Katholieke
Universiteit
Leuven

**Master of Engineering Science**
**Artificial Intelligence**

# PRIVACY IMPACT ASSESSMENT

Macless-Haystack

**Lili De Bruyn (r0848166)**
**Annemerel Bobbaers (r0897155)**
**Magnus De Maerteleire (r0889456)**
**Jeff Jambe (r0902205)**

Academic year 2025–2026

# Contents

# 1 Introduction

This report presents a Privacy Impact Assessment (PIA) for Macless-Haystack [1], an open-source project that enables users to access Apple's Find My network without requiring Apple hardware. As location tracking systems become increasingly prevalent, understanding the privacy implications of such technologies is crucial for both developers and end users.

The assessment examines how Macless-Haystack collects, processes, and stores location data, identifies potential privacy risks, and proposes technical solutions to mitigate these concerns. We analyze the system from multiple perspectives: the tracker devices themselves, the Macless-Haystack application and API, and Apple's Find My infrastructure that the project relies upon.

This report is structured in three main parts. First, we provide a detailed description of the application's functionality, stakeholders, and data collection practices. Second, we conduct a privacy impact assessment that includes a threat model, identification of privacy issues, and analysis of specific threats. Finally, we present recommendations for addressing the identified privacy concerns, with particular focus on technical implementations that can be tested and verified.

# 2 Application Description

## 2.1 Functionality of the Application

The project we are analyzing is called Macless-Haystack, an open-source implementation to use Apple's 'Find My' network without needing access to an Apple device. It builds upon the OpenHaystack [2] project, which reverse engineered the Apple 'Find My' protocol that enables tracking of Bluetooth devices without the need of an internet connection.

At a high level, the Find My trackers broadcast identifiers, which can be detected by nearby Apple devices. These apple devices then anonymously upload the location of the detected signal to iCloud. The owner can later retrieve the location of their device by requesting reports containing the trackers identifier.

The diagram above visually represents the four main stages of the application:

1. **Pair through initial setup**: The process begins with the owner pairing their device, this means generating a unique private-public key pair. These keys are programmed onto supported Bluetooth devices.
2. **Broadcast**: After setup, the tracker begins broadcasting Bluetooth advertisements that are based on its public key. Apple devices nearby act as a finder device and can pick up on these advertisements.
3. **Upload encrypted location reports**: When a finder device detects an advertisement, it generates a report containing its current GPS location. This location is then encrypted using the advertisement key broadcasted by the device, making it inaccessible to everyone except the owner. The encrypted report is uploaded to Apple's servers, with no way for Apple to decipher the actual location.
4. **Download and decrypt location reports**: Any device authenticated with the owner's Apple ID can request location reports linked to the tracker's keys. Using the matching advertisement key, the owner can decrypt these reports and see each location on a map.

Macless-Haystack gets rid of the need for any Apple hardware. All you need is an Apple ID with two-factor authentication via SMS and either a linux device or a supported bluetooth enabled microcontroller.

Add technical architecture diagram

## 2.2 The Stakeholders

The primary stakeholders are the end users who want to track their own Bluetooth devices via Apple's Find My network using Macless-Haystack's web browser. They provide an Apple ID, password, and SMS-based two-factor authentication once when setting up the Macless-Haystack endpoint container, and then interact with the system via a browser to import device key JSON files, view locations, filter by date, and export data.

End users are responsible for generating and managing tracker keys and deciding which devices are associated with their Apple ID; they are the primary beneficiaries of the system's tracking functionality.

The Macless-Haystack endpoint operator is the party running the backend container that logs into Apple's Find My service and periodically fetches encrypted location reports, typically on a server with Docker installed. In many cases this is the same party as the end user.

The Anisette server operator runs the separate Anisette-compatible service that Macless-Haystack connects to in order to emulate a real Apple device during login and communication with Apple. This role can again coincide with the endpoint operator in a self-hosted setup, but when a public or third-party Anisette instance is used, that operator becomes a distinct stakeholder who may see sensitive authentication-related traffic and must therefore be explicitly trusted.

