\substack{n=N^{te}_{T,te,z}\\z=M_{T,te}}}$
	$\left(N^{te}_{T,te,z}\right)_{z=1}+\cdots+\left(N^{te}_{T,te,z}\right)_{z=M_{T,te}}=N^{te}_{T,te}$		

2. Extraction of the likelihood ratios of the target data points based on

$$LR\left(\mathbf{x}^{te,n}_{T,te,z}\right)_{\substack{1\leq z\leq M_{T,te}\\1\leq n\leq N^{te}_{T,te,z}}}=\frac{p\left(\mathbf{x}^{te,n}_{T,te,z}\middle|W_{pos}\right)}{p\left(\mathbf{x}^{te,n}_{T,te,z}\middle|W_{neg}\right)} \tag{A.18}$$

3. Sorting the likelihood ratios of the target class data points in ascending order.

$$\begin{aligned} &LR^{\min}_{T}\leq\cdots\leq LR^{\max}_{T}\\ LR^{\min}_{T}&=\min\left\{LR\left(\mathbf{x}^{te,n}_{T,te,z}\right)_{\substack{1\leq z\leq M_{T,te}\\1\leq n\leq N^{te}_{T,te,z}}}\right\}\\ LR^{\max}_{T}&=\max\left\{LR\left(\mathbf{x}^{te,n}_{T,te,z}\right)_{\substack{1\leq z\leq M_{T,te}\\1\leq n\leq N^{te}_{T,te,z}}}\right\}\end{aligned} \tag{A.19}$$

4. Extraction of the TPR (also known as probability of detection or p_d) for each of the extracted threshold τ values. Each TPR relates to the percentage of cases that the target data points are correctly classified. For example, $TPR = 0.6$ means that 60 percent of the target data points has a higher probability $p\left(\mathbf{x}^{te,n}_{T,te,z}\middle|W_{pos}\right)_{\substack{1\leq z\leq M_{T,te}\\1\leq n\leq N^{te}_{T,te,z}}}$ than the corresponding parameter values $\tau_{p_{FA}=0.6}\times p\left(\mathbf{x}^{te,n}_{T,te,z}\middle|W_{neg}\right)_{\substack{1\leq z\leq M_{T,te}\\1\leq n\leq N^{te}_{T,te,z}}}$, as shown in Eq. (A.20).

$$LR\left(\mathbf{x}^{te,n}_{T,te,z}\right)_{\substack{1\leq z\leq M_{T,te}\\1\leq n\leq N^{te}_{T,te,z}}}\geq\tau_{p_{FA}=0.6} \tag{A.20}$$

To obtain the related p_d for each corresponding p_{FA}, the following steps are done.

- Attributing an index to each of the sorted LRs, as shown in Eq. (A.21).

$$\left(LR_j\right)_{j=1} = LR_T^{\min} \leq \cdots \leq \left(LR_j\right)_{j=N_{T,te}^{te}} = LR_T^{\max} \tag{A.21}$$

- Using the extracted τ values, the corresponding TPR (p_d) for each FPR is extracted. For this goal, after applying the extracted threshold to each of the corresponding FPRs, the percentage of the correct classified target data points are obtained.

  Using this mechanism, through the calculation and use of the τ values, the corresponding TPR (or p_d) and FPR (or p_{FA}) values for the ROC plot are obtained.

Appendix B

LSTM Based Autoencoders

Recurrent neural networks (RNNs) are a specific kind of models that includes layers which take advantage of the time-dependency of the data as a key factor for feature extraction [111]. Using an integrative loop over the different time steps of the input sequence, an RNN layer can extract the information from a time-step and memorize it for the next steps [115]. Long short-term memories as a specific kind of layers to support time-dependency in the sequences of the input data were used in combination with an autoencoder in [116] in 2015. The application of the LSTM based autoencoder in [116] was to reconstruct and predict the video frames using the features extracted from the input data. In [116], the input dataset was used to train the model for predicting the next sequence. With this strategy, simultaneously training of the reconstruction and prediction output guarantees to provide meaningful features at the output of the encoder for good classification performance.

B.1 How to implement an LSTM based autoencoder in Keras

Designing an LSTM based autoencoder in Keras for feature extraction and data reconstruction is done through implementing an autoencoder including a recurrent layer using a recurrent neural network (RNN), long short-term memory (LSTM), gated recurrent unit (GRU), CuDNN long short-term memory (CuDNNLSTM), or any other types of recurrent layers [115]. In this section we intended to design an LSTM based autoencoder, using an LSTM layer in Keras, to provide a review on the model design procedure. But as seen in Table 5.1, the recurrent layer type which is used is an CuDNNLSTM layer. It is worth mentioning that the CuDNNLSTM and LSTM layers are similar, and the only difference is that CuDNNLSTM is the fast implementation of the LSTM layers which works on GPU with the TensorFlow backend [117].

