Tracking Trackers: ML-based Personal Tracker Detection in Crowded IoT Environments

Katharina O.E. Müller[1], Stefan Saxer[1], Daria Schumm[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|stiller]@ifi.uzh.ch, stefanrichard.saxer@uzh.ch, bruno.rodrigues@unisg.ch

***Abstract*—Bluetooth Low Energy (BLE) personal trackers offer convenient item-finding services, but they also pose severe privacy risks when misused for stalking in everyday life and crowded environments, such as train stations. Robust, vendor-agnostic detection mechanisms are thus paramount. Our preliminary work demonstrated binary Machine Learning (ML) approaches that distinguish trackers from non-trackers, but these methods fall short in heterogeneous, high-density environments.**

This paper introduces a categorical ML-based classifier that identifies popular BLE trackers and their operational states. Using a 200-hour dataset, several ML models were trained and compared, achieving over 99% accuracy. The final model was validated in a real-world, high-traffic central station, demonstrating scalability and robustness in urban scenarios. Our findings confirm that accurate, vendor-agnostic classification of BLE trackers in heterogeneous IoT environments is feasible.

***Index Terms*—Internet of Things (IoT), Privacy and Trust, Bluetooth Low Energy (BLE), Personal Tracker, Crowdsourced Finding Network (CFN), Device Classification**

I. Introduction

Everyday environments increasingly consist of dense constellations of interconnected devices, from smartphones to Bluetooth Low Energy (BLE)-based personal trackers, such as Apple's AirTag. With their surge in popularity [1] and the crowdsourced finding network (CFN) backbone enabeling convinient worldwide tracking, their pervasivness has amplified real-world risks from stalking [2] to murder [3]. Illustrating that privacy threats from this technology are not hypothetical but immediate, tangible, and potentially life-threatening, and can be exploited on a worldwide scale [4].

These devices exhibit a dual nature: they must be discoverable to enable user alerts, yet the same discoverability can be exploited for malicious tracking. Current countermeasures, such as Apple's brand-specific detection app [5], provide only partial protection. They fail to generalize across devices from other manufacturers or account for non-tracker BLE devices, such as headphones, that can be repurposed in similar ways [4]. This lack of general and scalable detection leaves users vulnerable in the heterogeneous, high-density IoT ecosystems.

Prior work has broadly mapped privacy risks in IoT [6] and analyzed BLE protocols and behaviors in depth [7], [1]. Smartphone-based defenses [8], [9], [10] enhance the device's warning capabilities but remain platform-specific or limited in coverage, relying on reverse engineering and hardcoded rules. Thus, efficient for known devices but brittle against new trackers on the market or evolving communication patterns, highlighting the need for a broader adaptive approach.

This paper presents a new Proof-of-Concept (PoC) approach based on Machine Learning (ML) leveraging the communication patterns and information of BLE advertisement packets (link-layer) for tracker detection. The proposed PoC goes beyond binary classification to categorically identify trackers and their operational states, critical in stalking cases. This methodology allows distinguishing known devices and lays the foundation for identifying previously unseen trackers in real-world scenarios and expansion of the approach to any BLE-enabled device. ML was selected over Deep Learning for its lightweight, resource-efficient suitability in constrained IoT deployments; energy efficiency is left for future work. By processing BLE advertisement streams at scale, the proposed models advance vendor-agnostic detection accuracy and demonstrate the potential for scalable, adaptable, real-time detection. The key contributions of this work are:

- Novel approach for personal tracker identification and detection in dense and crowded environments;
- Comparison of three ML models for tracker and state classification, trained on a 200-hour BLE dataset [11], achieving over 99% accuracy;
- Inference validation in a dense, real-world environment, confirming robustness and reliability, achieving a weighted average confidence level above 83%.

This work acknowledges a dual-use dilemma, from an ethical perspective: the same technology that enables protective detection of trackers could also inform adversaries. Ignoring this risk would leave it unaddressed; therefore, this work explicitly considers it to inform balanced countermeasures for personal trackers. All data consisted solely of publicly broadcast anonymous BLE advertisement packets with fully encrypted payloads and no personally identifiable information. In line with the University of Zurich's ethical guidelines [12], no explicit approval was required. Nevertheless, this paper underscores that even anonymous network metadata, when analyzed at scale, can present significant privacy risks.

This paper is organized as follows: Section II outlines BLE fundamentals, Section III reviews related work, Section IV describes the dataset, Section V present the approach and modeling, Section VI and Section VII report results and real-world validation, and Section VIII concludes.

