TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 04/14 pp862–872
DOI: 10.26599/TST.2022.9010059
Volume 28, Number 5, October 2023

Bluetooth Low Energy Device Identification Based on Link Layer Broadcast Packet Fingerprinting

Jinghui Zhang, Xinyang Li, Junhe Li, Qiangsheng Dai, Zhen Ling, and Ming Yang*

Abstract: With the rapid development of the Internet of Things (IoT), wireless technology has become an indispensable part of modern computing platforms and embedded systems. Wireless device fingerprint identification is deemed as a promising solution towards enhancing the security of device access authentication and communication process in the IoT scenario. However, the extraction of features from the network layer and its upper layers often confront restrictions from specific devices: the association with a certain wireless network and the access to the plaintext of the payload. Meanwhile, Bluetooth Low Energy (BLE) packets have been encrypted above the link layer, which makes those features difficult to extract. To tackle these problems, we introduce a novel method to identify BLE devices based on the fingerprint features in the data link layer. Initially, the BLE packets are collected through a receiver based on software-defined radio technology. Then, fields that reflect device differences in BLE broadcast packets are extracted through traffic analysis. Finally, a MultiLayer Perceptron (MLP) model is employed to recognize the category of BLE devices. An experimental result on a dataset with 15 types of BLE devices shows that the identification accuracy of the proposed method can reach 99.8%, which accomplishes better performance over previous work.

Key words: link layer fingerprint; software-defined radio; Bluetooth Low Energy (BLE)

1 Introduction

According to International Data Corporation's forecast[1], by 2025, there will be 41.6 billion IoT devices worldwide, generating 79.4 zettabytes of data. As the Internet of Things (IoT) continues to grow and evolve, wireless technology is becoming an essential component of modern computing platforms and embedded systems[2, 3]. As a result, there is a growing necessity to precisely identify devices in wireless networks. Figure 1 illustrates that without device identification systems, IoT can be easily attacked through object emulation[4]. Several security solutions have been proposed to detect abnormal network behaviors[5]. During the phase of wireless devices accessing to a network, traditional cryptography-based authentication protocols are difficult to implement due to the limited computing resources of IoT devices. Therefore, extracting unique and non-replicable device identifiers (e.g., device fingerprinting) becomes a possible solution for achieving authentication.

The basic idea of wireless device fingerprinting is to identify wireless devices by extracting their unique patterns from the target device during wireless communication in a passive or active manner. Various features can be extracted from both the link layer and the upper layers of the protocol stack to generate fingerprints

• Jinghui Zhang, Xinyang Li, Zhen Ling, and Ming Yang are with School of Computer Science and Engineering, Southeast University, Nanjing 211189, China. E-mail: {jhzhang, lixinyang, zhenling, yangming2002}@seu.edu.cn.
• Junhe Li is with School of Cyber Science and Engineering, Southeast University, Nanjing 211189, China. E-mail: lijunhe@seu.edu.cn.
• Qiangsheng Dai is with Research Institute, State Grid Jiangsu Electric Power Company Ltd., Nanjing 210024, China. E-mail: day_qs@163.com.
* To whom correspondence should be addressed.
Manuscript received: 2022-11-24; accepted: 2022-12-10

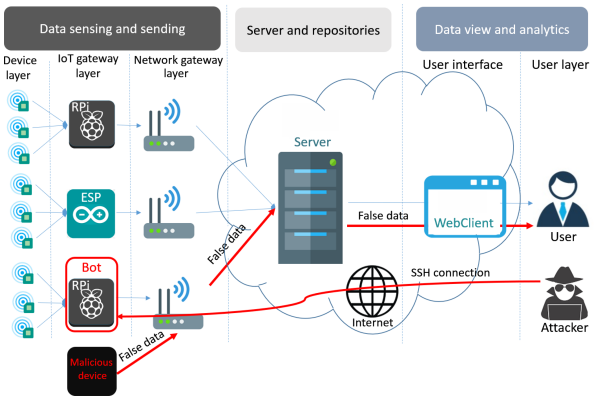

Fig. 1 Object emulation attack and Botnet attack model in IoT.

for each layer. The fingerprint above the link layer extracts recognizable keywords and their sequences, packet lengths, and other information from the traffic generated at the transport and application layers as features. The extraction of upper layer fingerprints often requires that the device to be identified is associated with a wireless network and has access to the payload plaintext. However, many wireless protocols are encrypted above the link layer. For example, the current mainstream Bluetooth Low Energy (BLE) protocol uses LE encryption. Many recognition systems based on upper-layer fingerprints become narrowly applicable or even difficult to implement.

Although recent years have witnessed research efforts devoting to link layer fingerprint, these works face some shortcomings. Firstly, it is noteworthy that current research on link layer fingerprinting primarily concentrates on the classical Bluetooth protocol while neglecting other wireless protocols in the IoT domain, such as the BLE protocol. Hence, a critical gap exists in our understanding of link layer fingerprinting techniques for these other protocols. Secondly, real-time performance continues to be a significant challenge in link layer fingerprinting research. This is primarily attributed to the requirement for certain features to necessitate device traffic periods for recognition, which consequently makes fingerprint recognition algorithms that utilize these features time-consuming. Thirdly, the identification of unconnected Bluetooth devices through fingerprinting remains a persistent challenge. To utilize the arrival time of Bluetooth packets as a feature, both master and slave devices must be connected before the fingerprint can be extracted. Considering the aforementioned points, a fingerprint acquisition system is essential for real-time identification of unconnected BLE devices in link layer.

