# Protocol-agnostic IoT Device Classification on Encrypted Traffic Using Link-Level Flows

Gabriel Morales, Adam Bienek-Parrish, Patrick Jenkins, and Rocky Slavin
Department of Computer Science, University of Texas at San Antonio
San Antonio, Texas, USA
{firstname.lastname}@my.utsa.edu,rocky.slavin@utsa.edu

## ABSTRACT

Convenience is a strong driver for the evolution of technology. Such efforts have given rise to the Internet-of-Things (IoT), defined as the network of everyday devices (i.e., "things") ranging from light bulbs to smart speakers, connected to the Internet and each other. IoT devices frequently transmit data wirelessly which can be passively collected by an adversary. In this work we present a methodology with which to perform device classification on encrypted traffic in a protocol-agnostic manner by applying network flow analysis to link-level data. Our evaluation demonstrates successful device classification for 15 device categories with an overall weighted F1-Score of 95% on a dataset consisting of Wi-Fi, Bluetooth, and Zigbee traffic. Furthermore, we explore model transferability between encrypted and decrypted datasets on these three networking protocols and present our flow generation tool, ProtoFlow.

## CCS CONCEPTS

• **Networks** → *Intermediate nodes*; *Link-layer protocols*; • **Security and privacy**; • **Computing methodologies** → Machine learning approaches;

## KEYWORDS

Internet-of-Things, IoT, Traffic Flow, Network Analysis, Networking Standards, Classification

**ACM Reference Format:**
Gabriel Morales, Adam Bienek-Parrish, Patrick Jenkins, and Rocky Slavin. 2023. Protocol-agnostic IoT Device Classification on Encrypted Traffic Using Link-Level Flows. In *Cyber-Physical Systems and Internet of Things Week 2023 (CPS-IoT Week Workshops '23), May 09–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3576914.3587487

## 1 INTRODUCTION

Our world is becoming increasingly connected. As technology and society move toward automation and convenience, innovations in wireless technology continue to contribute to an increasingly dense Internet of Things (IoT). IoT can be defined as the connection of a variety of heterogeneous devices including everyday existing objects to communicate and integrate with each other to collect, generate, process, and exchange data [1]. These technologies give rise to smart homes, which "incorporate common devices that control features of the home" [23]. With IoT becoming more prevalent, the attack surface expands to more devices within the home for exploitation due to weak security, insecure network services, insecure update mechanisms, and more [21, 28]. One such notable instance is the Mirai attack in 2016: a wide-scale distributed denial-of-service (DDoS) attack propagated by the use of default passwords present in IoT devices [3].

Since IoT devices communicate frequently with one another and services over the Internet, the risk to private information is high. Even with encrypted traffic, inference attacks can be performed through collection and observation of publicly-viewable data. For example, passive network observation and prior knowledge of human behavior can be combined to infer home occupant activities [5]. Recent works have shown traffic information, even when encrypted over TCP, can be classified based on communication patterns, most of which include traffic flow [9, 10, 13]. However, these approaches do not include flow creation and classification on the link-level with various networking protocols, as we do.

Traffic flows represent communications between devices. For traffic on the link-level whose higher-level information is encrypted, only MAC-like addresses and quantitative packet data such as sizes are present. However, the communications between devices are still visible within packet captures. As such, one can still derive statistical information from the quantitative packet data, without network association. We study flows in an *agnostic* manner (i.e., equivalent features) across three networking protocols to extract information which spans more devices, instead of studying them separately. We utilize traffic from the link-level on these protocols to experiment with classification of devices, noting predictable behaviors from their communications and sizes [25].

Our contributions are as follows:

(1) **Flow-based link-level device classification:** We describe a flow-based approach to IoT device classification of encrypted traffic which can be generalized to different wireless communication protocols/standards. The method is evaluated on 15 Zigbee, Wi-Fi and Bluetooth Low Energy device types on the link-level achieving an overall, weighted F1-score of 95% for the best model.
(2) **Study of interchangeability of encrypted and unencrypted traffic for device classification:** We demonstrate that packet sizes between the decrypted and unencrypted versions of a packet capture remain the same. Then, we explore the application of machine learning (ML) between these two versions of a packet capture to classify device types.



*CPS-IoT Week Workshops '23, May 09–12, 2023, San Antonio, TX, USA*

ACM ISBN 979-8-4007-0049-1/23/05.
https://doi.org/10.1145/3576914.3587487

(3) **ProtoFlow:** An extensible open-source tool for the generation protocol-agnostic flow tables from standard packet captures.

