

Towards Fast Detection of Suspicious Bluetooth Trackers using Anomaly Detection

Orobosa Ekhatior
Portland State University
Portland, Oregon, USA
orobosa@pdx.edu

Dylan Conklin
Portland State University
Portland, Oregon, USA
dconklin@pdx.edu

Primal Pappachan
Portland State University
Portland, Oregon, USA
primal@pdx.edu

Roberto Yus
University of Maryland,
Baltimore County
Baltimore, Maryland, USA
ryus@umbc.edu

Abstract

Detecting malicious Bluetooth Low Energy (BLE) trackers remains challenging because existing approaches rely on fixed time and distance thresholds that are brittle across environments. These heuristics produce false positives in dense environments and require long observation windows before flagging a device. To address these limitations, we present BL(u)E CRAB, a cross-platform (iOS/Android) mobile system that represents nearby devices using three risk factors derived from BLE scan data. Our detection model adapts Clustering-Based Local Outlier Factor (CBLOF) to BLE tracker detection and adds a gap-thresholding mechanism to separate high-scoring outliers from the benign majority. Across micro-benchmarks and end-to-end case studies, CBLOF reduces false positives up to 77% and false negatives up to 20% compared to the state of the art. In our case studies, suspicious trackers are typically detected within 5 minutes of scanning, improving practical usability for real-world deployment.

CCS Concepts

• **Security and privacy** → **Intrusion/anomaly detection and malware mitigation.**

Keywords

Bluetooth Low Energy, Anomaly Detection, BLE Trackers

ACM Reference Format:

Orobosa Ekhatior, Dylan Conklin, Primal Pappachan, and Roberto Yus. 2026. Towards Fast Detection of Suspicious Bluetooth Trackers using Anomaly Detection. In *Proceedings of the 19th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '26)*, June 30–July 03, 2026, Saarbrücken, Germany. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3765613.3811690>

1 Introduction

In recent years, many different Bluetooth Low Energy (BLE) trackers have hit the market. Although these trackers are inexpensive and offer benefits such as reduced energy consumption and reliable wireless connections, they are also susceptible to malicious use, such as unauthorized tracking and stalking [7]. Previous works have sought to protect people against malicious trackers. Apple’s approach to tackling this issue, Find My network [8], is efficient in

detecting when an unknown Find My tracker-compatible device is following a user. Despite its advantages, its functionality is limited to BLE trackers that use the Find My network. Vendor-agnostic frameworks such as AirGuard [6] and BLE-Doubt [2] provide compatibility across a wider array of heterogeneous BLE trackers. However, they are limited in their effectiveness in dynamic scenarios due to their reliance on static heuristic thresholds based on time (observed for the last 10 minutes) and distance thresholds (e.g., at least 300 meters of co-movement). These heuristics or static thresholds are brittle and do not adapt well to different environments (e.g., driving, public transit), where many benign devices may transiently appear along with the user.

To address these challenges, we introduce BL(u)E CRAB, a cross-platform mobile application that detects malicious BLE trackers within a few minutes. Our approach leverages anomaly detection techniques to identify suspicious devices as outliers. Specifically, we use clustering-based local outlier factor (CBLOF) [5], which considers different risk factors derived from BLE scan data and identifies anomalies based on the intuition that malicious devices exhibit behavior different from the majority. To improve robustness, we use a gap-based thresholding mechanism to flag suspicious devices across different environments. To the best of our knowledge, clustering-based anomaly modeling has not been previously explored for BLE tracker detection. Our approach is motivated by the observation that suspicious trackers are statistical outliers in a feature space where most devices are benign.

Our key contributions are: (1) Development of BL(u)E CRAB, a cross-platform (iOS and Android) scanner that computes privacy-preserving risk factors from BLE advertisements; (2) Implementation of an unsupervised CBLOF-based detector with a gap-based thresholding mechanism that avoids fixed time/distance heuristics and avoids assuming a fixed “contamination” rate, (a parameter that specifies the expected percentage of anomalies in the dataset); and (3) A comprehensive evaluation, including end-to-end case studies in real-world settings and micro-benchmarks across both existing datasets and newly collected real-world datasets. Our results demonstrate reductions in false positives (77%) and false negatives (20%), and achieve a higher F1 score (0.89) than the state of the art.

2 Overview

In this section, we formalize the problem of identifying suspicious devices and describe the threat model considered in our work.