The characteristics of broadcast packets are leveraged to enable fast BLE device identification in link layer, including frequent broadcasting, plaintext transmission, rich device information, and passive listening and acquisition[6–8]. When considering which features to utilize as fingerprints, estimated broadcast interval time is a potential choice. However, the hopping frequency can affect the accuracy of the estimated device broadcast interval, rendering the device less resilient to forgery. Presently, research involving Bluetooth broadcast packets relies on explicit identifiers, such as Media Access Control (MAC) addresses[9], which are highly susceptible to falsification. To mitigate the issue of vulnerability to deception, we adopt link layer broadcast packet fingerprinting to identify BLE devices with a binary granularity of $\langle device\ type, vendor \rangle$.

In addition, it is important to identify unconnected devices and address the challenges on using broadcast time interval as a fingerprint[10]. Thus, we employ software radio technology[11] to capture BLE broadcast packets on the Bluetooth broadcast channel. Traffic analysis is conducted to explore the fields in the Bluetooth broadcast frames that reflect device differences[12]. Subsequently, the commonly used broadcast packets of types ADV_IND and ADV_NONCONN_IND are filtered to generate fingerprints[13]. Given that explicit identifiers can be easily falsified, we eliminate all the explicit identifiers such as MAC address, as well as device name and member UUID[14, 15].

Our contributions include exploring fields in the link layer frames that reflect the differences between devices from different vendors, studying and designing relevant algorithms for feature extraction and explicit identifiers removal at the link layer, and generating fingerprints of target devices to complete device identification at the link layer. Using this approach, the accuracy of identification of 15 types of BLE devices can reach up to 99.8%.

2 Related Work

2.1 Link layer device fingerprinting characteristics

Numerous studies have been dedicated to extracting features by analyzing link layer information to generate device fingerprints. This approach is advantageous because it does not require specialized hardware devices and is relatively simple to implement. Link layer features are typically derived from details that are unspecified

in the protocol, and their implementation is left up to the vendors. Consequently, link layer features often correspond to specific vendors.

For Bluetooth, the clock offset characteristics can be extracted from the device's clock. Huang et al.[16] exploited the temporal characteristics of Bluetooth frequency hopping, which extracts the transmitter clock offset by measuring the rate at which the received packets deviate from the time slot boundary. Experiments show that their identification system can accurately distinguish between 94.3% of the 56 Bluetooth devices. The arrival interval distribution of Bluetooth packets can also be used as a feature. The packet arrival interval vector was calculated by Aksu et al.[17] from the Bluetooth packet dataset. The authors then generated the vector's density distribution, and finally converted the distribution into a histogram, in which the height of each point was regarded as a feature.

There are a few issues with the aforementioned techniques. Firstly, extracting fingerprints using these techniques is a time-consuming process, so the techniques are useless when there is a need for faster throughput. Secondly, all the techniques focus on the classical Bluetooth protocol and exclude BLE, which is now increasingly popular in the IoT. Thirdly, the aforementioned techniques require the master and slave devices to establish a connection before fingerprint extraction. As a result, it is impossible to identify unconnected devices using the approaches mentioned above.

2.2 Feature processing and fingerprint algorithms

After the feature extraction stage, the extracted features for generating fingerprints will be processed and then used to recognize device category. As the extracted features may exhibit inter-dependencies, it is commonplace to apply dimensionality reduction techniques in order to decrease the feature space.

Dimensionality reduction can be applied either prior to classification or in conjunction with a classifier. A number of various dimensionality reduction analysis methods performed before classification have been applied to the field of Radio Frequency (RF) fingerprinting[18–20], including entropy analysis[21], Kolmogorov-Smirnov test[22, 23] and Principle Component Analysis (PCA)[24, 25].

Fingerprint generation and recognition algorithms generally fall into two categories: similarity-based and classification-based. Similarity-based algorithms generally require quantification of the extracted features to represent the device's fingerprints as a vector. In order to make recognition, it calculates the similarities between the new fingerprint and each fingerprint in the database, and sets a reasonable threshold. A common similarity measure for vectors is cosine similarity, and other similarity measures are also used depending on the nature of the features. Classification-based recognition algorithms treat device fingerprints as specific distribution[26]. For example, the PARADIS system proposed by Brik et al.[27] uses classification algorithms to identify IEEE 802.11 Network Interface Controllers (NIC)[28]. They extract features of the signal in the modulation domain and use classical classification algorithms, including k-Nearest Neighbor (kNN) and Support Vector Machine (SVM), to identify more than 130 NICs.

