Okay Google, Where's My Tracker? Security, Privacy, and Performance Evaluation of Google's Find My Device Network

Leon Böttger
lboettger@seemoo.de
Secure Mobile Networking Lab
Technical University of Darmstadt

Alexander Matern
amatern@seemoo.de
Secure Mobile Networking Lab
Technical University of Darmstadt

Dennis Arndt
darndt@seemoo.de
Secure Mobile Networking Lab
Technical University of Darmstadt

Matthias Hollick
mhollick@seemoo.de
Secure Mobile Networking Lab
Technical University of Darmstadt

Abstract

In April 2024, Google launched the Find My Device Network (FMDN), an Offline-Finding Network (OFN) that allows lost Bluetooth devices, such as trackers or headphones, to be located using billions of Android devices as finders. Similarly to Apple's Find My network, it is activated by default on all modern Android devices. Google promises end-to-end encryption for all location updates and claims to protect the privacy of finder devices as well as owners of lost devices and trackers. Although Android is open-source, FMDN is part of Google Play Services and is only partially publicly specified. We reverse-engineer the proprietary parts of the network, document its behavior, and analyze its privacy, security, and performance. We find several security and privacy issues, including denial-of-service attacks and a potential linkage attack on Android. We further implement a custom app, porting Google's trackers from Android to iOS while also extending the features of the FMDN.

Keywords

Bluetooth, Offline-Finding Networks, Tracker, Security, Privacy, Find My Device

1 Introduction

Offline-Finding Networks (OFNs) allow people to track valuable items by attaching small trackers to them. Trackers can be as small as a coin, are available in several shapes, such as a credit card, and usually have an independent battery life of up to one year. These trackers are offline, have no means of connecting to the internet, and do not use a Global Navigation Satellite System (GNSS) sensor to determine their location. Hence, they use Bluetooth Low Energy (BLE) advertisements to broadcast beacons, which so-called finder devices can receive (see Figure 1). A finder device can be a smartphone with an active internet connection and a GNSS sensor or other means to determine its location. When a finder device scans for BLE advertisements and finds a tracker, it fetches its location using GNSS and forwards its location to the tracker manufacturer. As a result, the owner can now view the tracker's location.

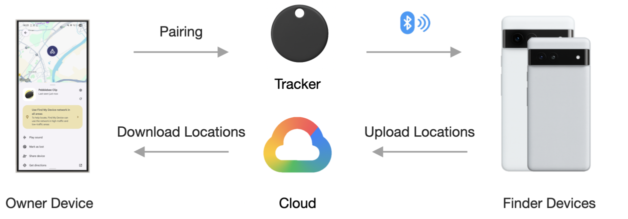


Figure 1: Offline-finding workflow for Google Find My Device Network (FMDN) trackers [61].

Apple was the first smartphone manufacturer to introduce an OFN, using all its iPhones and other Apple devices to act as finder devices. This created a massive worldwide network that can locate trackers with high accuracy and deliver updates with an average delay of only 15 minutes [33]. Apple reported that its network consists of up to a billion finder devices [2]. Samsung followed shortly after and implemented an OFN using Samsung smartphones to locate devices and trackers. The main difference with Apple's network is that Samsung used an opt-in approach, while Apple's network is only opt-out. Therefore, the last reported number of finder devices was 300 million [55]. A comparison between both networks has shown a similar accuracy of 30 m to 100 m [35].

In any case, an OFN deals with highly sensitive location data. As recent examples have confirmed, location data in the wrong hands can be used to identify individuals, build social graphs, create profiles for targeted advertising, and facilitate Unwanted Tracking (UT) [20, 33, 37, 69]. Existing OFNs have been found to be vulnerable to a range of attacks: Tile's network has been leaking email addresses since 2019 [37, 69], an issue that still exists today. Furthermore, their network was exploited in 2024, giving the attacker access to location data of all users [20]. Although Apple's OFN employs end-to-end encryption, researchers were able to access location data on macOS using an unprivileged macOS app that could download and decrypt the user's location [33].

Even if an adversary does not exploit the network, they may still track individuals by hiding a tracker in their belongings. These

Proceedings on Privacy Enhancing Technologies 2025(4), 605–619
© 2025 Copyright held by the owner/author(s).
https://doi.org/10.56553/popets-2025-0147

UT and cyberstalking attacks have become a major issue in our society [19, 39, 41]—especially since Apple and Samsung have created OFNs that deliver accurate and timely information. Previous research has analyzed these stalking incidents and found that victims are stalked through a plethora of methods, with unwanted location tracking used in 44.28 % of cases [34]. To inform victims about potential stalking attacks, tracker manufacturers have started implementing tracking detection methods. If a tracker follows a user for a certain time, the smartphone will warn them by sending a notification. Additionally, the research community has developed several apps to detect trackers for iOS and Android [13, 31, 46].

In April 2024, Google launched the Find My Device Network (FMDN), a new OFN for the largest mobile operating system Android. The FMDN is now also known as the *Find Hub* network. Any device running Android 9 or newer with Google's Play Services installed can now act as a finder device, creating a massive network with billions of devices. Since its launch, several third-party manufacturers have built trackers for the FMDN. Google has been more open than other manufacturers. It documents the behavior of trackers, including the applied end-to-end encryption methods [25]. Furthermore, it has learned from the mistakes of others – Google implemented UT detection on Android and, in cooperation with Apple, on iOS [38]. Additionally, it claims to implement rate-limiting to restrict stalkers' ability to get real-time updates on their victim's location [11]. By default, the location of finder devices is further protected by combining multiple location reports and only showing an approximate location to the owner of a tracker. Users can configure settings on the finder device to always report a precise location instead of relying on combined reports.

Google's network seemingly had a promising start in setting higher security and privacy standards. Nevertheless, since the code is not part of the Android open-source project, we asked the following research questions:

- **RQ1:** How is sensitive data protected against external access in Google's FMDN?
- **RQ2:** Is it possible to circumvent Google's tracking detection mechanisms?
- **RQ3:** How accurate are the locations reported by the network?

To summarize our contributions:

(1) We reverse-engineer the closed-source parts of the FMDN.
(2) We publish and document the entire FMDN protocol, from registration of a tracker to finder devices' location reports.
(3) We identify several security issues, allowing us to take over a previously shared tracker and perform a linkage attack that could be executed by a Google-internal attacker.
(4) We discover four different methods to circumvent current tracking detection mechanisms.
(5) We evaluate the network's performance and compare it to Apple's Find My OFN.

2 Background & Related Work

OFNs use BLE advertisements to locate lost devices and miniaturized trackers. In this section, we detail the basics of an OFN, introduce Google's FMDN, and summarize the related work.

2.1 Offline-Finding Networks

An OFN consists of four parts: the tracking device, the finder devices, the OFN provider cloud, and another owner device, which may be used to access the location of the tracking device. In this example, we explain the OFN based on a Bluetooth tracker, but it could just as well be a lost smartphone, laptop, or any other device with BLE capabilities. Figure 1 demonstrates this using the example of the Google FMDN.

To set up the tracker, the owner pairs it with their smartphone using the app provided for the OFN. The pairing process usually exchanges key material, ensuring that only the owner may access location reports and perform privileged actions, like playing a sound. The tracker can then broadcast BLE advertisements on known BLE channels. Each advertisement may contain up to 31 bytes of data. Extended advertisements in Bluetooth 5.0 even support up to 254 bytes of data. Data can be formatted as manufacturer-specific data, starting with a company ID, followed by the actual data, or as service data, indicating a publicly known service UUID [12] and service-specific data. In any case, the data sent by the tracker is used by the network to identify it.

Finder devices, usually other smartphones, scan for BLE advertisements of trackers at regular intervals. If they find a tracker, they fetch their own location information using GNSS and send a location report for the tracker to the OFN provider cloud. The owner can download the location reports using the app for the OFN. Some OFNs use end-to-end encryption, which ensures that only the owner of the device can decrypt the location, even if the location database were breached.

2.2 Google Find My Device Network

Similar to previous OFNs, Google's FMDN supports a number of tracking devices: Bluetooth trackers, smartphones, and headphones. Most of the tracking devices are built by third-party manufacturers who collaborate with Google.

However, contrasting existing OFNs, Google's FMDN operates in multiple modes that can be switched by the user. The default setting is called "With Network in High-Traffic Areas Only". If this setting is enabled, location reports of multiple finder devices are required and will be aggregated. In Section 4.3.1, we detail the properties of such a "High Traffic" area. In the alternative network mode "With Network in All Areas," reported locations of trackers are forwarded to the owner of a tracker, even if no other finder device detected it. This setting needs to be manually enabled.

2.3 Related Work

Research on OFN started in 2020, with the first publication analyzing several tracking networks for their security and privacy properties [69]. They found multiple data leaks, ranging from email addresses to location data. Finally, they invented a new OFN using end-to-end encryption. A similar research paper analyzed a different set of tracking networks for defined security properties, such as spoofing protection [22]. None of the analyzed networks adhered to all properties.

Apple's Find My network was also reverse-engineered and analyzed [33]. The network uses end-to-end encryption for all location data, protecting the privacy of finder devices and owner devices.

Nevertheless, several oversights allowed researchers to access location data from an unprivileged macOS app. Furthermore, custom trackers based on Apple's Find My network were developed by the same team [32].

In 2024, the Samsung SmartThings Find network was scrutinized [70]. Researchers found several issues with the tracker hardware itself, allowing them to identify trackers and rendering MAC address randomization useless. Furthermore, they were able to spoof fake location reports by relaying advertisements or extracting a token from a legitimate finder device.

As every tracker may also be used for UT, detecting a malicious tracker is important for potential victims. Three Android apps were built by the research community to detect trackers [13, 31, 46]. The only research-community-developed app available on public app marketplaces is AirGuard [31], which is also the only app for tracking detection on iOS [16]. Additionally, Android and iOS offer integrated tracking detection for Apple's Find My network and Google's FMDN [2, 11]. Furthermore, commercial products exist and are sold between USD 499 and USD 3500 [7, 8]. However, researchers found existing tracking detection mechanisms provided by manufacturers to be insufficient [68]. Tracking detection mechanisms were analyzed, and in all cases, an adversary who could create a custom tracker would be able to bypass them [44]. Approaches to counter these malicious custom trackers have also been proposed [43]. At the time of writing this paper, a standardization process at the Internet Engineering Task Force (IETF) for detecting unwanted location trackers is ongoing. It was initially proposed by Apple and Google to create a unified protocol that location trackers should use to be detectable if used for unwanted tracking [38].