Table B.1: An example LSTM based autoencoder structure introduced in [111].

```
1   from numpy import array
2   from keras.models import Model
3   from keras.layers import Input
4   from keras.layers import LSTM
5   from keras.layers import Dense
6   from keras.layers import RepeatVector
7   from keras.layers import TimeDistributed

    # Defining the Input Data
8   seq_in = array([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])
9   n_in = len(seq_in)
10  seq_in = seq_in.reshape((1, n_in, 1))

    # Defining the Encoder part of the Model
11  visible = Input(shape=(n_in,1))
12  encoder = LSTM(320, return_sequence=False, activation='relu')(visible)

    # Defining the Decoder part of the Model
13  decoder_1 = RepeatVector(n_in)(encoder)
14  decoder_2 = LSTM(320, activation='relu', return_sequence=True)(decoder_1)
15  decoder = TimeDistributed(Dense(1))(decoder_2)

    # Making the Model
16  model = Model(inputs=visible, outputs=decoder)

    # Compiling the Model
17  model.compile(optimizer='adam', loss='mse')

    # Training the Model
18  model.fit(seq_in, seq_in, epochs=100, verbose=0)

    # Testing the Model
19  seq_in_hat = model.predict(seq_in, verbose=0)
```

B.1.1 LSTM based autoencoder example in Keras

Let us look at an example shown in Table B.1, including the encoder and reconstruction decoder derived from an example called the composite LSTM autoencoder in [111]. Comparing the structure of the LSTM based autoencoder in Table B.1 with the structure introduced in Fig. 2.2, it can be seen that the general idea is the same. In Table B.1, lines 11 and 12 are responsible for extracting the features from the input data [118]. In these two lines, the output contains the summary of the information related to the input sequence for selected time steps [118]. In the decoder part, lines 13 to 15 are allocated to the data reconstruction. In line

Table B.2: Example model summary in lines 11 to 15 in Table B.1 [111].

Index	Layer Name	Layer Type	Output Shape	Active Func.	Parameters No.
0	visible	InputLayer	(None, 9, 1)	-	0
1	encoder	LSTM	(None, 320)	Relu	412160
2	decoder_1	RepeatVector	(None, 9, 320)	-	0
3	decoder_2	LSTM	(None, 9, 320)	Relu	820480
4	decoder	TimeDistributed(Dense)	(None, 9, 1)	-	321
Total parameters:		1,232,961			
Trainable parameters:		1,232,961			

13, the output sequence from line 12 is repeated n_in (which is the length of the seq_in and equal to time steps). Finally, line 14 runs an LSTM layer on the sequence to make it ready for the reconstructed input sequence (seq_in_hat) at the output of line 15. The summary of the model in lines 11 to 15 of Table B.1 is shown in Table B.2. Finally, lines 16 to 19 are responsible for making, compiling, training, and testing the model. This is the general principle for designing the autoencoders, as discussed in Chapter 2.

Input size to the autoencoder in Table B.1 Let us have a deeper look at lines 8 to 11 in Table B.1, since the main purpose here is to explain how to prepare the input for the LSTM layer. In line 8, the input data is as follows, with an array size of 9 $(9,)$[1] and length (n_in) equal to 9 in line 9:

$$seq\_in = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9] \tag{B.1}$$

After reshaping in line 10, the input data is as in Eq. (B.2) with size $(1, 9, 1)$.

$$seq\_in = \begin{bmatrix} \begin{bmatrix} [0.1] \\ [0.2] \\ [0.3] \\ [0.4] \\ [0.5] \\ [0.6] \\ [0.7] \\ [0.8] \\ [0.9] \end{bmatrix} \end{bmatrix} \tag{B.2}$$

Input size to the LSTM layer in Table B.1 The size of the input sequence to the LSTM layer in Table B.2 (seq_in) is equal to $(\text{None}, 9, 1)$, which is equal to the output size of the 'InputLayer'. Referring to [119], the first argument in the layer input shape $(\text{None}, 9, 1)$

[1] $(9,)$ is a Python programming notation to show an array with 9 elements.

which is equal to 'None', which is a non-specific value and points at the number of data points, second number (9) illustrates the number of time steps in each data point, and the third argument points at the number of samples in a time-step (equal to 1).

Output size from the LSTM layer in Table B.1 As can be seen in Table B.1, the structure of the LSTM layer in the encoder part in line 12 and also the decoder part in line 14 includes an element named 'return_sequence', a logical parameter equal to 'True' or 'False'. If this element is equal to 'False', the LSTM layer returns the last output of the output series, such as what has happened as the output of the line 12 with $N_{unit} = 320$ cell units (with the output size $(\text{None}, 320)$, as shown in Table B.2), and if this element is equal to 'True', the output of the LSTM layer includes the complete output series for all (9) time steps, similar to what has happened as the output of the line 14 (with the output size $(\text{None}, 9, 320)$, as depicted in Table B.2). In both cases of these two lines (lines 12 and 14) the number of cell units assigns the number of dimensions of the output [117]. The latter case is similar to the output of the CuDNNLSTM layer in Table 5.1 (with 4800 time steps, the number of cell units (N_{unit}) equal to 320, and the output size $(\text{None}, 4800, 320)$).

