AirTagged: A Dataset and Processing Framework for Heterogeneous High-Density IoT Environments

Katharina O. E. Müller[1], Stefan Saxer[1], Daria Schumm[1], Weijie Niu[1], Bruno Rodrigues[2], Burkhard Stiller[1]
[1]Communication Systems Group CSG, Department of Informatics IfI, University of Zurich UZH, Switzerland
[2]Embedded Sensing Group ESG, School of Computer Science SCS, University of St. Gallen HSG, Switzerland
E-mail: [mueller|schumm|niu|stiller]@ifi.uzh.ch, stefanrichard.saxer@uzh.ch, bruno.rodrigues@unisg.ch

***Abstract*—Personal Bluetooth Low Energy (BLE) trackers such as AirTag help locate lost items but can be misused for stalking. Research indicates that BLE trackers can be identified through their transmitted packets, thus offering potential for machine learning (ML) solutions. However, current packet datasets lack the scale and diversity needed for real-world applicability.**

This paper presents an open-source 200-hour BLE advertisement packet dataset focused on personal tags, enabling future ML-based device detection approaches. Additionally, introduces the first large-scale BLE data preprocessing framework for efficient and modular BLE packet preprocessing. The Framework is showcased on the dataset, demonstrating feature extraction, labelling, and dynamic plotting. This paper lays the groundwork for IoT device detection in high-density, heterogeneous environments, enabling future advances in BLE device classification.

***Index Terms*—Internet of Things, Application-Layer Traffic, Dataset, Classification, Privacy**

I. Introduction

Personal BLE tracking devices, such as Apple's AirTag, have become popular for locating lost items [1]. However, their small size and discreet design make them susceptible to misuse, particularly for stalking [2]. Trackers can be easily concealed in and under vehicles or in personal belongings, making visual detection nearly impossible [3]. These privacy risks are not just theoretical; documented stalking cases [4], [5] involving BLE trackers have even escalated to murder [6]. To mitigate these concerns, manufacturers such as Apple have introduced smartphone applications to detect only their own trackers [7], [8]. However, these solutions are vendor-specific and inadequate for comprehensive privacy protection [9], [10]. Thus, a universal detection approach is needed, not only for personal trackers but also for the growing range of BLE-enabled IoT devices (e.g., headphones, laptops, wearables) that broadcast similar advertisements and coexist in dense environments. Distinguishing trackers from this broader IoT landscape is essential for realistic detection and classification.

ML-driven detection presents a promising approach but requires large-scale data collection, particularly given the heterogeneous landscape of millions of IoT devices [11]. Thus, providing diverse, high-quality training data to distinguish between legitimate devices and unauthorized trackers, as well as distinguish trackers from all other IoT devices, is crucial. Moreover, real-world packet captures in high-traffic environments (*e.g.*, train stations) are necessary to ensure model robustness and generalizability. Since exhaustive data collection across all Bluetooth Low Energy (BLE) devices is unfeasible, a structured dataset is needed to facilitate feature analysis and extraction, anomaly detection, and the development of privacy-preserving tracking countermeasures.

Thus, this paper presents the first open-source, large-scale labeled dataset of BLE advertisement packets for ML, alongside a real-world dataset and a robust preprocessing tool tailored for high-volume BLE data. Captured in high-density public environments, the dataset offers diverse and representative signals critical for training scalable and reliable ML models. The preprocessing tool enables efficient feature extraction and data handling at scale. Together, these resources establish a foundation for developing vendor-agnostic, privacy-preserving ML solutions to detect unauthorized BLE tracking across diverse real-world settings. This paper contributes the following:

- Labeled BLE Advertisement Packet Dataset: 13 million BLE packets collected over 200 hours, including isolated and combined device- and state-specific traces[1].
- Unlabeled BLE Advertisement Packet Dataset: Two train station recordings, with 588,021 packets combined[1].
- Open Source Task-Group-Framework: A modular, scalable pipeline for efficient large-scale data preprocessing, labeling, dynamic visualization and feature extraction[2].
- BLE packet dataset analysis and feature definition using the Task-Group Framework and its visualization extension for structured data processing and representation.

This paper is structured as follows: Section II reviews related datasets; Section III outlines BLE advertisement packet structure. Section IV details dataset collection and processing, followed by an overview of the Task-Group-Framework in Section VI. Section VII presents a comprehensive dataset analysis, with a summary and future work in Section VIII.

II. Related Work

Research on BLE varies from device identification and network traffic analysis to studies focusing on the reverse engineering of the Apple [12] and Samsung [13] ecosystem. Several published BLE datasets have focused on localization and indoor positioning by leveraging Received Signal Strength Indicator (RSSI) data [14], [15], [16]. While valuable for

[1]kaggle.com/datasets/stefansaxer/ble-packets-from-tracking-devices
[2]github.com/stsaxe/Dataset-and-Task-Group-Framework-for-Heterogenous-High-Density-IoT-Environments

978-3-903176-75-1

spatial inference, these datasets publish only RSSI values, not the entire needed advertising packets.

