Deep Learning-Powered Radio Frequency Fingerprint Identification: Methodology and Case Study

Guanxiong Shen, Junqing Zhang, *Member, IEEE* Alan Marshall, *Senior Member, IEEE*

***Abstract*—Radio frequency fingerprint identification (RFFI) is an authentication technique that identifies wireless devices by analyzing the characteristics of the received physical layer signals. In recent years, RFFI has been significantly enhanced by deep learning. A neural network (NN) is often leveraged to predict device identity. As a data-driven approach, deep learning requires the collection of large amounts of data for NN training. In addition, the RFFI system should be evaluated on datasets collected under various conditions to assess the system's robustness. However, only a few RFFI datasets are publicly available, and there are no clear guidelines for building an RFFI testbed for data collection. This paper presents a tutorial to build both closed-set and openset RFFI systems. A LoRa-RFFI testbed is created as a case study and the implementation details are described in depth. The LoRa-RFFI testbed involves 60 commercial-off-the-shelf (COTS) LoRa development boards as devices to be identified, and a USRP N210 software-defined radio (SDR) platform for physical layer signal reception. Experiments are carried out using the implemented LoRa-RFFI testbed, and the collected datasets are made publicly available online. It is anticipated that this work can aid the research community in constructing RFFI testbeds and facilitate the development of RFFI research.**

***Index Terms*—Wireless security, device authentication, radio frequency fingerprint, deep learning**

I. Introduction

Wireless devices are becoming ubiquitous in our daily lives, with billions of wireless sensors/devices embedded in cities, homes, and factories [1]. Wireless security becomes a pressing concern due to the massive number of devices. Authentication is the first defense for any wireless network, which prevents unauthorized devices from gaining access to the network and confidential resources. Conventional authentication schemes rely on cryptographic algorithms, which require keys to be installed or agreed upon between legitimate parties. However, it is challenging to ensure the security of the key distribution and management. Moreover, cryptographic authentication schemes rely on software addresses such as media access control (MAC) addresses, which however can be tampered with. Therefore, a lightweight, low-cost and reliable authentication mechanism is needed for wireless security.

Radio frequency fingerprint identification (RFFI) is a potential authentication method suitable for low-cost wireless devices [2]. RF signals are generated by the analog front-end of wireless transmitters, which consists of mixers, oscillators, power amplifiers, antennas, etc. These components inevitably deviate from nominal specifications due to variations in manufacturing processes. Therefore, each transmitter contains device-specific RF hardware impairments, termed *RF fingerprint (RFF)*. The RF impairments cause unique distortion to the emitted RF signals, which allows us to extract RFF from the physical layer signal captured at the receiver and uniquely identify from which device the packet is sent. Traditional RFFI systems rely on handcrafted features, which depend heavily on the quality of the designed feature extraction algorithm. The algorithms design requires expert knowledge of the underlying communication protocols.

Recently, deep learning (DL) is widely employed in RFFI due to its excellent feature extraction capabilities [3], [4]. Although a DL-based RFFI system can achieve excellent identification performance, it is data-hungry as a large number of labeled signals are required to train a well-performing neural network (NN). The lack of comprehensive public datasets and testbed construction tutorials has greatly limited the development of RFFI research. Firstly, there are only a few public RFFI datasets available online, but most of them only focus on specific scenarios. The implemented RFFI system should be tested on the datasets collected in various environments and conditions to evaluate system robustness. Secondly, there are no detailed instructions on how to construct an RFFI system. This has left many researchers unable to quickly build an RFFI testbed or use existing datasets to evaluate the designed algorithms, thus greatly limiting the expansion of the RFFI research community.

This article aims to offer a comprehensive tutorial to guide researchers to build an RFFI testbed. The necessary knowledge of hardware/software requirements for implementing an RFFI system is elaborated, and the collection settings are detailed. The collected dataset is made public for researchers to rapidly evaluate their designed RFFI protocol. The RFFI testbed construction tutorial as well as the closed-set and openset protocols are covered in Section II. Then we use LoRa as a case study to illustrate how to create an RFFI testbed from

Manuscript received xxx; revised xxx; accepted xxx. Date of publication xxx; date of current version xxx. The work was in part supported by UK Royal Society Research Grants under grant ID RGS/R1/191241 and National Key Research and Development Program of China under grant ID 2020YFE0200600. The review of this paper was coordinated by xxx. *(Corresponding author: Junqing Zhang.)*

G. Shen, J. Zhang and A. Marshall are with the Department of Electrical Engineering and Electronics, the University of Liverpool, Liverpool, L69 3GJ, United Kingdom. (email: Guanxiong.Shen@liverpool.ac.uk; junqing.zhang@liverpool.ac.uk; alan.marshall@liverpool.ac.uk)