In addition to UT, OFNs have been misused for other unintended purposes. For example, using Apple's Find My for data transfer has been studied [6, 29, 66]. Depending on the protocol used, transfer speeds of up to 12.5 bps were achieved. Recently, scientists also found that it was possible to create BLE advertisements sent from Windows, Linux, and Android, replicating the Apple Find My format and allowing an adversary to track the affected device [17].

3 Methodology

A combination of investigative approaches was used to understand the FMDN protocol. For starters, we used Google's developer documentation, which already documents large parts of the tracker firmware [25]. Then, we used dynamic [47, 52] and static [50, 60, 67] application reverse engineering programs to study the inner workings of various undocumented protocols involved in FMDN, analyzed various logs produced by Android and applications [21, 24], and examined both BLE and network traffic between devices and Google servers [40, 48?]. We omit further details, as this methodology is common for reverse engineering and vulnerability research.

Devices involved in the analysis include a laptop with macOS 15, and a number of Android devices running Android versions 9–14, rooted and unrooted. We list all the devices in Table 1. The rooted devices ran Google Play Services 24.23.25, and the Find My Device (FMD) app 3.1.148 (now called *Find Hub*). Unrooted devices were frequently updated to use the latest version of the Play Services app and the FMD app.

Table 1: Used Android devices for testing.

Device	Android	Rooted
Samsung Galaxy S5 Plus	9	✓
Samsung Galaxy S5	12	✓
Google Pixel 2 XL	11	✓
Samsung Galaxy S21 Ultra	14	×
Samsung Galaxy A55	14	×
ASUS ROG Phone 3	12	×
Samsung Galaxy S8	9	×

Table 2: Evaluated trackers that support the FMDN.

Model	Manufacturer	Type
ONE Point	Chipolo	Tracker
CARD Point	Chipolo	Tracker
Moto Tag	Motorola	Tracker
Clip	Pebblebee	Tracker
WH-1000XM5	Sony	Headphones

We evaluated the FMDN with all trackers that were available in our region at the time of writing. The trackers and their tracker manufacturers are listed in Table 2.

4 The Google Find My Device Network Protocol

In this section, we present the protocol's registration process, Bluetooth advertisements, location reports, as well as retrieving network locations by the owner. Some parts are publicly documented [25], while other protocol details were obtained through reverse engineering.

4.1 Firmware

As mentioned earlier, Google already described details of the firmware of FMDN trackers meant for tracker manufacturers [25, 26]. We briefly summarize Google's documentation in this section.

4.1.1 Fast Pair. FMDN trackers are initially paired with Google's Fast Pair protocol. The pairing process is handled by Google Play Services, a system-level app installed on most Android devices. The specification for Fast Pair is partly available; implementation guidelines for Fast Pair-compatible peripherals are described in Google's developer documentation [25].

Fast Pair uses a pre-shared key to protect against Machine-in-the-middle (MitM) attacks during pairing [26]. This key is a NIST-256 elliptic curve key pair, unique for every Fast Pair device model, and generated when the manufacturer registers the Fast Pair model with Google. The private part of this key is stored on the Fast Pair device itself, while the public part is uploaded to Google's server. If an Android smartphone attempts to pair with a Fast Pair device, both devices must prove that they possess the respective part of the pre-shared key. Fast Pair accessories advertise the Fast Pair service FE2C. Android devices read out the Fast Pair Model ID from the advertisement and send it to Google. Google then returns the public part of the pre-shared key. An elliptic curve Diffie-Hellman

(ECDH) key exchange is performed, which is later used to encrypt succeeding keys using `AES-ECB` [26]. The Fast Pair procedure ends with the exchange of the *Account Key*, which is a 16-byte value randomly generated on the Android device, starting with `0x04`. Further Bluetooth communication uses this key for authentication.

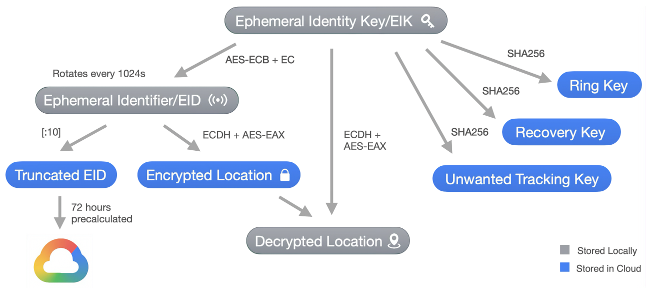


Figure 2: Keys derived during pairing.

4.1.2 *Tracker Pairing and Key Exchange.*

After the Account Key is exchanged using Fast Pair, the FMDN-specific pairing process starts. This involves calculating and deriving the keys shown in Figure 2. These keys are needed for the end-to-end encryption of the network. The most important key is the 32-byte Ephemeral Identity Key (EIK), functioning as a master key. The key is unique for every tracker, randomly generated on the phone, and not known to Google. During the pairing procedure, the EIK is sent to the tracker using an encrypted channel based on the *Account Key* [25].

Three keys are derived from the EIK using the 16-byte prefix of the SHA-256 hash of the EIK appended with an identifier. The identifier is `0x01` for the Recovery Key, `0x02` for the Ring Key, and `0x03` for the UT Key [25]. The Recovery Key can be used to recover the EIK from the tracker. The Ring Key can be used to start a sound on the tracker. The UT Key is used to enable UT mode. We discuss UT mode in Section 4.1.6, and describe its activation in Section 4.2.3. Those keys are stored in the cloud and accessible to Google.

4.1.3 *Tracker Advertisement.*

If the pairing process is complete, the FMDN tracker starts sending BLE advertisements using the BLE Service `0xFEAA`, allowing finder devices to discover it and report its location. Table 3 shows the contents of the advertisement. The Ephemeral Identifier (EID) is a public key allowing finder devices to end-to-end encrypt location reports. The EID can either be 20 or 32 bytes long. The UT byte indicates whether this tracker is in UT mode. The hashed flags store the battery status and the UT mode status. They are encrypted using the EIK and can only be accessed by the owner [25]. Trackers do not implement an activation lock, and they can be forcibly reset. This procedure varies by manufacturer, but always requires physical interaction, such as pressing a button.

4.1.4 *EID Calculation.*

The EID is calculated on the tracker. It is generated as follows [25]:

(1) A byte array, as defined in Table 4, is constructed.
(2) The data is then encrypted using AES-ECB-256 with the EIK.
(3) The result is interpreted as an unsigned integer r' in Big Endian.

Table 3: BLE advertisement data sent from Google FMDN trackers (20-byte EID).

Bytes	Content
0-4	BLE flags and payload length
5-6	`0xFEAA` (Service UUID)
7	UT byte: `0x40` = off; `0x41` = on
8-27	EID
28	Hashed flags

Table 4: Byte array used to create the EID. K defines the rotation period exponent, representing when the EID rotates.

Bytes	Content
0-10	`0xFF` * 11
11	K
12-15	`ts_bytes`
16-26	`0x00` * 11
27	K
28-31	`ts_bytes`

The default key rotation is set to $1024s = 2^K s = 2^{10} s$. `ts_bytes` represents the timestamp of the device, counting the seconds since its initial provisioning, with the lowest K bits set to 0.

(4) The unsigned integer r' is mapped to the finite field F_p of SECP160R1 (20 bytes) or SECP256R1 (32 bytes) by performing a modulo operation with the order of the curve. The result is r.
(5) Finally, the point on the curve R is calculated by multiplying r by the generator of the curve, and the x-coordinate of the point R (R_x) is returned as the EID.

The counter (`ts_bytes`) determines the current advertisement. For this calculation, the lowest K (= 10) bits are always set to 0, leading to a change of the EID every 1024 seconds on average. A slight randomization is implemented, which we describe in more detail in Section 6.2.2.

4.1.5 *End-to-End Encryption.*

The EID can be transformed to a public key that is used to encrypt data using the following method [25]:

(1) The correct curve is chosen: SECP160R1 for 20-byte EIDs or SECP256R1 for 32-byte EIDs.
(2) The finder device generates a random ephemeral key pair on the respective curve, (d_e, P_e).
(3) The public key P_{eid} is extracted from the EID. The EID represents the x-value of P_{eid}, while the y-value is derived by substitution in the curve equation.
(4) An ECDH key exchange $s = d_e * P_{eid}$ is performed.
(5) A 256-bit key is derived as $k = \text{HKDF-SHA256}(s_x)$.
(6) The nonce is calculated by appending the lower 8 bytes of the x-value of P_{eid} and the lower 8 bytes of the x-value of P_e.
(7) The message is encrypted using `AES-EAX-256` with the nonce and k as the key.
(8) The result is the encrypted message with a 16-byte tag.
(9) The final result is sent to Google alongside the x-value of P_e.

4.1.6 *Unwanted Tracking Protection Mode.*

UT (Protection) Mode is designed to mitigate the risk of trackers being misused for stalking. During normal operation, a tracker's BLE MAC address rotates with the EID, ca. every 1024 seconds. If UT mode is enabled, the MAC address rotation is reduced to one rotation every 24 hours. The EID rotation behavior remains unchanged. As a result, the built-in tracking detection mechanisms of Android and iOS can determine if a FMDN tracker is following a user.

4.2 Finder Devices

In the following, we present the aspects of FMDN trackers that are unrelated to their firmware. This information is not officially documented and we obtained it through reverse engineering. We describe web requests and data encoding strategies in the Appendix.

4.2.1 *Client-Side Scanning and Delaying.*

All Android devices running Android 9 or later, with Google Play Services 24.12.14 or newer [56], can act as finder devices. Google Play Services continuously scans for FMDN-compatible trackers via Bluetooth, detecting new trackers almost immediately. Even if Bluetooth is disabled, Play Services can scan for trackers if the Android location services setting *Bluetooth Scanning* is enabled. If a Bluetooth advertisement from a FMDN-compatible tracker is detected, the finder device queries its location, with a timeout of 150 seconds. If the device's current location is within 100 m of the owner's home location, sightings of foreign trackers are discarded (see Section 7.4).

If a valid location was retrieved, non-discarded sightings are passed to the `SightingAggregator`, which tracks both uploaded sightings and those pending upload. Sightings are grouped by the `SightingKey`, which corresponds to either the tracker's EID or the device ID of an owned tracker. Only one sighting per `SightingKey` (based on time and accuracy) is uploaded, with a maximum of 100 sightings per upload. If a location has already been uploaded for a `SightingKey`, a re-upload is allowed if the location changes by at least 40 m, the accuracy improves by at least 50%, or the time difference is at least 300 seconds (foreign trackers) or 60 seconds (owned trackers).

If the finder's screen is off, location uploads for foreign trackers are delayed by at least 5 minutes. In power-saving mode, this delay extends to 15 minutes. If the Android device's screen is on or an owned tracker is detected, locations are uploaded immediately.