[17] studied hybrid approaches to device fingerprinting in indoor environments using both BLE and Wi-Fi data, but again relying on RSSI for environment-based localization. In contrast, IoT device classification has seen more progress in Ethernet-based networks, where analysis of IP and MAC layer traffic provides meaningful insights [18]. Unfortunately, such methods are not transferable to BLE, due to layer and packet structure incompatibility.

A limited number of datasets examine BLE traffic for machine learning applications, only focusing on physical-layer signal data or modulation characteristics [19], [20]. Recent studies have also investigated commercial BLE-based trackers (*e.g.*, AirTag), to assess localization and privacy risks across various regions [21], [22]. These datasets often include GPS coordinates and aggregate counts of detected devices, offering valuable macro-level insights into tracker deployment. However, they do not include detailed link-layer metadata or advertisement packets, limiting their usefulness for low-level protocol analysis and device classification. In summary, while a wide range of BLE-related datasets and studies exist, none focus on analyzing advertisement metadata at the link layer, especially for the purpose of identifying and classifying BLE peripherals such as personal trackers.

III. Background

In BLE, Peripherals (*e.g.*, trackers) broadcast advertisements, while Centrals (*e.g.*, smartphones) scan for them [23].

A. BLE Devices, Trackers, and States

BLE centrals and peripherals periodically transmit advertisements, but personal trackers advertise continuously, adjusting frequency by state. The studied devices operate in 2–4 states with different privacy implications: e.g., AirPods alternate between *nearby* (low-frequency, low risk) and *lost* (high-frequency, high risk). Find My trackers add an *unpaired* state, while Tile and SmartTag include a *searching* state triggered by user activation. Transmission intervals range from >1 s in *nearby* mode to 100–250 ms in visibility-critical states (*lost*, *searching*, *unpaired*). Non-tracker devices are instead categorized as *online* (trackable via BLE and cellular) or *offline* (BLE only).

B. BLE Advertisement Packet Structure

The link layer packet structure follows a predefined format, see Figure 1, focusing specifically on the packet differences in the payload structure of the *Advertising Protocol Data Unit* (ADVPDU) as features for classification. An ADVPDU consists of a header and a variable-length payload ($\leq$ 37 bytes) containing the source address and one or more *advertisement data types*. These blocks may repeat, vary in length, or be absent. Examples include *manufacturer specific data* (company ID, *e.g.*, , 0x004C = Apple, plus vendor-defined content), Service UUIDs (basic or with Service Data of fixed length), as well as Flags (four bits) or an 8-bit TX Power Level.

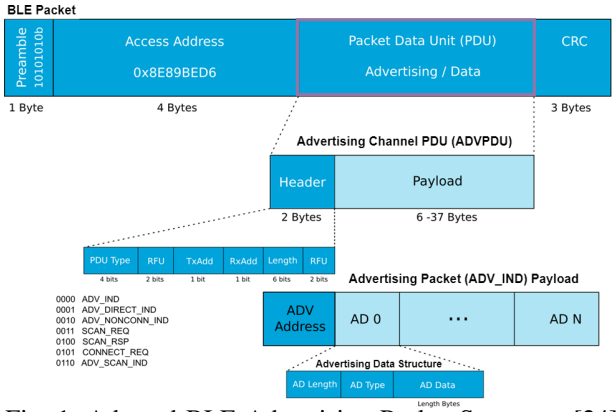


Fig. 1: Adapted BLE Advertising Packet Structure: [24]

IV. Dataset Collection and Processing

This section describes the dataset requirements, device choice, dataset collection, feature extraction, and data labeling.

A. Dataset Requirements

The first step in collecting data is to define key requirements to ensure the dataset supports effective model training. It is essential that the data covers multiple device states (*cf.* Section III-A), since models must be able to distinguish between them; partial state coverage is insufficient. The dataset should include both conventional BLE trackers (*e.g.*, AirTags) and other BLE-enabled devices that can act as trackers (*e.g.*, iPhones). Non-tracker BLE devices common in public transit spaces (e.g., phones, tablets, laptops) were included, while smart-home devices were omitted. The dataset must include sufficient samples per device and state to capture features such as packet rate over time. For instance, aggregating 10 packets/s into 15-s intervals reduces 600 samples/min to just four. As no existing dataset met these criteria (see Section II), a custom dataset was created.

B. Device Choice Overview

Devices are split into three categories:

1) ***Conventional BLE Trackers****:* Trackers were selected based on their popularity in the consumer tracking market, determined by surveying available BLE trackers on the largest national online marketplace and summarized in Table I.

TABLE I: BLE Trackers, Manufacturers, and Class Labels

Tracker	Manufacturer	Network	Class Label
AirTag	Apple	Apple Find My	`"AirTag"`
SkyTag	4Smarts	Apple Find My	`"SkyTag"`
One	Chipolo	Apple Find My	`"Chipolo"`
SmartTag	Samsung	Samsung Galaxy Find	`"SmartTag"`
Mate	Tile	Tile Network	`"Tile"`

Vendors such as Tile, offer many similar tracker models, thus, a single model per vendor was selected, assuming they use the same tracking approach and packet rates.