Problem Statement. A smartphone device runs BL(u)E CRAB and scans for nearby Bluetooth devices. Let $\mathbb{D} = \{d_1, \dots, d_n\}$ be the set of all devices scanned by the smartphone device. For each scanned device $d \in \mathbb{D}$, we define a sequence of observations: $O_d = \{(l_1, t_1, r_1),$



This work is licensed under a Creative Commons Attribution 4.0 International License. *WiSec '26, Saarbrücken, Germany*

© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2201-1/2026/06
<https://doi.org/10.1145/3765613.3811690>

$(l_2, t_2, r_2), \dots, (l_k, t_k, r_k)\}$, where $l_i = (lat_i, long_i)$ represents the latitude and longitude of scanned device d at time t_i and r_i is the Received Signal Strength Indicator (RSSI) of d at that moment. Note that the scanner does not know a device's true location; we use the phone's GPS location at scan time as a proxy for device location. The observation sequence O_d from a device d is mapped into a multi-dimensional feature vector $x_d \in \mathbb{R}^m$ by a mapping function $f: x_d = f(O_d)$. In our work, $m = 3$ and x_d is composed of three features: $x_d = (Time_d, Dist_d, RSSI_d)$ (described in Section 4).

Our goal is to learn an anomaly detection function $A(x_d)$ such that it partitions the set of all scanned devices $\mathbb{D}$ into two disjoint sets: suspicious/anomalous (S) and non-suspicious ($\mathbb{D} \setminus S$). As the majority of the scanned devices in the environment are assumed to be non-suspicious $|S| \ll |\mathbb{D}|$, their feature distribution will be dominant in the environment. Thus, the feature vector distribution of suspicious devices will significantly deviate from the dominant distribution. Given a set of scanned devices and their sequence of observations, we aim to identify the subset $S \subseteq \mathbb{D}$ such that: $S = \{d \in \mathbb{D} | A(x_d) \geq \delta\}$ where δ is the decision threshold (also called `gap_threshold`) and is described in more detail in Section 4.

Threat Model. We call the malicious user who is deploying the Bluetooth Low Energy (BLE) device to track a user without their consent as the *adversary*. We assume that this adversary tracks the user using only commercially available off-the-shelf BLE trackers (e.g., AirTag, Tile, and SmartTag). Previous studies have shown that these trackers are frequently used for malicious purposes, such as stalking a person or mode of transportation [6]. The tracker device utilizes an object location network (OLN) that utilizes the location of the victim's smartphone to report as its own location. While we trust the OLN provider (e.g., Apple, Google) and believe that they do not actively collude with the adversary, the adversary utilizes the provider's legitimate infrastructure for malicious ends. We assume that our adversary hasn't modified the hardware or software of the tracker device and utilizes the manufacturer-specific application to track its location. The adversary has physical access to the tracker only during the initial setup and placement phase. Post-placement, the adversary has remote access to location updates via the companion app but cannot communicate directly with the tracker. In this work, we do not consider location tracking via a network of permanently placed high-powered Bluetooth beacons that may be able to re-identify the users in their vicinity. Additionally, non-BLE-based devices, such as GPS-based trackers, are not considered in our threat model.

3 Related Work

Apple's Find My network is a system designed to help users locate lost items. While this system is beneficial, the release of AirTag, an inexpensive tracker supported by this nearly ubiquitous network, has introduced significant risks when misused for stalking. Although Apple provides a stalking detection feature within its ecosystem, billions of Android users remain vulnerable. In 2021, Apple released the Android app, "Tracker Detect" [10], to help Android users scan for nearby malicious trackers. However, practical evaluations and user reports suggest that the application provides limited real-world protection due to inconsistent detection performance

and a lack of reliability as a standalone defense mechanism [1]. Consequently, several studies have proposed third-party Android-based apps to mitigate stalking risks associated with Bluetooth trackers.