4.2.2 *End-to-End Encryption of Location Data.*

If locations should be uploaded, the location is encoded using Google's Protobuf. Latitude and longitude are stored as signed 32-bit integers. To convert decimal coordinates to integers, a factor of 1.0×10^7 is applied. After encoding the location in Protobuf format, it is serialized and encrypted using the EID. In addition to the encrypted location and the ephemeral public key P_e, the *Truncated EID* is sent, representing the 10-byte prefix of the EID a tracker advertises. A precomputed list of future precomputed Truncated EIDs is regularly uploaded by the tracker's owner to Google's server, allowing the server to identify which tracker and Google account a location report belongs to. Location reports also contain further unencrypted metadata, such as the UNIX timestamp when the tracker was found, the accuracy of the location, and the status of the UT mode.

The end-to-end encrypted payload and the additional metadata will then be encrypted again using a public key of Google and the NIST P-256 curve. The encrypted payload is sent alongside the so-called DroidGuard results, which attest to the integrity of the Android device (see Section 6.2.5).

4.2.3 *Final Report.*

To send the report to the server, the HTTPS request `UploadScans` is used. The so-called `X-Goog-Spatula` header is used to authenticate the client (see Section 6.3.2)

The server's response to this request contains an array of actions the client should perform on the foreign devices reported to the server. Each action object contains an array of Truncated EIDs to identify the device the server is referring to. There are three actions available: the first indicates whether the sighted tracker was explicitly marked as lost by its owner. In this case, the Android device evaluates the current Bluetooth signal strength of the tracker. If the tracker is roughly 1 meter or less away, a notification will be displayed, allowing the user to contact the owner. Second, the Ring Key may be sent, instructing the finder device to play a sound on the sighted tracker by connecting using Bluetooth and sending the Bluetooth request detailed in [25]. Third, the UT key may be sent. In this case, the finder device is instructed to enable UT mode on the sighted device by using the Bluetooth request from [25]. The server will be implicitly notified if the write operation was successful, since the next finder device location report contains the status of UT mode for a reported tracker. Therefore, enabling UT mode requires a nearby Android phone and an active internet connection, which contrasts previous OFN implementations that enable UT mode using a timer on the tracker [31, 70]. Disabling UT mode requires the EIK and an Android device belonging to the owner to be near the tracker.

4.3 Server

Another unique aspect of the FMDN in comparison to other networks such as Apple's Find My and Samsung's SmartThings is the implementation of certain privacy protection techniques on the server, which are also advertised by Google [11, 28].

4.3.1 *High-Traffic Areas Only.*

As mentioned before, the owner of a finder device can configure how to participate in the FMDN. By default, devices are configured in the "High Traffic" mode. Reports that originated from finder devices that have this mode enabled are not supposed to be shown to the user directly, but shall be aggregated by averaging multiple locations. The server cannot aggregate the locations directly due to end-to-end encryption. Therefore, aggregation is performed locally on the owner device (see Section 4.4). Most importantly, the server will only send these location reports if at least two of them are available from at least two *distinct* devices. In addition, we found that in most cases, the server will only cache the four most recent "High Traffic" locations, i.e., it will purge older locations.

4.3.2 *In All Areas.*

Finder devices can also be manually set to the "In All Areas" mode. Reports uploaded by such a device do not require aggregation on the client. This is also how other networks, such as Apple's Find My [33] or Samsung's SmartThings Find [70], handle location reports. However, to receive "In All Areas" reports, the owner also needs to enable the "In All Areas" contributor mode, otherwise those reports will appear as "High Traffic" reports. Also, similar to the purging implemented for "High Traffic" locations, retrieval of a new "In All Areas" location will remove previous, older locations, including "High Traffic" reports.

4.3.3 Owner Reports. Similarly, if a self-report is sent from the owner, all previous network locations are dropped from the server. In addition, all "In All Areas" reports up to 10 minutes after the last owner report will not be stored. "High Traffic" reports are kept but remain unavailable until at least 10 minutes have passed. As a result, the FMDN does not provide a location history as seen in other OFNs.

4.3.4 Rate Limiting and Throttling. In addition to the contributor mode and data purging, the server implements rate limiting and throttling. Rate limiting restricts the number of times an Android device can report a *particular* tracker. If an Android device sends multiple location reports for a specific tracker, succeeding reports can be discarded by the server. We found that only one location report per tracker is permitted roughly every nine hours.

The FMDN also implements throttling, which limits how often the owner can receive an update from *any* finder device. We observed that throttling is initially disabled when a new tracker is paired, and is enabled one hour after continuously retrieving new location reports[1]. If disabled, locations can be queried without limits. With throttling enabled, an "In All Areas" report blocks further updates for 10 minutes, while "High Traffic" reports blocks them for 5 minutes from the latest report timestamp.

4.4 Owner Smartphone

Similar to previous OFN implementations, the owner's smartphone interacts with the network in multiple ways: it reports the locations of the owner's trackers to Google in the background, and the user may register new trackers or view the location of their trackers. We describe additional details in the Appendix and publish open-source code to fetch tracker locations [15].

4.4.1 Own Tracker Location Reports. Locations of nearby owned trackers are uploaded by the Play Services app and the Google FMD app. In either case, the location is encrypted symmetrically using the EIK. This process is performed as follows:

(1) A random 12-byte IV is generated.
(2) The EIK is hashed using SHA-256 mode, producing a 32-byte key.
(3) The Protobuf location data is encrypted with the hashed EIK using AES-GCM, with a tag length of 128 bits.

After encryption, the API request `UploadOwnerScans` is used to upload sightings for nearby owned devices. For identifying the tracker, the device ID is included in the location report instead of the Truncated EID.

In addition to the smartphone of the owner, Google Home devices can report the location of trackers owned by the same user. We detail related issues in Section 6.3.3.

4.4.2 Retrieving Location Reports. The retrieval of locations is implemented in the Google FMD application. The FMD app uses the Nova API (see Section A) to retrieve location reports for trackers. This request contains the contributor type of the owner, used to determine which locations the server sends (see Section 4.3.2). Interestingly, the encrypted locations will not be included in the response of the HTTPS request's API call, but will be delivered using Google's proprietary Firebase Cloud Messaging (FCM) protocol. The FCM protocol has already been reversed in [63] and requires client applications to register their installation. This process involves generating and exchanging several public cryptographic secrets with Google's Firebase server, which will be used to encrypt sent and received FCM messages later on. Finally, the device receives a registration ID, which FCM can use to identify this device to send a notification to it. The registration ID is shared with the server when requesting location updates.

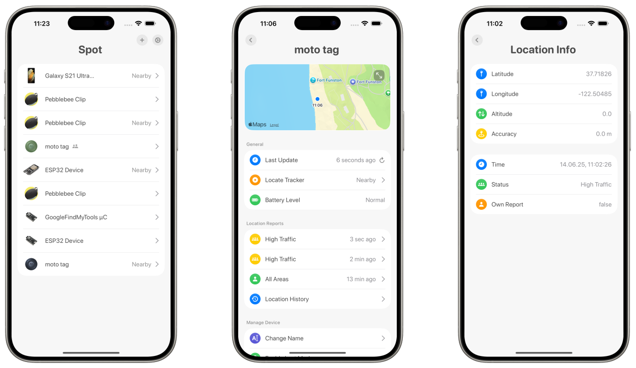
4.4.3 Location Report Payload. FCM delivers the locations with the same data as in Section 4.2.2, i.e., the end-to-end encrypted location report, the accuracy, and the discovery time. In addition, the server sends the tracker's timestamp `ts_bytes`, which determines the time in seconds since the tracker was first set up, rounded down to the last multiple of 1024. Furthermore, the server sends a status field indicating from which contributor type the location report came. A "Last Known Location" is a report that was generated by an owned device. A "Crowdsourced Location" report is a report that was uploaded by a foreign device set to the "In All Areas" contributor mode. "Aggregated Location" reports are sent from "High Traffic" contributors (or "In All Areas", see Section 4.3.2).

4.4.4 Decryption of Locations. If a "Crowdsourced Location" or multiple "Aggregated Location" reports are received, the reverse operation to the public key encryption process from Section 4.1.5 is applied to decrypt the location data. Encrypted location reports uploaded by one of the *owner* devices are decrypted by reversing the encryption operation described in Section 4.4.1. The result of this procedure contains the location (Latitude, Longitude, Altitude) represented as a Protobuf object.

4.4.5 Client-Side Location Aggregation. After decryption, received reports are post-processed depending on their type. If a non-aggregated report is sent from the network ("Last Known Location" or "Crowdsourced Location"), this report is directly returned and shown to the user in the app. In contrast, multiple "Aggregated Locations" are filtered on-device and aggregated, requiring at least 4 locations in our evaluation. The minimum number of locations is determined by the server.

In addition, all locations need to be within a certain radius varying between 324 and 644 meters. If locations with the minimum size are found where the latest location is newer than the time of the last report, and all locations are detected within 60 minutes, the average location is calculated by taking the average of the locations' latitude values and the average of the locations' longitude values. The timestamp of the final aggregated location is the time of the most recent location report, rounded to the last 10 minutes. Afterward, this location is encrypted symmetrically with the EIK and re-uploaded by the FMD app.

[1]We tested this in a busy area, retrieving a new location every 30 seconds.



Spot

Galaxy S21 Ultra...	Nearby
Pebblebee Clip	
Pebblebee Clip	Nearby
moto tag	
ESP32 Device	Nearby
Pebblebee Clip	
GoogleFindMyTools µC	
ESP32 Device	
moto tag	Nearby

moto tag

General

Last Update	6 seconds ago
Locate Tracker	Nearby
Battery Level	Normal

Location Reports

High Traffic	3 sec ago
High Traffic	2 min ago
All Areas	13 min ago
Location History	

Manage Device

Change Name

Location Info

Latitude	37.71826
Longitude	-122.50485
Altitude	0.0
Accuracy	0.0 m
Time	14.06.25, 11:02:26
Status	High Traffic
Own Report	false

(a) Devices List (b) Location Report View (c) Raw Location Data

Figure 3: Screenshots of the Spot app on iOS, allowing the use of Google FMDN trackers with an Apple smartphone [61].

5 GoogleFindMyTools and the Spot App

To gain a better understanding of the FMDN and to verify our results from the functionality analysis, we reimplemented the client side of the network and also created an app we named *Spot*.

5.1 Design

Initially, we implemented the location retrieval and decryption process using Python and open-sourced it in the *GoogleFindMy-Tools* repository [15]. Our scripts enable a closer inspection of the network since they are not bound to the client-side aggregation from Section 4.4.5. All "High Traffic" location reports can be retrieved in their raw form as soon as two distinct finder devices shared a location report for the same tracker (see Section 4.3.1). In addition, we implemented the ability to register new FMDN trackers on Google's server as described in Section 4.1.1 and released firmware for ESP32-based devices that allows them to be used as custom-built FMDN trackers [15].