B.1.2 LSTM based autoencoder in Table 5.1

Input size to the LSTM layer in Table 5.1 The number of data points in the dataset in Table B.1 is equal to 1. If the dataset size is larger than 1, such as what was used in Chapters 4 and 5, referring to Table 5.1, the size of the input to the LSTM layer in the fourth row (with the layer name as EN.layer_3) is (None, 4800, 1). This means that for a non-specific number of data points (None), the number of time steps will be equal to $K_{RNN} = 4800$, and in each time-step, there is a single sample, as illustrated in Eq. (B.3).

$$seq\_in = \begin{bmatrix} \begin{bmatrix} [a_{1,1}] \\ \vdots \\ [a_{1,4800}] \end{bmatrix} \\ \vdots \\ \begin{bmatrix} [a_{\text{None},1}] \\ \vdots \\ [a_{\text{None},4800}] \end{bmatrix} \end{bmatrix} \tag{B.3}$$

where $a_{n,k}$ shows the k^{th} sample from n^{th} data point for $n \in \{1, \cdots, \text{None}\}$ $(1 \leq \text{None} \leq N)$ and $k \in \{1, \cdots, K_{RNN} = 4800\}$.

It is worth mentioning that if we compare the input size of the LSTM layer in Table 5.1 (None, 4800, 1) with 4800 time steps, to that of the LSTM layer in Table B.1 (equal to

(None, 9, 1)) with 9 time steps, we can see that both have the same format. As shown in Eq. (B.3), the input to the LSTM layer is vertical, which means that each row in a vertical format is counted as a time-step. Then in Eq. (B.3), for each data point, the number of time steps is $k_{RNN} = 4800$, including one sample in each time-step, which emphasizes on the fact that the model is taking benefit from the time-dependency of the input data point sequence. This is the same as the input for the CuDNNLSTM layer in Table 5.1 (with the input size $(\text{None}, 4800, 1)$).

Output size from the LSTM layer in Table 5.1 An LSTM layer (such as LSTM layer in row EN.layer_3 of Table 5.1) has a cell with N_{unit} units, equal to 320, as the same as EN.layer_3 in Table 5.1. Focusing on a single data point from Eq. (B.3), the size of an input vector ($\mathbf{x}_n^{RNN}$) is $(4800, 1)$, the same as the size of a single data point as the input to the EN.layer_3 of Table 5.1. Then, based on Eq. (2.7), we have:

$$\mathbf{x}_n^{RNN} = [a_1, a_2, \cdots, a_{4800}]^\intercal \tag{B.4}$$

where a_k $(1 \leq k \leq K_{RNN} = 4800)$ are scalar values. In such a manner, input values at time $1, 2, \cdots, k = 4800$ are $x_{n,1}^{RNN} = a_1, x_{n,2}^{RNN} = a_2, \cdots, x_{n,4800}^{RNN} = a_{4800}$, respectively. Then, the size of the output $\mathbf{o}_{n,k}^{RNN}$ vectors of an RNN cell with 320 units ($N_{unit} = 320$) are $(1, 320)$. These cell units at time-step k are applied to the input data, to make the LSTM output at time-step k from all 320 cell units ($\mathbf{o}_{n,k}^{RNN}$). In addition, as mentioned before, if we leave the 'return_sequence'=False, the size of the output matrix O_n^{RNN} from an LSTM layer after $K_{RNN} = 4800$ time steps is equal to the size of $\mathbf{o}_{n,k}^{RNN}$ at the last time-step $k = K_{RNN} = 4800$. The size of the overall output O_n^{RNN} for this case if $(1, 320)$. On the other hand, if we set 'return_sequence'=True, the size of output matrix O_n^{RNN} from a cell unit after $K_{RNN} = 4800$ time steps is $(4800, 320)$, which has the same shape as the O_n^{RNN} in Eq. (2.7).

Absence of an LSTM layer in the decoder structure of Fig. 5.3 and Table 5.1 As can be seen in Table 5.1, line 14 is allocated to an LSTM layer in the decoder part of the autoencoder, responsible for making the sequence ready for reconstruction. But as shown in Fig. 5.3 and Table 5.1, there is no such LSTM layer in the decoder part of the model proposed in Chapter 5. The reason for this modification is that, as explained in Section 2.2, the application of the decoder part is just at the training stage. In such a case, training the decoder and classifier outputs together guaranties that the features provided by the encoder part for the classifier are meaningful enough to lead to a high classification accuracy. Then, the responsibility of the decoder is just to reconstruct the data, as accurate as possible, at the training stage, to help the encoder to find a meaningful feature collaborating to the success of the classifier in its classification duty; and after training, at the testing stage, this part does not exist in the model anymore. Therefore, keeping this basic rule in mind, the LSTM layer in the encoder part is kept for time-dependency extraction from the input data, and in order to reduce the computational complexity of the model at the training stage, the LSTM structure in the decoder part is removed in Table 5.1. The MSE error values in Figs. 5.5 and 5.6 for both the RAW and DDWT datasets prove that removing the LSTM layer from the decoder does not affect the decoder accuracy in the reconstruction of the input dataset.