2) ***Other BLE Tracking Capable Devices****:* Include BLE devices, which are part of a crowdsourced finding network, such as Apple FindMy, and can also be tracked within that system. These devices include: an iPhone 11, a 3rd generation

iPad Pro 12.9", a 2019 MachBook Pro 15", and 3rd generation Airpods. Devices supporting Tile's tracking network include HP and Lenovo laptops, which were excluded due to limited access and lesser relevance. As of May 2024, Google introduced its Find My Device network for Android devices [25]. However, Google's tracking network was excluded as it launched after our data collection period. Therefore, to minimize inference bias, the dataset focused on capturing as many Apple devices as possible.

*3) **Other BLE Devices**:* To reduce bias toward trackers and Apple devices, data was also collected from a diverse set of BLE-enabled devices. These included a Lenovo Yoga laptop, Lenovo Tab 12 Pro (with Bluetooth pencil and keyboard), Samsung Galaxy S23 Ultra smartphone, Ultimate Ears Boom 2 speaker, JBL BT 510 headphones, Logitech K810 keyboard, MX Anywhere 2S mouse, and an Xbox One controller. Although our dataset cannot include all IoT device types, it covers representative categories (mobile, audio, input, peripherals) frequently encountered in public environments.

C. Data Collection and Setup

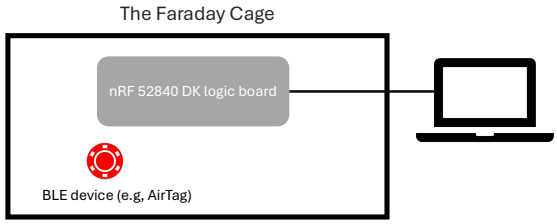
The actual data was collected in two separate environments: (1) a controlled basement with a aluminium lined metal box to minimize interference as much as possible and isolate specific devices or combination of devices from all others to create the labeled dataset of 13 million packets, and (2) unfilterred Zurich main station to create a real-world unlabeled dataset to test against, with as many unknown devices and interference as possible. To do so, both data collection setups utilized an nRF52840 Development Kit (DK) from Nordic Semiconductor to passively capture BLE advertisement packets.

During data collection, the DK sequentially cycles between channels 37, 38, and 39 [26], which are the three channels reserved for advertising in BLE communication.

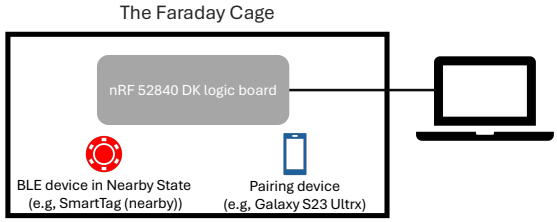
For environment (1) isolating the targeted device(s) from all other devices was paramount, as they can rapidly switch states when an owner device is nearby. For instance, a tracker can transition from a "lost" to a "nearby" state almost instantly. As such, to accurately collect data for all device states, three setups were necessary.

First, each device had to be individually placed within the Faraday cage, with only the DK and a flat USB micro B cable leading out of the closed box to a PC with Bluetooth turned off to send the captures to, see Figure 2a. Since there was only one BLE device in the cage, the packets could be captured without interference from other devices and directly labeled. This allowed the capturing of the AirPods and conventional trackers in the "lost", "unpaired", and, where possible, "searching" states. As well as the individual captures for the iPhone, MacBook, and iPad in both the "online" and "offline" states.

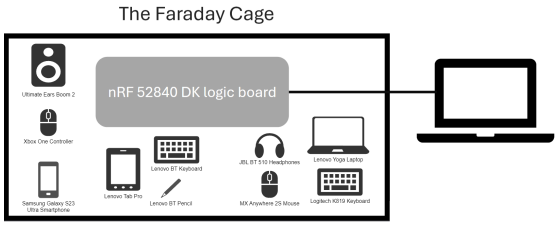
Second, to collect BLE packets in the nearby state, in addition to the AirPods and conventional trackers, the paired owner device needed to be placed in the Faraday cage (*cf.* Figure 2b). For example, a SmartTag (nearby) and a Samsung Galaxy S23 Ultra (online) were included in the Faraday cage together, which is necessary as a device in a "nearby" state will go to a "lost" state if the owner device is out of range.



(a) Setup 1: "unpaired", "lost", "searching", "online", and "offline" states



(b) Setup 2: "nearby" state



(c) Setup 3: all other devices

Fig. 2: Environment 1 Data Collection Setups

Last, the setup in Figure 2c was needed to collect the garbage class, or more specifically, all other BLE devices (*cf.* Section IV-B3), which needed to be collected simultaneously.

All collected data was saved in CSV and PCAP format and published on Kaggle; see a summary of all datasets, including device name, state, and capture duration, in Table II.

V. Data Labeling Approaches

Assigning class labels to captured packets is essential for preprocessing and subsequent ML approaches. Labeling source addresses directly would add unnecessary complexity, as packet labeling depends on the number of devices captured simultaneously.