The rest of this paper is organized as follows. Section 2 discusses the technical background. Section 3 presents closely related work. Section 4 provides insight to how we structure and prepare our data, our tool, and the relationship between encrypted and unencrypted data. Section 5 describes our experiment design while Section 6 discusses the classification results. Finally, Section 7 presents future work and conclusions.

## 2 BACKGROUND

In this section we discuss the wireless technology standards we evaluated (Wi-Fi, Bluetooth Low Energy, and Zigbee) and the concept of intra-device data flows.

### 2.1 Wireless Networking Protocols

IoT devices communicate through various networking protocols. These networking protocols have different behaviors depending on factors such as hardware constraints or locale of operation, which requires them to be analyzed separately. However, their unification would enable their analysis to take place on a macro-level for the smarthome environment. For our work, we study Wi-Fi, Bluetooth Low Energy (BTLE), and Zigbee, based on their popularity [8, 16, 26]. The Open Systems Interconnection (OSI) model is commonly used to describe the standard for network communication stacks [18]. Here, we operate on the link-level, or layer two. Data is packaged at the link-level and handles physical transfer and hardware addressing [18, 27]. Furthermore, most encryption techniques generally operate on higher levels, leaving link-level data intact.

Different wireless technologies can be used depending on the application. Wi-Fi enables wireless connection to the Internet and a large area for local communication [6]. Using Wi-Fi, IoT devices can be easily introduced into the home to connect with existing devices and the Internet to communicate with their cloud-services [16, 29]. BTLE enables data transfer and audio streaming, but is low-power, uses low data rates, and can support large-scale mesh networks [16]. Zigbee[1] is designed for low-power communication. However, it transmits at a lower data rate than Bluetooth and operates for a very long time [11]. Furthermore, it also supports a large number of nodes, easy deployment, interoperability, and self-healing [22, 24].

### 2.2 Intra-Device Data Flows

A traffic flow can be defined as a series of related attributes in a communication between two devices. With such information, both a uni- and bidirectional flow can be represented where unidirectional flow is the series of communications between a source and its destination, and bidirectional flow is the series of communications between both the source and destination, and the destination and source (its inverse) [9, 13, 15]. A collection of flows can be aggregated into a flow table data structure.

[1]https://csa-iot.org/all-solutions/zigbee/

**Table 1: Flow Entry Features**

| Flow Entry Features | |
|---|---|
| Source MAC | Dest MAC |
| Source OUI | Dest OUI |
| Bidirectional Total Packets | Bidirectional Total Bytes |
| Source to Dest Total Bytes | Dest to Source Total Bytes |
| Source to Dest Total Packets | Dest to Source Total Packets |
| Source to Dest First Seen Time (ms) | Dest to Source First Seen Time (ms) |
| Source to Dest Last Seen Time (ms) | Dest to Source Last Seen Time (ms) |
| Source to Dest Total Duration (ms) | Dest to Source Total Duration (ms) |
| Bidirectional Total Duration (ms) | Source to Dest Min Packet Size |
| Source to Dest Max Packet Size | Source to Dest Mean Packet Size |
| Source to Dest Stdev Packet Size | Dest to Source Min Packet Size |
| Dest to Source Max Packet Size | Dest to Source Mean Packet Size |
| Dest to Source Stdev Packet Size | Bidirectional min Packet Size |
| Bidirectional Max Packet Size | Bidirectional Mean Packet Size |
| Bidirectional Stdev Packet Size | Source to Dest Transmission Rate (ms) |
| Dest to Source Transmission Rate (ms) | Bidirectional Transmission Rate (ms) |
| Source to Dest Transmission Rate Bytes (ms) | Dest to Source Transmission Rate Bytes (ms) |
| Bidirectional Transmission Rate Bytes (ms) | |

## 3 RELATED WORK

As mentioned in Section 1, existing works explore the classification of devices using link-level. We discuss some here and highlight the differences in our contributions.

IoThound [2] classifies device types and performs anomaly detection. Their work considers Bluetooth, Zigbee, and Wi-Fi using packet information at various levels above, and including, the link layer. They perform *separate* evaluations for each protocol, and achieve above 94% accuracy for Wi-Fi, 97% for Zigbee, and 90% for Bluetooth. The work focuses on an unsupervised clustering approach. In contrast, we utilize three supervised machine learning algorithms, and compare their performance as a *whole* over traffic from all protocols. Furthermore, our approach requires only link-level information.