Furthermore, we ported the Python code to Swift, making it possible to use the FMDN with an iPhone. This also benefited our network evaluation presented in Section 9, as our iOS app *Spot* allows for continuous reception of location updates in the background and stores the location history. Figure 3 shows three screenshots of the app.

5.2 Implementation

The Python implementation uses the `firebase-messaging` [63] library to set up the FCM endpoint required for receiving location reports (see Section 4.4.2). Furthermore, we use `gpsoauth` [59] to retrieve the OAuth2 tokens needed for interactions with the server. We ported these libraries to Swift to use the same procedure for our iOS app.

Custom Trackers. To make it possible to use custom trackers with the FMDN, we chose a simple approach. We did not implement any of the Fast Pair procedures described in Section 4.1.1. In addition, we decided not to use the advertisement generation detailed in Section 4.1.4. Instead, our firmware keeps a static advertisement and MAC address. Our code generates a new random EIK and calculates the first advertisement (time counter = 0) as described in Section 4.1.4. This advertisement is then statically written to the ESP32. In the registration process from Section 4.1.1, we duplicate this advertisement for all announced Truncated EIDs. If we receive a report, we always use the time counter of 0 to decrypt the location report. As a result, we can now deploy fake trackers to receive location reports. A drawback is that the Truncated EIDs must be announced regularly to Google's server to continue receiving location reports. The server only allows 4 days of announced Truncated EIDs in advance. Therefore, our code regularly announces new Truncated EIDs, similar to Google's FMD app.

6 Security & Privacy Analysis

For **RQ1**, we evaluated the security and privacy of the network. We analyzed the network structure, focusing on the robustness of client-side encryption, the handling of locations, metadata, anti-tracking measures, as well as the robustness of the server-side interfaces.

6.1 Cryptography

The end-to-end location encryption uses the NIST P-160R1 curve by default (see Section 4.1.5). We believe this curve was chosen due to the limited size of Bluetooth advertisements. A larger 192-bit curve would require at least 24 bytes, which is 4 bytes more than for the NIST P-160R1 curve. However, the FMDN Bluetooth frame format only permits 2 more bytes, making the 160-bit curve the maximum that can be used for regular Bluetooth advertisements. The use of NIST curves, such as the P-160R1 curve, is discouraged by some researchers due to the lack of rigidity, i.e., due to its unknown parameter origin [10]. In addition, this curve has a key length of 160 bits. Usually, key lengths of at least 250 bits for ECDH are recommended [14], which makes P-160R1 a weak curve.

6.1.1 *The Alternative NIST P-256R1 Curve.*

As mentioned in Section 4.1.5, the NIST P-256R1 curve can be used alternatively. Due to the larger key size, trackers using this curve must use Bluetooth Extended Advertising, which is only supported in Bluetooth 5.0 or newer. Similar to the P-160R1 curve, the P-256R1 curve lacks rigidity [10]. Alternative curves such as Curve25519, which have the same key length, offer more transparency in their parameter selection [9]. The security level of 128 bits is acceptable by today's standards [14] and significantly increases security compared to the P-160R1 curve.

We did not find this curve implemented in any of the tested trackers (see Table 2). The only device we found using P-256R1 was Sony's WH-1000XM5 headphones. The reason many tracker manufacturers may have decided not to use this curve is compatibility, since not all FMDN-compatible Android devices support Extended Advertising, such as the Galaxy S9+ [51].

6.1.2 *End-to-End Encryption Process.*

The end-to-end location encryption technique from Section 4.1.5 is an ECDH key exchange. We did not find any structural problems with this cryptographic process: The random ephemeral key pair d_e is generated using a secure random source (Java's `SecureRandom`). A shared secret is calculated using ECDH. As described above, this is secure if the elliptic curve's Computational Diffie-Hellman assumption holds. No practical attacks against ECDH key exchange targeting the NIST P-160R1 or the NIST P-256R1 curve are publicly known.

The AES-EAX key k is derived using HKDF-SHA256, which is considered a secure hash algorithm [14]. AES-EAX is considered a secure mode of encryption [5]. No attacks are known that would compromise AES-EAX security. Furthermore, a 16-byte nonce is used for AES-EAX, consisting of 8 bytes from P_{EID}'s x-value and 8 bytes from P_e's x-value. This is secure for AES-EAX, which supports nonces of any length but typically uses 16 bytes [53]. Since the same EID is reused for multiple encryptions, 8 bytes remain fixed. However, the ephemeral public key changes with each encryption, ensuring that the nonce remains unique.

6.2 Privacy Protections

In this section, we detail how the privacy protection mechanisms of Google's FMDN work.

6.2.1 Key Diversification. During our investigation, we did not find any structural issues with the EID calculation process as described in Section 4.1.4. In essence, the advertisement is secured twice - by 1) encrypting the time-counter-based byte array with the EIK, interpreting the result as an elliptic curve private key, and 2) generating the associated public key. Since the EIK is generated randomly, the result of using AES-ECB as a pseudo-random function will also produce a random value. After constructing the public key, the result will still be random. Therefore, the advertisement is random as well and cannot be used to retrieve the EIK, nor can it be used to track the tracker for more than one key rotation cycle.

6.2.2 Device Tracking. Due to the time counter of the EID, both the MAC and the EID change at the same time every 1024 seconds on average. A slight randomization is implemented to avoid tracking over long periods based on the key change interval. The implementation of the randomization is up to the tracker manufacturer. Google does not enforce a specific implementation, but recommends that the next rotation time should be set to a multiple of the rotation period plus 1 to 204 seconds [25]. We found that all tested trackers implement this randomization appropriately.

6.2.3 Restoring End-to-End Encrypted Keys. When Google's FMD app is set up on a new device, the end-to-end encrypted keys, such as the EIK, are not directly available. Instead, they need to be retrieved from Google's server. We found that the key recovery process is very similar to the process implemented for Android backups researched by NCC Group [30]. The user is asked to enter the unlock PIN code of a device that previously had access to the end-to-end encryption keys. In our testing, we found that instead of SHA-256 [30], Scrypt was used to hash the PIN/Lock Screen Knowledge Factor (LSKF). The key retrieval procedure is implemented in a way that Google cannot learn the plaintext LSKF, and brute-force attacks are prevented.

6.2.4 Advertisement Collision. Requesting end-to-end encrypted location data is only possible for previously announced Truncated EIDs that are linked to trackers registered to a Google account. Even though it is possible to announce arbitrary Truncated EIDs to the server when registering a new tracker, it is not possible to retrieve encrypted location reports from other accounts. In the case of an advertisement collision, i.e., two or more Google accounts register the same Truncated EID for a similar time period, neither account will receive location reports.

6.2.5 Location Report Spoofing. As mentioned in Section 4.2.2, DroidGuard data is required to send location reports for foreign trackers. This data consists of a base64-encoded string generated from the location report payload with DroidGuard. DroidGuard evaluates whether a device is secure and has not been tampered with, detecting modifications such as root access, an unlocked bootloader, or a Custom ROM [64]. DroidGuard checks certain system parameters to generate an encrypted, obfuscated string. This string is sent to Google's server, which uses it to evaluate the security level of the device. We found that foreign location reports sent from devices with an unlocked bootloader are discarded. We were unsuccessful in sending location reports on rooted devices using tools such as PlayIntegrityFix [18] and TrickyStore [1], which spoof certain device parameters to circumvent DroidGuard's root detection. In addition, bypassing DroidGuard by sending empty DroidGuard strings or sending foreign location reports to the `UploadOwnerScans` API is not possible. We conclude that Google's safety measures to prevent spoofed location reports on-device are adequate. However, note that spoofing location reports is still possible by duplicating a Bluetooth signal near an unmodified Android phone. This could enable DoS attacks, as detailed in [33].

6.3 Privacy Issues

Although Google has implemented many privacy protection measures, we found several issues, which we detail in this section.

6.3.1 Tracker Sharing. When a tracker is shared, the EIK and the Fast Pair Account Key are exchanged between the owner of the tracker and the person with whom it is shared. After sharing is completed, both parties possess the same EIK and Account Key. To protect the privacy of tracker owners, location reports sent before the sharing procedure are inaccessible to invitees. Invitees will only be shown a location if the tracker is nearby, the network reports a new location, or the owner or other invitees report a location for the tracker. Similarly, if a person the tracker was shared with is removed from sharing, location updates are no longer available. In this case, the server also indicates that the EIK should be rotated as soon as possible. The network requests for issuing a sharing invitation are sufficiently safeguarded. Issuing a sharing request can only be executed by the owner. If an invitee attempts to issue an invitation in this way, the server responds with an error and does not update its state.

The key exchange protocol is based on ECDH. As mentioned above, this method is considered secure. However, again the NIST P-256 curve is used, leading to the same rigidity problem detailed in Section 6.1.1. Also, Google still has to be trusted to a certain degree. Since the exchange of ECDH keys takes place via Google's server, Google could carry out a MitM attack by exchanging the invitee's public key with Google's own public key. This would allow Google to decrypt the EIK and thus gain access to the locations of the tracker's owner. In theory, MitM attacks such as this one should be avoided by comparing the confirmation PIN, which is a hash of the public keys and shared secret displayed on the owner's and invitee's phones. However, in practice, comparing confirmation PIN codes is usually not done by people due to lack of understanding, automation, and low usability [57].

6.3.2 Linking Location Reports. Google claims that location reports cannot be linked to the account they were sent from [11]. Although we couldn't fully verify this claim, it sounds plausible since authentication for finder reports (X-Goog-Spatula authentication header) differs from regular FMDN API requests (OAuth2 token). Nevertheless, we don't know exactly how the X-Goog-Spatula header is generated. However, *owners* of FMDN trackers could still be targeted even if location reports cannot be directly linked to a user. As mentioned in Section 4.2.2, only latitude, longitude, and altitude are end-to-end encrypted. Further metadata, such as the frequency of location uploads, the timestamp of location reports, and the accuracy, can be read by Google. In addition, Google knows which advertisements belong to which Google account. Information such as accuracy (low, e.g., walking; high, e.g., driving) and frequency of location reports (low, e.g., at home; high, e.g., in the city) and their timestamps could be used to determine likely locations, especially if an owner is specifically targeted.