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier xxx

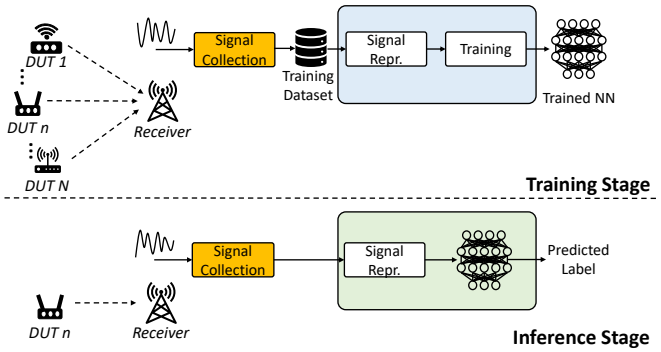

Fig. 1: Overview of a closed-set RFFI system.

scratch, which is presented in Section III. Specifically, we employ 60 LoRa development boards as devices under test (DUTs), and a USRP N210 software-defined radio (SDR) platform as the receiver to capture physical layer signals. In Section IV, a series of experiments are conducted with the implemented LoRa-RFFI testbed and the collected datasets are made public. Section V details the designed closed-set and openset LoRa-RFFI protocols, and results on the collected dataset are provided. Section VI reviews the existing public RFFI dataset and Section VII concludes the work. The testbed implementation and dataset are based on our previous work [2]. The dataset and code are accessible online [2].

II. RFFI System

This section will first introduce the appropriate hardware platforms for an RFFI testbed, including DUTs and receivers. Then the signal collection program is described. The openset and closed-set RFFI protocols are discussed.

A. RFFI Testbed

As shown in Fig. 1, a basic RFFI testbed consists of N transmitters (DUTs) to be identified and a receiver running the signal collection program and RFFI protocol.

1) Transmitters (DUTs): Depending on their programmability, DUTs can be categorized into commercial off-the-shelf (COTS) consumer electronics & IoT devices, IoT development kits, and SDR devices.

Consumer electronics and commercial IoT devices are usually equipped with wireless modules, e.g., smartphones and laptops have WiFi/Bluetooth connectivity. They do not require any programming thus transmission can be easily enabled. However, their transmission parameters are difficult to control such as power, packet interval, payload information, etc.

IoT development kits are also excellent candidates for constructing RFFI testbeds. They are programmable and therefore allow precise control of some transmission parameters. They are highly recommended from the research perspective because they are more flexible compared to consumer electronics and only require a small amount of programming effort.

SDR devices and waveform generators can be customized for any RF waveform. Their RF front-end components are usually more expensive and of higher quality than COTS IoT devices [4]. Furthermore, by using these types of equipment, the characteristics of the transmitted waveform can be modified to emulate DUTs with various hardware impairments. For instance, the authors in [5] use SDRs to emulate DUTs with different values of I/Q mismatch.

2) Receiver: The RFFI systems leverage the physical layer signal, i.e., IQ samples, for identification, which is however inaccessible in most COTS receivers. Therefore, SDR devices are often utilized for signal collection in RFFI research. The SDR platforms capture radio signals and then convert them to baseband IQ samples. All the rest procedures of the communication system, such as packet detection and decoding, are implemented by software. Note that the employed SDR platform must be compatible with the target communication protocol. The main parameters to consider include the range of receiving operating frequency and the bandwidth. For instance, ADALM-PLUTO SDR cannot be used for WiFi 802.11a that operates in the 5 GHz band because it only supports receiving frequencies ranging from 325 MHz to 3.8 GHz. In addition to the SDR platforms, the vector signal analyzer (VSA) can also be used to capture physical layer IQ samples.

B. Signal Collection

The SDR receiver will capture a buffer of IQ samples, which may be composed of valid wireless packets as well as useless Gaussian noise. Therefore, we need to develop a signal collection program to extract the region of interest, i.e., IQ samples of wireless packets, and discard the noise part. The signal collection program mainly includes three necessary steps, namely packet detection, synchronization and carrier frequency offset (CFO) compensation. The packet detection algorithm is first executed at the receiver to determine whether a valid wireless packet is present in the captured IQ samples. If a packet exists, a synchronization algorithm is performed to accurately locate the start point of the received packet. Finally, the CFO of the received signal should be estimated and compensated since the oscillator frequency is sensitive to temperature variations [6].

The signal collection program varies depending on the communication technologies due to different preamble structures. Different SDR boards can utilize the same program to collect wireless packets of a particular protocol.

C. RFFI Protocol

Depending on whether rogue devices are present in the inference stage, RFFI can be categorized into closed-set identification and open-set identification. This section first introduces the signal representation module that is included in both closed-set and openset protocols. Then the protocol details are presented.