AirGuard proposed in [6] focuses on mitigating abuse by Apple tracking devices by reverse engineering Apple's automatic tracking system. This approach employs strict detection criteria to minimize false positives, requiring a scanned device to be observed for more than the predefined minimum duration to travel a minimum distance with the user, and to be detected at least three times before being flagged. Despite these strict criteria, AirGuard may flag devices belonging to nearby individuals in shared environments that temporarily move with the user as suspicious, even though there is no malicious intent. **BLE-DOUBT** [2] utilizes a trajectory-based classifier in their paper as its detection mechanism. This method employs time and distance thresholds in order to partition a scanned device's trajectory into sub-trajectories called ϵ -components. While this classifier outperformed prior approaches, achieving a higher F1 score, its effectiveness depends on collecting BLE scan data from a suspicious device over a long period of time. Under default settings, a device has to be scanned over a period of 10 minutes and a distance of 300 meters before BLE-DOUBT flags them as suspicious.

These approaches highlight the limitations of relying solely on trajectory and proximity thresholds, which may not generalize well across diverse real-world scenarios. This motivates our exploration of anomaly detection techniques, where suspicious BLE trackers can be conceptualized as anomalies/outliers. In this work, we investigate the applicability of the Cluster-Based Local Outlier Factor (CBLOF) for detecting abnormal Bluetooth tracker behavior. The CBLOF model has been successfully applied in domains such as banking [4] and social media-based crime detection [9]. Although CBLOF has not been previously explored for detection of suspicious BLE trackers, its capability to identify anomalous patterns makes it a promising approach for this purpose.

4 Our Approach

Our system consists of three stages: (i) scanning the environment for nearby devices, during which multiple observations are recorded for each device; (ii) transforming these observations into risk factors; and (iii) inputting the risk factors into the detection model to identify suspicious devices. The overview of this process is shown in Figure 1. During scanning, BL(u)E CRAB on the scanner device captures metadata about nearby devices from their advertising Bluetooth packets. When these packets are received by the scanner device, BL(u)E CRAB records a timestamp of when the packet was received (obtained from the user's device), the location of the device (based on the current location of the user's device), and the signal strength at that moment. This information is used to reconstruct the user's path and derive risk factor values from it. Optionally, the packets include device name, manufacturer name, platform (e.g., Android, iOS), and a unique device identifier. We do not use this in our analysis to protect the privacy of benign devices.

Risk Factors. The raw data captured from scanned Bluetooth data is transformed into quantifiable risk factors, serving as input to the detection model. We focus on three primary risk factors to assess potential threats. These risk factors are generated per device.

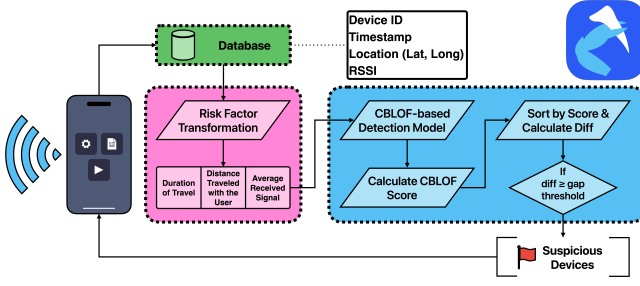


Figure 1: Architecture of BL(u)E CRAB

- **Duration of travel ($Time_d$):** This risk factor quantifies the duration of time that a device has been near the user while isolating active connection periods. This is done by calculating the intervals between timestamps, removing duration intervals that exceed a predefined threshold to eliminate long-time gaps between consecutive detections, and calculating the sum of the remaining intervals, measured in seconds.
- **Distance traveled with the user ($Dist_d$):** This risk factor quantifies the distance that a device has traveled near the user during active connection periods to ensure that spatial data correlate with active exposure. This is done by calculating the intervals between timestamps, removing duration intervals that exceed a predefined threshold to eliminate long-time gaps between consecutive detections, and calculating the sum of the Haversine distance between user locations (latitude and longitude) at neighboring timestamps.
- **Average Received Signal ($RSSI_d$):** Measures the average signal strength for each device over the scan duration¹

Our approach for risk factor transformation takes as input a device d , its observation sequence O_d sorted by time, and the time threshold τ which is used to decide whether two successive observations should be part of the same sub-sequence. The transformation function starts by looping through pairs of observations (o_i, o_{i+1}) , where the difference in timestamps is less than the threshold ($\theta \leq \tau$). The approach then aggregates time in seconds and computes the distance between them using the geodesic distance formula. The RSSI values from these time windows are averaged to derive $RSSI_d$. Each device is subsequently represented by an m -dimensional risk-factor feature vector. The feature vectors for all devices are stored in feature_matrix $X \in \mathbb{R}^{n \times m}$, where each row corresponds to a device, and each column corresponds to a risk factor.