Also, Google can still infer information through linking. If a user uploads location reports for different trackers with only a minimal time difference, Google knows which trackers have likely been at the same location at the same time. Linking several reports is possible because the `X-Goog-Spatula` header is reused for multiple location reports. We verified this by inspecting the network calls of the Galaxy S5 from Table 1. Since Google knows which report belongs to which tracker, it can also identify the owner of the tracker. If one can now deanonymize the location of one of the trackers, e.g., by recording BLE advertisements at a demonstration, multiple follow-up attacks are possible. A law enforcement agency could send the EIDs to Google and request the names of the owners. As Truncated EIDs are always uploaded to the Google cloud, Google can identify the owner of the tracker. In addition, it can infer which other people have been at or near the demonstration at the same time by linking the reports made by the same finder device around the same time. This attack must not only be linked to a specific event like a demonstration. It also allows Google to build social graphs of all users in the FMDN. Therefore, even with flight mode, users could be deanonymized if they own a tracker or if their phone is part of the Google FMDN and also sends FMDN advertisements.

6.3.3 Semantic Locations. In addition, smart home products can leak location information. If products such as the Google Home Mini are connected to the owner's Google Account, location reports may not include an encrypted location, but an unencrypted string such as "Alice's Home Mini" or "Alice's Home". This would enable Google to know when users are currently home or at other locations equipped with Smart Home products, opposing Google's claim that locations are always end-to-end encrypted. We used a Google Nest Audio with our Python application from Section 5.2 to verify that Semantic Locations are stored without using end-to-end encryption.

6.4 Bypassing Server Restrictions

During our testing, we discovered that the server-side restrictions from Section 4.3.4 (rate limiting, throttling) are implemented per registered tracker. This makes it possible to circumvent those restrictions by registering a new tracker, and only sending the first Truncated EID of the tracker to Google's server. Afterward, register another tracker, but only announce the second Truncated EID to the server, and continue this process for every new advertisement of the tracker. As a result, rate limiting and throttling are reset whenever the tracker's advertisement rotates and a new tracker is registered, i.e., every 1024 seconds. This guarantees new location updates, provided that finder devices are near the tracker. We successfully implemented this approach using a modified version of our Python program from Section 5.2.

6.5 Security Issues

In addition to the privacy issues, we also found some minor security issues during our investigation.

6.5.1 Location Report Denial of Service. As mentioned in Section 6.2.4, the server will handle advertisement collisions by dropping location reports. This can be abused to effectively stop all surrounding trackers from being able to be located by their owners. This attack would be executed as follows: An adversary could consistently monitor surrounding FMDN advertisements. If a new advertisement is detected or changed, the adversary could register a new (fake) tracker and spoof the request in such a way that the detected advertisement is included in the announced Truncated EIDs for the registered device. As Google's server does not allow duplicate EIDs, finder reports from surrounding Android devices will be dropped. The location of those trackers will not be visible to the owner. We implemented this approach using a modified version of our Python program from Section 5.2. We queried a tracker on a different Google account and verified that we would not receive any location update for this tracker, even though FMDN Android devices were nearby.

6.5.2 High Traffic Definition. As mentioned in Section 2.2, the FMDN operates in the "High Traffic" mode by default. In the FMDN settings, the feature's description states that "location info from your device is only used if others in the network also detect the item". We found that the description of the "High Traffic" setting is misleading in two ways.

First, one could believe that locations are only uploaded if other Android finder devices are also in proximity. However, locations are uploaded all the time, regardless of whether other devices also discovered the item. We verified this by inspecting the network calls of the Galaxy S5 which was not nearby any other Android device. By using the unencrypted timestamps and the contributor type from the location reports, the server decides if location reports should be forwarded to the owner of the tracker or if they should be dropped. Even though sent locations are end-to-end encrypted, certain attacks are still possible, such as the linkage attack mentioned in Section 6.3.2. As a result, the description of the "High Traffic" mode may lead to a false sense of security among users.

Second, we observed that the network does not differentiate between unique Google accounts when aggregating "High Traffic" reports. Such a location report can be generated by positioning multiple devices near the tracker, all logged into the same Google account. We tested this by positioning the four unrooted devices with the same Google account near a tracker registered to a different Google account. The unrooted devices were all set to the "High Traffic" contributor mode. Afterward, we could retrieve a location by the network. One could argue that this contradicts Google's claim in the settings that "*others* in the network also [have to] detect the item".

6.5.3 Re-Registering a Shared Tracker. As mentioned in Section 6.3.1, the FMD app should rotate the EIK after sharing with a person was stopped. However, we found that it often fails to do so, e.g., because the tracker is not nearby, the Bluetooth connection timed out, etc. This allows an attacker who stored the EIK and account key to execute Bluetooth actions on the tracker that require those keys. Most critically, the attacker can change the EIK itself. This operation requires the device that requested the EIK change to be in possession of the old EIK and the current account key [25]. Both keys are available to the attacker from the previous sharing operation. If the attacker proceeds to rewrite the EIK, the advertisements also change. The attacker can then register a new tracker in their account, announcing the new advertisements. This makes it possible to continue receiving location reports, even though the attacker is no longer invited. We were able to reproduce this issue with a Samsung Galaxy S21 Ultra running Android 14 (Patch 2025.01) and the FMD app 3.1.205-1. We used the Python implementation from Section 5.2 to re-register the tracker on Google's server.

6.6 Transforming Any Device Into a Tracker

Chen et al. [17] demonstrated that they have been able to transform a Linux computer, a Windows computer, and an Android smartphone into an Apple Find My tracker, allowing them to accurately locate the device and potentially track the owner of this device. This attack is based on the notion that a benign-looking app has code-execution rights on the system and access to BLE. Then, the app starts sending BLE advertisements following Apple's Find My protocol [33]. As Apple's Find My protocol also uses the MAC address, they must find a matching public key for the current device's MAC address.

However, Google's FMDN only uses the content of the advertisement to send the EID, which is then used by nearby finder devices to encrypt location reports. Therefore, a similar attack as presented by [17] can be launched with Google's FMDN. Here, the adversary registers a new tracker on their own account. Then, they configure the app on the victim's machine to send BLE advertisements with a EID belonging to the tracker. The adversary can keep MAC address randomization enabled and can set UT mode in the advertisement to off. Now, the adversary can track the victim based on the malicious app installed on one of their systems. This attack could work on any modern operating system (OS) with Bluetooth support. We successfully implemented the attack with a normal Android app (Android 15) and a Linux program (Ubuntu 25.04). macOS, iOS, and Windows do not allow sending BLE advertisements containing service data without root-level access [3, 45, 65]. Similar to the work of Chen et al., a limitation of this attack is that some OS use advanced permission systems that require the user to give consent if an app tries to access BLE for the first time.

7 Unwanted Tracking Protection

In this section, we present Google's protection mechanisms against UT. To answer **RQ2**, we also show several methods that we found to circumvent them. For all of the following attacks, the adversary would hide a tracker that they own in the belongings of their victim. Through the OFN, the adversary receives regular updates on where their victim is residing.

7.1 Unwanted Tracking Protection Mode

The most obvious measure against UT is the UT mode. As mentioned before, UT mode slightly changes the behavior of the tracker: the tracker will rotate its MAC address only every 24 hours, not every 1024 seconds and the tracker advertises a byte indicating that UT mode is enabled. Tracking detection implementations in Android, iOS, and the AirGuard app [31] can then detect that the same tracker is following the user and warn them. A unique aspect of UT mode in the FMDN is that the determination of whether UT mode should be enabled is decided on the server, not on trackers (see Section 4.1.6). This approach makes it possible to update the behavior of UT mode without needing firmware updates for trackers. However, this approach also comes with several drawbacks, which we will present in the following.

7.1.1 Adversary Models. We define two adversary models for the next attacks. Both adversaries share the same goal: they want to ensure that their tracker may not be detected by current tracking detection methods [4, 27, 31]. This will allow them to continue tracking their victim and reduce the risk that the tracker will be found by the victim.

A1: This adversary has the capability to program simple software. They purchase an off-the-shelf tracker and try to modify the tracker's behavior through their programming skills. They do not know about hardware development and cannot build custom tracker devices.

A2: This adversary also has the capability to program Bluetooth hardware, such as an ESP32 or an nRF5 microcontroller. These microcontrollers have a similar size to a tracker, and many of these trackers are based on an nRF5 chip [54]. They use their capabilities to build custom trackers that do not have to adhere to Google's specifications for the FMDN [25].

7.1.2 Keeping Unwanted Tracking Mode Disabled. Attacker **A1** may keep UT mode disabled, even when they are *not near* their tracker. As mentioned in Section 4.1.6, UT mode is initiated by the server if the tracker has been separated from its owner for at least 30 minutes, and a finder device has reported its location. The determination of how long the tracker has been separated from its owner is based on the last owner report. If the owner sends an owner report with any location to Google's server every 30 minutes or less, UT mode is never activated. The attacker must not need to be nearby the tracker to send these arbitrary location updates. We successfully tested this procedure for more than 48 hours with a modified version of our Python implementation and a Pebblebee Clip tracker. The phone of the owner was turned off, and other

FMDN-registered Android devices were nearby. In this scenario, **A1** continues to receive location updates from finder devices for their tracker. If they hide the tracker in a victim's belongings, the victim will never receive a notification from one of the known tracking detection systems.

The best solution to mitigate this issue would be a firmware update for FMDN trackers. This update would need to implement a timer that activates UT mode automatically without requiring a finder device to activate it, as implemented by Apple's Find My network [33]. Another possibility would be that the server randomly tries to enable UT mode on any tracker. If the owner is actually nearby, the owner's Android device would immediately notice that UT mode is enabled and disable it. However, if the owner's report was spoofed, UT mode stays enabled.

7.1.3 Spoofing the Unwanted Tracking Key. An even more robust way for adversary **A1** to avoid the UT mode is by modifying the tracker registration process. When the tracker is registered on the server, the UT key is sent, as detailed in Section 4.1.2. If this value is set to a random, invalid value, the tracker will never transition to UT mode, even if finder devices try to enable it with a Bluetooth request. As detailed in [25], the tracker only accepts the request and enables UT mode if the correct key is provided. Similar to the previous attack, **A1** continues to receive location updates, while their victim will never receive a notification for the malicious tracker. We verified this attack using a modified version of our Python implementation, a Pebblebee Clip, and the same setup described for the last attack.

7.1.4 Keeping the MAC Address Rotating. The two approaches listed above can be performed with an off-the-shelf, unmodified FMDN tracker. However, adversary **A2** has stronger capabilities and can build custom trackers that do not adhere to Google's specification [25]. Tracking protection mechanisms, such as Google's and Apple's implementations, detect trackers by their MAC addresses. If the MAC address does not stay static but keeps rotating when UT mode is enabled, those algorithms cannot identify the tracker over longer periods. The default MAC address rotation interval is 1024 seconds, which is too short to trigger a tracking notification in any of the anti-tracking algorithms [4, 27, 31]. Therefore, implementing a custom tracker that continuously rotates its MAC address would be undetectable. We successfully tested this attack by moving with an ESP32 running a modified version of our firmware from Section 5.2 and a Galaxy S21 Ultra with Android 14 (Patch 2025.02) and an iPhone Xs running iOS 18.3.