1) Signal Representation: The signal representation module converts the collected complex-number IQ samples to appropriate feature representations as NN inputs. Numerous signal representations have been designed in prior RFFI studies. For instance, complex IQ samples can be decomposed into two independent branches, i.e., I and Q, to form a real-number matrix as the NN input [5]. In some studies, the time-domain IQ samples are transformed to frequency spectrum [6], [7], which

can make the signal characteristics more evident and enhance identification performance. The design of signal representation should also take the employed communication technology into consideration. For example, LoRa devices communicate with chirp signals whose frequency linearly changes over time, thus time-frequency domain spectrogram is an appropriate signal representation for LoRa-RFFI systems [6].

2) Closed-Set Identification Protocol: As shown in Fig. 1, closed-set RFFI consists of two stages: training and inference. All DUTs in the inference stage are assumed to be already involved in the training process.

Training: A training dataset is first created by capturing a large number of packets from the N legitimate devices. The corresponding transmitter labels are also saved to the training dataset since the NN is typically trained in a supervised manner. The collected packets are then transformed into designed signal representations as NN inputs. When sufficient packets are available in the training dataset, a NN is constructed and its parameters are updated with the training data. The model needs to be re-trained when new devices join the communication network.

There are many varieties of NNs investigated in previous RFFI studies, including convolutional neural network (CNN) [2], [8], [9], recurrent neural network (RNN) [10], long short-term memory (LSTM) [6], [8] network and transformer. The characteristics of the signal representation should be taken into account when designing the neural network. For instance, 1D CNNs can be used to process IQ samples because they are efficient for handling time series data [9], whereas 2D CNNs are more appropriate for spectrograms as they are effective for image-like data [2]. The NNs can be implemented by numerous DL libraries such as PyTorch, TensorFlow, MATLAB Deep Learning Toolbox, etc.

Inference: The collected IQ samples are converted to the same signal representation as that in the training stage and fed into the NN, with the output being the predicted device label.

3) Openset Identification Protocol: The openset RFFI protocol designed in [2] is illustrated in Fig. 2. It is composed of three stages, namely training, enrollment, and inference. In contrast to the closed-set RFFI system, the openset RFFI system can identify DUTs that are not included during training. This is significant for RFFI since the data from rogue devices and attackers are never available during training.

Training: We first collect signals from M DUTs to train an NN-based feature extractor instead of a classification NN. The input to the feature extractor is the signal representation transformed from the IQ samples and the output is the extracted RFF. Note that the M DUTs used during training are not necessarily involved in the latter stages. Therefore, the training stage can be carried out by a third party with a large number of DUTs to enhance the generalization of the NN-based feature extractor. In the openset protocol, the NN serves as a feature extractor but not a classifier, and thus is not needed to be re-trained when new devices join.

Enrollment: An enrollment stage is additionally introduced between the training and inference stages. We collect signals from the N legitimate DUTs and extract their RFFs with the trained feature extractor. The extracted RFFs and corresponding transmitter labels are then saved into an RFF database. The enrollment stage should be conducted in a controlled environment where only legitimate DUTs exist.

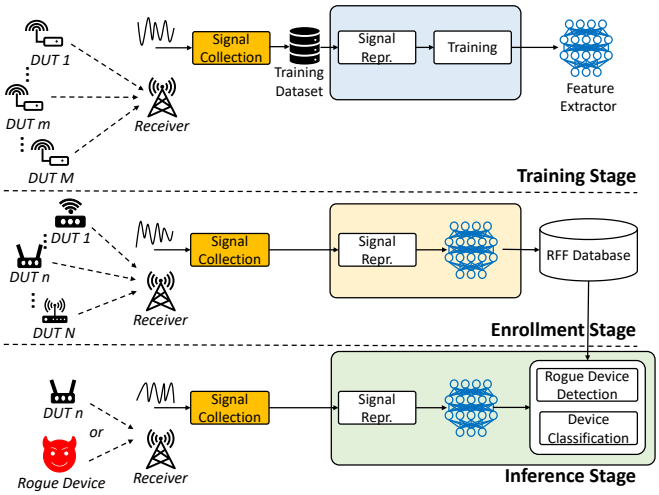

Fig. 2: Overview of an openset RFFI system.

Inference: The inference stage of an openset RFFI protocol is required to accomplish two tasks, namely rogue device detection and device classification. The first task determines whether the received signal is from a legitimate device, and the device classification module further predicts the specific identity of a legitimate DUT. Note that the inference stage can be implemented by simple distance measurement such as the k-nearest neighbor (kNN) algorithm [2].