5 Detection Model

To classify a device as suspicious, our detection model analyzes the risk factors of the scanned devices. The core novelty of our detection framework lies in adapting the Clustering-Based Local Outlier Factor (CBLOF) [5], an unsupervised machine learning model, to the domain of Bluetooth metadata. Unlike conventional methods that rely on static, hard-coded thresholds for time and distance [2, 6], CBLOF dynamically groups (clusters) devices based on similarities

in a multi-dimensional feature space. By calculating the deviation of specific devices from these established "normal" clusters, the model can robustly distinguish benign devices from suspicious devices, enabling high-precision detection without requiring prior knowledge of specific attack signatures. This approach is well-suited to a standard Bluetooth environment, where most devices are benign, and only a handful are suspicious. This observation aligns with the assumption in [3] that normal data belong to large clusters while anomalies belong to small clusters. Following this assumption, we apply CBLOF, which divides the established clusters into Large Clusters LC , containing normal devices, and Small Clusters SC , containing anomalous devices.

To implement this framework, we formalize the detection logic using Clustering-Based Local Outlier Factor (CBLOF). Let the feature_matrix $X \in \mathbb{R}^{n \times 3}$ represent n detected devices, where each row corresponds to a 3-dimensional feature vector. The detection process then proceeds into the clustering and scoring phases. The feature_matrix X is input into the K-Means Algorithm to produce a set of clusters $C = \{C_1, C_2, \dots, C_k\}$ in the sequence such that: $|C_1| \geq |C_2| \geq \dots \geq |C_k|$. We define a boundary b for Large Clusters LC and Small Clusters SC if one of the following holds:

$$\sum_{i=1}^b |C_i| \geq |X| \cdot \alpha \quad \text{and} \quad \frac{|C_b|}{|C_{b+1}|} \geq \beta \quad (1)$$

where $\{C_i \mid i \leq b\}$ becomes the set of Large Clusters LC which contains benign devices and $\{C_j \mid j > b\}$ becomes the set of Small Clusters SC containing suspicious devices. Parameters α and β control the determination of large clusters and the large-to-small cluster size ratio, respectively.

After cluster partitioning, the framework quantifies the suspicious level of each device by scoring each device based on its deviation from normal behavior. Let μ_i and μ_j be the cluster centroids of any cluster C_i and C_j . In implementing this model, we adopt the unweighted configuration of CBLOF to avoid bias introduced by cluster size weighting. The CBLOF score for each device is computed using only the distance to the centroid.

$$CBLOF(x) = \begin{cases} \min_{C_j \in LC} \text{distance}(x, \mu_j), & x \in C_i, C_i \in SC; \\ \text{distance}(x, \mu_i), & x \in C_i \text{ and } C_i \in LC \end{cases} \quad (2)$$

As previously established, if any suspicious devices exist, they are likely to be in SC , and therefore making devices in this cluster yield the highest CBLOF scores because of the distance calculation in Equation 2. The CBLOF scores for all devices are first sorted in descending order, where the difference between consecutive scores is computed. The first instance where the difference exceeds the calibrated gap threshold δ , which is used as the anomaly boundary, devices with scores above this boundary are classified as suspicious. This gap-based thresholding mechanism effectively separates high-scoring anomalous devices from the continuum of benign devices. In Algorithm 1, the CBLOF model takes as input the feature_matrix X , the number of clusters, α , β , and the gap threshold δ . In lines 4-8, K-means is first applied to form clusters, which are then sorted by size in descending order. Parameter α is then used to determine the large cluster by accumulating cluster sizes from largest until their total covers at least α fraction (e.g., 90%) of all data points.

¹RSSI value is jittery and impacted by the environment of the scanner device. Hence, we average it over the scan duration for that device.