7.1.5 Spoofing the Unwanted Tracking Mode Bit. Another approach for **A2** is to always advertise with UT mode disabled. This works since tracking detection algorithms on Android and iOS will only trigger a notification if the tracker has UT mode enabled. If the mode is constantly disabled in the advertisement, the tracker will never be considered for delivering a notification. We validated this attack also by moving with an ESP32 running a modified version of our firmware from Section 5.2 and the devices from the previous attack.

7.2 Detecting Unwanted Location Trackers

Detecting Unwanted Location Trackers (DULT) is a proposed IETF standard with the collaboration of Apple and Google [38]. The standard defines specific BLE advertisements that trackers should send to be identifiable by nearby smartphones. Trackers of any OFN should follow the same specification. To allow detection of UT, the tracker should implement UT mode and certain BLE actions, like playing a sound, must be possible while the tracker is in UT mode. For Google's FMDN, implementing the DULT standard is mandatory [25]. We found that the actions demanded by DULT, such as looking up more information about the owner of a tracker, are implemented and also work on an iPhone.

7.2.1 Unwanted Tracking Sound Playback. The DULT paper [38] also mandates that trackers should play a sound if they are in UT mode for several hours to warn potential stalking victims. Similar to UT mode, we observed that the implementation of this feature is server-dependent. As described in Listing 4.2.3, the response of a location report upload may also contain the Ring Key. If the field is present, the finder device will try to connect to the tracker to play a sound. Therefore, this server-side implementation is also prone to the issues discussed in Section 7.1. Additionally, since alerts by the sound maker require reporting by an Android device, iPhone users are never alerted if no Android device is nearby.

7.3 Server-Side Measures Against Unwanted Tracking

In addition to UT mode, Google's FMDN also uses a variety of additional server-side measures to mitigate the risk of abuse. What we consider positive is rate limiting, explained in Section 4.3.4. Rate limiting prevents the Android device of a stalked individual from continuously sending reports for the tracker the individual is being tracked with. As mentioned in Section 4.3.4, a single Android device can only report every few hours. This limits the network reports available for an attacker to those of non-victim devices, making abuse of the network harder. In addition to rate limiting, we also believe that throttling, detailed in Section 4.3.4, is positive for anti-abuse. Since this feature restricts the location history and frequency at which an attacker can request a new location from the network, it makes using Google's network less attractive for stalkers. However, note that more sophisticated **A1** attackers can still easily circumvent all of these protection features, as mentioned in Section 6.4.

7.4 Notable Locations

As mentioned in Section 4.2, Android devices will not upload finder reports if they are currently near the user's home location. This feature should prevent leaking home locations of stalking victims. Even if the intention behind this feature is good, it has a few weak points. First, the home location needs to be manually added to the Google Account, limiting the users who benefit from this feature. Second, the radius of 100 m is rather small. If the victim's Android device sends a report when moving in/out of the radius of the home location, an attacker could go to the last known location and use the Bluetooth signal to precisely find the home location of the victim later. We therefore demand a larger radius of at least 500 m.

Furthermore, it is unlikely that the victim's phone is the only device near the victim's home. Neighbors or passing cars are also able to pick up the signal and report to the attacker. Therefore, the feature may create a false sense of security and privacy.

8 Responsible Disclosure

We responsibly disclosed all issues described in Section 6 and 7 to Google. According to the procedure used by Google's Project Zero team, we allowed Google 90 days to implement fixes for the issues before publishing. We reported issues through the official Google Bug Hunters platform [**?**]. Each vulnerability report contained a detailed technical description, an assessment of the impact, reproducible steps, and a proof of concept with code (where applicable).

Despite acknowledgment, most of the problems remain unresolved as of mid June 2025. Some reports were accepted but have yet to be addressed. The issue *Transforming Any Device Into a Tracker* was accepted on 06.05.25, but is still pending a fix. Similarly, *Keeping Unwanted Tracking Mode Disabled* was acknowledged on 02.04.25, yet no mitigation has been implemented. In the case of *Spoofing the Unwanted Tracking Key*, Google closed the bug report without providing a fix, stating that they are already aware of the issue.

Other issues were closed after being classified as either intended behavior or infeasible to address. The report on *High Traffic Definition* was initially accepted by Google, but was then closed on 25.01.25. Google stated that it is infeasible to fix since they claim that location reports cannot be tied to individual Google accounts. Likewise, the *Re-Registering a Shared Tracker* report was closed on 06.03.25 as "infeasible". Google claims that the issue closely resembles factory resetting a tracker. However, we believe that this attack is more applicable than a tracker reset, as it only requires a BLE connection instead of physical access, and execution of the attack does not alert the victim, unlike a tracker reset which triggers a sound on the device. Reports about firmware-based attacks, such as *Keeping the MAC Address Rotating* and *Spoofing the Unwanted Tracking Mode Bit*, were also marked "infeasible" on 03.04.25. These issues were described by Google as "known risks". However, they told us that they might introduce additional ways based on server-side signals for preventing malicious use of the FMDN.

Several additional reports were acknowledged by Google as abuse risks but are still under investigation. This includes *Linking Location Reports*, which was last updated on 02.05.25 without a fix. Similarly, the issue of Section *Semantic Locations* was also last updated on 02.05.25, and no fix has been implemented yet. The issue *Bypassing Server Restrictions* received its most recent update on 02.05.25, where it was also recognized as an abuse risk, but is still unresolved. The *Location Report DoS* was last updated on 05.05.25, again recognized as an abuse risk, but not fixed yet.

9 Evaluation

For **RQ3**, we performed a quantitative evaluation of Google's FMDN, comparing it to Apple's Find My. This has two goals: first, from an attacker's perspective, it helps us understand how well Google's FMDN could be abused for stalking. Second, from a user's perspective, it provides evidence of how reliable the location reports are for finding lost devices.

9.1 Methodology

For this evaluation, we compare the number of reports and their accuracy with a recorded GNSS trace. We use two ESP32-based microcontrollers of the same model. One uses OpenHaystack [32], software that makes it possible to use custom devices such as an ESP32 with Apple's Find My. The other ESP32 uses our custom firmware from Section 5.2, which advertises FMDN frames. To avoid bias, we performed the experiments with both ESP32's with the same BLE advertisement parameters. We set both devices to the highest possible TX Power (9 dBm) and an advertisement frequency of 20 ms. These settings are different from normal trackers but allow us to compare the network performance under identical conditions.

9.1.1 Tracker Data Collection. As mentioned in Section 4.3, Google's FMDN discards network reports if newer ones are available. Therefore, we use a simple polling-based approach implemented in our Spot app, querying location data for the ESP32 device every 30 seconds. The app logs when it receives a new location report (the publish timestamp) and logs the location report itself. Note that due to this polling-based approach, the publish timestamp might not be precise to the second. Unlike Apple's server, Google's server does not indicate when the finder device found the tracker. In addition to Spot, we used the iOS app "Open GPX Tracker" [36] on an iPhone Xs to record the route we traveled during our measurements. The app records the GPS location, including timestamps, which we used to determine the accuracy of the network. We used linear, second-based interpolation with a rolling mean with a window size of 10 to smooth out the GPS Exchange Format (GPX) track. We disabled Apple's Find My and Google's FMDN on our personal devices to avoid self-reporting during the experiment.

To test the performance of the networks, we traveled with our testing setup to examine how well the respective network can reconstruct the route. The first route was a 50-minute trip using public transportation. The route included a 5-minute ride with a tram through a medium-sized city (>100,000 inhabitants), and a 45-minute ride with regional trains. The second route was a 1-hour walk through a large city (>500,000 inhabitants). We performed the evaluation for both networks at the same time. Both routes were recorded during midday.

9.1.2 Parameters. To compare our datasets, we calculate the following parameters: (1) Number of reports (2) Distance to actual location (3) Advertised accuracy (4) Time between availability and timestamp of location report. For the number of reports, we only consider reports received within the timespan during which the GPX recording was performed. Note that due to the polling-based approach for FMDN, certain reports may be dropped by the server (see Section 4.3). The distance from the actual location compares the GPX location data with the reports received from both networks. The distance is compared to the GPS location with the same timestamp (precise to the second). The advertised accuracy is the accuracy sent with the location report. In theory, this should align with the previous metric. The time between availability and the timestamp of the report measures the delay of the network. Previous research has shown that Apple's Find My network delays network reports [33, 58].

Table 5: Comparison of the FMDN and Apple's network for Route 1 (Public Transport).

Metric	Google	Apple
Number of Locations	166	906
Average Accuracy (actual)	1215.45 m	713.53 m
Median Accuracy (actual)	121.79 m	57.45 m
Average Accuracy (advertised)	187.20 m	132.22 m
Median Accuracy (advertised)	148.07 m	119.00 m
Average Location Date to Retrieval Date	72.95 s	1246.75 s
Max Time Without Location	273.00 s	134.00 s

Table 6: Comparison of the FMDN and Apple's network for Route 2 (Walking).

Metric	Google	Apple
Number of Locations	242	1218
Average Accuracy (actual)	130.74 m	63.02 m
Median Accuracy (actual)	49.08 m	33.13 m
Average Accuracy (advertised)	122.04 m	98.97 m
Median Accuracy (advertised)	93.97 m	96.00 m
Average Location Date to Retrieval Date	78.05 s	1244.05 s
Max Time Without Location	429.00 s	95.00 s

9.2 Results

In the following, we present the evaluated data for the two routes.

9.2.1 Route 1 (Public Transport). The results of this measurement are depicted in Table 5. With over 906 locations, Apple's Find My network provided a significantly higher number compared to Google, while also providing more accurate locations. In both networks, the actual accuracy is worse than the advertised accuracy. The delay for the FMDN is significantly lower compared to Apple's Find My.

9.2.2 Route 2 (Walking). The results of this measurement are depicted in Table 6. Again, Apple's Find My network provided a higher number of reports compared to Google, while also providing more accurate locations. The accuracy for both networks is better than in the public transport route. Compared to the public transport route, the delay is similar for both networks, and Google's network again has a significantly lower delay than Apple's Find My.