On the other hand, as explained in Section B.1.1, if the 'return_sequence' for an LSTM layer is set to 'True' (as what is set for the layer EN.layer_3 in Table 5.1), then the LSTM layer returns the complete series for all time steps (equal to $K_{RNN} = 4800$) in Chapter 5. In such a case, we can prevent the LSTM layer from forgetting the long time dependencies of the output signal, and feed the classifier with all possible time dependencies from the complete $K_{RNN} = 4800$ time steps.

Bibliography

[1] M. Tillman, “What is Zigbee and why is it important for your smart home?” https://www.pocket-lint.com/smart-home/news/129857-what-is-zigbee-and-why-is-it-important-for-your-smart-home, accessed on: March 23, 2022, publication date: 26 May, 2021.

[2] “A Closer Look to Zigbee Technology Applications,” https://www.orapada.com/zigbee-technology-applications/, accessed on: March 23, 2022.

[3] X. Guo and J. Zhu, “Research on security issues in Wireless Sensor Networks,” in *Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology*, vol. 2. IEEE, 2011, pp. 636–639.

[4] K. Islam, W. Shen, and X. Wang, “Security and privacy considerations for wireless sensor networks in smart home environments,” in *Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2012, pp. 626–633.

[5] M. A. B. Karnain and Z. B. Zakaria, “A review on ZigBee security enhancement in smart home environment,” in *2015 2nd International Conference on Information Science and Security (ICISS)*. IEEE, 2015, pp. 1–4.

[6] F. Callegati, W. Cerroni, and M. Ramilli, “Man-in-the-Middle Attack to the HTTPS Protocol,” *IEEE Security and Privacy*, vol. 7, no. 1, pp. 78–81, 2009.

[7] P. Arana, “Benefits and vulnerabilities of Wi-Fi protected access 2 (WPA2),” *INFS*, vol. 612, pp. 1–6, 2006.

[8] Wi-Fi Alliance, 2019, “WPA3 specification version 2.0,” https://www.wi-fi.org/download.php?file=/sites/default/files/private/WPA3_Specification_v2.0.pdf, Accessed on: March 23, 2022.

[9] W. E. Cobb, E. D. Laspe, R. O. Baldwin, M. A. Temple, and Y. C. Kim, “Intrinsic physical-layer authentication of integrated circuits,” *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 14–24, June 2011.

[10] M. A. Haji Bagheri Fard, J.-Y. Chouinard, and B. Lebel, "Rogue device discrimination in ZigBee networks using wavelet transform and autoencoders," *Annals of Telecommunications*, pp. 1–16, 2020.

[11] K. Nohl and C. Paget, "GSM: SRSLY?(2009)," in *26th Chaos Communication Congress (26C3), Berlin*, 2009.

[12] I. W. Group et al., "IEEE standard for local and metropolitan area networks—Part 15.4: Low-rate wireless personal area networks (LR-WPANS)," *IEEE Std*, vol. 802, pp. 4–2011, 2011.

[13] E. Rodríguez, B. Otero, N. Gutiérrez, and R. Canal, "A Survey of Deep Learning Techniques for Cybersecurity in Mobile Networks," *IEEE Communications Surveys Tutorials*, vol. 23, no. 3, pp. 1920–1955, 2021.

[14] N. R. Kumar, C. Bhuvana, and S. Anushya, "Comparison of ZigBee and Bluetooth wireless technologies-survey," in *2017 International Conference on Information Communication and Embedded Systems (ICICES)*. IEEE, 2017, pp. 1–4.

[15] E. Fodor, "Zigbee vs. Bluetooth: Choosing the Right Protocol for Your IoT Application," https://www.digi.com/blog/post/zigbee-vs-bluetooth-choosing-the-right-protocol, accessed on: March 23, 2022, publication date: 5 March, 2020.

[16] M. Khasawneh, I. Kajman, R. Alkhudaidy, and A. Althubyani, "A survey on Wi-Fi protocols: WPA and WPA2," in *International Conference on Security in Computer Networks and Distributed Systems*. Springer, 2014, pp. 496–511.

[17] D. Zhan, "ZigBee vs Wi-Fi, Which is Better?" https://itead.cc/sonoff/zigbee-vs-wi-fi-which-is-better/, accessed on: March 23, 2022, publication date: 17 May, 2021.

[18] O. G. Aju, "A survey of ZigBee wireless sensor network technology: Topology, applications and challenges," *International Journal of Computer Applications*, vol. 130, no. 9, pp. 47–55, 2015.

[19] P. Tracy, "What is Zigbee and where does it fit into the IoT?)," https://enterpriseiotinsights.com/20160824/5g/zigbee-iot-tag31-tag99, accessed on: March 23, 2022, publication date: 24 August, 2016.