Algorithm 1: CBLOF-Based Detection Model

```

1 Input: feature_matrix  $X$ , number of clusters  $k$ , gap_threshold  $\delta$ ,
   CBLOF parameters  $\alpha$  and  $\beta$ 
2 Output: Suspicious devices  $S$ 
3  $C \leftarrow K\text{-Means}(X, k)$ 
4 Sort clusters such that  $|C_1| \geq |C_2| \geq \dots \geq |C_k|$ 
5 Determine boundary point  $b$  using  $\alpha$  and  $\beta$  from Eq. 1
6  $LC \leftarrow \{C_1, \dots, C_b\}; SC \leftarrow \{C_{b+1}, \dots, C_k\}$ 
7 for  $x \in X$  do
8   if  $x \in SC$  then
9      $score[x] \leftarrow \min_{C_j \in LC} distance(x, \mu_j)$ 
10  else
11     $score[x] = distance(x, \mu_i)$ 
12 Sort scores in descending order:  $Z = \{z_1 \geq \dots \geq z_n\}$ 
13  $stop \leftarrow -1; S \leftarrow \emptyset$ 
14 for  $i \leftarrow 1$  to  $|Z| - 1$  do
15    $\theta \leftarrow z[i] - z[i + 1]$ 
16   if  $\theta \geq \delta$  then
17      $stop \leftarrow i$ 
18     break
19 if  $stop \neq -1$  then
20    $S \leftarrow \{Z[1], \dots, Z[stop]\}$ 
21 return  $S$ 

```

The parameter β looks for a large drop in cluster size between two consecutive clusters to help separate large (LC) and small (SC) clusters. In lines 9-14, the CBLOF scores are then computed for each device in the small and large clusters and sorted. In lines 15-25, the difference between consecutive scores is computed, and the first instance where the difference exceeds δ defines the anomaly boundary. Devices with scores above this boundary are classified as suspicious; if no difference exceeds δ , no devices are flagged. The time complexity of this algorithm is $O(n \cdot K \cdot i \cdot f)$ where n is the number of devices, K is the number of clusters, i is the number of K-means iterations until convergence, and f is the dimensionality of the feature_matrix X . In Section 7, we empirically show that the detection model is efficient and runs in a few minutes even with hundreds of scanned devices. There, we also discuss how to appropriately set the different parameters α , β , δ , and K involved in the detection model.

6 Implementation

BL(u)E CRAB² is an open-source mobile application developed using the Dart programming language and the Flutter development framework for iOS and Android that collects data from the user's environment to detect malicious BLE devices. Flutter provides a cross-platform framework that allows us to develop BL(u)E CRAB on multiple platforms from a single codebase. BL(u)E CRAB supports four modes to accommodate developers, end users, and researchers with a user-friendly interface and a wide set of features. The detection model is implemented in Python and invoked by BL(u)E CRAB as an external command.

²<https://github.com/DIPrLab/BLuE-CRAB/>

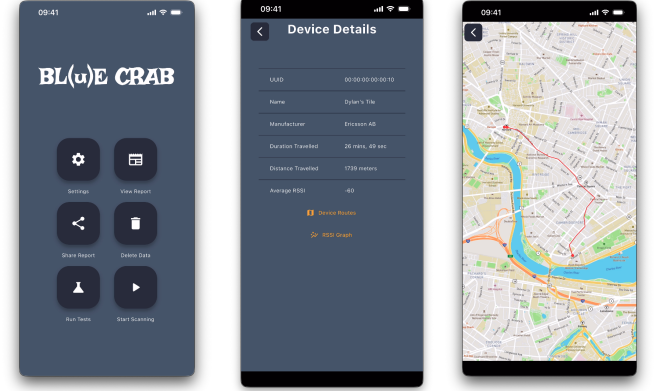


Figure 2: Screenshots of BL(u)E CRAB displaying the scanner view (left), device detail view displaying identifiers along with risk factors (center), and a map of where the selected device traveled with the user (right).

BL(u)E CRAB provides a simple and compact format for storing the collected data by separating them into device data and location history. There is one entry per device, which contains the device's identifier, name, platform, manufacturer, and list of detections. The detections consist of a timestamp rounded down to the nearest second and an array of RSSI values received from the associated device within that second. Storing multiple values per second allows us to estimate the rate of packet transmissions and prevent outlier RSSI values from being used. The location history consists of timestamps with associated location data, which avoids duplicated data by adding entries for the user's location when they move instead of storing entries for every possible timestamp. When reading device and location data from storage, BL(u)E CRAB references the location history by finding the last known location of the user at each timestamp the user saw a device.

7 Evaluations

We evaluate the detection model in BL(u)E CRAB with end-to-end experiments and micro-benchmarks. The end-to-end study evaluates how our app performs in realistic scenarios, and micro-benchmarking evaluates the detection performance and robustness of our detection model across heterogeneous datasets. Our evaluation addresses two important questions: (1) Does anomaly detection achieve fast detection of suspicious devices in real-world settings?, and (2) Does CBLOF accurately detect suspicious devices?