Kostas et al. propose IoTDevID [17], a system designed to classify the model and brand of device. Performance of six different models are evaluated on two widely used Wi-Fi datasets based on their strong feature selection process. Using feature importance scoring, followed by a Genetic Algorithm, the resulting features relate to basic packet flags, sizes, and timings. The best results reach 94% accuracy and a 93% F1-score. Similarly, Maiti et al. implement a framework, PrEDeC [20], which performs device type classification on link-level Wi-Fi traffic. Traffic is collected passively using a commercial off-the-shelf radio. Classification is performed in real-time to yield device types for devices in the immediate vicinity. Due to the focus on Wi-Fi-specific features, both IoTDevID and PrEDeC are limited to the classification of Wi-Fi devices whereas our approach is generalizable to BTLE and Zigbee with comparable results.

To our knowledge, no other work has performed device type classification on the link-level for various protocols in an agnostic manner.

## 4 METHODOLOGY

To achieve protocol-agnostic flow generation for classification purposes, we observe that the captures between networking protocols share the following information outside of the payload: timestamps, MAC-like addresses, and packet length. These features are the only ones available at the link-level. For instance, Wi-Fi traffic generally

has information such as IP addresses and ports, but not on the link-level.

Flow generation tools exist to perform the task of traffic flow creation for Wi-Fi LAN traffic [4, 12]. For link-level traffic, however, these tools produce incorrect results since they are designed for Wi-Fi above the link-level. For this reason, we developed ProtoFlow[2] as an extensible, open-source solution for generating traffic flows from link-level (layer two) information, which is not hidden when the traffic is encrypted at a higher level. The tool is capable of generating flows for any traffic capture that is recognizable by a standard packet parsing library. This enables a versatile and customizable tool that can create flows for any networking protocol, which in our case includes Wi-Fi, BTLE, and Zigbee.

As input, ProtoFlow takes a packet capture and iterates through the packets to calculate statistical attributes for each flow. To detect the networking protocol being used, each packet contains the protocols visible within the frame. PyShark is used to parse the packets, followed by the use of NumPy to perform the statistical calculations. As flows are parsed, a table is accumulated with flow entries which represent device communications and their statistical attributes (e.g., average packet size, standard deviation, minimum, maximum, etc.). The tool will output a CSV file which represents the final flow table for the capture.

## 4.1 Traffic Capturing

As introduced in Section 2, traffic is captured on layer two of the OSI model, the link-level. This allows for all traffic to be observable passively without association using the appropriate sensors for Wi-Fi, Bluetooth, and Zigbee. Here, we describe how ProtoFlow handles each.

*4.1.1 802.11 Wi-Fi.* Traffic can be captured on the link-level by using network hardware available on most computers by putting the network card into monitor mode. MAC address 2-tuples are used to represent a flow. A forward-direction tuple is constructed as: (*Source_MAC*, *Destination_MAC*) and a backward-direction tuple is the inverse. These two tuple types constitute a bidirectional flow pair, which is identified on the first such encountered pair.

*4.1.2 Bluetooth & Bluetooth LE.* Bluetooth and BLE link-level traffic can be captured using the Ubertooth One[3] which enables one to sniff and follow active Bluetooth and BLE connections. Differences exists in how the source and destination are referred, depending on if the communication is a broadcast or a communication between two devices:

- **Broadcast packet:** Source is an advertising address (`FF:FF:FF:FF:FF:FF` is still the broadcast destination)
- **Source to destination communication packet:** The source address is a scanning address and the destination address is an advertising address.

To create an entry from these cases, the sources and destinations are organized as such:

- **Broadcast:** Source = Scanning Address, Destination = `FF:FF:FF:FF:FF:FF`. In other words, an entry is still created for this pair.

[2]https://github.com/Gabriel-Morales/ProtoFlow
[3]https://greatscottgadgets.com/ubertoothone/

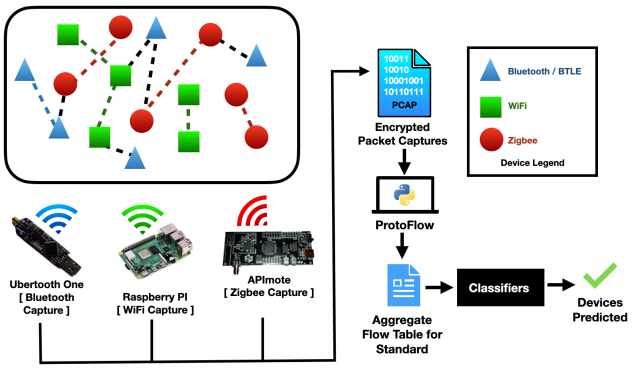



**Figure 1: Flow to ML Pipeline**

- **Source-to-Destination:** Source = Scanning Address, Destination = Advertising Address

*4.1.3 Zigbee.* Zigbee traffic can be sniffed via the APImote[4]. Source and destination pairs are typically represented as either two byte hex value (16 bits), or an eight-byte extended address (64 bits). In the case of a two-byte or eight-byte address, the (*Source*, *Destination*) tuple structure is still used. For example: $(0x743F, 0xFF3B)$.