3 Approach

3.1 Methodology overview

Considering the characteristics of BLE broadcast packets, like frequent broadcast, plaintext transmission, rich device information, and passive listening and acquisition, we decide to use BLE broadcast packets for link layer identification of BLE devices. An overview of the BLE device identification system is shown in Fig. 2. The detailed procedure is listed as follow:

(1) Acquisition of BLE packets. In order to acquire the link layer data of BLE packets, we build a Gaussian filtered Minimum Shift Keying (GMSK) I/Q demodulation receiver based on Universal Software Radio Peripheral (USRP) and GNU Radio using software radio technology. The receiver listens on channels (i.e., 37, 38, and 39) at certain time intervals. The BLE signal goes through frame detection[29], Bluetooth signal filtering and extraction, and GMSK demodulation to the link layer. Finally Pcap files are

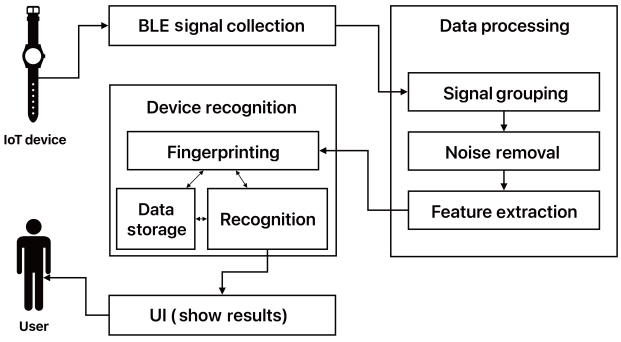

Fig. 2 Overview of the BLE device identification system.

generated and the acquisition of BLE broadcast packets is achieved.

(2) Analysis of broadcast packet characteristics. In order to determine whether the BLE broadcast packets can provide enough information for device identification, we use the collected broadcast packets, combined with the protocol specification, to analyze the differences in broadcast content and protocol implementation of different types of BLE devices from various vendors in IoT. In addition, we look for other features by observing how the devices apply different broadcast packet types and the meaning of broadcast packet header and different types of broadcast data. We find rich information relating to devices in the metadata fields of the packet header, flags, service data, service UUID, and manufacturer specific.

(3) Recognition of BLE device based on link layer broadcast packet fingerprint. Our goals include automating feature extraction, making full use of the structural features defined by manufacturer specific carrier, and improving the robustness in data variation of packets from devices of the same kind. Therefore, we choose a lightweight deep learning model to complete the classification and identification. Furthermore, we remove the explicit identifiers that can be easily forged from broadcast packets of types ADV_IND and ADV_NONCONN_IND and process the data accordingly. After that, a frame of the packet is converted into a 200-dimensional vector and input into a MultiLayer Perceptron (MLP) model with two hidden layers. A Dropout layer is added between the hidden layers to prevent overfitting. Finally, the trained model can make a recognition on the device type for each broadcast packet frame by a BLE device.

3.2 USRP-based BLE packet capture

In order to capture BLE broadcast packets, we first chose USRP to capture the BLE signal. Then, a BLE receiver is implemented on a PC by GNU Radio to demodulate and decode the BLE broadcast packets to generate Pcap files. The USRP device consists of a main board, a digital-to-analog converter, an analog-to-digital converter, and a daughter board, which is capable of receiving and transmitting signal on a specific frequency band. The daughter board is locked to the 2.48 GHz ISM band, where BLE devices operate.

The BLE receiver implemented by the open source framework GNU Radio is intended to demodulate the BLE physical layer and decode it to the link layer. The first step is to detect the presence of each frame according to the signal power. The frequency conversion of the received signal is then performed to extract the frequency band within which the BLE signal is located. Afterwards, GMSK demodulation is applied to demodulate the discrete signal represented by complex numbers to generate a byte stream, and finally decode it to the link layer. We use the ble_dump project[29], including osmocom source, simple squelch, frequency xlating FIR filter, GMSK demod, and unpacked to packed modules. The signal reception and demodulation process is shown in Fig. 3.

We use the BLE receiver to capture the BLE packets of 15 types of devices, as shown in Table 1. 401 341 broadcast packets were collected for these devices in one month.

3.3 BLE broadcast packet analysis

In this section, the collected BLE broadcast packets will be used to analyze the BLE broadcast packet types and the existence of features in their fields, respectively. After analysis, we conclude that broadcast packets of types ADV_IND and ADV_NONCONN_IND can be used to achieve the identification of BLE device type. The reasons are as follows: Firstly, devices frequently transmit these two types of broadcast packets, so the packets are conducive to the real-time discovery of the surrounding BLE devices for identification. Secondly, both types of broadcast packets are actively broadcast by the device and can be passively listened to and collected. Thirdly, various types of Advising Data (AD) in these two types of broadcast packets have characteristics, which are shown in Table 2, which can provide enough information for the classification and identification of BLE devices.

3.3.1 Feature field analysis—BLE broadcast packet header

The type field of the Physical Data Unit (PDU) and the ChSel field in the broadcast packet header reflect the broadcast packet type of the device and the supported channel selection strategy, and thus the characteristics exist. The type field of the PDU indicates the type to which the broadcast physical channel PDU belongs. For

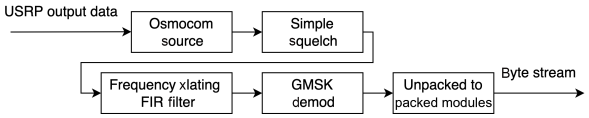

Fig. 3 BLE receiver flow chart.

Table 1 BLE device capture list.