II. Background

BLE defines two roles: *Centrals* (*e.g.*, smartphones, laptops) that scan for signals, and *Peripherals* (*e.g.*, personal trackers) that broadcast advertisement packets to initiate connections [13]. These packets, which contain only metadata and encrypted payloads, can be analyzed for classification.

A. BLE Devices, Personal Trackers, States, and Stalking

Personal trackers advertise continuously, varying transmission frequency by state, which makes them consistently detectable through packet sniffing. This paper's devices operate in 2-4 states with different privacy implications. For example, AirPods switch between a *nearby* state with infrequent packets and a *lost* state with rapid broadcasts. Find My trackers (AirTag, SkyTag, Chipolo) add an *unpaired* state, while Tile and SmartTag include a user-triggered *searching* state. Non-tracker devices are instead categorized as *online*, combining BLE with other connectivity such as cellular, or *offline*, relying solely on BLE advertisements.

Accurate device state identification is essential for detecting personal trackers abused for stalking. In such cases, *e.g.*, AirTags are typically in the lost state, where the device is paired to an owner's Apple ID but out of BLE range of the owner's device. Although intended for item recovery, this state allows location tracking via the Find My CFN and can be exploited to covertly track a victim's belongings worldwide, especially when the attacker cannot accompany the victim. Transmission intervals vary: in *unpaired* or *searching* state, trackers advertise every 100 ms; in *nearby* state, intervals extend to around 1 s to conserve energy; and in *lost* state, packets are broadcast every 100–250 ms to maximize visibility [11]. Although values differ by manufacturer, frequent advertising is typical of states that require device discovery.

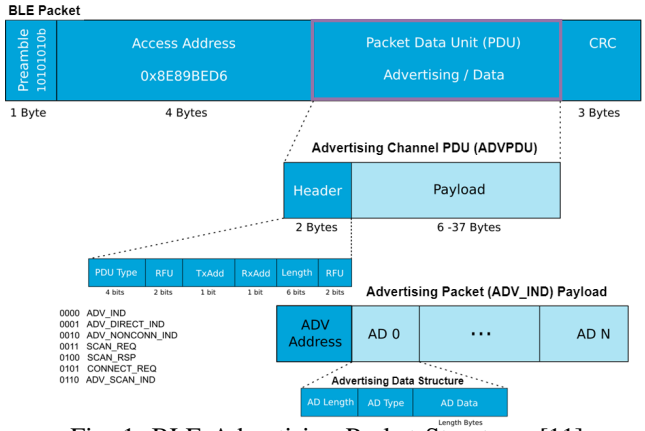


Fig. 1: BLE Advertising Packet Structure: [11]

B. BLE Advertisement Packet Structure

At the link layer, BLE advertisement packets follow a fixed structure, illustrated in Fig. 1. The central element is the *Advertising Protocol Data Unit* (ADV_PDU), consisting of a header that encodes type and length, followed by a variable-length payload. The payload contains the device address and one or more *advertisement data types*, whose presence, length, and order may vary but cannot exceed 37 bytes. These features are highly relevant for classification tasks.

For example, *manufacturer-specific data* includes a company identifier (*e.g.*, 0x004C for Apple) along with arbitrary manufacturer-defined bytes constrained by PDU size. Service UUIDs uniquely identify a device or service and appear either as a simple Service UUID or as a Service Data UUID variant, which combines the identifier with associated data (16-bit, 32-bit, or 128-bit). Additional fields may include Flags or an 8-bit TX Power Level. By analyzing the presence, structure, and distribution of such elements, devices can be effectively distinguished and categorized.

III. Related Work

Research on IoT device classification has explored behavioral fingerprinting [14], clustering [15], deep learning for unseen devices [16], and deep packet inspection [17]. These approaches work in controlled or home-scale IoT settings with 1–18 devices and rely on Ethernet or Wi-Fi traffic, with well-defined packet structures. They do not generalize to pervasive environments with thousands of heterogeneous devices, where short, encrypted BLE advertisements dominate.

BLE-specific work has examined advertising packets [18], protocol reverse engineering [19], and MAC randomization vulnerabilities [20]. Apps such as AirGuard [8] and Homescout [9] show that detection is possible in practice, but their hardcoded, vendor-specific rules limit adaptability and scalability. Similarly, BLE datasets used in ML focus mainly on physical-layer signal features or RSSI values [21], [22], [23], which support localization but not link-layer device classification. Even commercial tracker studies [24], [25] provide only aggregate counts or GPS-level insights, not the detailed advertisement metadata required for classification.

As a result, existing solutions either operate at the wrong protocol level (IP/TCP flows), the wrong layer (physical signal characteristics), or reduce BLE to simple RSSI-based localization. None provides a vendor-agnostic classification framework that can handle dense, real-world deployments with thousands of known and unknown devices. Although IoT privacy risks are well recognized [6], practical countermeasures remain limited. This work addresses that gap by presenting an operational, ML-based prototype that classifies BLE trackers from advertisement metadata with high accuracy, validated against a real-world train station scenario.