## 4.2 Protocol Agnostic Link-level Classification

Any packet capture obtained through means of link-level collection can be passed to ProtoFlow to output their respective flow tables. When parsing a packet capture binary file, ProtoFlow inserts an auxiliary column in the feature list entitled 'protocol', denoting the networking protocol a flow is captured under. The values for this column are `Z` for Zigbee, `B` for BTLE, and `W` for Wi-Fi.

Figure 1 displays the workflow to capture and create a flow table CSV file using ProtoFlow. Once the flow table is generated for each of the protocols individually, an aggregate flow table is created through a Pandas DataFrame.

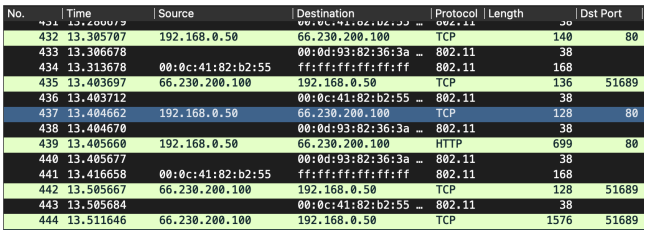
The structure of the data is uniform across all networking protocols passed through ProtoFlow. This means that all features, listed in Table 1, are uniform and ensures that all data can be merged together seamlessly. Each flow entry is labeled based on its source device (i.e., the sender). Table 2 includes the 15 device types used in our experiments and their detail. The category of the device, networking protocol, and the device are listed.

IoT devices behave differently based on the networking protocol. For instance, a BTLE device may only transmit data in short bursts and go back to sleep, while some Wi-Fi devices may be on constantly. Due to these factors, the flows for each device can be different across the protocols, posing a challenge to protocol-agnostic device classification. To address this, we abstract the specific device classes to a broader category, which spans across these networking protocols. For example: `Bulb_A` operates on Wi-Fi, and `Bulb_B` operates on Zigbee. Instead of labeling each flow as `Bulb_A` or `Bulb_B`, the label is abstracted to `Bulb`, thus enabling larger label coverage.

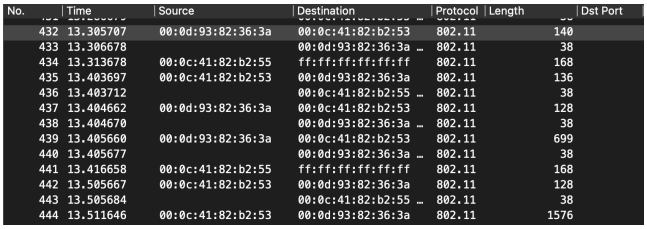
[4]https://apimote.com/

**Table 2: Base link-level Lab Devices Captured**

| Category | Wireless Standard | Detail |
|---|---|---|
| streaming stick | Wi-Fi | Amazon Fire TV Stick Streaming Media Player<br>Google Chromecast |
| router | Wi-Fi | Asus Router RT-N12<br>Asus RT-AC1200GE |
| router | Wi-Fi & Zigbee | Tp-Link Kasa Router |
| smart lock | Bluetooth | 2x August Smart Lock |
| ereader | Wi-Fi | Barnes & Noble Nook<br>Amazon Kindle |
| smart speaker | Bluetooth & Wi-Fi | Bose Home Speaker 300 |
| smart speaker | Bluetooth & Wi-Fi | Sonose One SL |
| smart speaker | Wi-Fi & Zigbee | Amazon Echo with Hub |
| smart switch | Wi-Fi | 5x C by GE 3-Wire On/Off Toggle |
| smart assistant | Wi-Fi & Zigbee | Amazon Echo with Hub |
| fitnesss tracker | Bluetooth | Samsung Galaxy Wear Active<br>Fitbit Charge 4 Health & Fitness Tracker |
| smartphone | Wi-Fi & Bluetooth | Galaxy A21<br>Google Pixel 4a |
| smart vacuum | Wi-Fi | iRobot Roomba |
| smart pet feeder | Wi-Fi | PETIKIT Wi-Fi-enabled feeders |
| smart bridge | Zigbee & Wi-Fi | Philips Hue Bridge |
| smart bulb | Zigbee | x3 Philips Hue Bulb |
| smart plug | Zigbee | x10 Generic Smart Plug |
| smart camera | Wi-Fi | x2 Blink mini camera<br>x2 Kasa Spot |




a. LAN Traffic Capture (Unencrypted)




b. 802.11 link-level Traffic Capture (Encrypted)