Apple is an indirect but essential stakeholder, as its Find My and iCloud infrastructure receives the encrypted location reports from nearby Apple devices and serves them to the Macless-Haystack endpoint authenticated with an Apple ID.
Add stakeholders diagram

## 2.3 Types of Data Collected and Purposes

### 2.3.1 Data Collection in the Tracker

The tracker itself does not collect any data. It operates solely using the public key that is provided to it.

### 2.3.2 Data Collection in Apple's System

Because Apple's implementation is closed-source, we can only make informed assumptions about what data is collected. For completeness, we assume that Apple collects all data that is technically available to it.
Crowd-sourced Apple devices collect only three pieces of information when they detect a tracker:

- the advertisement key,
- the device's location at the moment of detection,
- and the Bluetooth signal strength at that time.

This data is normally deleted quickly, but users can delay deletion by placing their device in airplane mode. This data is placed on the apple servers, and used for people to retrieve the location of their tracker reference.

Apple stores the advertisement key together with its corresponding location report on their servers. From this, Apple can infer how many trackers are in use and estimate the general region in which they are being used, based on the uploader's IP address. Apple also knows the Apple ID associated with each device that uploads these reports. This means, at least in theory, that Apple could link a user to their tracker because the user's own Apple device will likely often be the one uploading the tracker's location reports.

Fetching location reports requires authentication. As a result, Apple can see which Apple account is hosting a macless-haystack endpoint and which trackers are used through that endpoint. This could allow Apple to terminate the Apple ID associated with the endpoint. The requirement for Apple ID authentication is likely in place to prevent abuse such as users uploading spam, overwhelming Apple's servers, or attempting to download large volumes of location reports that aren't their own.

### 2.3.3 Data Collected by the Macless-Haystack Application

The Macless-Haystack frontend stores very little data. The single-page browser interface keeps the private keys of imported trackers in secure local storage. It also stores the username and password used to authenticate with the macless-haystack endpoint; however, these credentials are stored in plain text.

#### 2.3.4 Data Collected by the Macless-Haystack API

The API stores only minimal data, but it logs a significant amount of sensitive information. This includes every request made by the application, which means that if the logs are reviewed, the advertisement keys of all Macless-Haystack trackers can be seen.

# 3 Privacy Impact Assessment

## 3.1 Threat Model

The system's privacy properties depend on several trust assumptions. Users must trust Apple's infrastructure to provide end-to-end encryption (though Apple has metadata visibility), the endpoint operator to protect stored credentials and logs, and any third-party Anisette server operator to handle authentication securely. The primary trust boundaries exist at: (1) Bluetooth broadcast between tracker and nearby devices, (2) network communication between frontend and backend (currently vulnerable when HTTP is used), (3) authenticated connections to Apple's servers (which reveal query metadata), and (4) filesystem access on the server (where plaintext credentials are stored). These boundaries represent the main points where information can be compromised, as detailed in the threat analysis below.

## 3.2 LINDDUN Privacy Threat Analysis

The following table presents a systematic analysis of privacy threats using the LINDDUN framework, which categorizes threats across privacy dimensions: Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness, and Non-compliance. Many compliance-related threats stem from the project's reverse-engineering of Apple's proprietary protocol and operation outside Apple's official ecosystem, which cannot be fully resolved without changing the project's purpose.