9.3 Discussion

Our evaluation shows that Apple's Find My network delivers 5–6 times more locations than FMDN, regardless of the route. In addition, the locations of Apple's network were consistently more accurate. However, Google's network is significantly quicker at delivering up-to-date locations. In comparison, Apple's location reports are on average significantly more delayed. This aligns with the findings of Heinrich et al. [33], who discovered that Apple devices often accumulate reports instead of sending them directly. Since Android devices send finder reports immediately when the screen of the Android device is turned on, this gives Google's network an advantage in this metric.

9.3.1 Device vs. Data Distribution. In the evaluated region, Android devices have a market share of 66%, whereas Apple devices account for 33.5% [62]. This proportion is not reflected in our test data. One reason for this could be the polling-based approach to querying locations of Google's FMDN. If many locations were detected between two location query requests, some may have been dropped by the server, as mentioned in Section 9.1. In addition, rate limiting and throttling (see Section 4.3) reduce the network's quality. Finally, we believe that FMDN is not yet activated on every supported Android device. We observed that the network was often not enabled when setting up new Android devices.

10 Conclusions

To the best of our knowledge, we performed the first public in-depth evaluation of Google's new FMDN. We analyzed the security and privacy properties of the network. The network has a solid foundation and uses end-to-end encryption for all location reports from finder devices (**RQ1**). Furthermore, Google implemented rate limiting and throttling, which actively weaken their network's performance but also result in a weaker incentive for misuse. Nevertheless, we found several issues that range from a linking attack that allows Google to create social graphs of network users to transforming any Bluetooth device into a Google FMDN tracker, potentially allowing an adversary to track them. We also analyzed the protection mechanisms against unwanted tracking and found four different approaches to hiding a tracker from current tracking detection implementations (**RQ2**).

At last, we compared the performance of the new FMD network with Apple's Find My network in a small-scale study. The first results show that Google's network achieves a median accuracy between 49.08 m and 121.79 m. However, Apple's network has been more accurate and has also exhibited more location reports (**RQ3**).

We believe that Google has taken a promising approach by focusing on privacy and implementing UT protection measures from the beginning. As Google has more insights on the server than Apple, they may hopefully improve their measures even further in the future.

Acknowledgments

We thank the anonymous reviewers and the editor for their constructive feedback, which has improved the final version of this paper. This work has been funded by the German Federal Ministry of Education and Research and the Hessian State Ministry for Higher Education, Research, and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE. We used AI-based text tools, such as Grammarly and GPT-4o, throughout this paper to enhance writing by correcting grammar errors and typographical errors.

References

[1] 5ec1cff. 2024. *TrickyStore.* https://github.com/5ec1cff/TrickyStore
[2] Apple. 2021. *Apple introduces AirTag.* https://www.apple.com/newsroom/2021/04/apple-introduces-airtag/
[3] Apple Inc. 2025. Apple CoreBluetooth - startAdvertising(_:). https://developer.apple.com/documentation/corebluetooth/cbperipheralmanager/startadvertising(_:)
[4] Apple Inc. 2025. Detecting Unwanted Trackers. https://support.apple.com/guide/personal-safety/detecting-unwanted-trackers-ips139b15fd9/web

[5] Mihir Bellare, Phillip Rogaway, and David Wagner. 2004. The EAX mode of operation. In *International Workshop on Fast Software Encryption*. Springer, 389–407.

[6] Alex Bellon, Alex Yen, and Pat Pannuto. 2023. TagAlong: Free, Wide-Area Data-Muling and Services. In *Proceedings of the 24th International Workshop on Mobile Computing Systems and Applications (HotMobile '23)*. Association for Computing Machinery, New York, NY, USA, 103–109. https://doi.org/10.1145/3572864.3580342

[7] Berkeley Varitronics Systems, Inc. 2025. BlueSleuth-Lite BLE Tag Detector. https://www.bvsystems.com/product/bluesleuth-lite-ble-tag-detector/

[8] Berkeley Varitronics Systems, Inc. 2025. BlueSleuth-Pro Bluetooth and BLE Device Locator. https://www.bvsystems.com/product/bluesleuth-pro-bluetooth-and-ble-device-locator/

[9] Daniel J Bernstein. 2006. Curve25519: new Diffie-Hellman speed records. In *Public Key Cryptography-PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, April 24-26, 2006. Proceedings 9*. Springer, 207–228.

[10] Daniel J. Bernstein and Tanja Lange. 2017. SafeCurves: Choosing Safe Curves for Elliptic-Curve Cryptography. https://safecurves.cr.yp.to.

[11] Google Security Blog. 2024. *How we built the new Find My Device network with user security and privacy in mind*. https://security.googleblog.com/2024/04/find-my-device-network-security-privacy-protections.html

[12] Bluetooth SIG. 2025. Assigned Numbers. https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Assigned_Numbers/out/en/Assigned_Numbers.pdf

[13] Jimmy Briggs and Christine Geeng. 2022. BLE-doubt: Smartphone-based Detection of Malicious Bluetooth Trackers. In *2022 IEEE Security and Privacy Workshops (SPW)*. 208–214. https://doi.org/10.1109/SPW54247.2022.9833870

[14] Cryptographic Mechanisms Bsi. 2020. Cryptographic Mechanisms: Recommendations and Key Lengths. *BSI—Technical Guideline* (2020).

[15] Leon Böttger. 2024. *GoogleFindMyTools*. https://github.com/leonboe1/GoogleFindMyTools

[16] Leon Böttger, Alexander Heinrich, and Matthias Hollick. 2024. *AirGuard for iOS: Tracking Protection*. https://github.com/seemoo-lab/AirGuard-iOS

[17] Junming Chen, Xiaoyue Ma, Qiang Zeng, and Lannan Luo. 2025. Tracking You from a Thousand Miles Away! https://cs.gmu.edu/~zeng/papers/2025-security-nrootgag.pdf

[18] chiteroman. 2024. *PlayIntegrityFix*. https://github.com/chiteroman/PlayIntegrityFix

[19] Samantha Cole. 2022. *Police Records Show Women Are Being Stalked With Apple AirTags Across the Country*. https://www.vice.com/en/article/y3vj3y/apple-airtags-police-reports-stalking-harassment

[20] Joseph Cox. 2024. Hacker Accesses Internal 'Tile' Tool That Provides Location Data to Cops. *404 Media* (June 2024). https://www.404media.co/hacker-accesses-internal-tile-tool-that-provides-location-data-to-cops/

[21] Android Developers. 2025. Android Studio. https://developer.android.com/studio Accessed: 2025-02-28.

[22] Chinmay Garg, Aravind Machiry, Andrea Continella, Christopher Kruegel, and Giovanni Vigna. 2021. Toward a Secure Crowdsourced Location Tracking System. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '21)*. Association for Computing Machinery, New York, NY, USA, 311–322. https://doi.org/10.1145/3448300.3467821 arXiv:2106.00217

[42]]googlebughunters Google. [n. d.]. Google Bug Hunters. https://bughunters.google.com/. Accessed: 2025-05-06.

[24] Google. 2025. Google Chrome. https://www.google.com/chrome/

[25] Google Developers. 2023. *Find My Device Network Accessory Specification v1.3*. https://developers.google.com/nearby/fast-pair/specifications/extensions/fmdn

[26] Google Developers. 2024. *Fast Pair Procedure*. https://developers.google.com/nearby/fast-pair/specifications/service/gatt

[27] Google LLC. 2025. Find Unknown Trackers - Android Help. https://support.google.com/android/answer/13658562?hl=en

[28] Google LLC. 2025. How Find My Device Protects Your Data - Help. https://support.google.com/product-documentation/answer/14796936?hl=en

[29] Max Granzow, Alexander Heinrich, Matthias Hollick, and Marco Zimmerling. 2024. Poster: Leveraging Apple's Find My Network for Large-Scale Distributed Sensing. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services (MOBISYS '24)*. Association for Computing Machinery, New York, NY, USA, 666–667. https://doi.org/10.1145/3643832.3661412

[30] NCC Group. 2018. *Android Cloud Backup/Restore*. Technical Report. https://web.archive.org/web/20221013203959/https://research.nccgroup.com/wp-content/uploads/2022/04/NCC_Group_Google_EncryptedBackup_2018-10-10_v1.0.pdf Archived version.

[31] Alexander Heinrich, Niklas Bittner, and Matthias Hollick. 2022. AirGuard–Protecting Android Users From Stalking Attacks By Apple Find My Devices. *arXiv preprint arXiv:2202.11813* (2022).

[32] Alexander Heinrich, Milan Stute, and Matthias Hollick. 2021. OpenHaystack: A Framework for Tracking Personal Bluetooth Devices via Apple's Massive Find My Network. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '21)*. Association for Computing Machinery, New York, NY, USA, 374–376. https://doi.org/10.1145/3448300.3468251

[33] Alexander Heinrich, Milan Stute, Tim Kornhuber, and Matthias Hollick. 2021. Who Can Find My Devices? Security and Privacy of Apple's Crowd-Sourced Bluetooth Location Tracking System. *Proceedings on Privacy Enhancing Technologies* 2021, 3 (July 2021), 227–245. https://doi.org/10.2478/popets-2021-0045

[34] Alexander Heinrich, Leon Würsching, and Matthias Hollick. 2024. Please Unstalk Me: Understanding Stalking with Bluetooth Trackers and Democratizing Anti-Stalking Protection. *Proceedings on Privacy Enhancing Technologies* (2024). https://doi.org/10.56553/popets-2024-0082

[35] HyunSeok Daniel Jang, Hazem Ibrahim, Rohail Asim, Matteo Varvello, and Yasir Zaki. 2025. A Tale of Three Location Trackers: AirTag, SmartTag, and Tile. https://doi.org/10.48550/arXiv.2501.17452 arXiv:2501.17452

[36] Juan Manuel Merlos. 2023. Open GPX Tracker. https://apps.apple.com/de/app/open-gpx-tracker/id984503772

[37] Simon Kurz. 2024. Simon Kurz' Blog. https://blog.simonkurz.de/posts/

[38] Brent Ledvina, David Lazarov, Ben Detwiler, and Siddika Parlak Polatkan. 2024. *Detecting Unwanted Location Trackers Accessory Protocol*. Internet-Draft draft-ietf-dult-accessory-protocol-00. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-dult-accessory-protocol/00/ Work in Progress.