D. Summary

As can be observed, all the RFFI operations are implemented at the receiver side and no modification is required at the transmitters. This unique feature makes RFFI extremely suitable for legacy and future IoT networks, as RFFI can be deployed with a specialized receiver without upgrading the transmitter firmware. In addition, RFFI does not require a dedicated packet but can piggyback on existing packets, which will not drain transmitter energy and can achieve a per-packet authentication.

III. LoRa-RFFI Case Study: Testbed

LoRa is taken as a case study to illustrate how to construct an RFFI testbed. Both the hardware equipment and signal collection program are illustrated. The implemented LoRa-RFFI testbed is based on the work in [2].

A. LoRa-RFFI Testbed

1) LoRa DUTs: We use 60 COTS LoRa development boards as DUTs to be identified, which are shown in the upper part of Fig. 3. To make the testbed more representative, four models of LoRa development boards are employed since there are likely a variety of transmitters in a LoRa network. Specifically, we use 45 Pycom LoPy4, five Mbed SX1261 shields, five Pycom FiPy, and five Dragino SX1276 Shields

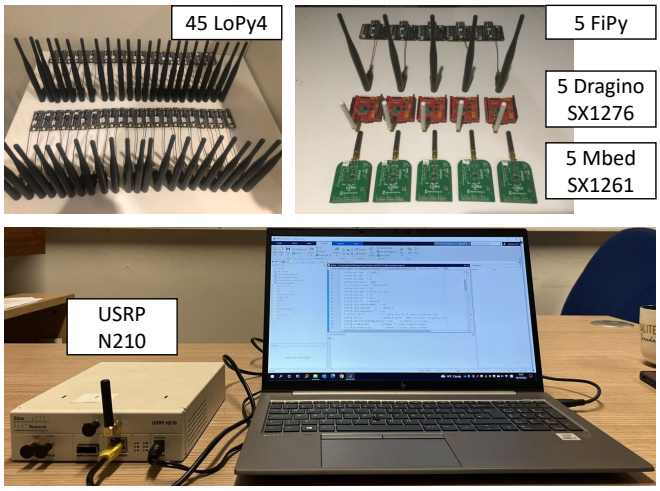

Fig. 3: Experimental devices. 60 LoRa DUTs and a USRP N210 SDR connected to a PC.

for experiments. The LoPy4 and FiPy are micropython-programmable. The Mbed SX1261 and Dragino SX1276 are based on Mbed and Arduino platforms, respectively, and are programmed in C/C++ language.

The spreading factor (SF) and bandwidth of LoRa devices are configured as seven and 125 kHz, respectively. They continuously transmit LoRa packets at 868.1 MHz with 'Hello' as the payload. Note that the payload content is not relevant in this work because only the preamble part is leveraged for identification, which is consistent for all the LoRa packets.

2) SDR Receiver: A USRP N210 SDR platform with a UBX 40 full-duplex daughterboard is used to implement the LoRa receiver, which is shown in the lower part of Fig. 3. The USRP devices support various software development frameworks, such as Universal Hardware Driver (UHD), MATLAB, GNU Radio, LabVIEW, etc. The users can design customized signal processing algorithms with these platforms.

The sampling rate of USRP N210 is configured as 1 MHz, which is eight times oversampling compared to the transmission bandwidth of 125kHz. The receiver gain is set to 0 dB.

B. Signal Collection

In our testbed, the USRP N210 SDR is connected to a Windows PC. MATLAB is used to access the USRP N210 and implement the signal collection program. The MATLAB Communications Toolbox provides a system object *comm.SDRuReceiver* to configure radio parameters and receive data from the USRP board, which is based on the UHD driver. According to our setting, this system object returns a column vector of 375,000 complex numbers once it is called, which is the captured RF signal.

The signal collection module aims to extract valid LoRa packets from the captured IQ samples. Three steps are needed to complete the signal collection, namely packet detection, synchronization, and CFO compensation. Then the packet preamble part is extracted for RFFI. The algorithms are explained in [6], which are specially designed for LoRa signals. The designed MATLAB signal collection program can be used for any SDR platform by simply replacing the *comm.SDRuReceiver* system object with the adopted SDR platform. Note that it is crucial to verify the signals are correctly captured. This can be achieved by comparing the collected signals to the simulated LoRa preamble waveform.

IV. LoRa-RFFI Case Study: Dataset

This section elaborates on the datasets collected with the LoRa-RFFI testbed, including the experimental environments and dataset information.

A. Dataset Overview

The dataset consists of 16 sub-datasets that are collected in numerous conditions and thus can be leveraged for various evaluation purposes. Each sub-dataset corresponds to an HDF5 file, and the details are shown in Table I. The considered collection conditions include which DUTs are involved, environments, whether LOS is included, channel condition, and antenna polarization direction. In addition to the real-collected datasets, several augmented test datasets are also provided. Data augmentation can emulate more channel conditions by feeding the collected signals into a wireless simulator, which is detailed in Section IV-B2.