No.	Device type	Device manufacturer	Model	Number of devices	Number of frames	Number of frames of types 0 and 2
1	Smart band	Xiaomi	Mi Band 5	1	1848	1701
2	Smart band	Huawei	4-506	6	4109	3779
3	Smart band	Garmin	vivosmart3	1	14 886	13 836
4	Smart band	ADZ	A9	1	582	538
5	Smart band	Polar	A370	1	975	888
6	Smart watch	Garmin	010-02120-41	1	9676	8970
7	Smart watch	Xiaomi	Mi Color BCAA	1	11 673	10 834
8	Smart watch	Polar	IGNITE	1	997	874
9	Smart oximeter	Konsung	W01LT	1	944	885
10	Smart oximeter	Viatom	O2Ring	1	1343	1247
11	Smart phone	Apple	iPhone8/6s	2	501	474
12	PC	Apple	MBP 2020	1	1983	1873
13	Headphone	Sony	WH-1000XM3	2	1312	816
14	Smart humidifier	Xiaomi	CJXJSQ02ZM	1	903	590
15	PC	Microsoft	ASUS A550L	2	643	643

Table 2 BLE device characteristics field.

Field characteristic	Description
Packet header	The type of PDU and ChSel flags in the header reflect the broadcast packet type of the device and the supported channel selection strategy.
AdvData type	Different devices choose to carry different types of AdvData when broadcasting.
Flags AD	Reflect the broadcast mode of the device and its compatibility with the protocol.
Service UUID AD	Reflect the functionality of the device. The size the device chooses to carry and whether it broadcasts the full service UUID vary by vendor implementation.
Tx power level AD	Reflect the transmit power of the device to transmit broadcast packets.
Service data AD	Reflect the services being provided by the equipment.
Appearance AD	Reflect the appearance of the device.
Manufacturer specific AD	Fields contain vendor-defined data; the structure of the specific data payload will vary depending on vendor-specific implementation.

ADV_IND and ADV_NONCONN_IND, their values are 0000 and 0010, respectively. The field value can be well distinguished from that of Windows devices because Windows devices broadcast ADV_NONCONN_IND in the normal state, which is different from other devices. For ADV_IND, the ChSel field indicates the channel selection strategy to be used for data communication after connection, and for ADV_NONCONN_IND, the ChSel field is reserved. Two channel selection strategies are observed on the collected packets, and the channel selection strategies used for each type of device are determined, as shown in Table 3. According to the BLE protocol, if the broadcast device supports BLE channel selection strategy 2, the ChSel field should be set to 1.

3.3.2 Feature field analysis—AdvData

The AdvData field on ADV_IND and ADV_NONCONN_IND payloads consists of significant part and non-significant part in the format shown in Fig. 4. The non-significant part is used to fill the packet with 0 bytes when necessary. The significant part

Table 3 Channel selection strategy for capturing BLE devices.

Channel selection strategy	Device
1	(band, Huawei), (band, Polar), (oximeter, Konsung), (oximeter, Viatom), (PC, Apple), (smartphone, Apple), (watch, Xiaomi), (humidifier, Xiaomi), (headphone, Sony)
2	(band, Xiaomi), (band, Garmin), (watch, Garmin), (watch, Polar), (band, ADZ)

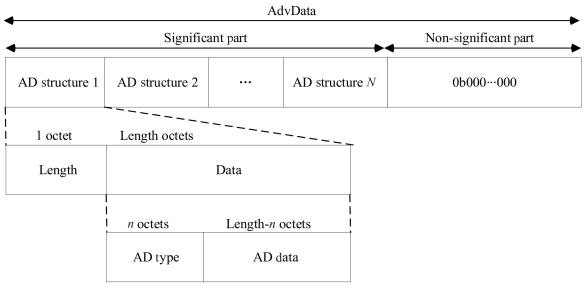

Fig. 4 Broadcast data AdvData format.

consists of a series of AD structures, the first byte of which indicates the length of the data field. The first byte of the data field indicates the type of AD structure, and the remaining bytes are the data corresponding to the AD type. The AD types carried in each device broadcast packet are shown in Table 4, and the meaning of each AD type is analyzed below.

(1) Flag type reflects the broadcast mode of the device and its compatibility with the protocol; hence, it can be used as a characteristic. It is 1 byte long, and each bit represents a Boolean value. The first to the fifth bits indicate limited discoverable mode, general discoverable mode, BR/EDR not supported, whether the Bluetooth controller layer supports both LE and BR/EDR, and whether the Bluetooth host layer supports both LE and BR/EDR. The last three bits are reserved fields. According to the protocol, when the bits of the flag have a value other than 0, they must be carried in the broadcast packets but otherwise can be ignored.

(2) 16-bit service UUIDs (incomplete), 16-bit service UUIDs (complete), and 128-bit service UUIDs (incomplete) all belong to the service UUID type in AD type. The AD data of service UUID reflect the device function and the member of Bluetooth Special Interest Group (SIG)[30] to which the device belongs, thus can be applied as a characteristic. This type of AD data is used by broadcast devices to inform other devices of their own available services or service classes. Service UUID can be classified into 9 types including 16-bit member UUID[31], GATT characteristic and object type, GATT descriptor, GATT service, GATT unit, protocol identifier, SDO GATT service, and service classes and profiles. Whether the device broadcast carries service UUID and which type of service UUID it carries vary from vendor to vendor. The service UUIDs carried by different types of devices like bands and oximeters naturally have obvious differences because of the different services provided.