1) Single-device labeling: When capturing a single device in isolation, all packets receive the same class label, regardless of source address randomization. This straightforward method was used for most of the collected data.

2) Multiple devices with one label: If several devices share a class label, all packets can be labeled accordingly. This applied only to the "Other BLE Devices," which were captured together in one PCAP file.

3) Multiple devices with different labels: When multiple devices with distinct class labels are captured together, transmissions overlap, making labeling challenging. This occurs, for example, when trackers operate in the "nearby" state or AirPods connect to another device.

A **naïve labeling** approach uses source addresses as identifiers, but frequent randomization restricts this method to short time frames and renders it unsuitable for large-scale datasets.

TABLE II: Overview of All Collected and Published Datasets

Find My Trackers (AirTag, SkyTag, Chipolo One)	
Tracker_lost	12h
Tracker_nearby	12h
Tracker_nearby_3h	3h
Tracker_nearby_labeled_training	10m
Tracker_nearby_labeled_evaluation	10m
Tracker_unpaired	12h, 4min, 60s respectivly
Samsung SmartTag	
SmartTag_lost	12h
SmartTag_nearby	12h
SmartTag_nearby_3h	3h
SmartTag_nearby_labeled_training	short
SmartTag_nearby_labeled_evaluation	short
SmartTag_unpaired	5m
SmartTag_searching	12s
Tile Mate	
Tile_lost	12h
Tile_nearby	12h
Tile_nearby_3h	3h
Tile_nearby_labeled_training	13m
Tile_nearby_labeled_evaluation	23m
Tile_unpaired	60s
Tile_searching	10s
Apple AirPod	
AirPod_lost	12h
AirPod_nearby	12h
AirPod_nearby_3h	3h
AirPod_nearby_labeled_training	5m
AirPod_nearby_labeled_evaluation	5m
Other Apple Devices (iPhone, MacBook, iPad)	
iDevice_online	12h
iDevice_offline	12h
Other Devices	
Other_Device	2.5h

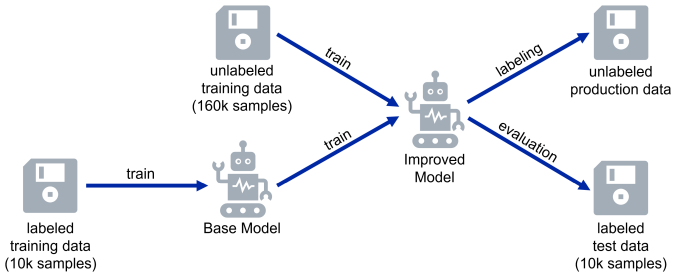


Fig. 3: Semi-Supervised Learning

To overcome this, we propose a **semi-supervised ML labeling** approach focused on two-device scenarios. Small labeled datasets are created using the **naïve labeling** method, a neural network is trained, and then refined with self-training on larger unlabeled data. The resulting model is evaluated on a test set labeled using the **naïve labeling** approach (cf. Figure 3).

This **semi-supervised ML labeling** process enables scalable labeling of device-state combinations and lays the foundation for extending to more complex multi-device scenarios.

A. Feature Extraction

Feature extraction is key for all datasets, both labeled and unlabeled, as ML models rely on extracted features. For labeled datasets, extraction can occur before or after labeling. The Task-Group-Framework can, for example, be utilized for feature extraction methods, such as analyzing captured data, as showcased in Section VII, and modeling. The **automatic labeling** approach requires the training of two models for each device combination with respect to the two presented extraction methods. In total 14 features were identified from the BLE packet fields. For personal tag distinction and labeling, two key features can be combined effectively. First, the Company ID provides manufacturer-level identification (e.g., Apple, Samsung, Tile), which narrows down the device family. Second, manufacturer-specific fields refine the labeling: for Apple devices, the Continuity Type (CT) byte distinguishes AirTags from other Find My accessories, while for Samsung devices, the SmartTag Type bits in the Service Data separate different SmartTag states. Together, these features enable accurate differentiation and labeling of personal tracking tags across ecosystems.

B. Automatic Labeling: Semi-Supervised Learning

For the **automatic labeling** approach, shown in Figure 3, features are first extracted, and then the **naïve labeling** approach is used to create a small labeled dataset with a few thousand samples. A neural network is then trained on this data using scikit-learn's default model, which consists of one hidden layer with 100 neurons and a ReLU activation function. This model was chosen for its simplicity, fast training time, and suitability for structured tabular data such as BLE packet features. It offers a good balance between capacity and overfitting risk, making it a practical choice for this relatively simple binary classification task. To improve robustness, the trained model is further refined using self-training on a much larger, unlabeled dataset via scikit-learn's semi-supervised learning implementation. Although self-training can enhance the model's generalization to unseen data, it may not always improve test set performance. In such cases, the original base model is used. Despite the limited size of the labeled dataset, the straightforward nature of the classification task allows the model to achieve reliable results. Once trained, the model can label other unlabeled datasets containing the same devices in the same states as those seen during training. Lastly, the model's performance can be evaluated using the labeled test set. In most cases, the labeling models achieved an excellent accuracy of over 99%, see Table III for a summary. Due to the excellent accuracy, no further models were tested.