[20] M. Brown, "Philips Hue Bluetooth + Zigbee smart bulbs review: The best smart lighting just got better (but no less expensive)," https://www.techhive.com/article/3410037/philips-hue-bluetooth-zigbee-smart-bulb-review.html, accessed on: March 23, 2022, publication date: 19 July, 2019.

[21] L. Rosencrance, “ZigBee),” https://internetofthingsagenda.techtarget.com/definition/ZigBee, accessed on: March 23, 2022, publication date: June, 2017.

[22] Ashwak, “Zigbee v/s Bluetooth,” https://www.electronicshub.org/zigbee-vs-bluetooth/, accessed on: March 23, 2022, publication date: 23 April, 2021.

[23] , “Bluetooth,” https://en.wikipedia.org/w/index.php?title=Bluetooth&action=history, accessed on: March 23, 2022, June, 2021.

[24] ——, “Wi-Fi,” https://en.wikipedia.org/wiki/Wi-Fi, accessed on: March 23, 2022, June, 2021.

[25] ——, “ZigBee,” https://en.wikipedia.org/wiki/Zigbee, accessed on: March 23, 2022, June, 2021.

[26] X.-g. Huang, L. Shen, and Y.-h. Feng, “A user authentication scheme based on fingerprint and USIM card,” in *2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. IEEE, 2008, pp. 1261–1264.

[27] K. Zhang, X. Liang, R. Lu, and X. Shen, “Sybil attacks and their defenses in the internet of things,” *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 372–383, 2014.

[28] Z. Akram, M. A. Saeed, and M. Daud, “Real time exploitation of security mechanisms of residential WLAN access points,” in *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. IEEE, 2018, pp. 1–5.

[29] D. He and S. Zeadally, “An analysis of RFID authentication schemes for internet of things in healthcare environment using elliptic curve cryptography,” *IEEE internet of things journal*, vol. 2, no. 1, pp. 72–83, 2014.

[30] J. Lim, H. Oh, and S. Kim, “A new hash-based RFID mutual authentication protocol providing enhanced user privacy protection,” in *International Conference on Information Security Practice and Experience*. Springer, 2008, pp. 278–289.

[31] Y. Tu, Z. Zhang, Y. Li, C. Wang, and Y. Xiao, “Research on the Internet of Things device recognition based on RF-fingerprinting,” *IEEE Access*, vol. 7, pp. 37 426–37 431, 2019.

[32] D. Schatz, R. Bashroush, and J. Wall, “Towards a more representative definition of cyber security,” *Journal of Digital Forensics, Security and Law*, vol. 12, no. 2, pp. 53–74, 2017.

[33] L. Kappelman, V. L. Johnson, C. Maurer, K. Guerra, E. McLean, R. Torres, M. Snyder, and K. Kim, “The 2019 SIM IT Issues and Trends Study.” *MIS Quarterly Executive*, vol. 19, no. 1, 2020.

[34] "NDIA 2019 Cybersecurity Report," https://www.ndia.org/policy/cyber/2019-cybersecurity-report, accessed on: March 23, 2022.

[35] K. Merchant, S. Revay, G. Stantchev, and B. Nousain, "Deep learning for RF device fingerprinting in cognitive communication networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 160–167, 2018.

[36] Z. Chen, L. Peng, A. Hu, and H. Fu, "Generative adversarial network-based rogue device identification using differential constellation trace figure," *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, no. 1, pp. 1–27, 2021, publisher=SpringerOpen.

[37] A, Kevin et al., "That 'Internet of things' thing," *RFID journal*.

[38] M. Gregg, S. Watkins, G. Mays, C. Ries, R. Bandes, and B. Franklin, *Hack the stack: Using snort and ethereal to master the 8 layers of an insecure network*. Elsevier, 2006.

[39] Y. Xu, Y. Jiang, C. Hu, H. Chen, L. He, and Y. Cao, "A balanced security protocol of wireless sensor network for smart home," in *2014 12th International Conference on Signal Processing (ICSP)*. IEEE, 2014, pp. 2324–2327.

[40] T. Shang, F. Huang, J. Chen, and J. Liu, "A security-enhanced key distribution scheme for AODVjr routing protocol in Zig-Bee networks," *Chinese Journal of Electronics*, vol. 22, no. 3, pp. 577–582, 2013.

[41] I. M. R. Verbauwhede, *Secure integrated circuits and systems*. Springer, 2010.

[42] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002.

[43] G. DeJean and D. Kirovski, "RF-DNA: Radio-frequency certificates of authenticity," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2007, pp. 346–363.

[44] W. C. Suski II, M. A. Temple, M. J. Mendenhall, and R. F. Mills, "Radio frequency fingerprinting commercial communication devices to enhance electronic security," *International Journal of Electronic Security and Digital Forensics*, vol. 1, no. 3, pp. 301–322, 2008.

[45] M. Barbeau, J. Hall, and E. Kranakis, "Detection of rogue devices in Bluetooth networks using radio frequency fingerprinting," in *Proceedings of the 3rd IASTED International Conference on Communications and Computer Networks, CCN*. Citeseer, 2006, pp. 4–6.