IV. Dataset

Two complementary datasets were created [11] to support the classification and evaluation tasks in this paper. Through advertisment channel packet sniffing with an nRF52840 development kit, acquired in two steps: (1) isolated and controlled basment environment to allow the distinction of specific packet information,*cf.* Fig. 2 (left), and (2) a rush hour central train station for data with as much interference and real-world heterogeneity, unknown devices, and radio frequency issues as possible, *cf.* Fig. 2 (right).

TABLE I: Feature Usability and Use in BLE Packet Processing

Packet Field	Description	Feature	Use
(00)Time	UTC time in ms when captured in Wireshark. Essential for modeling packet rates and aggregating packets, but not a direct feature.	✗	Aggregating packets for ML
(0)Source	Source address of the transmitting Bluetooth device. Used for modeling packet rates and inference, but not as a feature.	✗	Identifying and labeling devices
(1)Destination	Destination address of the packet. Converted to a one-hot encoded column where 1 indicates a broadcast (FF:FF:FF:FF:FF:FF) and 0 represents any other address.	✓	Identifying broadcast packets
(2)Protocol Type	Indicates the protocol used for transmission. No further preprocessing required.	✓	Differentiating communication protocols
(3)Channel	Advertising channel used for transmission. No further preprocessing required.	✓	Identifying transmission channels
(4)Packet Length	Length in bytes on the nRF 52840 DK layer, always 26 bytes longer than the header length. Not related to the BLE link layer packet length.	✓	Understanding data transmission size
(5)Header Length	Length in bytes on the BLE link layer, always 26 bytes shorter than the packet length on the nRF 52840 DK layer.	✓	Identifying BLE packet structure
(6)Advertisement Data Type	List of advertising data types used in the packet, separated by commas. No preprocessing required.	✓	Categorizing advertisement data
(7)Company ID	Company ID found in the manufacturer-specific data. Null values replaced with "None", and common company names simplified (*e.g.*, "Apple, Inc." → "Apple").	✓	Identifying device manufacturers
(8)Manufacturer Specific Data	Raw manufacturer-specific data payload as a string. Null values are replaced with an empty string. The length of the raw data in bits was extracted as a feature.	✓	Extracting unique device information
(9)UUID	UUIDs present in the packet, separated by commas. Null values replaced with "None", and common UUIDs simplified (*e.g.*, "Tile, Inc." → "Tile"). Relevant for one-hot encoding.	✓	Identifying device services
(10)Service Data	Raw service data payload as a string. Null values are replaced with an empty string. The length of the raw service data in bits was extracted as a feature.	✓	Extracting service-related information
(11)PDU	PDU type of the packet. No preprocessing required.	✓	Classifying packet type
(12)Continuity Type (CT)	First byte of Apple's manufacturer-specific data indicating continuity service type (*e.g.*, 0x12 for Find My). Extracted only for Apple devices.	✓	Differentiating Apple tracker services and states
(13)SmartTag Type (ST)	Bits 5–7 of Samsung's Service Data indicating tracker state. Extracted only for Samsung devices.	✓	Classifying SmartTag states
(14)Malformed Packet	Flag indicating whether the packet is malformed, based on extended PDU labels.	✓	Identifying corrupted packets

Fig. 2: Data Acquisition Setups; (left) isolated basement with Faraday cage, (right) open view train station bench

(1) Controlled labeled dataset. To generate reliable ground truth, advertisement packets were captured from a diverse set of BLE devices. This dataset spans over 200 hours of captures, resulting in more than 13 million labeled packets. It covers the following device categories: (i) conventional BLE trackers (*e.g.*, AirTag, SmartTag, Tile, Chipolo), (ii) BLE-enabled devices participating in crowdsourced finding networks (*e.g.*, smartphones, laptops, tablets, earbuds), and (iii) other BLE peripherals (*e.g.*, headphones, speakers, keyboards, mice). Critically, all operational states were recorded (*cf.* II-A). Packets were labeled using a combination of isolation captures and feature-based heuristics.

(2) Unlabeled real-world dataset. To evaluate scalability and inference in realistic conditions, anonymous, unassociated BLE packets were collected at a public train station with extremely high device density and heterogeneity, totalling $\approx 580,000$ packets and split into a shorter and a longer dataset. Unlike the labelled dataset, this collection is intentionally unfiltered and unlabeled, serving as the basis for inference experiments.