B. Dataset Details

Each HDF5 file contains a number of LoRa signals, i.e., IQ samples of the preamble part, as well as device labels. Some HDF5 files contain the real-collected signals, others contain the signals augmented by a wireless channel simulator.

1) Real-Collected Dataset: The variations of wireless channels can degrade the RFFI system performance, posing a significant obstacle for RFFI development [2], [9]. The impact of wireless channels is reflected in two aspects, namely the multipath effect and the Doppler effect. The multipath effect is determined by the RF signal propagation environment, while the Doppler effect is caused by the movement of RF transceivers and/or surrounding objects. Therefore, it is expected that the dataset contains signals collected in various locations and in moving scenarios.

We first collect some sub-datasets in a residential room. The LoRa transmitter and USRP N210 receiver are roughly placed half a meter apart, with line-of-sight (LOS) between them. The surrounding objects remain static during signal collection. This is nearly the ideal case since both multipath and Doppler effects are not strong.

The signal collection is also carried out in an office building, whose floor plan is given in Fig. 4. As shown in the figure, the receiver is located in an office room, and the LoRa transmitters are in turn placed at six locations A-F, respectively. Both LOS and non-line-of-sight (NLOS) scenarios are involved. We further consider three scenarios.

- Stationary scenario: the transmitter and receiver, as well as the surrounding objects, remain static. The channel coherence time is sufficiently long and therefore the channel can be assumed constant during signal collection.
- Mobile scenario: the USRP N210 receiver is static, but the LoRa transmitter is carried by a person walking at

TABLE I: Dataset Information

Sub-dataset Name	DUT Index	# Packets Per Dev	Environment	LOS/NLOS	Channel Condition	Ant. Polar.	Augmentation
dataset_training_aug.h5	1 - 30	1,000	Residential	LOS	Stationary	Linear	Yes
dataset_training_aug_0hz.h5	1 - 30	1,000	Residential	LOS	Stationary	Linear	Yes
dataset_training_no_aug.h5	1 - 30	500	Residential	LOS	Stationary	Linear	No
dataset_seen_devices.h5	1 - 30	400	Residential	LOS	Stationary	Linear	No
dataset_rogue.h5	41-45	200	Residential	LOS	Stationary	Linear	No
dataset_residential.h5	31-40	400	Residential	LOS	Stationary	Linear	No
dataset_other_device_type.h5	46-60	400	Residential	LOS	Stationary	Linear	No
A.h5	31-40	200	Office loc. A	LOS	Stationary	Linear	No
B.h5	31-40	200	Office loc. B	LOS	Stationary	Linear	No
C.h5	31-40	200	Office loc. C	LOS	Stationary	Linear	No
D.h5	31-40	200	Office loc. D	NLOS	Stationary	Linear	No
E.h5	31-40	200	Office loc. E	NLOS	Stationary	Linear	No
F.h5	31-40	200	Office loc. F	NLOS	Stationary	Linear	No
B_walk.h5	31-40	200	Office loc. B	LOS	Object moving	Linear	No
F_walk.h5	31-40	200	Office loc. F	NLOS	Object moving	Linear	No
moving_office.h5	31-40	200	Office	LOS	Mobile	Linear	No
moving_meeting_room.h5	31-40	200	Office	NLOS	Mobile	Linear	No
B_antenna.h5	31-40	200	Office loc. B	LOS	Stationary	Orthogonal	No
F_antenna.h5	31-40	200	Office loc. F	NLOS	Stationary	Orthogonal	No
A_aug_0hz.h5	31-40	200	Office loc. A	LOS	Stationary	Linear	Yes
A_aug_10hz.h5	31-40	200	Office loc. A	LOS	Stationary	Linear	Yes
A_aug_30hz.h5	31-40	200	Office loc. A	LOS	Stationary	Linear	Yes
A_aug_50hz.h5	31-40	200	Office loc. A	LOS	Stationary	Linear	Yes
A_aug_100hz.h5	31-40	200	Office loc. A	LOS	Stationary	Linear	Yes

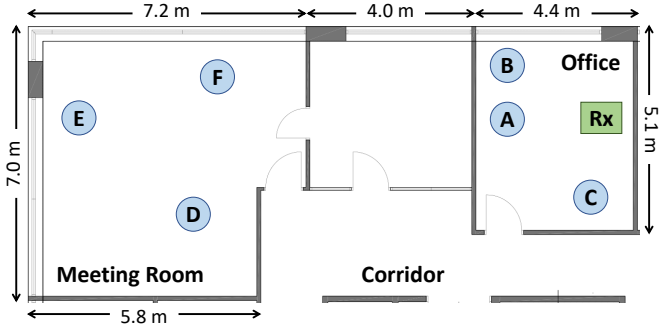

Fig. 4: Floor plan.

a speed of v =2 m/s. The channel coherence time is calculated to be around $c/(v \times f_c) = 0.1728$ s, where c is the light of speed in free space, and f_c is the carrier frequency which is 868.1 MHz in this paper.