7.1 Experimental Setup

Dataset. We use the real-world datasets from BLE-DOUBT [2], which were collected over two weeks across different scenarios, as well as newly collected datasets obtained using the data collection mode of our BL(u)E CRAB application. Our datasets were collected in urban and suburban neighborhoods while walking and using public transportation. To mimic realistic malicious trackers, trackers (Tile and AirTags) were placed in the researchers' close proximity (pockets and backpacks) during data collection³. The statistics about our dataset are shown in Table 1.

³The anonymized version of this dataset is publicly available in our GitHub Repo.

ID	Duration (H:MM)	Movement	Tracker Location
1	0:31	Walking	Pocket
2	0:23	Walking	Backpack
3	0:30	Bus	Backpack

Table 1: Datasets Collected using BL(u)E CRAB

Ground truth. We use the ground truth file provided by BLE-DOUBT to evaluate our model’s performance. In addition, we include a corresponding ground truth file for our own datasets to enable consistent evaluation.

Model Configuration. We implement CBLOF with a gap-based thresholding mechanism for detection. The model requires selecting parameters, which we determined through systematic empirical exploration during system development. We use k-means as the clustering algorithm. The α and β parameters, controlling the separation between large and small clusters, were explored across a range of candidate values, and configurations that produced stable clustering behavior across multiple datasets were selected. Similarly, the gap-based threshold was determined by evaluating candidate values and choosing settings that yielded consistent detection behavior. The resulting parameter and threshold configuration was then fixed and applied uniformly across all experiments. The configuration $k = 5, \alpha = 0.9, \beta = 5$, and $\delta = 2.4$ produced the best balance between precision and recall, achieving an F1 score of 0.89. We observe that performance remains stable across nearby parameter settings, showing that our model is not overly sensitive to small parameter variations. This robustness indicates that our parameter configuration is generalized across datasets rather than being over-fitted to a specific dataset.

Baselines. We compare the CBLOF detection model in BL(u)E CRAB against the state-of-the-art system, BLE-DOUBT. Specifically, we evaluate our detection model against BLE-Doubt’s best-performing topological classifier. We adopted the parameter values for BLE-DOUBT based on what is suggested in the original paper (duration = 10 minutes, diameter = 300 meters, and $\epsilon = 60$ seconds). Apple and Android’s Find My systems are not included as baselines because their detection mechanisms are proprietary and not publicly documented. We also did not use AirGuard as it is primarily designed to detect the abuse patterns of known trackers, whereas BL(u)E CRAB is designed to detect suspicious BLE trackers irrespective of the manufacturer’s brands.

Metrics Used. The metrics used are Recall, Precision, and F1 Score to evaluate the performance of our app. *Recall* measures how many of the suspicious devices our detection model flags, *Precision* indicates how many of the flagged devices are in fact suspicious, and *F1 score* captures how well the model balances flagging all suspicious devices (Recall) with being accurate when it does (Precision).

7.2 Micro-benchmarks

Detection Performance. We evaluate our CBLOF-based detection model against BLE-DOUBT topological classifier using BLE-DOUBT dataset for fair comparison. Our model achieves 0.94 precision, 0.86 recall, and 0.89 F1 score. Aggregate results for CBLOF-based

Detection Model and BLE-Doubt topological classifier are shown in Table 2. Compared to the baseline, our approach reduces false positives by approximately 77% and false negatives by 20%. We also compare the F1 scores of our detection model and BLE-Doubt topological classifier across highlighted scenarios in the BLE-DOUBT dataset as shown in Figure 4(a). Both models achieve the same performance in the walking scenario. However, our model demonstrated better performance in the driving and train scenarios. In the train scenario, the BLE-DOUBT topological classifier failed to detect the suspicious devices because none of the sub-trajectories of the suspicious devices satisfied the predefined thresholds.

	TP	FP	TN	FN	Precision	Recall	F1 Score
BLE-DOUBT Classifier	47	13	12135	10	0.78	0.83	0.80
CBLOF-based Detection Model	49	3	12145	8	0.94	0.86	0.89

Table 2: Aggregate Results for CBLOF-based Detection Model and BLE-DOUBT topological classifier. (TP= True Positive, FP= False Positive, TN= True Negative, FN= False Negative)

Robustness across Heterogeneous Datasets. We evaluate the robustness of our detection model using datasets collected from different real-world settings. We observe that the number of devices observed during scanning does not appear to significantly affect detection effectiveness, suggesting that the proposed model is resilient to variations in device density.