Together, these two datasets enable accurate model training on well-defined classes and robust evaluation in uncontrolled IoT environments, demonstrating that the results generalize beyond lab-scale IoT setups.

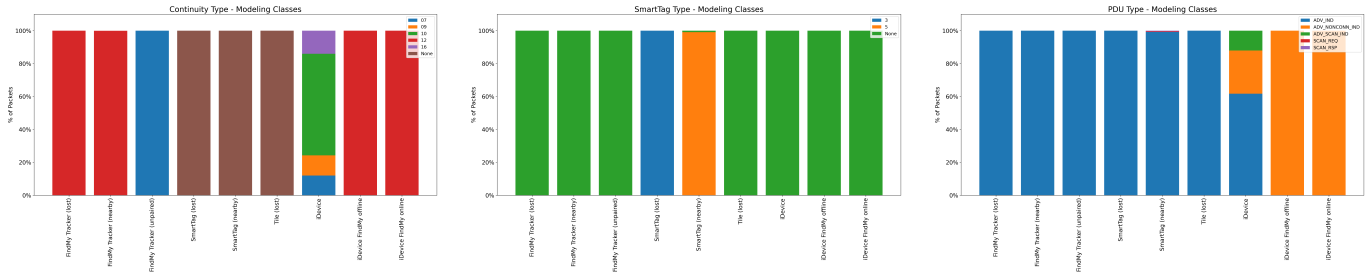
Data Features For ML analysis of BLE advertisement packets, identifying the most relevant features is paramount. Among the 16 available packet fields, 14 were deemed suitable as features (*cf.* Table I). *Time* and *Source* were excluded because they serve only as contextual information for aggregation and device identification.

V. Preprocessing and Modeling Class Analysis

The following classes were defined after a packet analysis based on the dataset and insights from [11]:

TABLE II: Defined Classes

Class (state)	Personal Tracker Packets
FindMy Tracker (lost)	AirTag, Chipolo, and SkyTag
FindMy Tracker (nearby)	AirTag, Chipolo, and SkyTag
FindMy Tracker (unpaired)	AirTags only
SmartTag (lost)	SmartTag
SmartTag (nearby)	SmartTag
Tile (lost)	Includes *nearby*, as Tile does not distinguish
iDevice	Non–Find My Apple device and AirPods
iDevice FindMy (offline)	Find My packets without a public key
iDevice FindMy (online)	Find My packets with a public key
other Device	All remaining packets



(a) Continuity Type - Modeling Classes (b) SmartTag Type - Modeling Classes (c) PDU Type - Modeling Classes

Fig. 3: Simple Features per Modeling Class

A. Data Preprocessing

The data preprocessing pipeline has the following steps:

1) Features: The same 12 features described in Table I were used, with feature (12) used to identify corrupted packets rather than for classification.

2) Conversion to Dummy Variables: Categorical variables were converted: all non-numerical features must be transformed using one-hot encoding, using a custom executor to ensure deterministic selection and ordering. Limiting the number of features to only those useful for device distinction improves training and inference efficiency.

3) Labeling: All packets are labeled; this can be done either by simply labeling all packets with the same label for single device captures or using the ML-based approach [11].

4) States and Continuity Type (CT): For some devices, class labels were extended to include state information, such as the *nearby* state. Since CT labeling is only needed for Find My packets, state labeling is simplified. The final label combines class and state, *e.g.*, "SmartTag (lost)".

5) Drop Labels: Certain class labels, primarily owner devices of trackers in the *nearby* state, were removed from the training data since their packets were unnecessary for modeling and risked introducing noise. All Find My packets from AirPods were also excluded because of their similarity to tracker traffic, which could cause misclassification.

6) Modeling: Datasets are prepared by dropping unnecessary columns (*e.g.*, timestamps) and defining resampling for packet rate modeling. The resampling interval choice balances accuracy and usability: longer intervals improve accuracy, while shorter ones increase practicality.

B. Analysis of Modeling Classes: Separability Indicators

First, the number of packets per modeling class confirmed sufficient data across all classes. To balance the dataset, either undersampling or oversampling was considered. Undersampling, limiting each class to the size of the smallest class ("SmartTag (nearby)" with 24,000 samples), results in a **balanced dataset of $\approx$ 240,000 samples**, suitable for this papers ML models. In contrast, oversampling to match the largest class ("other Device" with 1.8 million samples) would yield an 18-million-sample dataset, which is infeasible due to resource constraints. Feature analysis showed that the average source address interval is sufficiently large for packet rate modeling across all classes. Header length (*i.e.*, BLE packet length) effectively distinguishes between *lost* and *nearby* states of Find My trackers and between *online* and *offline* states of iDevices, as only *offline* and *lost* packets contain the long private key. However, header length alone cannot separate other classes. Similarly, while the SmartTag and Tile tracker have unique UUIDs not shared by other devices, the SmartTag's UUID does not distinguish between its two states and serves only to identify the device.