- Object moving scenario: the transceivers are kept static but there is one person moving between them at a normal speed of around 2 m/s. The Doppler effect in this setting is weaker than that in the mobile scenario, therefore the coherence time should be less than 0.1728 s.

These channel conditions cover most indoor communication situations, which can be used to evaluate the LoRa-RFFI performance in the indoor environment.

Another factor to consider is the polarization direction of the transmitter and receiver antennas. The RFFI accuracy decreases when the antenna polarization in the training and test datasets differs. This has been investigated in earlier transient feature-based studies but is often overlooked in recent DL-based work. It is critical to explore how antenna polarization affects signal characteristics and how to improve the system's robustness in response to this. The dataset contains signals collected under both orthogonal and linear polarization directions. Linear polarization refers to the two antennas being aligned and having the same polarization, while Orthogonal polarization means that the two antennas have orthogonal polarization directions.

2) Augmented Dataset: Some channel conditions are difficult to create by conducting experiments. For instance, the rapid movement of LoRa transceivers can cause severe Doppler effects, but it is challenging to conduct experiments at such a high speed. Alternatively, we can feed the collected data into a wireless channel simulator, which can emulate a variety of channel conditions.

In the data augmentation, each signal is sent through a tapped delay line (TDL) channel model with various multipath and Doppler effects. More specifically, the multipath is described by the exponential power delay profile (PDP) while the Doppler effect can be depicted by Jake's Doppler power spectrum. The details/mathematical expressions of the wireless channel simulator can be found in [2].

V. LoRa-RFFI Case Study: Protocols and Results

This section describes the designed closed-set and openset LoRa-RFFI protocols and the results on the collected dataset. The code and dataset used in this section are available in [2].

A. Signal Representation

The channel-independent spectrogram proposed in [2] is used as the signal representation, which can mitigate the channel effects. The impact of wireless channels is a huge challenge in the development of RFFI research. The RFFI performance can be severely degraded when channel conditions are inconsistent with the training data, which is unacceptable given that most wireless devices are mobile. To overcome this, we design mitigation algorithms in the time-frequency domain. More details about the derivation of channel-independent spectrogram can be found in [2].

B. Closed-Set Identification

1) Protocol Implementation: The closed-set RFFI protocol is implemented with a classification CNN.

Training: We leverage the TensorFlow library to construct a classification CNN. A softmax-activated fully connected layer is subsequently connected to the deep learning model proposed in [2] to perform the classification function. The CNN is updated by the RMSprop optimizer with an initial learning rate of 0.001. The learning rate decreases by 80% when the validation loss does not reduce for 10 epochs. The training stops when validation loss plateaus for 30 epochs.

Inference: The input to the CNN is the channel-independent spectrogram converted from the received IQ samples, and the output is a probability vector over all the predictable DUTs. The input and output dimensions are (102, 62, 1) and (30, 1), respectively. The prediction result is the DUT with the highest probability.

2) Results: We use the IQ samples and device labels stored in 'dataset_training_aug.h5' to train the CNN. 10% of the signals are used for validation and the rest are used for training. The saved IQ samples are converted to channel-independent spectrograms as NN inputs. Then we use 'dataset_seen_devices.h5' to test the trained CNN. The classification result is depicted in Fig. 5. It is a confusion matrix often used to visualize the classification results. The overall accuracy reaches 99.6% which means the trained CNN can classify DUTs 1-30 with high accuracy.

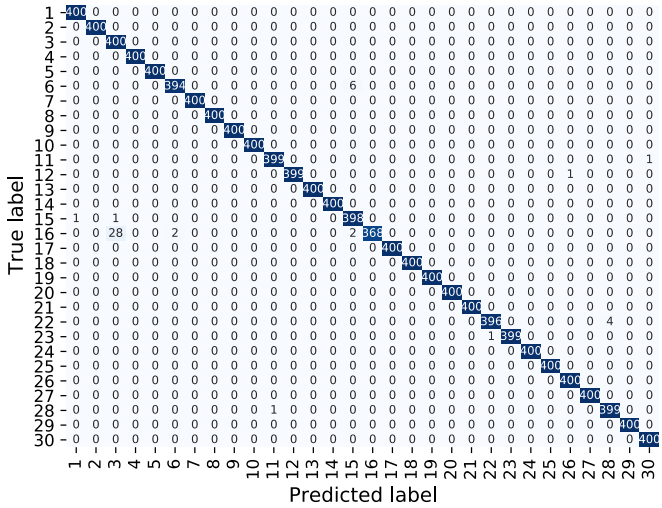

Fig. 5: Classification result of the closed-set RFFI system. The overall accuracy is 99.6%

C. Openset LoRa-RFFI Protocol

1) Protocol Implementation: The openset RFFI protocol is implemented with an NN-based feature extractor, an RFF database, and the kNN algorithm.