[46] B. Danev and S. Capkun, "Transient-based identification of wireless sensor nodes," in *2009 International Conference on Information Processing in Sensor Networks*. IEEE, 2009, pp. 25–36.

[47] B. Danev, T. S. Heydt-Benjamin, and S. Capkun, "Physical-layer Identification of RFID Devices," in *USENIX security symposium*, 2009, pp. 199–214.

[48] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable PUFs," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 2, no. 1, pp. 1–33, 2009.

[49] A. Rahnama, M. Beheshti-Atashgah, T. Eghlidos, and M. R. Aref, "An Ultra-Lightweight RFID Mutual Authentication Protocol," in *2019 16th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)*, vol. , no. , 2019, pp. 27–32.

[50] M. T. Hammi, E. Livolant, P. Bellot, A. Serhrouchni, and P. Minet, "A lightweight IoT security protocol," in *2017 1st cyber security in networking conference (CSNet)*. IEEE, 2017, pp. 1–8.

[51] X. Liang, Y. Lv, B. Zhao, Y. Liu, Z. Sun, and W. Cen, "A RFID Mutual Authentication Security Protocol Design and Analysis," in *2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, vol. , no. , 2014, pp. 508–512.

[52] D. R. Reising, D. C. Prentice, and M. A. Temple, "Real-time RF fingerprinting using field programmable gate arrays," in *Proc. 2011 IEEE Int. Conf. on Commun. (ICC11)*, 2011, pp. 1–6.

[53] M. D. Williams, S. A. Munns, M. A. Temple, and M. J. Mendenhall, "RF-DNA fingerprinting for airport WiMax communications security," in *2010 Fourth International Conference on Network and System Security*. IEEE, 2010, pp. 32–39.

[54] M. Williams, M. A. Temple, and D. R. Reising, "Augmenting bit-level network security using PHY Layer RF-DNA fingerprinting," in *Proc. IEEE Global Comm. Conf. Wireless Inf. Netw. and Syst. (GLOBECOM10)*, 2010, pp. 2168–2173.

[55] O. Ureten and N. Serinken, "Wireless security through RF fingerprinting," *Canadian Journal of Electrical and Computer Engineering*, vol. 32, no. 1, pp. 27–33, 2007.

[56] D. R. Reising, M. A. Temple, and M. J. Mendenhall, "Improved wireless security for GMSK-based devices using RF fingerprinting," *International Journal of Electronic Security and Digital Forensics*, vol. 3, no. 1, pp. 41–59, 2010.

[57] W.E. Cobb, E. W. Garcia, M. A. Temple, R. O. Baldwin, and Y. C. Kim, "Physical layer identification of embedded devices using RF-DNA fingerprinting," in *2010-MILCOM 2010 Military Communications Conference*. IEEE, 2010, pp. 2168–2173.

[58] D. R. Reising, M. A. Temple, and J. A. Jackson, "Authorized and Rogue Device Discrimination Using Dimensionally Reduced RF-DNA Fingerprints," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1180–1192, 2015.

[59] H. Patel, "Non-parametric feature generation for RF-fingerprinting on ZigBee devices," in *2015 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*. IEEE, 2015, pp. 1–5.

[60] T. J. Bihl, K. W. Bauer, and M. A. Temple, "Feature selection for RF fingerprinting with multiple discriminant analysis and using ZigBee device emissions," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, pp. 1862–1874, 2016.

[61] C. Dubendorfer, B. W. Ramsey, and M. A. Temple, "ZigBee device verification for securing industrial control and building automation systems," in *International Conference on Critical Infrastructure Protection*. Springer, 2013, pp. 47–62.

[62] M. A. Haji Bagheri Fard, J.-Y. Chouinard, B. Lebel, and Paul Fortier, "Wireless Device Authentication using LSTM based Autoencoders," in *Biennial Symposium on Communications*, 2021.

[63] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proceedings of the 14th ACM international conference on Mobile computing and networking*, 2008, pp. 116–127.

[64] N. O. Tippenhauer, K. B. Rasmussen, C. Pöpper, and S. Čapkun, "Attacks on public WLAN-based positioning systems," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*, 2009, pp. 29–40.

[65] R. Klein, M. A. Temple, M. J. Mendenhall, and D. R. Reising, "Sensitivity Analysis of Burst Detection and RF Fingerprinting Classification Performance," in *2009 IEEE International Conference on Communications*. IEEE, 2009, pp. 1–5.

[66] R. W. Klein, M. A. Temple, and M. J. Mendenhall, "Application of wavelet-based RF fingerprinting to enhance wireless network security," *Journal of Communications and Networks*, vol. 11, no. 6, pp. 544–555, 2009.

[67] M. D. Williams, M. A. Temple, and D. R. Reising, "Aeronautical Mobile Communications System Development Status," *Proceedings of 2010 IEEE Global Communications Conference (GLOBECOM10)*, May 2010.