Table 4 AD types of captured BLE devices.

AD type	Type of AD structure	Serial number of the devices carrying this type of AD
0x1	Flag	1–14
0x2	16-bit service UUIDs (incomplete)	5, 8, 9
0x3	16-bit service UUIDs (complete)	4
0x7	128-bit service UUIDs (incomplete)	3
0x9	Complete local name	4, 7, 10, 13
0xA	Tx power level	11
0x16	Service data	2
0x19	Appearance	4
0xFF	Manufacturer specific	1–3, 5, 6, 8, 10–14, 15

(3) Transmission (Tx) power level type of AD data reflects the transmission power of the broadcast packet containing that data type and therefore has characteristics. According to the protocol, the path loss can be calculated by the value of the transmitter power carried by Tx power level of the AD data. In the identification scenario, this type of AD data reflects the power of the Bluetooth chip RF end when the BLE device broadcasts. The type of the **service data** reflects the service provided by the device and thus is distinctive. This type of AD data consists of the service UUID and the data associated with that service. **Appearance** type of the AD data reflects the appearance of the BLE device transmitting the packet and hence can be distinguishing. The structure of the **manufacturer specific** of the AD data is implemented by the manufacturer and therefore has features. Figure 5 shows the structure of this type of AD data, which is used for manufacturer specific data. The data consist of a two-byte company identifier code and an manufacturer specific data payload. It is worth noting that the company identifier code here is not the same value as that of the member UUID of the service UUID corresponding to the same manufacturer as described above.

3.4 Fingerprinting based on link layer broadcast packets

3.4.1 Data pre-processing

First of all, we need to filter the types of ADV_IND and ADV_NONCONN_IND of broadcast physical channel PDUs from the collected BLE broadcast packets to extract device fingerprints. The type of the broadcast physical channel PDUs can be judged by the PDU type flags in the header. For ADV_IND and ADV_NONCONN_IND, the field values are 0x0 and 0x2, respectively.

Second, the explicit identifiers present in these two types of packets should be removed, including the broadcast MAC address, the company identifier，and length field in the manufacturer specific AD data, the

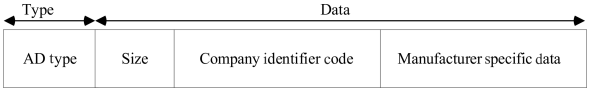

Fig. 5 Format of manufacturer specified type AD.

member UUID in the 16-bit service UUID class AD data, and the device name field. The flow of the entire algorithm is shown in Algorithm 1.

Finally, the BLE packets are zero-padded and normalized after removing the explicit identifiers. As shown in Fig. 6, this is done by converting the hexadecimal values into numeric data types and then dividing them by 16 for normalization. A 200-dimensional vector is extracted and put into the MLP to train a device classifier for device identification.

Algorithm 1 Extraction of header flag and the removal of explicit identifiers for the BLE broadcast packet

Input: ADV_IND and ADV_NONCONN_IND types of BLE link layer broadcast packets *blell*
Output: Link layer data with header flag and without explicit identifiers *blell_strip*

```
blell_strip = list();
step = blell[FirstAdvData_index];
i = FirstAdvData_index;
while (i+3 != len(blell)) do
    type =blell[i+1];
    switch (type) do
        case Device_Name do
            blell = blell[:i]+blell[i+step+1:];
            break
        case Manufacturer_Specific do
            blell = blell[:i] + bytes(0x00) + blell[i+1:
            i+2] + blell[i+2+2:];
            i = i + step + 1 − 2;
            break
        case 16-bit_Service_UUIDs do
            num_of_uuid = (blell[i] − 1)/2;
            UUID_changeFlag = False;
            for j in range(num_of_uuid) do
                UUID_2 = blell[i + 2 + j × 2 + 1];
                if (UUID_2 && 0xf0) ⩾ 0xf0 then
                    blell = blell[:i + 2 + j × 2]+
                    blell[i + 2 + j × 2 + 2:];
                    blell[i] = blell[i] − 2;
                    i = i + step + 1 − 2;
                    UUID_changeFlag = True;
            if UUID_changeFlag == False then
                i = i + step + 1;
        case default do
            i = i + step + 1;
blell_strip = blell[HeaderFlag_start :
HeaderFlag_end] + blell[FirstAdvData_index :];
return blell_strip.
```

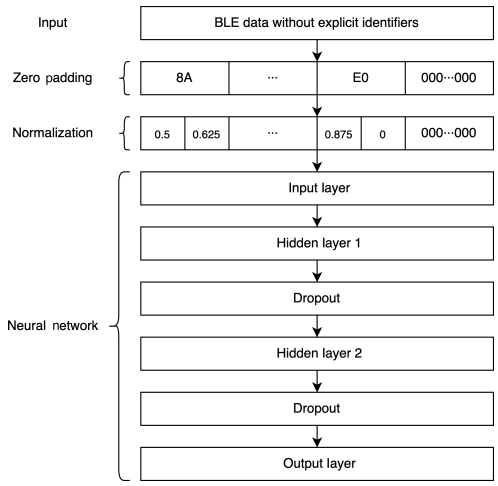

Fig. 6 Structure of the BLE classification model.