TABLE III: Device Classification Confusion Matrix Results

Device 1	Device 2	Accuracy Device 1 (%)	Accuracy Device 2 (%)	Misclassification Rate (%)
SmartTag	Galaxy S23	99.34	100.00	0.66
Chipolo	iPhone	99.65	100.00	0.35
AirTag	iPhone	99.92	100.00	0.08
AirPod	iPhone	100.00	100.00	0.00
Tile	iPhone	99.81	100.00	0.19
SkyTag	iPhone	99.90	100.00	0.10

VI. Task-Group-Framework

The Task-Group-Framework is a custom device-agnostic tool and thus directly applicable to other IoT data once collected. It handles data preprocessing and visualization,

featuring configurable, deterministic task pipelines with multi-threading support. As discussed in Section IV, data processing involves steps such as feature extraction, labeling, and appending states to class labels (e.g., "AirTag" becomes "AirTag (lost)"). While some steps like feature extraction can be reused across datasets, others such as labeling and state assignment are dataset-specific. To avoid code duplication and improve efficiency, the Task-Group-Framework enables a single, adaptable pipeline that uses runtime parameters to activate or skip steps as needed. This approach eliminates the need for separate pipelines for each device-state combination, ensuring scalability and maintainability.

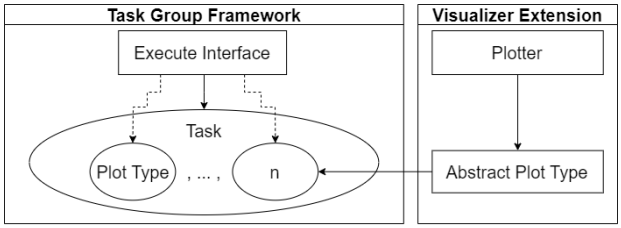


Fig. 4: Task-Group-Framework and Visualizer Components

The Task-Group-Framework structures data processing and visualization through modular TaskGroups that handle preprocessing, labeling, and state assignment. Each TaskGroup may include multiple Tasks or nested TaskGroups, enabling scalable, reusable workflows (Figure 4). Visualization is provided by dedicated Tasks based on a common Plotter superclass, supporting bar, box, stacked, grouped, and line plots. Plots are automatically refined with captions, labeled axes, sorted data, and optimized formatting for clarity and quality.

VII. Results and Evaluation

This Section outlines results and insights, covering dataset collection and processing using the Task-Group-Framework.

A. Overview of Generated Datasets

The generated datasets vary by device, reflecting their unique characteristics, with a full overview in Table II. Each Environment 1 dataset was captured continuously without combining separate recordings. Production datasets span 12 hours, ensuring sufficient data for ML models. With a 10-second aggregation/sequence length, this results in 4,320 samples, with a 1,080-sample test set (25%), providing an evaluation accuracy granularity of 0.1%. Additionally, two Environment 2 datasets were created at the Zurich central station, a real-world high-density IoT environment, "_V1," is 10 minutes long, while the second, "_V2," is 35 minutes long with over half a million unknown BLE advertisement packets.

B. Labeling, States, and CT

Packet labeling was performed using the Task-Group-Framework, with automatic labeling achieving over 99% test accuracy (Table III). After class labeling, device states were appended to create device-state categories for analysis.

Apple's continuity services, such as the Find My network, use structured manufacturer-specific data with service-specific encodings [12]. All continuity packets include a leading byte, the CT, which identifies the service (e.g., 0x12 for Find My) [27]. For Apple devices, CTs help distinguish tracker types. For example, long 0x12 packets include a public key and indicate an offline (BLE-only) device, while shorter ones indicate online devices (BLE with cellular/Wi-Fi). To capture this distinction, the Task-Group-Framework appends the CT to the class label before the state, e.g., "iPhone CT 10 (offline)," enabling more precise classification of trackable vs. non-trackable packets. Similarly, Samsung encodes device state in its Service Data [13], specifically in bits 5–7, referred to as the SmartTag type. This field is extracted to accurately label tracker states.

C. Plots and Analysis

For this paper, 24 plot types were devised to analyze each identified feature across 9 device types and their various states, which could be up to 4, as in the case of the Tile, thus resulting in hundreds of plots. For brevity, only the most relevant to *Conventional BLE Tracker* distinction will be discussed, with the full analysis available on GitHub.

Figure 6 shows the number of packets found for AirTag, Chipolo, and SkyTag across their 3 states, with the packet count remaining consistent across all three trackers in the "nearby" and "lost" states, indicating a stable packet rate. However, in the "unpaired" state, it varies significantly, highlighting its potential as a feature for tracker distinction; nevertheless, due to the different capture duration and therefore significantly reduced number of packets, it is questionable whether the Chipolo tracker and the SkyTag in their "unpaired" state can be used for modeling.