[68] S.-C. Pei and J.-J. Ding, "Relations between Gabor transforms and fractional Fourier transforms and their applications for signal processing," *IEEE Transactions on Signal Processing*, vol. 55, no. 10, pp. 4839–4850, 2007.

[69] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *International conference on engineering applications of neural networks*. Springer, 2016, pp. 213–226.

[70] M. Schmidt, D. Block, and U. Meier, "Wireless interference identification with convolutional neural networks," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. IEEE, 2017, pp. 180–185.

[71] C. Zhao, X. Wu, L. Huang, Y. Yao, and Y.-C. Chang, "Compressed sensing based fingerprint identification for wireless transmitters," *The Scientific World Journal*, vol. 2014, 2014.

[72] Y. Yuan, Z. Huang, F. Wang, and X. Wang, "Radio Specific Emitter Identification based on nonlinear characteristics of signal," in *2015 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*. IEEE, 2015, pp. 77–81.

[73] B. Lebel, L.-N. Bélanger, M. A. Haji Bagheri Fard, J.-Y. Chouinard, "RF Fingerprinting for 802.15.4 Devices: Combining Convolutional Neural Networks and RF-DNA." Barcelona, Spain: ThinkMind, Nov. 2017, pp. 70–74.

[74] V. Baranova, O. Zeleniy, Z. Deineko, and V. Lyashenko, "Stochastic Frontier Analysis and Wavelet Ideology in the Study of Emergence of Threats in the Financial Markets," in *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*. IEEE, 2019, pp. 341–344.

[75] C. Bertoncini, K. Rudd, B. Nousain, and M. Hinders, "Wavelet fingerprinting of radio-frequency identification (RFID) tags," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 12, pp. 4843–4850, 2012.

[76] N. Benvenuto, F. Piazza, and A. Uncini, "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, 1993, pp. 232–236.

[77] F. Mkadem and S. Boumaiza, "Physically inspired neural network model for RF power amplifier behavioral modeling and digital predistortion," *IEEE Transactions on Microwave Theory and Techniques*, vol. 59, no. 4, pp. 913–923, 2011.

[78] Y. Wu and C. Chen, "The Noise-like Disguised Scheme for Physical Layer Security Using Phase Rotation And Wavelet Transform," in *2018 5th International Conference on Systems and Informatics (ICSAI)*. IEEE, 2018, pp. 823–827.

[79] M. Dehghani, M. Ghiasi, T. Niknam, A. Kavousi-Fard, E. Tajik, S. Padmanaban,and H. Aliev, "Cyber attack detection based on wavelet singular entropy in AC smart islands: False data injection attack," *IEEE Access*, vol. 9, pp. 16 488–16 507, 2021.

[80] M. Nayeripour, A. H. Rajaei, M. M. Ghanbarian, and M. Dehghani, "Fault detection and classification in transmission lines based on a combination of wavelet singular values and fuzzy logic," *Cumhuriyet Üniversitesi Fen-Edebiyat Fakültesi Fen Bilimleri Dergisi*, vol. 36, no. 3, pp. 69–82, 2015.

[81] C. Cattani, "Haar wavelet-based technique for sharp jumps classification," *Mathematical and Computer Modelling: An International Journal*, vol. 39, no. 2-3, pp. 255–278, 2004.

[82] D. Chanda, N. K. Kishore, and A. K. Sinha, "Application of wavelet multiresolution analysis for classification of faults on transmission lines," in *TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region*, vol. 4. IEEE, 2003, pp. 1464–1469.

[83] M. J. Reddy and D. K. Mohanta, "A wavelet-fuzzy combined approach for classification and location of transmission line faults," *International Journal of Electrical Power & Energy Systems*, vol. 29, no. 9, pp. 669–678, 2007.

[84] M. Mufti and G. Vachtsevanos, "Automated fault detection and identification using a fuzzy-wavelet analysis technique," in *Conference Record AUTOTESTCON'95.'Systems Readiness: Test Technology for the 21st Century'*. IEEE, 1995, pp. 169–175.

[85] P. Jokar, N. Arianpoo, and V. C. M. Leung, "Spoofing prevention using received signal strength for ZigBee-based home area networks," in *2013 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2013, pp. 438–443.

[86] I. Chakeres and L. Klein-Berndt, "AODVjr, AODV simplified," *Mobile Computing and Communications Review*, vol. 6, pp. 100–101, 06 2002.

[87] G. Dini and M. Tiloca, "Considerations on security in ZigBee networks," in *2010 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*. IEEE, 2010, pp. 58–65.

[88] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press Cambridge, 2016.

[89] Z. Ming, J. Chazalon, M. M. Luqman, M. Visani, and J-C Burie, "Simple triplet loss based on intra/inter-class metric learning for face verification," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. IEEE, 2017, pp. 1656–1664.

[90] Q. Meng, D. Catchpoole, D. Skillicom, and P. J. Kennedy, "Relational autoencoder for feature extraction," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 364–371.

[91] X. Cao, “A practical theory for designing very deep convolutional neural networks,” *Technical Report*, 2015.