7.3 End-to-End Case Studies

We conducted two case studies to evaluate the effectiveness and performance of CBLOF in realistic scenarios. Each scenario was carried out by a different researcher to demonstrate generalizability.

Case Study 1 - Walking. The first case study consists of two walking scenarios. In the first scenario, we evaluate CBLOF in an indoor walking environment inside the university library for 30 minutes. To simulate a malicious tracker, the researcher placed a tracker in their backpack. During the experiment, the researcher moved across all five floors of the library, encountering approximately 60 nearby devices per floor. After scanning, a total of 300 devices were observed. At the end of the scan, the Tile tracker was flagged as suspicious. To further analyze this result, we examined the F1 score over time (as shown in Figure 3(a)). Within the first five minutes of scanning, our model correctly identifies the suspicious tracker. Between minutes 11 and 20, the model’s performance degraded briefly as an influx of new devices caused the tracker’s feature vector to appear indistinguishable from the benign devices. Detection accuracy was restored after minute 21, when the tracker’s feature vector once again became statistically distinct from other devices.

In the second scenario, the researcher walked around a suburban neighborhood for approximately 25 minutes, placing four trackers (2 Tiles, 1 AirTag, 1 SmartTag) in their pockets. A total of 586 nearby devices were observed during the experiments. At the end of scanning, all four trackers were accurately detected. The F1 score over scanning duration is illustrated in 3(b). From the analysis, the sudden dip occurring between minutes 22 and 24 appears to be a temporary detection instability, possibly caused by reduced cluster separability in the feature space. From minute 25 onward, the tracker became clearly separated from other devices, allowing the model to correctly flag it as suspicious.

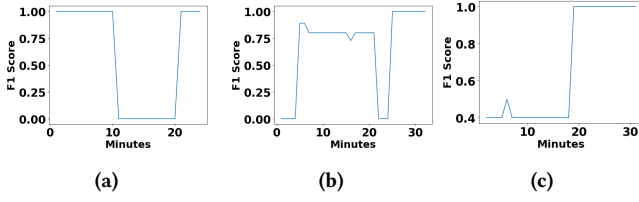


Figure 3: F1 score over time. (a) shows when the researcher walked inside the university library. (b) shows when the researcher walked around their suburban neighborhood, and (c) shows when the researcher took the bus commute.

Case Study 2 - Bus. In the second case study, the researcher rode a bus for 30 minutes. One Tile tracker was placed in the researcher’s backpack during the commute. As the bus was busy, about 1,400 devices were observed during scanning. Several stops were made during transit, causing some nearby devices to appear briefly. At the end of scanning, the Tile tracker was detected as suspicious, as seen in Figure 3(c).

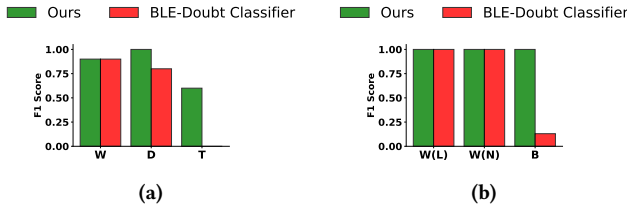


Figure 4: (a) Comparing F1 Scores across different scenarios (W = Walking, D = Driving, T = Train) in BLE-DOUBT dataset (b) Comparing F1 Scores across our Case Studies (W(L)= Walking inside the Library, W(N)= Walking around the neighborhood, B = Bus Commute)

From these case studies, we observe that our model is effective across different real-world scenarios. Figure 4(b) presents a comparison of F1 scores across different datasets. We evaluate the performance of our detection model against BLE-DOUBT’s best performing topological classifier. In the walking case study, both methods achieve an F1 score of 1. However, in the bus commute case study, our detection model demonstrated better performance. BLE-DOUBT topological classifier falsely flagged 13 other devices, which is attributed to its detection criterion of constructing sub-trajectories to satisfy predefined thresholds, causing several non-suspicious devices to be classified as suspicious. We also observe that as the feature distribution stabilizes over time, CBLOF finally determines whether a device is suspicious.