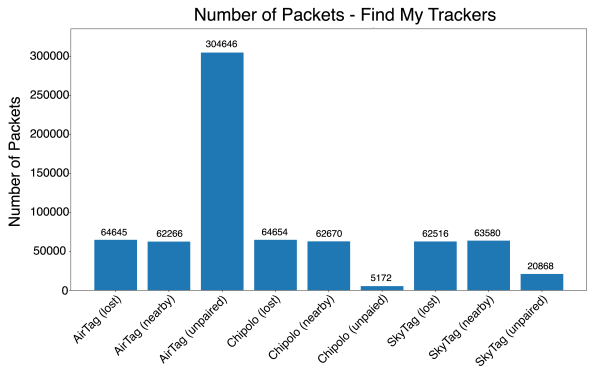
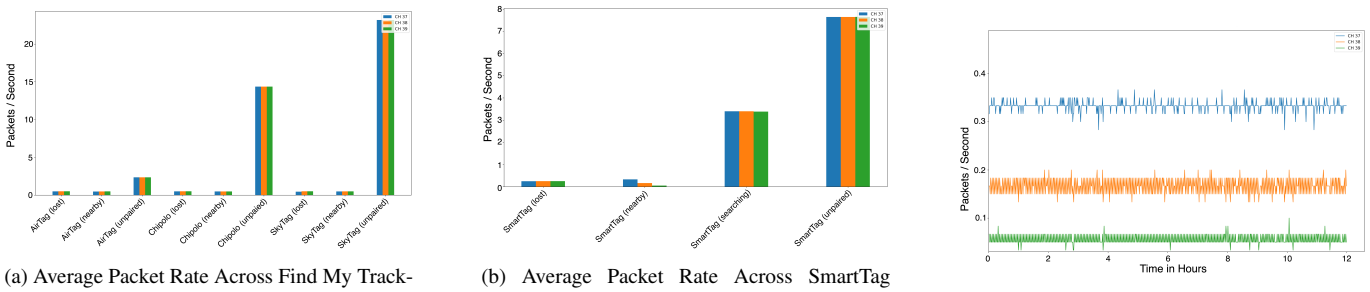


Fig. 6: Number of Packets - Find My Tracker

The box plot in Figure 7 reveals clear differences in manufacturer-specific data across tracker states. The collapsed boxes (*i.e.*, orange median lines) indicate tightly clustered values, with nearly all data points identical except for a few outliers (*i.e.*, black circles), suggesting highly consistent packet lengths in most states, especially "lost" and "unpaired." Closer analysis shows length differences between the "nearby" and "lost" states are due to the presence or absence of the public key, consistent with variations in header and packet length. Chipolo and SkyTag lack manufacturer-specific data



(a) Average Packet Rate Across Find My Trackers and States

(b) Average Packet Rate Across SmartTag States

(c) Graph of Packet Rate of SmartTag (nearby)

Fig. 5: Packet Rates for Find My Trackers, AirTag, and SmartTag

in the "unpaired" state, highlighting differences from AirTag. Outliers in the "nearby" state are due to labeling errors, or match the "lost" state's median, indicating state switching.

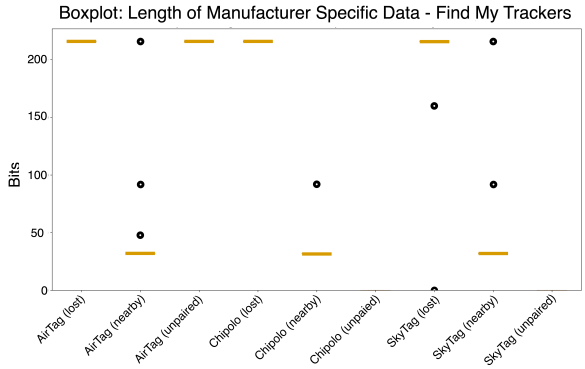


Fig. 7: Boxplot of Length of Manufacturer Specific Data

AirTag packet rate analysis reveals three key findings. First, packet rates are generally stable, except for Chipolo and SkyTag in the "unpaired" state, likely due to short captures (Figure 5a). Second, no major differences are observed across BLE channels. Third, outliers in the "unpaired" state show packet rates up to ten times the median, occurring only during the first 600 seconds after activation (Figure **??**). This temporary high-rate phase, triggered by pulling the security latch, is likely intended to expedite pairing and serves as a clear indicator for state identification.

Compared to AirTags, SmartTag packets are harder to distinguish across states. All states share identical structure, fixed 160-bit Service Data, and consistent use of UUIDs and flags. Only the status bits differ, with "unpaired" and "searching" sharing values. Unlike AirTags, SmartTag PDU type and packet structure offer little variation. Packet rate, especially in the "nearby" state, remains the most distinguishing feature, with it varying significantly across states, see Figure 5b. In "nearby," channel usage is uneven, with lower channels showing higher packet rates, while other states are uniform. Additionally, Figure 5c reveals that the packet rate in the "nearby" state differs between each channel over the entire captured time, and correlates with the distribution of packets among the channels. The higher the packet rate, the higher the relative share of packets.