[92] H. Nielsen, “Theory of the backpropagation neural network,” in *International 1989 Joint Conference on Neural Networks*, vol. 1, no. , 1989, pp. 593–605.

[93] Y. Yu, X. Si, C. Hu, and J. Zhang, “A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures,” *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[94] S. Sathyanarayana, “A Gentle Introduction to Backpropagation,” *Numeric Insight, Inc Whitepaper*, p. , 07 2014.

[95] A. A. O. Tay and T. Y. Lin, “Influence of temperature, humidity, and defect location on delamination in plastic IC packages,” *Components and Packaging Technologies, IEEE Transactions on*, vol. 22, pp. 512 – 518, 2000.

[96] V. Nithya and B. Chandran, “Performance Analysis of IEEE 802.15.4 RF Zigbee Transceiver in an Indoor and Outdoor Environment,” *Indian Journal of Science and Technology*, vol. 9, 2016.

[97] C. K. Dubendorfer, B. W. Ramsey, and M. A. Temple, “An RF-DNA verification process for ZigBee networks,” in *2012 Military Communications Conference (MILCOM 2012)*. IEEE, 2012, pp. 1–6.

[98] U. Pesovic, D. Gliech, P. Planinsic, Z. Stamenkovic, and S. Randic, “Implementation of coherent IEEE 802.15.4 receiver on software defined radio platform,” *2015 23rd Telecommunications Forum Telfor (TELFOR)*, pp. 224–227, 2015.

[99] K. Benkic, P. Planinsic, and Z. Cucej, “Custom wireless sensor network based on ZigBee,” 10 2007, pp. 259–262.

[100] B. W. Ramsey, “Improved Wireless Security through Physical Layer Protocol Manipulation and Radio Frequency Fingerprinting,” Ph.D. dissertation, 2014.

[101] D. R. Reising and M. A. Temple, “WiMAX Mobile Subscriber Verification using Gabor-based RF-DNA Fingerprints,” in *Communications (ICC), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1005–1010.

[102] G. Shi and K. Li, “Fundamentals of ZigBee and Wi-Fi,” in *Signal Interference in WiFi and ZigBee Networks*. Springer, 2017, pp. 9–27.

[103] S. Mallat, “Multifrequency channel decompositions of images and wavelet models.” *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 37, no. 12, pp. 2091–2110, 1989.

[104] Y. Nievergelt and Y. Nievergelt, *Wavelets made easy*. Springer, 1999, vol. 174.

[105] R. Merry and M. Steinbuch, “Wavelet theory and applications,” *Literature study, Eindhoven university of technology, Department of mechanical engineering, Control systems technology group*, 2005.

[106] E. Alpaydin, *Introduction to Machine Learning*. MIT Press, 2020.

[107] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[108] B. W. Ramsey, T. D. Stubbs, B. E. Mullins, M. A. Temple, and M. A. Buckner, “Wireless infrastructure protection using low-cost radio frequency fingerprinting receivers,” *International Journal of Critical Infrastructure Protection*, vol. 8, pp. 27–39, 2015.

[109] D. Lazard, “Sequence,” https://en.wikipedia.org/w/index.php?title=Sequence&oldid=1057769072, Accessed on: March 23, 2022.

[110] A. Leon-Garcia, *Probability, Statistics, and Random Processes For Electrical Engineering*. Pearson Education, 2011. [Online]. Available: https://books.google.ca/books?id=-aYuAAAAQBAJ

[111] J. Brownlee, “A Gentle Introduction to LSTM Autoencoders,” https://machinelearningmastery.com/lstm-autoencoders/, accessed on: March 23, 2022, publication date: 2018.

[112] "K. M. Ting", *”Confusion Matrix”*. "Boston, MA": "Springer US", "2017", pp. "260–260". [Online]. Available: "https://doi.org/10.1007/978-1-4899-7687-1_50"

[113] T. Fawcett, “An introduction to ROC analysis,” *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[114] W. E. Cobb, M. A. Temple, R. O. Baldwin, E. W. Garcia, and E. D. Laspe, “Intrinsic physical-layer authentication of integrated circuits,” May 2015, US Patent 9, 036, 891.

[115] F. Chollet, “Keras,” https://keras.io/guides/working_with_rnns/, accessed on: March 23, 2022, publication date: 2015.

[116] N. Srivastava, E. Mansimov, and R. Salakhutdinov, “Unsupervised Learning of Video Representations using LSTMs,” *CoRR*.

[117] F. Chollet, “Keras,” https://keras.io/ja/layers/recurrent/, accessed on: March 23, 2022, publication date: 2015.

[118] ——, “Keras,” https://blog.keras.io/building-autoencoders-in-keras.html, accessed on: March 23, 2022, publication date: 2015.

[119] J. Brownlee, “How to Reshape Input Data for Long Short-Term Memory Networks in Keras,” https://machinelearningmastery.com/reshape-input-data-long-short-term-memory-networks-keras/, accessed on: March 23, 2022, publication date: 2017.