| Code | Threat Category | Description | Affected Interaction(s) | Severity | Mitigation |
|---|---|---|---|---|---|
| NC.3 / II.1 | Non-Compliance | Apple ID and password can be stored in plain text in config.ini (optional but supported). Exposes credentials to anyone with filesystem access (to the macless-haystack server). Similar for auth.json | Interaction 8, 14, 13 | Critical | Remove plain text password storage; require interactive login only |
| DD.4.2 / NC.2 | Data Disclosure / Non-Compliance | Storage of private and public keys in plain text. Keys should be stored using platform-specific secure storage (Apple Keychain, Android Keystore, GNOME Keyring, Windows Credential Manager). | Interaction 2, 3b | High | Implement secure key storage using platform APIs (e.g., flutter_secure_storage) |

| Code | Threat Category | Description | Affected Interaction(s) | Severity | Mitigation |
|---|---|---|---|---|---|
| L.1.1 / I.2.2.b | Linkability / Information Disclosure | Static advertisement keys allow tracking of device presence. The system broadcasts the same public key continuously, enabling anyone to determine if the tracked device (and by extension, its owner) is nearby. | Interaction 4 | High | Implement key rotation mechanism (firmware currently rotates every 30 min for multi-key setups) |
| NC.3.c | Non-Compliance | Weak authentication: no authentication required by default, or single username/password shared among all users of an endpoint instance. | Interaction 15, 16, 17, 24 | High | Implement per-user authentication; support OAuth or token-based auth |
| U.1.2 | Unawareness | Users are not informed that sharing their private key with others enables complete location tracking of their device. Lacks clear privacy warnings. | Interaction 2 | Medium | Add explicit privacy warnings in documentation and UI when exporting keys |
| NC.1.2.a | Regulatory Non-Compliance | Violation of Apple's EULA by creating custom AirTag-like devices using Apple's Find My network infrastructure. | Interaction 3a, 4 | Medium | Add legal disclaimer; users assume responsibility |
| D.3 | Detectability | Static advertisement keys allow querying Apple servers to determine if a tracker is still active by checking for new location reports, even without decryption. | Interaction 4, 20 | Medium | Key rotation partially mitigates this; inherent to Find My protocol |
| NR.1.1.a | Non-Repudiation | Apple ID is attached to all location report requests, allowing Apple to identify and track Macless-Haystack server hosters and their query patterns. | Interaction 8, 11, 12, 13, 20 | Medium | Inherent to Apple's authentication; no mitigation without protocol changes |

| Code | Threat Category | Description | Affected Interaction(s) | Severity | Mitigation |
|---|---|---|---|---|---|
| NC.1.2.b | Regulatory Non-Compliance | Reverse engineering and accessing iCloud authentication from non-Apple devices violates Apple's EULA | Interaction 11, 12, 13 | Medium | Legal disclaimer; users assume legal responsibility |
| L.2.2.1 | Linkability | Sending advertisement keys to potentially public endpoint links user's IP address with their tracked devices. | Interaction 15, 16 | Medium | Only use trusted, self-hosted endpoints; implement end-to-end encryption for API |
| NC.3.b | Non-Compliance | HTTP access is supported (non-mandatory HTTPS), exposing advertisement keys during transmission. Man-in-the-middle attackers can capture key. | Interaction 15, 16, 24 | Medium | Enforce HTTPS mandatory; disable HTTP in production |
| I.1.2 | Information Disclosure | Apple servers receive query metadata (timestamps, query frequency, advertisement key hashes) revealing usage patterns. | Interaction 20, 21 | Medium | Inherent to protocol; rate limiting and query batching can reduce exposure |

## 3.3 GDPR Compliance Considerations

Macless-Haystack processes location data, which constitutes personal data under GDPR when relating to identifiable EU individuals. For personal use, GDPR obligations are minimal under the household exemption (Article 2(2)(c)). However, organizational deployments require full GDPR compliance, including appropriate security measures (Article 32), data subject rights support (Articles 15-22), and privacy by design (Article 25). The current implementation fails to meet several GDPR requirements: plaintext credential storage and optional HTTP violate security obligations (Article 32), while the lack of data export/deletion mechanisms prevents users from exercising their rights (Articles 15-22).