**Figure 2: Wi-Fi encrypted vs decrypted**

### 4.3 Encryption-Decryption Packet Transferability

Packet sizes are consistent when comparing encrypted packet capture data and its respective unencrypted data. We hypothesize that, since this relationship exists and a majority of the features are statistical, flows generated from publicly-available unencrypted data sets could be used to infer device types from encrypted data. Furthermore, this implies that the same flow tables should be produced by ProtoFlow for the encrypted and unencrypted versions of the same packet captures.

Figure 2 shows the comparison for an identical packet capture on Wi-Fi. Figure 2 (a) displays a packet capture that has been decrypted using the network password, which then reveals details such as TCP, UDP, IP addresses, etc. Figure 2 (b) is the same packet capture entirely encrypted on the link-level. Each of the packets shown are the same sequentially, and are identical in size. When comparing the encrypted and decrypted versions of a packet capture for both Zigbee and Bluetooth, the packet sizes are also identical in the same manner as Wi-Fi.

## 5 EXPERIMENTAL DESIGN

In this section we describe the process and implementation of collecting our data. Next, we see if it is possible to use both types of data to perform device classification between each other. Consequently, we state the following research questions (RQs):

- **RQ1:** Can IoT devices, on various networking protocols, successfully be identified using machine learning on the link-layer?
- **RQ2:** Can statistical flows developed from networked (unencrypted) traffic be used to identify device flows developed from link-layer (encrypted) traffic?

### 5.1 Link-Layer Device Classification

A Raspberry Pi (RPI) 400 is set into monitor mode to capture traffic without being connected to the LAN, Bluetooth traffic is captured using an Ubertooth One, and Zigbee traffic is captured using an Apimote. We interact with each lab device to generate more traffic. All traffic is periodically uploaded to an internal server for a week, then passed through ProtoFlow to create the CSV flow tables for each capture. The CSV files are merged into a large flow table, then labelled by cross-referencing a device database and thier identifiers. The labels are abstracted using the technique from Section 4.2.

Three ML algorithms are employed: Random Forest (RF), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). As a preprocessing step, devices not in our database are removed and, to fix class imbalance, a random sampling of 7,500 routers is removed. In total, the data distribution is 2,726 samples. All models receive a train/test split through a series of parameter testing, reaching a performance peak at 50% split. The RF model has a max depth of 11. The SVM has 1250 max iterations and a regularization of 1.25. The KNN model retains the default scikit configuration.

### 5.2 Encryption-Decryption Packet Relationships and Model Transferability

Two independent datasets are used in order to perform training and testing separately. This helps measure transferability of the models not only between different labs, but also if it is possible to train models on unencrypted traffic, and use them to classify devices on link-level traffic. The link-level traffic is obtained from the dataset generated in Section 5.1. The unencrypted traffic used is the UNSW IoT Device dataset [14], with traffic flows generated using NFStream [4]. The UNSW dataset is processed the same as the encrypted traffic flows, with each device type being abstracted. The abstractions are the categories in Table 2.

**Table 3: Link-level Classification Results**

| | KNN | | | | RF | | | | SVM | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Device Class** | **Precision** | **Recall** | **F1-Score** | **Support** | **Precision** | **Recall** | **F1-Score** | **Support** | **Precision** | **Recall** | **F1-Score** | **Support** |
| ereader | 0% | 0% | 0% | 3 | 0% | 0% | 0% | 2 | 0% | 0% | 0% | 3 |
| fitness_tracker | 0% | 0% | 0% | 2 | 100% | 50% | 67% | 2 | 100% | 50% | 67% | 2 |
| router | 94% | 95% | 95% | 420 | 100% | 99% | 99% | 402 | 94% | 99% | 96% | 403 |
| smart_assistant | 27% | 50% | 35% | 14 | 73% | 53% | 62% | 15 | 58% | 44% | 50% | 16 |
| smart_bridge | 59% | 64% | 61% | 69 | 88% | 94% | 91% | 71 | 72% | 42% | 53% | 74 |
| smart_bulb | 95% | 86% | 90% | 22 | 95% | 95% | 95% | 22 | 0% | 0% | 0% | 22 |
| smart_camera | 94% | 98% | 96% | 582 | 100% | 99% | 100% | 595 | 80% | 100% | 89% | 594 |
| smart_lock | 40% | 33% | 36% | 6 | 83% | 83% | 83% | 6 | 75% | 86% | 80% | 7 |
| smart_pet_feeder | 43% | 20% | 27% | 15 | 58% | 64% | 61% | 11 | 50% | 25% | 33% | 12 |
| smart_plug | 84% | 98% | 91% | 94 | 99% | 100% | 99% | 93 | 92% | 53% | 67% | 93 |
| smart_speaker | 50% | 40% | 44% | 20 | 91% | 45% | 61% | 22 | 10% | 6% | 7% | 18 |
| smart_switch | 64% | 58% | 61% | 24 | 92% | 82% | 87% | 28 | 50% | 48% | 49% | 25 |
| smart_vacuum | 0% | 0% | 0% | 1 | 0% | 0% | 0% | 2 | 0% | 0% | 0% | 1 |
| smartphone | 35% | 23% | 28% | 60 | 56% | 85% | 68% | 59 | 68% | 31% | 42% | 62 |
| streaming_stick | 25% | 6% | 10% | 31 | 72% | 55% | 62% | 33 | 33% | 10% | 15% | 31 |
| Overall | 84% | 86% | 84% | 1363 | 95 | 94 | 95 | 1363 | 79 | 82 | 79 | 1363 |