[39] Ben Lovejoy. 2022. *AirTags with deactivated speakers being sold on eBay and Etsy; seller claims not for stalking*. https://9to5mac.com/2022/02/03/airtags-with-deactivated-speakers-being-sold/

[40] PortSwigger Ltd. 2025. Burp Suite. https://portswigger.net/burp

[41] Ryan Mac and Kashmir Hill. 2021. *Are Apple AirTags Being Used to Track People and Steal Cars?* https://www.nytimes.com/2021/12/30/technology/apple-airtags-tracking-stalking.html

[42]]FridaUnpinning masbog. [n. d.]. frida-android-unpinning-ssl. https://codeshare.frida.re/@masbog/frida-android-unpinning-ssl/

[43] Travis Mayberry, Erik-Oliver Blass, and Ellis Fenske. 2023. Blind My - An Improved Cryptographic Protocol to Prevent Stalking in Apple's Find My Network. *Proceedings on Privacy Enhancing Technologies* 2023, 1 (Jan. 2023), 85–97. https://doi.org/10.56553/popets-2023-0006

[44] Travis Mayberry, Ellis Fenske, Dane Brown, Jeremy Martin, Christine Fossaceca, Erik C. Rye, Sam Teplov, and Lucas Foppe. 2021. 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 (WPES '21)*. Association for Computing Machinery, New York, NY, USA, 181–186. https://doi.org/10.1145/3463676.3485616

[45] Microsoft. 2025. BluetoothLEAdvertisementPublisher Class (Windows.Devices.Bluetooth.Advertisement) - Windows Apps. https://learn.microsoft.com/en-us/uwp/api/windows.devices.bluetooth.advertisement.bluetoothleadvertisementpublisher?view=winrt-26100

[46] Katharina O. E. Müller, Louis Bienz, Bruno Rodrigues, Chao Feng, and Burkhard Stiller. 2023. HomeScout: Anti-Stalking Mobile App for Bluetooth Low Energy Devices. In *2023 IEEE 48th Conference on Local Computer Networks (LCN)*. 1–9. https://doi.org/10.1109/LCN58197.2023.10223406

[47] Youssef Noser. 2025. MagiskHluda. https://github.com/Exo1i/MagiskHluda

[48] NVISOsecurity. 2020. MagiskTrustUserCerts. https://github.com/NVISOsecurity/MagiskTrustUserCerts

[49] OpenMapTiles and OpenStreetMap. 2024. Basic GL Style using OpenMapTiles. https://openmaptiles.org/, https://www.openstreetmap.org/. Licensed under CC BY 4.0 (https://creativecommons.org/licenses/by/4.0/). Changes were made.

[50] Bob Pan and contributors. 2023. Dex2Jar. https://github.com/pxb1988/dex2jar

[51] Jack Price. 2019. *Not all Bluetooth 5-enabled smartphones are created equally, here's why*. https://www.xda-developers.com/check-bluetooth-5-all-features-supported/

[52] Frida Project. 2025. Frida. https://frida.re

[53] PyCryptodome. 2025. AES. https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html Accessed: 2025-03-01.

[54] Thomas Roth, Fabian Freyer, Matthias Hollick, and Jiska Classen. 2022. AirTag of the Clones: Shenanigans with Liberated Item Finders. In *2022 IEEE Security and Privacy Workshops (SPW)*. 301–311. https://doi.org/10.1109/SPW54247.2022.9833881

[55] Samsung Electronics Co., Ltd. 2023. Samsung SmartThings Find Rapidly Expands With Over 300 Million Nodes Helping to Locate Devices – Samsung Mobile Press. https://www.samsungmobilepress.com/press-releases/samsung-smartthings-find-rapidly-expands-with-over-300-million-nodes-helping-to-locate-devices

[56] Ben Schoon. 2024. *Android's Find My Device network settings start going live for some users*. https://9to5google.com/2024/04/03/android-find-my-device-network-live-early/

[57] Svenja Schröder, Markus Huber, David Wind, and Christoph Rottermanner. 2016. When SIGNAL hits the fan: On the usability and security of state-of-the-art secure mobile messaging. In *European Workshop on Usable Security. IEEE*. 1–7.

[58] Narmeen Shafqat, Nicole Gerzon, Maggie Van Nortwick, Victor Sun, Alan Mislove, and Aanjhan Ranganathan. 2023. Track You: A Deep Dive into Safety Alerts for Apple AirTags. *Proceedings on Privacy Enhancing Technologies* 2023, 4 (Oct. 2023), 132–148. https://doi.org/10.56553/popets-2023-0102

[59] Simon Weber. 2024. *gpsoauth.* https://github.com/simon-weber/gpsoauth
[60] Skylot and contributors. 2025. JADX. https://github.com/skylot/jadx
[61] Skymakers. 2025. MockUPhone. https://mockuphone.com/model/iphone-15-pro-max/. Licensed under CC BY 3.0 (https://creativecommons.org/licenses/by/3.0/). Changes were made..
[62] Statista. 2024. *Android vs iOS market share in smartphone sales in Germany.* https://www.statista.com/statistics/461900/android-vs-ios-market-share-in-smartphone-sales-germany/
[63] Steven B. 2024. *firebase-messaging.* https://github.com/sdb9696/firebase-messaging
[64] Romain Thomas. 2022. DroidGuard: A deep dive into SafetyNet. In *Symposium sur la sécurité des technologies de l'information et des communications (SSTIC).*
[65] Davide Toldo, Jiska Classen, and Matthias Hollick. 2020. Attaching InternalBlue to the proprietary macOS IOBluetooth framework. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (Linz, Austria) *(WiSec '20).* Association for Computing Machinery, New York, NY, USA, 328–330. https://doi.org/10.1145/3395351.3401697
[66] Leonardo Tonetto, Andrea Carrara, Aaron Yi Ding, and Jörg Ott. 2022. Where Is My Tag? Unveiling Alternative Uses of the Apple FindMy Service. In *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM).* 396–405. https://doi.org/10.1109/WoWMoM54355.2022.00059
[67] Connor Tumbleson and contributors. 2025. Apktool. https://github.com/iBotPeaches/Apktool
[68] K. Turk, Alice Hutchings, and A. Beresford. 2023. Can't Keep Them Away: The Failures of Anti-Stalking Protocols in Personal Item Tracking Devices. In *Security Protocols Workshop 2023.* https://www.semanticscholar.org/paper/Can%E2%80%99t-Keep-Them-Away%3A-The-Failures-of-Anti-Stalking-Turk-Hutchings/0756ebf656d4ca6d980d6c3943d6050f6c29863d?utm_source=alert_email&utm_content=AuthorCitation&utm_campaign=AlertEmails_DAILY&utm_term=LibraryFolder+AuthorCitation&email_index=0-0-0&utm_medium=22345888
[69] Mira Weller, Jiska Classen, Fabian Ullrich, Denis Waßmann, and Erik Tews. 2020. Lost and found: stopping Bluetooth finders from leaking private information. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks.* 184–194.
[70] Tingfeng Yu, James Henderson, Alwen Tiu, and Thomas Haines. 2024. Security and Privacy Analysis of Samsung's Crowd-Sourced Bluetooth Location Tracking System. In *33rd USENIX Security Symposium (USENIX Security 24).* USENIX Association, Philadelphia, PA, 5449–5466. https://www.usenix.org/conference/usenixsecurity24/presentation/yu-tingfeng

A API Requests and Endpoints

Spot API. The Spot API is used for managing owned trackers. The endpoint is spot-pa.googleapis.com/google.internal.spot.v1.SpotService, and it requires authentication with the Google Account using an OAuth2 Token with scope `googleapis.com/auth/spot`.

After registration, Android devices periodically (every 24 hours) send a request to `UploadPrecomputedPublicKeyIds` of the Spot API to upload the next 72 hours of precalculated Truncated EIDs to the server to pre-announce future advertisements of the tracker. Since the EIK is stored on-device, the smartphone does not need to have a connection to the tracker to refresh the Truncated EIDs.

Spot Report API. The endpoint for the Spot Report API is spot-pa.googleapis.com/google.internal.spot.v1.SpotReportingService. The `UploadScans` request of the Spot Report API is used to upload locations structured in the following Protobuf format:

```
message Location {
    sfixed32 latitude = 1;
    sfixed32 longitude = 2;
    int32 altitude = 3;
}
```

The Spot Report API is also used for the `UploadOwnerScans` API call, which does not require DroidGuard.

Nova API. The Nova API is used to access location reports with the FMD app. It uses the URL android.googleapis.com/nova. Similar to the Spot API, the Nova API requires an OAuth2 Token with the `android_device_manager` scope.

For querying what devices are paired with the user's Google account, the API request `nbe_list_devices` is used. The response will contain metadata for every tracker, such as the device ID (so-called *Canonic Device ID*) and the encrypted user secrets (such as the EIK). If the user wants to query location data for a tracker, the `nbe_execute_action` API call to the Nova API will be performed. This request contains the device ID, the last time the network mode was switched to "High Traffic", and the contributor type ("High Traffic" or "In All Areas"). The contributor type determines what locations are available; see Section 4.3. The timestamp is used to filter out all locations that were received by the server before the FMDN was enabled on the phone.

B Evaluation Maps

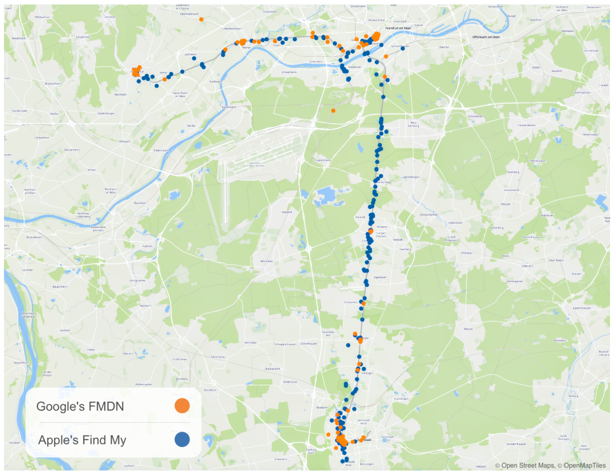


Figure 4: Route 1 (Public Transport) [49].

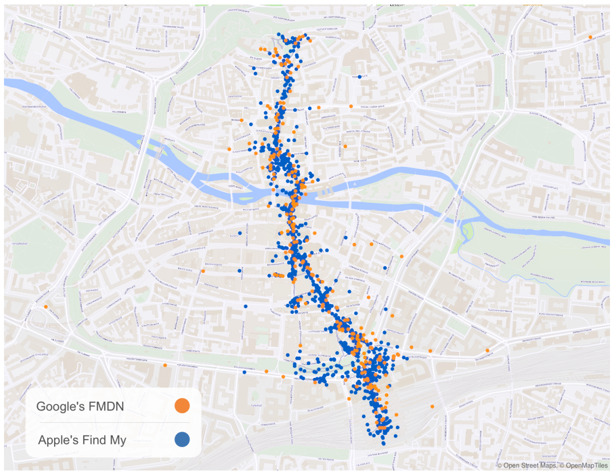


Figure 5: Route 2 (Walking) [49].

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009