Several features help distinguish packet classes. CT (*cf.* Fig. 3a) is vital: all Find My packets use type *0x12*, while iDevices show varied types due to their role as a catch-all for non-Find My Apple packets. Non-Apple devices, *e.g.*, SmartTag, carry no CT. A potential ambiguity arises with the "FindMy Tracker unpaired" class sharing type *0x07* with iDevice packets, though these are from AirPods and can be distinguished by their PDU type. The SmartTag type (ST) (*cf.* Fig. 3b) is crucial for differentiating its states, as no other class includes this feature. PDU type (cf. Fig. 3c) is useful to separate Find My packets from iDevices and trackers.

VI. Classifier: Training and Packet Modeling

Three packet classifiers were trained and evaluated. The balanced 240'000-sample dataset was split 75/25 into training and test sets, each maintaining class balance with 18'000 training and 6'000 test samples per class. A min-max scaler, trained on the training set and applied to both sets to prevent data leakage, was saved for later inference. Each model was trained on scaled data, saved for reuse, and evaluated using a confusion matrix and classification report.

A. Simple Neural Network (NN)

A basic Multi-layer Perceptron (MLP) classifier from scikit-learn, with one hidden layer of 100 neurons using ReLU activation, followed by a softmax output layer for class probabilities. The resulting network has roughly 4'000 parameters: *30 features * 100 hidden neurons* for the first fully connected layer plus *100 hidden neurons * 10 classes* for the second fully connected layer; ignoring the bias neurons. For 180'000 training samples, 4'000 parameters provide a favorable data-to-capacity ratio that reduces overfitting risk for a shallow MLP. Even though the training was done on the

CPU only, the training time did not exceed a few seconds, as early stopping was used to prevent overfitting.

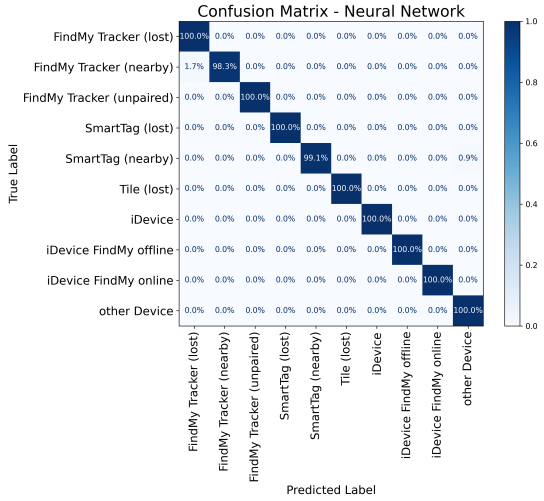


Fig. 4: Confusion Matrix - NN

The confusion matrix in Fig. 4 shows near-perfect accuracy across most classes, with the main exceptions being the *nearby* states of Find My trackers (98.3%) and the SmartTag (99.1%). For Find My trackers, misclassifications likely result from occasional state switching to *lost*.

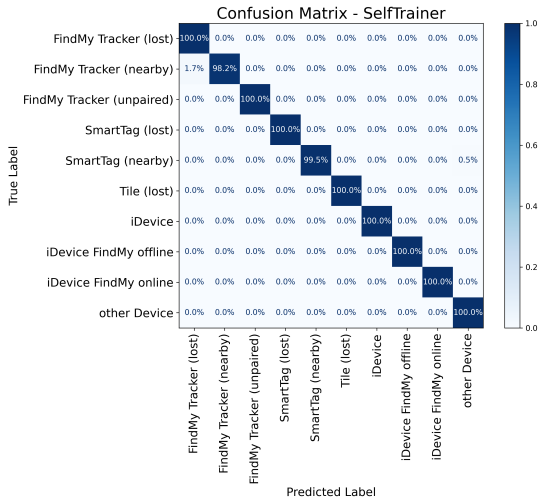


Fig. 5: Confusion Matrix - NN-SelfTrainer

B. NN Extended: Self Training Classifier

To improve generalization and real-world robustness, the NN was extended with a self-training classifier for semi-supervised learning. Labeled data is often scarce or noisy, so incorporating unlabeled data helps improve model reliability in uncontrolled environments. Extending the NN with a self-training classifier for the semi-supervised method had the singular purpose of increasing robustness rather than performance; most effective when the additional unlabeled training data is similar to the data seen during inference. Thus, the shorter dataset captured at the train station was utilized. The longer dataset was soley used for inference in Section VII. As expected, Fig. 5 shows performance comparable to the base NN on test data.