In comparison, Tile's "lost" and "nearby" states are identical in packet structure and static in BLE address, making them well-suited for packet rate modeling. However, the "searching" and "unpaired" states have too few packets for modeling. Unlike AirTag and SmartTag, Tile does not differentiate between "lost" and "nearby" states. Notably, "searching" packets occasionally include undocumented Apple CTs. Service Data length is fixed at 80 bits in active states, differing from SmartTag, which may aid classification. As with other trackers, PDU type remains constant and uninformative.

D. Overarching Insights and Limitations

While the analysis above highlights only key findings, the broader evaluation offers important insights and limitations for BLE tracker classification. Several features, such as malformed packet percentage, broadcast rate, and protocol type, show little variation and can be excluded. Find My trackers (AirTag, Chipolo, SkyTag) are technically similar, except AirTag differs in the "unpaired" state. Samsung SmartTag enables "lost" vs. "nearby" differentiation via Service Data status bits, but the "unpaired" and "searching" states are underrepresented. Similarly, Tile lacks a distinct "nearby" state, so "lost" and "nearby" should be merged, while its other states provide too few samples for modeling. These limitations highlight the need for cautious feature selection to avoid misleading model behavior.

Apple devices (iPhone, iPad, MacBook) all rely on continuity protocols, making device-level classification infeasible; instead, classification should rely on CT values. AirTags and iDevices can be separated by PDU type, while "online" and "offline" states differ in manufacturer-specific data length. AirPods act as hybrids, producing both AirTag-like and iDevice-like traffic, and therefore form a distinct class. However, overlapping features across devices raise the risk of models overfitting to dataset-specific artifacts rather than generalizable characteristics.

Overall, effective modeling requires complex preprocessing, which we address via the Task-Group-Framework pipeline. Relevant features range from basic BLE attributes (e.g., packet length, PDU type) to higher-level traits (e.g., CT and SmartTag type), with categorical variables encoded for automatic labeling. Yet, not all devices are fully separable; grouping similar ones (e.g., Find My trackers) improves robustness, but

may obscure device-specific nuances. Given the limited data in certain states and potential feature overlap, there remains a risk of overfitting, underscoring the importance of validating on diverse real-world datasets.

VIII. Summary and Future Work

This paper presents over 200 hours of BLE advertisement captures, yielding 13 million labeled packets from 10 devices across up to 4 states, plus two real-world inference datasets with more than half a million packets. We introduced a structured methodology for acquisition, preprocessing, feature extraction, and evaluation in high-density IoT environments, and showcased the Task-Group-Framework, the first large-scale BLE preprocessing tool supporting labeling, automatic labeling, and feature extraction. Together, these contributions establish a foundation for ML-based BLE tracker classification and, more broadly, for distinguishing trackers from heterogeneous IoT devices in dense environments.

Although this study focuses on trackers, the extracted features (e.g., advertisement rate, manufacturer fields, CT values) generalize to other BLE IoT devices. Validating this requires broader datasets, which we identify as a key next step. A further limitation is overlapping features and underrepresented states, which risk overfitting and limit inference, particularly in the **Other BLE Devices** class. Since collecting data from all IoT devices is infeasible, future work will target ML-based tracker classification, validation on diverse real-world datasets, and synthetic data generation (e.g., VAEs, GANs, rule-based methods) to expand coverage and improve robustness.

IX. Ethical Statment

The data collected in this study consists solely of BLE advertisement data, which is freely broadcast and publicly accessible metadata. It does not contain any personally identifiable information, thus anonymous, or patient data, and all payload is encrypted. Following UZH's ethical policy, such work does not require explicit ethical approval.

References

[1] G. Celosia and M. Cunche, "Saving private addresses: An analysis of privacy issues in the bluetooth-low-energy advertising mechanism," in *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2019, pp. 444–453.

[2] T. Mayberry, E. Fenske, D. Brown, J. Martin, C. Fossaceca, E. C. Rye, S. Teplov, and L. Foppe, "Who tracks the trackers? circumventing apple's anti-tracking alerts in the find my network," in *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*, 2021, pp. 181–186.

[3] "Class Action," Jun. 2022, [Accessed 15-06-2025]. [Online]. Available: https://www.classaction.org/media/hughes-et-al-v-apple-inc.pdf

[4] "Ex-partner Uses Apple AirTag to Stalk Ahmedabad Woman, Device Found Hidden Under Driver's Seat," [Accessed 14-06-2025]. [Online]. Available: https://bit.ly/4amwV9r

[5] D. Hoffmeyer, "Airtag-Stalking: Hat Apple die Gefahr unterschätzt?" *Neue Zürcher Zeitung*, Jun. 2022, [Accessed 16-06-2025]. [Online]. Available: https://www.nzz.ch/panorama/airtag-stalking-hat-apple-die-gefahr-unterschaetzt-ld.1688765

[6] M. Gault, "Woman Allegedly Used Apple AirTag to Track and Kill Her Boyfriend," Jun. 2022, [Accessed 14-06-2025]. [Online]. Available: https://www.vice.com/en/article/xgy8qz/woman-allegedly-used-apple-airtag-to-track-and-kill-her-boyfriend

[7] "Apple Tracker Detect," February 2022, accessed 10-06-2025. [Online]. Available: https://play.google.com/store/apps/details?id=com.apple.trackerdetect&hl=de

[8] C. Silva, "Google and Apple are closer to making AirTags stalker free," Dec. 2023, [Accessed 16-06-2025]. [Online]. Available: https://mashable.com/article/protection-privacy-airtag-google-apple

[9] A. Heinrich, N. Bittner, and M. Hollick, "Airguard-protecting android users from stalking attacks by apple find my devices," in *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2022, pp. 26–38.