3.4.2 Model train

We chose MLP as the classification model for the following three reasons: First, BLE broadcast packets vary among devices of the same type. This variation is difficult to measure in cases where the unique matching recognition method is used directly, resulting in poor recognition accuracy. Second, neural networks provide automation to extract features from each bit of the BLE broadcast packet without the need for manual feature selection. Third, according to the above analysis, the manufacturer specific class AD and the service data class AD in AdvData have manufacturer defined structural features. By means of deep learning, the features present in the upper field structure can be exploited for device identification. Since the packet structure follows a simple specification and does not have such complex structure as images and text, a lightweight MLP is chosen.

In order to balance the performance and computational cost, a lightweight neural network based on MLP was constructed, and the structure is shown in Fig. 6. The number of hidden layers was set to 2, the number of neurons in each hidden layer was set to 128, a Rectified Linear Unit (ReLU) activation function was selected, and Softmax was used at the output layer as the activation function. To prevent overfitting, a dropout layer was added between the hidden layers, its activation function value will change to 0 when the probability is P (P is set to 0.3), thereby stopping neurons in the previous layer from working.

4 Experiment

4.1 Evaluation metrics

In the experiments, we selected five representative evaluation metrics for comprehensive evaluation. Note that we are dealing with a multi-classification problem, so the commonly used evaluation metrics are as follows.

- **Accuracy** is defined the same as that in dichotomous classification. It is the proportion of the total samples that make correct predictions.
- **Macro precision** is precision calculated for each label and class, and then taken as an unweighted average.
- **Macro recall** is calculated separately for each label and then taken as an unweighted average.
- **Macro F1-Score** is calculated separately for each label and then taken as an unweighted average.
- **Micro F1-Score** is obtained by summing the True Negative (TN), False Positive (FP), and False Positive (FN) of n dichotomous evaluations, calculating precision and recall, and finally calculating F1-Score.

The precision, recall, and F1-Score are shown in the following:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

$$\text{F1-Score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{3}$$

where True Positive (TP) indicates the number of positive samples classified as positive cases, TN indicates the number of negative samples classified as negative cases, FP indicates the number of negative samples classified as positive cases, and FN indicates the number of positive samples classified as negative cases. In this section, we report the macro precision, macro recall, and macro F1-Score in the experiments.

4.2 Evaluation results

The identification granularity of the link layer fingerprint of the BLE devices based on broadcast packets is defined as a binary group consisting of device type and vendor. The experimental devices include smart bands, smart watches, smart oximeters, smart humidifiers, headphones, smartphones, and PCs. The manufacturers include Xiaomi, Huawei, Garmin, ADZ, Polar, Konsung, Viatom, Apple, Sony, and Microsoft. As shown in Table 1, a total of 15 device types were in the combination.

The software radio based BLE receivers described in Section 3.1 were used to collect BLE packets from the above devices on channels 37, 38, and 39. The BLE broadcast packets of ADV_IND and ADV_NONCONN_IND types were collected for each type of devices, totaling 48 623 packets. For each type of device, 315 frames were extracted to form a training set of 4725 data, and 90 frames were extracted from each type to form a validation set of 1350 data. For the device with the least number of frames, it is ensured that the ratio of training set, validation set, and test set can meet 7:2:1.

The trained MLP model achieves a recognition accuracy of 99.8% for the above 15 types of BLE devices on the testing set, with precision, recall, and F1-Score being 0.9903, 0.9890, and 0.9888, respectively. The normalized confusion matrix is shown in Fig. 7, and the recognition results of various types of devices are shown in Table 5. For comparison, Aksu et al.[17] used the arrival interval distribution of Bluetooth packets as a feature, and successfully identified six different types of devices with an average accuracy of 98.5%. We achieve better precision and faster speed when expanding the number of identifications to at least 15 for the BLE protocol.

5 Conclusion

We use USRP and GNU Radio to acquire BLE packets. A large number of BLE device packets were collected, and the fields that reflect device differences in broadcast packets were explored through traffic analysis. Finally, the common types of ADV_IND and ADV_NONCONN_IND of the broadcast packets were

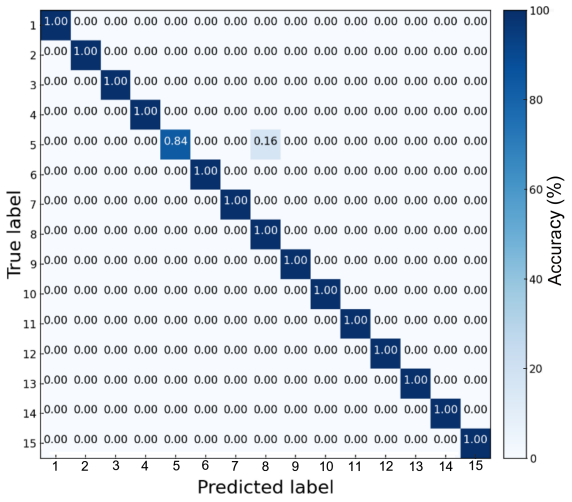

Fig. 7 Confusion matrix of the BLE device identification. As the color of squares on the diagnol gets darker and that of the others gets lighter, the prediction becomes more accurate.

Table 5 Identification results of various BLE devices.