C. Final Model: Decision Tree (DT)

TABLE III: Feature Importance and Classification Impact

Feature	RI (%)	Classification Impact
PDU ADV_IND	13.5	Distinguishes FindMy trackers from iDevices
Length MS Data	10.5	Helps separate FindMy states (lost vs nearby)
UUID Tile	10.5	Unique identifier for Tile trackers
ST 3	10.5	Critical for SmartTag state distinction
CT 12	10.5	Differentiates FindMy trackers vs iDevices
COMP Apple	10.5	Identifies Apple devices but prone to confusion with iDevices
UUID Samsung	10.0	Unique identifier for Samsung SmartTags
Length Header	10.0	Separates FindMy lost vs nearby and iDevice states
Length Packet	9.0	Supports state classification across FindMy and iDevices
Other Features	$\leq$0.1	Minimal classification impact

The final model was chosen for its speed, interpretability, and suitability given the distinct features of most classes. Using scikit-learn's default settings with no hyperparameter tuning, training completes in seconds. As shown by Fig. 6, it achieves accuracy comparable to the NN approaches, validating its effectiveness. The DT's interpretability enables analysis of relative feature importance (RI), see Table III, confirming that key features identified earlier, *e.g.*, the ST, are most influential in classification, compared to the less relevant, *e.g.*, channel number. Indicating the model effectively learned to classify based on unique device characteristics.

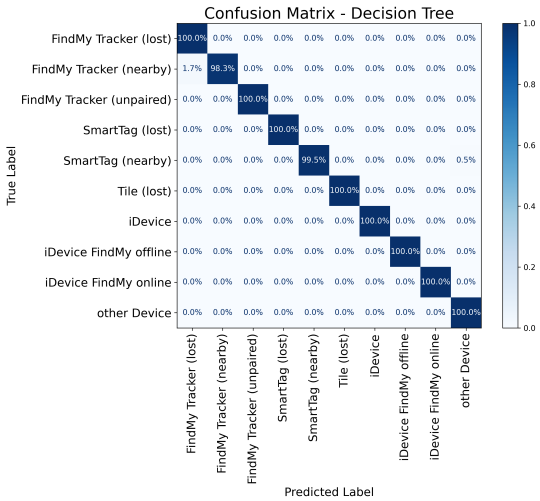


Fig. 6: Confusion Matrix - DT

To evaluate the impact of training data size, 7 DTs were trained on subsets from 1/64 to the full dataset and tested on a fixed set. Accuracy remained consistently at 99.9%, indicating that large data volumes, such as 30 million packets, are unnecessary for accurate BLE device classification.

D. Comparison of These Models

While all models achieved similarly high test accuracy, this result is largely attributable to the highly structured nature of BLE advertisement packets, which expose device- and vendor-specific features that are stable across transmissions. Consequently, near-perfect accuracy on balanced test data is expected and does not necessarily indicate overfitting.

However, in practice, tracker detection must operate in real time under high device density, heterogeneous traffic patterns, and with partial and noisy traffic. Under these conditions, robustness during inference becomes the primary differentiator.

Among these, NN-based models offer an advantage in pervasive settings due to their probabilistic outputs, enabling softmax-based confidence thresholding to suppress uncertain predictions and reduce false positives. In contrast, DTs, while lightweight and data-efficient, lack confidence-aware filtering and are therefore more susceptible to noise and mislabeled data during inference. Among the NN variants, the semi-supervised self-training model is expected to be more robust and is thus preferred.

VII. Inference: Central Station

Inference is performed using the NN self-training classifier on the longer train station dataset. As ground-truth labels are unavailable, evaluation relies on model confidence rather than accuracy metrics.

A. Preprocessing and Device Selection

Inference preprocessing mirrors the modeling pipeline but operates on unlabeled data. Packets are scaled using the pre-trained scaler and classified to obtain softmax probability vectors. To improve reliability, predictions from the same source address are aggregated by combining class frequencies and confidence values. For each source–class pair, the average softmax confidence and a weighted score are computed (*cf.* Table IV), and the class with the highest score is selected.

TABLE IV: Prediction Summary by Source and Class

Source Address	Predicted Label	Average Softmax Confidence	Score
2E-B0-D0-63-C2-26	other Device	90%	75%
2E-B0-D0-63-C2-26	iDevice	60%	25%
C9-2D-8B-7F-9B-A6	FindMy Tracker	100%	50%
C9-2D-8B-7F-9B-A6	SmartTag	30%	30%
C9-2D-8B-7F-9B-A6	Tile	40%	20%

To reduce false positives, a softmax confidence threshold is applied: source addresses with an average confidence below 98% are discarded. This cutoff was chosen empirically as the best trade-off between precision and coverage. In practice, approximately 20% of sources (mostly labeled as "other Device") are filtered out, retaining 80% for final analysis.