[10] K. O. Müller, L. Bienz, B. Rodrigues, C. Feng, and B. Stiller, "Homescout: Anti-stalking mobile app for bluetooth low energy devices," in *2023 IEEE 48th Conference on Local Computer Networks (LCN)*. IEEE, 2023, pp. 1–9.

[11] "IoT Connected Devices by Technology 2030," https://www.statista.com/statistics/1194688/iot-connected-devices-communications-technology/, [Accessed 19-03-2025].

[12] A. Catley, "Apple AirTag Reverse Engineering," February 2022, [Accessed 10-06-2025]. [Online]. Available: https://adamcatley.com/AirTag.html

[13] T. Yu, J. Henderson, A. Tiu, and T. Haines, "Privacy Analysis of Samsung's Crowd-Sourced Bluetooth Location Tracking System," October 2022, [Accessed 15-06-2025]. [Online]. Available: https://arxiv.org/abs/2210.14702

[14] G. M. Mendoza-Silva, M. Matey-Sanz, J. Torres-Sospedra, and J. Huerta, "BLE RSS Measurements Dataset for Research on Accurate Indoor Positioning," *Data*, vol. 4, no. 1, p. 12, 2019.

[15] E. Sansano-Sansano, F. J. Aranda, R. Montoliu, and F. J. Álvarez, "BLE-GSpeed: A new BLE-based Dataset to Estimate User Gait Speed," *Data*, vol. 5, no. 4, p. 115, 2020.

[16] M. Salimibeni, Z. Hajiakhondi-Meybodi, P. Malekzadeh, M. Atashi, K. N. Plataniotis, and A. Mohammadi, "IoT-TD: IoT Dataset for Multiple Model BLE-based Indoor Localization/Tracking," in *2020 28th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 1697–1701.

[17] A. N. Nor Hisham, Y. H. Ng, C. K. Tan, and D. Chieng, "Hybrid wi-fi and ble fingerprinting dataset for multi-floor indoor environments with different layouts," *Data*, vol. 7, no. 11, p. 156, 2022.

[18] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Z. Yang, "Automatic Device Classification from Network Traffic Streams of Internet of Things," *Conference on Local Computer Networks (LCN)*, vol. 43, 2018, accessed 20-07-2024. [Online]. Available: https://doi.org/10.1109/LCN.2018.8638232

[19] S. Kashani, S. Sherazi, A. Khokhar, S. W. Kim, and F. Nait-Abdesselam, "Bluetooth Low Energy (BLE) RF Dataset for Machine Learning in WBANs," in *2024 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2024, pp. 1–6.

[20] A. Jagannath, Z. Kane, and J. Jagannath, "Bluetooth and WiFi Dataset for Real World RF Fingerprinting of Commercial Devices," 2023. [Online]. Available: https://arxiv.org/abs/2303.13538

[21] H. Ibrahim, R. Asim, M. Varvello, and Y. Zaki, "I Tag, You Tag, Everybody Tags!" in *Proceedings of the 2023 ACM on Internet Measurement Conference*, ser. IMC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 561–568. [Online]. Available: https://doi.org/10.1145/3618257.3624834

[22] H. D. Jang, H. Ibrahim, R. Asim, M. Varvello, and Y. Zaki, "A tale of three location trackers: Airtag, smarttag, and tile," 2025. [Online]. Available: https://arxiv.org/abs/2501.17452

[23] Nordic Semiconductor, "Bluetooth LE Advertising Packet," https://academy.nordicsemi.com/courses/bluetooth-low-energy-fundamentals/lessons/lesson-2-bluetooth-le-advertising/topic/advertisement-packet/, 2024, accessed 14-06-2025.

[24] J. Wong, "Advertising Payload Format on BLE," August 2019, accessed 13-06-2025. [Online]. Available: https://jimmywongiot.com/2019/08/13/advertising-payload-format-on-ble/

[25] E. Kay, "5 ways to use the new Find My Device on Android," https://blog.google/products/android/android-find-my-device/, 2024, [Accessed 15-06-2025].

[26] Nordic Semiconductor ASA, "nRF Sniffer for Bluetooth LE v4.1.0," Tech. Rep., 2024, [Accessed 15-06-2025]. [Online]. Available: https://infocenter.nordicsemi.com/pdf/nRF_Sniffer_BLE_UG_v4.1.0.pdf

[27] FuriousMAC Research Group, "An apple continuity protocol reverse engineering project," September 2023, [Accessed 15-06-2025]. [Online]. Available: https://github.com/furiousMAC/continuity