Training: The architecture of the feature extractor is given in [2], which is implemented with the TensorFlow library as well. Triplet loss is used to train the feature extractor. Note that it is supervised training as the ground truth device labels are used to select positive, negative, and anchor samples. The utilized optimizer is RMSprop and the learning rate scheduler is exactly the same as that introduced in Section V-B. Consistent with the training settings for the closed-set RFFI protocol, 10% of the signals are used for validation and the rest are used for training. The trained feature extractor accepts the channel-independent spectrogram, i.e., a (102, 62, 1) tensor, and outputs a vector of 512 elements, i.e., RFF.

Enrollment: The N legitimate DUTs are required to enroll their RFFs into an RFF database. They transmit wireless signals to the receiver and the trained feature extractor is utilized to extract RFFs. The corresponding labels are also saved in the RFF database for use in the inference stage.

Inference: Th kNN algorithm is used to realize openset identification, which involves rogue device detection and device classification. The received IQ samples are first transformed into a channel-independent spectrogram and the RFF is extracted. A detection score is then calculated as the average Euclidean distance to the RFF's k nearest neighbors in the RFF database. The signal is considered to be from a rogue device when the detection score is higher than a predefined threshold. If the detection score is lower than the threshold, the signal is deemed to be from the most frequent DUT in its k nearest neighbors. The number of neighbors k is set to 15 according to our empirical optimization.

2) Results: We use 'dataset_training_aug.h5' to train the feature extractor and 'dataset_residential.h5' to create the RFF database in the enrollment stage. Then we use 'dataset_rogue.h5' to test the rogue device detection performance. The result shows that the area under the curve (AUC) reaches 0.9905 which indicates excellent rogue device detection ability. The dataset 'A.h5' is used to evaluate the device classification and the overall accuracy is over 95%.

VI. Review of Existing Datasets

There are currently four LoRa datasets available online [2], [7], [8], [11]. Elmaghbub *et al.* use a USRP B210 to collect signals from 23 LoPy4 and 2 FiPy boards. The authors carry out experiments in three environments. Al-Shawabka *et al.* collect packets from 100 FiPy development boards with a USRP N210. The data collection is carried out in both an

indoor wireless testbed and outdoor environment [8]. The authors in [7] conduct experiments within a building. The receiver is a USRP B210 and 22 LoRa transmitters of five different models are used as DUTs to be identified. Compared to the above-introduced datasets, the dataset presented in this paper additionally incorporates the signals collected in moving scenarios and takes into account the conditions of antennas [2].

Some WiFi-RFFI datasets are open source as well [5], [9], [12]. Sankhe *et al.* take 16 USRP X310 SDRs as DUTs to transmit WiFi 802.11a packets and use a USRP B210 for the reception [5]. Similarly, Al-Shawabka *et al.* employ SDRs as DUTs as well. Specifically, they use 13 USRP N210 and seven USRP X310 for transmitting and a USRP N210 for receiving [9]. In addition to the SDRs, the authors in [12] use 174 COTS WiFi NICs as DUTs, and 41 USRP SDRs of three models as the receivers.

There are also publicly available datasets based on other communication protocols besides LoRa and WiFi. The authors of [13] use RFFI to identify seven COTS DJI M100 UAVs that employ a proprietary communication protocol. Liu *et al.* use a USRP B210 to collect ADS-B signals from 130 flying aircrafts [14]. Piva *et al.* release an RFFI dataset for UHF RFID research as open source, containing signals collected from 200 COTS RFID tags [15].

VII. Conclusion

This article presents an easy-to-follow tutorial on constructing an RFFI testbed and building RFFI protocols. We list the required hardware devices, i.e., transmitters and receivers, and introduce both the closed-set and openset RFFI protocols. After that, LoRa is taken as a case study to show how to design RFFI systems in detail. We present the used hardware devices and the designed closed-set and openset LoRa-RFFI systems. We collect a number of datasets leveraging the implemented LoRa-RFFI testbed and make them open-source. Both the collection scheme and dataset are fully detailed. The collected dataset is used to evaluate the implemented closed-set and openset LoRa-RFFI systems. The results show that the designed LoRa-RFFI systems have excellent performance. Finally, we summarize the existing public dataset available for RFFI research.

References

[1] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6G networks: Use cases and technologies," *IEEE Commun. Mag.*, vol. 58, no. 3, pp. 55–61, 2020.