Our evaluation demonstrates how an anomaly-based detection model performs in real-world scenarios through case studies conducted in urban, suburban, and bus commute environments. The results show that CBLOF can successfully detect trackers under varying real-world conditions. One key advantage of CBLOF is that it does not require accumulating long trajectory segments before making detection decisions, unlike the topological classifier used in BLE-DOUBT. CBLOF is able to make detections regardless of

scanning duration, improving the practical usability in deployment settings where long-term scanning may not be feasible.

8 Conclusions & Future Work

We present BL(u)E CRAB, an open source cross-platform application designed to detect nearby BLE devices and determine whether they exhibit malicious behavior. Our approach combines a CBLOF-based detection model with a gap-thresholding mechanism, enabling accurate detection and achieving an overall F1 Score of 0.89 while outperforming the baseline BLE-DOUBT approach. Through both real-world case studies and micro-benchmark evaluations, we showed that CBLOF operates effectively across environments with varying device densities and remains reliable under practical conditions. These results demonstrate that the system is suitable for real-world deployment and can serve as a practical tool for everyday detection of suspicious BLE trackers.

Future work will focus on evaluating the model’s robustness across more diverse and large-scale deployment scenarios. Handling MAC address rotation behaviors in trackers is another challenge, as our model may incorrectly interpret them as distinct entities, leading to fragmented feature vectors. Furthermore, we plan to explore different density-based clustering techniques, such as DB-SCAN, and exploit tracker-specific features (e.g., Apple’s continuity protocol) to improve the detection accuracy even further.

9 Acknowledgments

This work is partially supported by the NSF Award #2451803.

References

- [1] Security Boulevard. 2021. Apple AirTag Android App is Absolutely Awful—Tracker Detect Fail. <https://securityboulevard.com/2021/12/apple-airtag-android-app-is-absolutely-awful-tracker-detect-fail/> Accessed Mar. 3, 2026.
- [2] Jimmy Briggs and Christine Geeng. 2022. BLE-Doubt: Smartphone-Based Detection of Malicious Bluetooth Trackers. In *43rd IEEE Security and Privacy, SP Workshops 2022, San Francisco, CA, USA, May 22-26, 2022*. IEEE, San Francisco, CA, USA, 208–214. doi:10.1109/SPW54247.2022.9833870
- [3] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Comput. Surv.* 41, 3 (2009), 15:1–15:58. doi:10.1145/1541880.1541882
- [4] Bharti Chugh, Nitin Malik, Deepak Gupta, and Badr S. Alkahtani. 2025. A probabilistic approach driven credit card anomaly detection with CBLOF and isolation forest models. *Alexandria Engineering Journal* 114 (2025), 231–242. doi:10.1016/j.aej.2024.11.054
- [5] Zengyou He, Xiaofei Xu, and Shengchun Deng. 2003. Discovering cluster-based local outliers. *Pattern Recognit. Lett.* 24, 9–10 (2003), 1641–1650. doi:10.1016/S0167-8655(03)00003-5
- [6] Alexander Heinrich, Niklas Bittner, and Matthias Hollick. 2022. AirGuard - Protecting Android Users from Stalking Attacks by Apple Find My Devices. In *WiSec '22: 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, San Antonio, TX, USA, 26–38. doi:10.1145/3507657.3528546
- [7] Alexander Heinrich, Leon Würsching, and Matthias Hollick. 2024. Please Unstalk Me: Understanding Stalking with Bluetooth Trackers and Democratizing Anti-Stalking Protection. *Proc. Priv. Enhancing Technol.* 2024, 3 (2024), 353–371. doi:10.56553/POPETS-2024-0082
- [8] Lewis Painter. 2021. Apple’s Find My app explained: Tracking items, devices and more. <https://www.techadvisor.com/article/743764/apples-find-my-app-explained-tracking-items-devices-and-more.html> Accessed Feb. 2, 2026.
- [9] Taahir Aiyob Patel and Clement N. Nyirenda. 2023. Semi-Supervised Anomaly Detection for the Determination of Vehicle Hijacking Tweets. *CoRR* abs/2308.10036 (2023). arXiv:2308.10036 doi:10.48550/ARXIV.2308.10036
- [10] Christian Wait. 2024. Tracker Detect Android - What is it, How to set up, and how to use it. <https://www.pcguides.com/android/android-how-to/tracker-detect-android/> Accessed Jan. 25, 2026.