B. Confidence per Device Type and State

Figure 7 shows distinct confidence decline patterns across device classes. "SmartTag (lost)" maintains 100% confidence throughout, which may initially suggest overfitting. However, this behavior is consistent with earlier observations that SmartTag lost-state packets expose highly specific and stable features. Manual inspection of the underlying packets confirmed that these features are consistently present and correctly classified, indicating strong class separability rather than memorization. Other classes, including "Tile (lost)," "iDevice Find My (offline/online)," "FindMy Tracker (nearby)," and "iDevice," remain stable at 100% confidence until the final 1–5% of packets, where confidence drops sharply. In contrast, "FindMy Tracker (unpaired/lost)" and "other Device" exhibit multi-step confidence declines. "SmartTag (nearby)" shows the most pronounced degradation, indicating frequent misclassification.

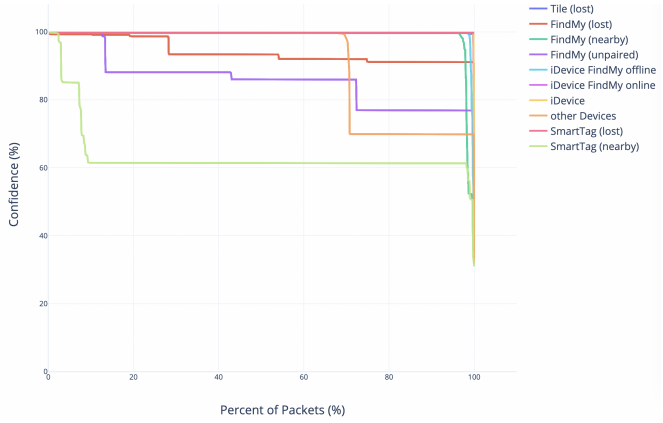


Fig. 7: Classification Confidence across Number of Packages

Weighted average confidence levels are shown in Table V. Nine of ten categories exceed 83% confidence, with six above 91%, suggesting reliable classification overall and highlighting the need for additional training data for "SmartTag (nearby)" exhibits substantially lower confidence (63.82%).

TABLE V: Weighted Average Confidence by Device/State

Device (state)	Confidence (%)
SmartTag (lost)	100.00
Tile (lost)	99.71
iDevice FindMy offline	94.92
iDevice FindMy online	93.22
iDevice	91.65
FindMy (nearby)	91.62
FindMy (lost)	89.95
Other Devices	87.13
FindMy (unpaired)	83.18
SmartTag (nearby)	63.82

C. Key Takeaways

Inference on the unlabeled central-station dataset shows that packet-based ML classification remains stable in dense, heterogeneous environments when confidence-aware filtering and source-level aggregation are applied. Most device and state classes exhibit high and stable confidence, with trackers in the *lost* state showing particularly strong separability in real-world traffic. Although "SmartTag (lost)" maintains

uniformly high confidence, manual packet inspection confirms this reflects stable packet features rather than overfitting. Confidence degradation is confined to the SmartTag *nearby* state, showing rapid decline and the lowest weighted confidence, while other classes remain stable until late-stage packet observations. The results indicate which device states generalize reliably and where classification uncertainty remains.

VIII. Summary and Future Work

This work demonstrates that BLE trackers can be reliably distinguished using lightweight, packet-based ML models, particularly in the *lost* state, where detection is most relevant for anti-stalking applications. Across three classifiers (DT, NN, and semi-supervised NN), test accuracy consistently exceeded 99%, while real-world inference in a dense urban environment confirmed robustness, with over 83% weighted confidence in 9 of 10 classes. These results show that data-efficient models are practical for deployment on resource-constrained devices and that BLE advertisement metadata alone poses tangible privacy risks, underscoring the need for protective anti-stalking mechanisms.

The primary limitation lies in reduced robustness for the SmartTag *nearby* state, driven by data imbalance, imperfect automatic labeling, and limited exposure to rare or transitional packet patterns. While softmax confidence thresholding effectively suppresses false positives, it restricts inference coverage to approximately 80% of observed devices. Residual mislabeling, despite preprocessing and filtering, likely contributes to the remaining misclassifications.

Future work focuses on data diversity rather than increasing model complexity. While additional real-world data would help, a more scalable, privacy-preserving solution is to develop a framework for synthetic BLE packet generation, enabling controlled modeling of device states and edge cases without relying on sensitive real-world captures. Further evaluation of energy efficiency and integration into real-time pervasive services will improve deployment readiness while ensuring ethical and responsible use.