[2] G. Shen, J. Zhang, A. Marshall, and J. R. Cavallaro, "Towards scalable and channel-robust radio frequency fingerprint identification for LoRa," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 774–787, 2022, Dataset and code are available: https://ieee-dataport.org/open-access/lorarffidataset, Last accessed on 2023-01-29.

[3] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury, "Deep learning convolutional neural networks for radio identification," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 146–152, 2018.

[4] M. Nair, T. A. Cappello, S. Dang, and M. A. Beach, "Rigorous analysis of data orthogonalization for self-organizing maps in machine learning cyber intrusion detection for LoRa sensors," *IEEE Trans. Microw. Theory Techn.*, 2022.

[5] K. Sankhe, M. Belgiovine, F. Zhou, S. Riyaz, S. Ioannidis, and K. Chowdhury, "ORACLE: Optimized radio classification through convolutional neural networks," in *Proc. IEEE INFOCOM*, 2019, pp. 370–378.

[6] G. Shen, J. Zhang, A. Marshall, L. Peng, and X. Wang, "Radio frequency fingerprint identification for LoRa using deep learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2604–2616, 2021.

[7] P. Robyns, E. Marin, W. Lamotte, P. Quax, D. Singelée, and B. Preneel, "Physical-layer fingerprinting of LoRa devices using supervised and zero-shot learning," in *Proc. ACM WiSec*, 2017, pp. 58–63.

[8] A. Al-Shawabka, P. Pietraski, S. B Pattar, F. Restuccia, and T. Melodia, "DeepLoRa: Fingerprinting LoRa devices at scale through deep learning and data augmentation," in *Proc. ACM MobiHoc*, Jul. 2021.

[9] A. Al-Shawabka, F. Restuccia, S. D'Oro, T. Jian, B. C. Rendon, N. Soltani, J. Dy, K. Chowdhury, S. Ioannidis, and T. Melodia, "Exposing the fingerprint: Dissecting the impact of the wireless channel on radio fingerprinting," in *Proc. IEEE INFOCOM*, Jul. 2020, pp. 646–655.

[10] D. Roy, T. Mukherjee, M. Chatterjee, E. Blasch, and E. Pasiliao, "RFAL: Adversarial learning for RF transmitter identification and classification," *IEEE Trans. on Cogn. Commun. Netw.*, vol. 6, no. 2, pp. 783–801, 2019.

[11] A. Elmaghbub and B. Hamdaoui, "LoRa device fingerprinting in the wild: Disclosing RF data-driven fingerprint sensitivity to deployment variability," *IEEE Access*, vol. 9, pp. 142 893–142 909, 2021.

[12] S. Hanna, S. Karunaratne, and D. Cabric, "WiSig: A large-scale WiFi signal dataset for receiver and channel agnostic RF fingerprinting," *IEEE Access*, vol. 10, pp. 22 808–22 818, 2022.

[13] N. Soltani, G. Reus-Muns, B. Salehihikouei, J. Dy, S. Ioannidis, and K. Chowdhury, "RF fingerprinting unmanned aerial vehicles with non-standard transmitter waveforms," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15 518–15 531, 2020.

[14] Y. Liu, J. Wang, J. Li, S. Niu, and H. Song, "Class-incremental learning for wireless device identification in IoT," *IEEE Internet Things J.*, vol. 8, no. 23, pp. 17 227–17 235, 2021.

[15] M. Piva, G. Maselli, and F. Restuccia, "The tags are alright: Robust large-scale RFID clone detection through federated data-augmented radio fingerprinting," in *Proc. ACM MobiHoc*, Jul. 2021.

Guanxiong Shen received the B.Eng. degree from Xidian University, China, in 2019. He is currently pursuing a Ph.D. degree at the University of Liverpool, U.K. His current research interests include the Internet of Things and wireless security.

Junqing Zhang is a Lecturer (an Assistant Professor) at the University of Liverpool, U.K. His research interests include the Internet of Things, wireless security, physical layer security, key generation, radio frequency fingerprint identification, and wireless sensing. He was a recipient of the U.K. EPSRC New Investigator Award.

Alan Marshall currently holds the Chair in communications networks at the University of Liverpool, where he is the Director of the Advanced Networks Group and the Head of the Department. He has spent over 24 years working in the telecommunications and defense industries. He has published over 250 scientific articles and holds a number of joint patents in the areas of communications and network security. He formed a successful spin-out company Traffic Observation and Management Ltd. His research interests include mobile and wireless network architectures and protocols, network security, and multisensory communications, including haptics and olfaction. He is a fellow of the Institution of Engineering and Technology and a Senior Fellow of the Higher Education Academy. He is also a Section Editor of the Computer Journal (British Computer Society) and an Editorial Board Member of the Journal of Networks.