## 3.4 Main Identified Privacy Threats

### 3.4.1 Advertisement Keys in Macless Haystack

In the current setup, each tracker continuously broadcasts a beacon containing the same static public key so that nearby Apple devices can hear it and create encrypted location reports. Any Apple device that receives such a beacon treats the advertisement key as a valid identifier, takes its own location and encrypts that location to send it to Apple's servers, where only the matching private key can decrypt it. Using a static public key introduces a serious linkability privacy problem. Because the same identifier is broadcasted in every packet, any nearby device with a simple BLE sniffer can record this constant over time and across different places, and then correlate all sightings as belonging to the same physical tracker. This means that anyone could build a detailed movement profile of the tagged object or person just by re-observing the same static key at different locations, even though they cannot decrypt the underlying Find My location reports. The root cause of the privacy threat is therefore not that the encryption is weak, it is that the identifier used as input to the encrypted reporting process is long-lived and public. The encryption protects the content of

each report from being read, but it does nothing to stop third parties from recognizing that "this is the same beacon as yesterday" and inferring sensitive patterns (home, workplace, daily routines) purely from repeated radio observations tied to an unchanging public key.

#### 3.4.2 Unit Test for Rolling Codes

A dedicated testing environment and accompanying test suite were created to evaluate and validate privacy-related behaviors in the tracker firmware, which had not previously undergone testing. To determine whether rolling keys were implemented, the run_openhaystack function is invoked repeatedly while a shim replaces the Bluetooth advertising routine, enabling direct inspection of the broadcast payloads. Rolling codes are deemed absent if each invocation of run_openhaystack consistently triggers the BLE controller with identical payload.

add picture of unit test

#### 3.4.3 Plain Text Credentials

The user's Apple ID credentials are stored in plaintext on the Macless server inside the auth.json file. This means that anyone with file access to the server can read the user's credentials. This represents a serious security vulnerability that must be addressed immediately, especially when the server and client are managed by different individuals. Additionally, the frontend website is not protected by HTTPS, and communication between the frontend and the Macless Haystack server is transmitted without encryption. This creates a severe security risk if a user enters their credentials for the macless haystack server into the frontend, as the information would be sent over the network unencrypted. While the current implementation does not require users to enter their credentials, since they are provided during server setup, an uninformed user might still attempt to do so. In scenarios where the server is configured by a third party, entering credentials through the website may even be necessary for the system to function, further increasing the risk.

Add unit test for plain text credentials

# 4 Recommendations

This section discusses how to address the privacy issues from Part 2.

## 4.1 Rolling Codes

Rolling codes (or rotating keys) address the linkability problem by changing the public key that is broadcasted over time, while still allowing the real owner to decrypt all the corresponding location reports. Instead of a single static key pair, the tracker starts from a master secret and uses it to derive a sequence of public-private key pairs, for example one per 15-minute time window. At any given moment, the beacon only broadcasts the current public key, and nearby Apple devices encrypt their location reports to that key exactly as before, then upload the location report to the server. To outsiders passively listening on Bluetooth, these public keys appear as unrelated random values: the identifier at 10:00 and the identifier at 10:15 look completely different, and there is no way for them to tell that both belong to the same physical tracker. In contrast, the owner, who knows the master secret, can recompute the full schedule of rolling keys for all past time slots, query the server for reports associated with each of those keys, and decrypt them with the corresponding private keys. In other words, rolling codes preserve end-to-end encryption for the owner but break the simple "always the same identifier" signal that enables third-party tracking with static keys. Adopting rotating keys would therefore significantly reduce the privacy risk: it becomes much harder for random observers, or for large sensor deployments, to follow a specific tracker over long periods, because the broadcast identifier they see keeps changing in a way they cannot link back to one device.

## 4.2 Solution to second threat

# 5 Conclusion

Add conclusion summarizing key findings and recommendations

# References

[1] dchristl. Macless-haystack. GitHub repository. Available at: https://github.com/dchristl/macless-haystack. Accessed: 20 December 2025.

[2] seemoo lab. Openhaystack. GitHub repository. Available at: https://github.com/seemoo-lab/openhaystack. Accessed: 20 December 2025.