UNSW data is used for training, which contains 946,655 samples. For testing, we use our link-level dataset, containing 10,227 total samples in the flow table. Our training set is much larger than the prior task, which enables us to spend more time training for each device flow. We will use *deep* learning which allows learning and mapping more complex feature sets [7, 19]. We deploy a Long Short Term Memory (LSTM) model, since flow data is sequential. Implemented in TensorFlow, the model is trained for 250 epochs on a batch size of 80, with scaled features.

*5.2.1 Zigbee Traffic Capture and Decryption.* A TI-CC2531 is used to capture Zigbee encrypted traffic. The TI-CC2531 displays the 'handshake' packets within a capture to decrypt it. We capture traffic using the Killerbee framework running on a RPI 400 and save it into a .pcap file. Using Wireshark to read the packet capture, we input a global Trust Center Key[5], to decrypt of the transport key exchange, which is captured when a new device joins the Zigbee network. This key is also input into Wireshark thereafter, allowing for traffic using that transport key to be visible.

*5.2.2 Bluetooth Low Energy (BTLE) Traffic Capture and Decryption.* Initially, all Bluetooth traffic was captured using an Ubertooth One. However, the Ubertooth One frequently hops between the Bluetooth channels, yielding incomplete captures, preventing proper decryption. To solve this, we implement the reverse procedure: convert the decrypted traffic to encrypted traffic using the same algorithm as the specification.

Using a Pixel 4a, Bluetooth traffic is captured and stored in the Host Controller Interface (HCI) logs on the phone. All data on the HCI logs are decrypted by default. We then extract these logs using the Android Debug Bridge (adb), and analyze them with Wireshark. The HCI log file is read and exported to the .pcap file format for manual parsing using our own scripts. The scripts are written in Python, with the libraries Scapy and PyCryptodome. Scapy is used to parse the packet captures and PyCryptodome is used to encrypt the payload of each packet to export an identical, but encrypted, version of the packet capture. The encryption mechanism used is the 128-bit AES in CCM mode, which uses the same encryption method for securing data[6].

**Table 4: LSTM Transferability Results**

| Device Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| ereader | 0% | 0% | 0% | 6 |
| fitness_tracker | 0% | 0% | 0% | 6 |
| router | 99% | 39% | 56% | 8325 |
| smart_assistant | 1% | 9% | 1% | 32 |
| smart_bridge | 16% | 28% | 20% | 126 |
| smart_bulb | 0% | 2% | 0% | 49 |
| smart_camera | 2% | 3% | 3% | 1171 |
| smart_lock | 2% | 7% | 4% | 15 |
| smart_pet_feeder | 1% | 12% | 1% | 26 |
| smart_plug | 0% | 2% | 1% | 181 |
| smart_speaker | 1% | 8% | 2% | 36 |
| smart_switch | 2% | 9% | 3% | 46 |
| smart_vacuum | 0% | 0% | 0% | 3 |
| smartphone | 11% | 4% | 6% | 134 |
| streaming_stick | 0% | 3% | 0% | 60 |
| Overall | 81% | 32% | 46% | 10216 |

## 6 EXPERIMENTAL RESULTS

This section discusses the results for our research questions obtained from the two classification tasks using various models.