Device	Precision	Recall	F1-Score
Band, Xiaomi	1.000	1.000	1.0000
Band, Huawei	1.000	1.000	1.0000
Band, Garmin	1.000	1.000	1.0000
Band, ADZ	1.000	1.000	1.0000
Band, polar	1.000	0.835	0.9102
Watch, Garmin	1.000	1.000	1.0000
Watch, Xiaomi	1.000	1.000	1.0000
Watch, polar	0.855	1.000	0.9220
Oximeter, Konsung	1.000	1.000	1.0000
Oximeter, Viatom	1.000	1.000	1.0000
PC, Apple	1.000	1.000	1.0000
Headphone, Sony	1.000	1.000	1.0000
Humidifier, Xiaomi	1.000	1.000	1.0000
PC, Microsoft	1.000	1.000	1.0000

filtered, and explicit identifiers, such as MAC address, device name, and member UUID, were eliminated. The MLP model was used to identify BLE devices by combining the device vendor and types into categories. An identification accuracy of up to 99.8% was achieved in 15 categories of BLE devices.

More accurate identification of BLE device classes was achieved based on data-link layer fingerprinting. As for future work, it is necessary to extract user information from upper layers of the protocol stack or from differences in device hardware manufacturing from the physical layer as features in order to identify individual devices instead of $\langle device\ \ type, vendor \rangle$ binary.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 61972085, 62072103, and 62232004), the Jiangsu Provincial Key R&D Program (Nos. BE2021729, BE2022680, and BE2022065-4), the Jiangsu Provincial Key Laboratory of Network and Information Security (No. BM2003201), the Key Laboratory of Computer Network and Information Integration of Ministry of Education of China (No. 93K-9), the Collaborative Innovation Center of Novel Software Technology and Industrialization, the Fundamental Research Funds for the Central Universities, the CCF-Baidu Open Fund (No. 2021PP15002000), and the Future Network Scientific Research Fund Project (No. FNSRFP-2021-YB-02).

References

[1] IDC, Worldwide global DataSphere IoT device and data forecast, https://www.iotplaybook.com/tags/worldwide-global-datasphere-iot-device-and-data-forecast-2019-2023, 2019.

[2] WiGLE, Statistics, https://wigle.net/stats, 2021.

[3] J. Ren, J. Li, H. Liu, and T. Qin, Task offloading strategy with emergency handling and blockchain security in SDN-empowered and fog-assisted healthcare IoT, *Tsinghua Science and Technology*, vol. 27, no. 4, pp. 760–776, 2022.

[4] P. Ma, B. Jiang, Z. Lu, N. Li, and Z. Jiang, Cybersecurity named entity recognition using bidirectional long short-term memory with conditional random fields, *Tsinghua Science and Technology*, vol. 26, no. 3, pp. 259–265, 2021.

[5] M. H. Haghighat and J. Li, Intrusion detection system using voting-based neural network, *Tsinghua Science and Technology*, vol. 26, no. 4, pp. 484–495, 2021.

[6] L. C. C. Desmond, C. C. Yuan, T. C. Pheng, and R. S. Lee, Identifying unique devices through wireless fingerprinting, in *Proc. 1st ACM Conf. on Wireless Network Security*, Alexandria, VA, USA, 2008, pp. 46–55.

[7] C. L. Corbett, R. A. Beyah, and J. A. Copeland, Using active scanning to identify wireless NICs, in *Proc. IEEE Information Assurance Workshop*, West Point, NY, USA, 2006, pp. 239–246.

[8] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, and J. Van Randwyk, Passive data link layer 802.11 wireless device driver fingerprinting, in *Proc. 15th Conf. on USENIX Security Symp.*, Vancouver, Canada, 2006, p. 12.

[9] F. Guo and T. C. Chiueh, Sequence number-based MAC address spoof detection, in *Proc. 8th Int. Workshop on Recent Advances in Intrusion Detection*, Seattle, WA, USA, 2005, pp. 309–329.

[10] S. Jana and S. K. Kasera, On fast and accurate detection of unauthorized wireless access points using clock skews, *IEEE Trans. Mobile Comput.*, vol. 9, no. 3, pp. 449–462, 2010.

[11] T. D. Vo-Huu, T. D. Vo-Huu, and G. Noubir, Fingerprinting Wi-Fi devices using software defined radios, in *Proc. 9th ACM Conf. on Security & Privacy in Wireless and Mobile Networks*, Darmstadt, Germany, 2016, pp. 3–14.

[12] C. Arackaparambil, S. Bratus, A. Shubina, and D. Kotz, On the reliability of wireless fingerprinting using clock skews, in *Proc. 3rd ACM Conf. on Wireless Network Security*, Hoboken, NJ, USA, 2010, pp. 169–174.

[13] C. L. Corbett, R. A. Beyah, and J. A. Copeland, Passive classification of wireless NICs during rate switching, *EURASIP J. Wirel. Commun. Netw.*, vol. 2008, p. 495070, 2007.

[14] S. Bratus, C. Cornelius, D. Kotz, and D. Peebles, Active behavioral fingerprinting of wireless devices, in *Proc. 1st ACM Conf. on Wireless Network Security*, Alexandria, VA, USA, 2008, pp. 56–61.

[15] P. Robyns, B. Bonné, P. Quax, and W. Lamotte, Noncooperative 802.11 mac layer fingerprinting and tracking of mobile devices, *Secur. Commun. Netw.*, vol. 2017, p. 6235484, 2017.