References

[1] G. Celosia and M. Cunche, "Saving Private Addresses: An Analysis of Privacy Issues in the Bluetooth-low-energy Advertising Mechanism," in *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 *20th Workshop on Workshop on Privacy in the Electronic Society*, 2021, pp. 181–186.

[3] M. Gault, "Woman Allegedly Used Apple AirTag to Track and Kill Her Boyfriend," Jun. 2022, [Accessed: 15.02.25]. [Online]. Available: https://www.vice.com/en/article/xgy8qz/woman-allegedly-used-apple-airtag-to-track-and-kill-her-boyfriend

[4] "Whisperpair: Hijacking bluetooth accessories using google fast pair," https://whisperpair.eu/, 2026, accessed: 2026-02-09. [Online]. Available: https://whisperpair.eu/

[5] "Apple Tracker Detect," February 2022, [Accessed: 15.02.25]. [Online]. Available: https://play.google.com/store/apps/details?id=com.apple.trackerdetect&hl=de

[6] P. Sun, S. Shen, Y. Wan, Z. Wu, Z. Fang, and X.-Z. Gao, "A Survey of IoT Privacy Security: Architecture, Technology, Challenges, and Trends," *IEEE Internet of Things Journal*, vol. 11, no. 21, pp. 34 567–34 591, 2024.

[7] A. Heinrich, M. Stute, T. Kornhuber, and M. Hollick, "Who can find my devices? security and privacy of apple's crowd-sourced bluetooth location tracking system," *Privacy Enhancing Technologies Symposium*, vol. 3, pp. 227–245, 2021.

[8] 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.

[9] K. O. Müller, L. Bienz, B. Rodrigues, C. Feng, and B. Stiller, "Homescout: Anti-stalking Mobile App for Bluetooth Low Energy Devices," in *48th Conference on Local Computer Networks (LCN)*. IEEE, 2023, pp. 1–9.

[10] J. Briggs and C. Geeng, "Ble-doubt: Smartphone-based detection of malicious bluetooth trackers," in *2022 IEEE Security and Privacy Workshops (SPW)*, 2022, pp. 208–214.

[11] K. O. E. Müller, S. Saxer, D. Schumm, W. Niu, B. Rodrigues, and B. Stiller, "Airtagged: A dataset and processing framework for heterogeneous high-density iot environments," in *Proceedings International Conference on Network and Service Management*. Bologna, Italy: IFIP/IEEE, Oct 2025, pp. 1–7.

[12] U. of Zurich UZH, "UZH Policy on the Ethical Review of Research Projects Involving Human Subjects (UZH Ethics Policy) ," [Online]. Available:https://www.research.uzh.ch/en/procedures/research-on-humans.html, 2025, Last visit January 10, 2026.

[13] Bluetooth SIG, "Bluetooth core specification: Low energy controller, link layer," https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-60/out/en/low-energy-controller/link-layer-specification.html, 2023, accessed: 2025-09-12.

[14] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Behavioral Fingerprinting of Iot Devices," in *2018 Workshop on Attacks and Solutions in Hardware Security*, 2018, pp. 41–50.

[15] F. Sawadogo, J. Violos, A. Hameed, and A. Leivadeas, "An Unsupervised Machine Learning Approach for Iot Device Categorization," in *2022 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*. IEEE, 2022, pp. 25–30.

[16] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Z. Yang, "Automatic Device Classification from Network Traffic Streams of Internet of Things," in *43rd conference on Local Computer Networks (LCN)*. IEEE, 2018, pp. 1–9.

[17] P. Khandait, N. Hubballi, and B. Mazumdar, "IoTHunter: IoT Network Traffic Classification using Device Specific Keywords," *IET Networks*, vol. 10, no. 2, pp. 59–75, 2021.

[18] V. Poirot, O. Harms, H. Martens, and O. Landsiedel, "BlueSeer: AI-driven Environment Detection via BLE Scans," in *59th ACM/IEEE Design Automation Conference*, 2022, pp. 871–876.

[19] J. Martin, D. Alpuche, K. Bodeman, L. Brown, E. Fenske, L. Foppe, T. Mayberry, E. C. Rye, B. Sipes, and S. Teplov, "Handoff all your Privacy: A Review of Apple's Bluetooth Low Energy Continuity Protocol," *arXiv preprint arXiv:1904.10600*, 2019.

[20] J. K. Becker, D. Li, and D. Starobinski, "Tracking Anonymized Bluetooth Devices." *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, pp. 50–65, 2019.

[21] 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.

[22] 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

[23] 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.

[24] 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

[25] 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