### 6.1 RQ1: Link-level Device Classification

Table 3 represents the results obtained from performing device classification on our fully-encrypted dataset on the link-level. RF obtains a weighted average F1-Score of 95%, with the best individual classes being the smart camera at 100% and routers at 99%. Classes with a smaller amount of support have a lower F1-Score in general. KNN has the second best weighted average F1-Score at 84%. Unlike the RF, the KNN does not reach an F1-Score of 100% for any category. However, it remains comparable to the RF with smart bulb, smart camera, smart plug, and router. The SVM is the worst with a weighted average F1-Score of 79%. The class for this model with the highest F1-Score is the router.

[5]https://www.hal9k.dk/sniffing-philips-hue-zigbee-traffic-with-wireshark/

[6]https://www.bluetooth.com/specifications/specs/?types=adopted&keyword=Core&filter=

### 6.2 RQ2: Encryption-Decryption Packet Transferability

Table 4 shows the results for training on UNSW data, and testing on our link-level data. The results are not as good as training and testing on a local lab level. Five out of the 15 classes obtain an F1-Score of 0%. There are two possible reasons for this: The patterns of network data collected and subsequent flows created are non-deterministic and not necessarily representative between environments. Second, the support for most of our data is also low. The overall F1-Score is 46%, which is improved mainly by the router class (56% F1-Score) and the smart bridge class (20% F1-Score). Aside from the router and smart bridge, eight out of 15 classes have F1 scores of a non-zero single digit. Based on the high F1-Scores for some classes, there is a potential for the other scores to be higher if we obtain more representative data from identical devices, and a larger breadth of data for both training and testing. This is more important if we seek to transfer models between environments. We have a lower distribution of classes in our encrypted dataset, which can also lower results; however, UNSW has a variety of devices of the same manufacturer as us, but not necessarily the same device type. Another way to boost the scores is extended training and better encrypted-decrypted feature selection.

## 7 CONCLUSION AND FUTURE WORK

In this paper we present methods with which to classify device types on fully-encrypted traffic from the link-level across three networking protocols: Wi-Fi, Zigbee, and Bluetooth. We successfully classify devices on these three networking protocols with a weighted average F1-Score reaching 95% with Random Forest. We then show that packet lengths remain the same for encrypted and unencrypted versions of the same data. As such, identical *statistical* flows based on packet length can be created. Results for this task show promise with more training data and potentially deep-learning techniques. Finally, we release an open-source tool capable of generating traffic flows for any parseable networking protocol.

For future work, we plan to revisit RQ2 with more representative data that contains specific device similarity, and identify additional features that are still derivable from the link-level to maintain protocol-agnosticism. Following classification success, we also find potential in novel generative modeling using state-of-the-art approaches. Such approaches may include variants of stable-diffusion models to support binary data as opposed to image generation related tasks. Using these models, rather than a Generative Adversarial Network or LSTM networks, one may generate unique traffic with a higher-quality training data distribution.

## REFERENCES


[1] Antar Shaddad Abdul-Qawy, PJ Pramod, E Magesh, and T Srinivasulu. 2015. The internet of things (iot): An overview. *International Journal of Engineering Research and Applications* 5, 12 (2015), 71–82.

[2] Prashant Anantharaman, Liwei Song, Ioannis Agadakos, Gabriela Ciocarlie, Bogdan Copos, Ulf Lindqvist, and Michael E. Locasto. 2020. IoTHound: Environment-Agnostic Device Identification and Monitoring. In *Proceedings of the 10th International Conference on the Internet of Things* (Malmö, Sweden) *(IoT '20)*. Association for Computing Machinery, New York, NY, USA, Article 7, 9 pages. https://doi.org/10.1145/3410992.3410993

[3] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. 2017. Understanding the mirai botnet. In *26th USENIX security symposium (USENIX Security 17)*. 1093–1110.

[4] Zied Aouini and Adrian Pekar. 2022. NFStream: A flexible network data analysis framework. *Computer Networks* (2022), 108719.

[5] Noah Apthorpe, Danny Yuxing Huang, Dillon Reisman, Arvind Narayanan, and Nick Feamster. 2019. Keeping the smart home private with smart (er) iot traffic shaping. *Proceedings on Privacy Enhancing Technologies* 2019, 3 (2019), 128–148.

[6] Sourangsu Banerji and Rahul SinghaChowdhury. 2013. Wi-Fi & Wi-MAX: A Comparative Study. *CoRR* abs/1302.2247 (2013). arXiv:1302.2247 http://arxiv.org/abs/1302.2247

[7] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. 2017. *Deep learning*. Vol. 1. MIT press Cambridge, MA, USA.