[16] J. Huang, W. Albazrqaoe, and G. Xing, BlueID: A practical system for Bluetooth device identification, in *Proc. IEEE Conf. on Computer Communications*, Toronto, Canada, 2014, pp. 2849–2857.

[17] H. Aksu, A. S. Uluagac, and E. S. Bentley, Identification of wearable devices with Bluetooth, *IEEE Trans. Sustainable Comput.*, vol. 6, no. 2, pp. 221–230, 2021.

[18] L. Peng, A. Hu, J. Zhang, Y. Jiang, J. Yu, and Y. Yan, Design of a hybrid RF fingerprint extraction and device classification scheme, *IEEE Internet Things J.*, vol. 6, no. 1, pp. 349–360, 2019.

[19] K. Merchant, S. Revay, G. Stantchev, and B. Nousain, Deep learning for RF device fingerprinting in cognitive communication networks, *IEEE J. Sel. Top. Signal Process.*, vol. 12, no. 1, pp. 160–167, 2018.

[20] 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 Trans. Inf. Forensics Secur.*, vol. 11, no. 8, pp. 1862–1874, 2016.

[21] P. Scanlon, I. O. Kennedy, and Y. Liu, Feature extraction approaches to RF fingerprinting for device identification in femtocells, *Bell Labs Tech. J.*, vol. 15, no. 3, pp. 141–151, 2010.

[22] C. Dubendorfer, B. Ramsey, and M. Temple, ZigBee device verification for securing industrial control and building automation systems, in *Proc. 7^{th} Int. Conf. on Critical Infrastructure Protection*, Washington, DC, USA, 2013, pp. 47–62.

[23] T. J. Bihl, K. W. Bauer, M. A. Temple, and B. Ramsey, Dimensional reduction analysis for Physical Layer device fingerprints with application to ZigBee and Z-Wave devices, in *Proc. 2015 IEEE Military Communications Conf.*, Tampa, FL, USA, 2015, pp. 360–365.

[24] O. Ureten and N. Serinken, Wireless security through RF fingerprinting, *Can. J. Electr. Comput. Eng.*, vol. 32, no. 1, pp. 27–33, 2007.

[25] Y. J. Yuan, Z. Huang, and Z. C. Sha, Specific emitter identification based on transient energy trajectory, *Prog. Electromagn. Res. C*, vol. 44, pp. 67–82, 2013.

[26] B. Chatterjee, D. Das, S. Maity, and S. Sen, RF-PUF: Enhancing IoT security through authentication of wireless nodes using *in-situ* machine learning, *IEEE Internet Things J.*, vol. 6, no. 1, pp. 388–398, 2019.

[27] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, Wireless device identification with radiometric signatures, in *Proc. 14^{th} ACM Int. Conf. on Mobile Computing and Networking*, San Francisco, CA, USA, 2008, pp. 116–127.

[28] IEEE 802.11-2016 IEEE standard for information technology–Telecommunications and information exchange between systems local and metropolitan area networks–Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, https://standards.ieee.org/ieee/802.11/5536/, 2022.

[29] drtyhlpr, SDR Bluetooth LE dumper, https://github.com/drtyhlpr/ble_dump, 2016.

[30] Bluetooth SIG, Bluetooth core specification, https://www.bluetooth.com/specifications/specs/core-specification-5-2/, 2019.

[31] Bluetooth SIG, 16-bit UUID numbers document, https://www.bluetooth.com/specifications/assigned-numbers/, 2023.

Jinghui Zhang received the BEng degree from Southeast University, China in 2005 and the PhD degree in computer science from Southeast University, China in 2014. He is an associate professor at the School of Computer Science and Engineering, Southeast University, China. His current research interests include edge computing, distributed machine learning, cloud computing, and network security.

Zhen Ling received the BEng degree and the PhD degree in computer science from Nanjing Institute of Technology, China in 2005, and Southeast University, China in 2014, respectively. He is a full professor at the School of Computer Science and Engineering, Southeast University, Nanjing, China. He won ACM China Doctoral Dissertation Award and China Computer Federation (CCF) Doctoral Dissertation Award in 2014 and 2015, respectively. His research interests include artificial intelligence of things, mobile system security, network security and privacy, and trusted computing.

Xinyang Li received the BEng degree in computer science and technology from Nanjing University of Science and Technology, Nanjing, China in 2022. Currently, he is a master student in electronic and information science at Southeast University, Nanjing, China. His current research interests include distributed machine learning, federated learning, and network security.

Ming Yang received the MEng and the PhD degrees in computer science and engineering from Southeast University, China in 2002 and 2007, respectively. He is currently a professor at the School of Computer Science and Engineering, Southeast University, Nanjing, China. His main research interests include network security and privacy, as well as the Internet of Things (IoT).

Junhe Li received the BEng degree in information engineering from Southeast University, Nanjing, China in 2018. Currently, he is a master student in cyberspace security at Southeast University, Nanjing, China. His current research interests include threat intelligence and malware detection.

Qiangsheng Dai received the PhD degree in electrical engineering from Tsinghua University, Beijing, China in 2020. He is now working at Research Institute, State Grid Jiangsu Electric Power Company Ltd. His research interests include deep learning and smart grids.