[8] Salim Jibrin Danbatta and Asaf Varol. 2019. Comparison of Zigbee, Z-Wave, Wi-Fi, and Bluetooth Wireless Technologies Used in Home Automation. In *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*. 1–5.

[9] Gerard Draper-Gil., Arash Habibi Lashkari., Mohammad Saiful Islam Mamun., and Ali A. Ghorbani. 2016. Characterization of Encrypted and VPN Traffic using Time-related Features. In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy - ICISSP,*. INSTICC, SciTePress, 407–414.

[10] Aviv Engelberg and Avishai Wool. 2021. Classification of Encrypted IoT Traffic Despite Padding and Shaping. arXiv:2110.11188 [cs.CR]

[11] Sinem Coleri Ergen. 2004. ZigBee/IEEE 802.15. 4 Summary. *UC Berkeley, September* 10, 17 (2004), 11.

[12] Arash Habibi Lashkari. 2018. CICFlowmeter-V4.0 (formerly known as ISCXFlowMeter) is a network traffic Bi-flow generator and analyser for anomaly detection. https://github.com/ISCX/CICFlowMeter. https://doi.org/10.13140/RG.2.2.13827.20003

[13] Arash Habibi Lashkari, Gerard Draper Gil, Mohammad Mamun, and Ali Ghorbani. 2017. Characterization of Tor Traffic using Time based Features. 253–262. https://doi.org/10.5220/0006105602530262

[14] Ayyoob Hamza, Hassan Habibi Gharakheili, Theophilus A Benson, and Vijay Sivaraman. 2019. Detecting volumetric attacks on lot devices via sdn-based monitoring of mud activity. In *Proceedings of the 2019 ACM Symposium on SDN Research*. 36–48.

[15] Matt Hayes. 2018. *What is a Network Traffic Flow?* Retrieved September 28, 2021 from https://mattjhayes.com/2018/09/26/what-is-a-network-traffic-flow/

[16] Oleh Horyachyy. 2017. *Comparison of Wireless Communication Technologies used in a Smart Home: Analysis of wireless sensor node based on Arduino in home automation scenario*. Ph. D. Dissertation. http://urn.kb.se/resolve?urn=urn:nbn:se:bth-14845

[17] Kahraman Kostas, Mike Just, and Michael A Lones. 2022. IoTDevID: A behavior-based device identification method for the IoT. *IEEE Internet of Things Journal* (2022).

[18] Sumit Kumar, Sumit Dalal, and Vivek Dixit. 2014. The OSI model: Overview on the seven layers of computer networks. *International Journal of Computer Science and Information Technology Research* 2, 3 (2014), 461–466.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[20] Rajib Maiti, Sandra Siby, Ragav Sridharan, and Nils Ole Tippenhauer. 2017. Link-Layer Device Type Classification on Encrypted Wireless Traffic with COTS Radios. 247–264. https://doi.org/10.1007/978-3-319-66399-9_14

[21] OWASP. [n. d.]. OWASP IoT Top 10. https://owasp.org/www-project-internet-of-things/

[22] C Muthu Ramya, M Shanmugaraj, and R Prabakaran. 2011. Study on ZigBee technology. In *2011 3rd International Conference on Electronics Computer Technology*, Vol. 6. IEEE, 297–301.

[23] Vincent Ricquebourg, David Menga, David Durand, Bruno Marhic, Laurent Delahoche, and Christophe Loge. 2006. The smart home concept: our immediate future. In *2006 1st IEEE international conference on e-learning in industrial electronics*. IEEE, 23–28.

[24] Stanislav Safaric and Kresimir Malaric. 2006. ZigBee wireless standard. In *Proceedings ELMAR 2006*. IEEE, 259–262.

[25] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. 2018. Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing* 18, 8 (2018), 1745–1759.

[26] Cristina Stolojescu-Crisan, Calin Crisan, and Bogdan-Petru Butunoi. 2021. An IoT-Based Smart Home Automation System. *Sensors* 21, 11 (2021). https://doi.org/10.3390/s21113784

[27] P Suresh. 2016. Survey on seven layered architecture of OSI model. *International Journal of research in computer applications and robotics* 4, 8 (2016), 1–10.

[28] Charles Wheelus and Xingquan Zhu. 2020. IoT Network Security: Threats, Risks, and a Data-Driven Defense Framework. *IoT* 1, 2 (2020), 259–285. https://www.mdpi.com/2624-831X/1/2/16

[29] Bharati Wukkadada, Kirti Wankhede, Ramith Nambiar, and Amala Nair. 2018. Comparison with HTTP and MQTT In Internet of Things (IoT). In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*. 249–253.